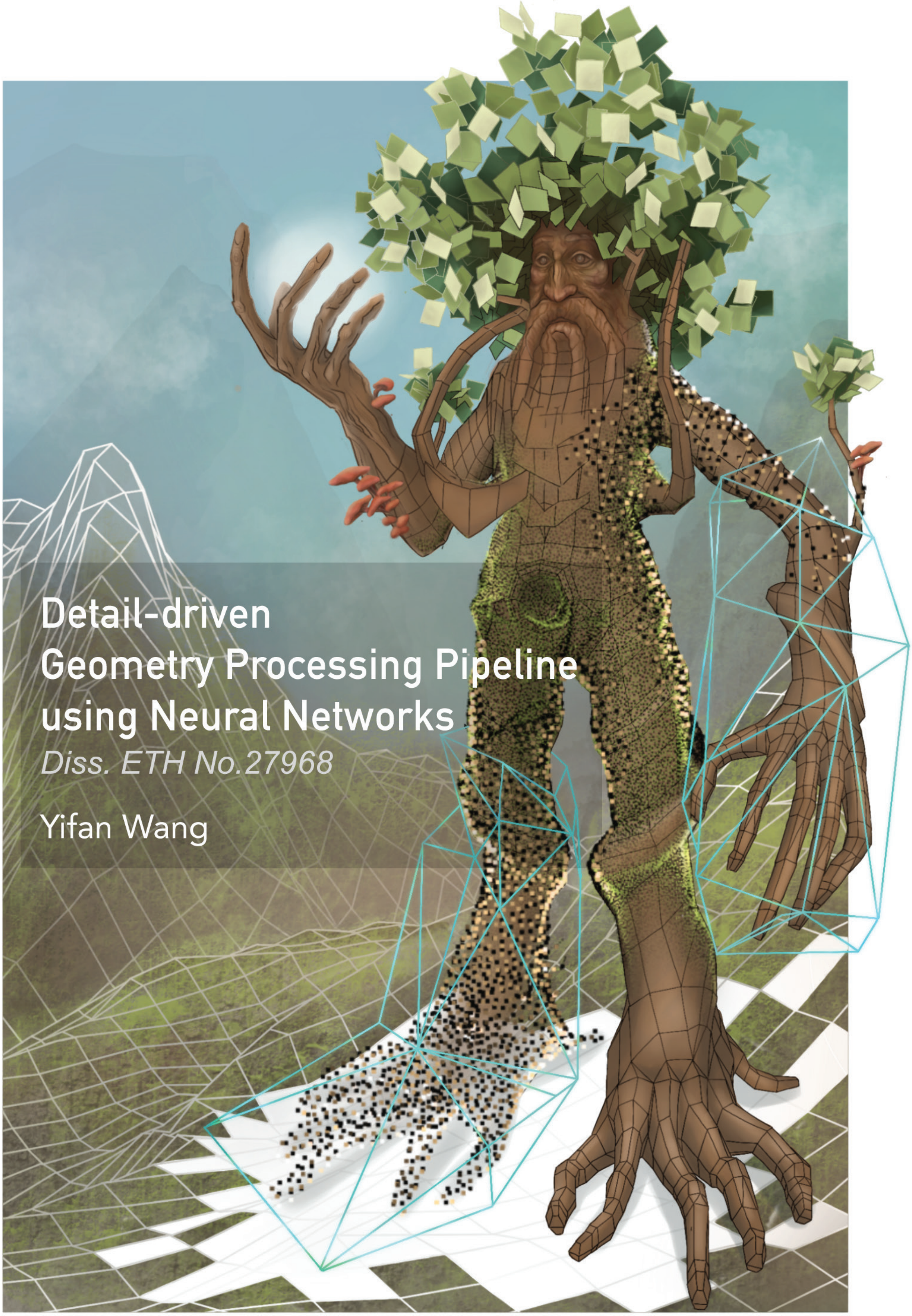


Detail-driven Geometry Processing Pipeline using Neural Networks

Diss. ETH No.27968

Yifan Wang

Diss. ETH No. 27968

Detail-driven Geometry Processing Pipeline using Neural Networks

A thesis submitted to attain the degree of
Doctor of Sciences of **ETH Zurich**
(Dr. sc. ETH Zurich)

presented by
Yifan Wang
MSc in Robotics, Systems and Control, ETH Zurich
born on 06.09.1990
citizen of China

accepted on the recommendation of
Prof. Dr. Olga Sorkine-Hornung, examiner
Prof. Dr. Cengiz Öztirelli, co-examiner
Prof. Dr. Gordon Wetzstein, co-examiner
Prof. Dr. Niloy Mitra, co-examiner

2021

Abstract

Geometry processing is an established field in computer graphics, covering a variety of topics that embody decades-long research. However, with the pressing demand of reality digitization arising in recent years, classic geometry processing solutions are confronted with new challenges.

For almost all geometry processing algorithms, a fundamental requirement is the ability to represent, preserve and reconstruct geometric details. Many established and highly-optimized geometry processing techniques rely heavily on educated user inputs and careful per-instance parameter tuning. However, fueled by the proliferation of consumer-level 3D acquisition devices and growing accessibility of shape modeling applications for ordinary users, there is a tremendous need for automatic geometry processing algorithms that perform robustly even under incomplete and distorted data. In order to transform existing techniques to meet the new requirements, a new mechanism is called for to distill the user expertise in algorithms.

This thesis offers a solution to the aforementioned challenge by utilizing a contemporary technology from the machine learning community, namely: deep learning. A general geometry processing pipeline includes the following key steps: raw data processing and enhancement, surface reconstruction from raw data, and shape modeling. Over the course of this thesis, we demonstrate how a variety of tasks in each step of the pipeline can be automated and, more importantly, strengthened by incorporating deep learning to leverage consistencies and high-level semantic priors from data.

Specifically, this thesis proposes two point-based geometry processing algorithms that contribute to the raw data processing step, as well as two algorithms involving implicit representations for the surface reconstruction step, and one shape deformation algorithm for the last shape modeling step of the geometry processing pipeline. We demonstrate that, by designing suitable deep learning paradigms and integrating them in the existing geometry processing pipeline, we can achieve substantial progress with little or no user guidance especially for challenging, e.g. noise-ridden, undersampled or unaligned, inputs. Correspondingly, the contributions in the thesis aim to enable autonomous and large-scale geometry processing and drive forward the ongoing transition to digitized reality.

Zusammenfassung

Geometrieverarbeitung ist ein etabliertes Feld in der Computergrafik, das eine Vielzahl von Themen abdeckt, hinter welchen jahrzehntelange Forschung steht. Mit der enormen Nachfrage nach der Digitalisierung der Realität in den letzten Jahren, stehen klassische Lösungen der Geometrieverarbeitung jedoch vor neuen Herausforderungen.

Für fast alle Geometrieverarbeitungsalgorithmen ist die Fähigkeit, geometrische Details darzustellen, zu erhalten und zu rekonstruieren, eine grundlegende Anforderung. Viele etablierte und hochoptimierte Geometrieverarbeitungstechniken verlassen sich stark auf geschulte Benutzereingaben und eine sorgfältige Parameterabstimmung pro Objekt. Angetrieben durch die Verbreitung von 3D-Erfassungsgeräten auf Verbraucherebene und die zunehmende Verfügbarkeit von Anwendungen zur Formmodellierung für normale Benutzer, entsteht ein enormer Bedarf an automatischen Geometrieverarbeitungsalgorithmen, die selbst bei unvollständigen und fehlerbehafteten Daten robust arbeiten. Um bestehende Techniken an die neuen Anforderungen anzupassen, ist ein neuer Mechanismus erforderlich, um das Fachwissen fortgeschrittener Benutzer durch Algorithmen zu ersetzen.

Diese Doktorarbeit bietet eine Lösung für die oben genannte Herausforderungen durch den Einsatz einer zeitgemäßen Technologie aus dem Bereich Machine Learning, nämlich Deep Learning. Eine allgemeine Geometrieverarbeitungspipeline umfasst die folgenden wichtigen Schritte: Rohdatenverarbeitung und -verbesserung, Oberflächenrekonstruktion aus Rohdaten und Formmodellierung. Im Verlauf dieser Dissertation zeigen wir, wie eine Vielzahl von Aufgaben in jedem Schritt der Pipeline automatisiert und, was noch wichtiger ist, durch die Integration von Deep Neural Networks verbessert werden können, um Konsistenzen und hochrangige semantische Muster in Daten auszunutzen.

Konkret schlägt diese Doktorarbeit zwei punktbasierte Algorithmen zur Geometrieverarbeitung von Rohdaten vor, zwei Algorithmen mit impliziten Darstellungen für die anschließende Oberflächenrekonstruktion und einen Deformationsalgorithmus für die abschliessende Formmodellierung. Indem wir geeignete Deep Learning Paradigmen konzipieren und diese intelligent in die bestehende Geometriebearbeitungspipeline integrieren, zeigen wir, dass wir mit geringer bis gar keiner Benutzerführung wesentliche Fortschritte erzielen können, besonders für anspruchsvolle, z.B. rauschbehaftete, spärliche oder nicht ausgerichtete Daten. Entsprechend treiben die Beiträge dieser Dissertation zur automatische Geometriebearbeitung grosser Datenmengen den fortschreitenden Übergang zur digitalisierten Wirklichkeit.

Acknowledgments

My greatest fortune is to be the daughter of an incredible woman. Mom is the very personification of “grit”. She taught me, in her action, to take pride in choosing the difficult path, to value perseverance more than cleverness, and to always proactively search for the next tougher challenge. Mom, I am who I am because of you and I'll be progressing from who I am also because of you. Thank you, mom.

I'm also extremely fortunate to be the academic daughter of another incredible woman, Professor Olga Sorkine-Hornung. She's my role model in practically every aspect I can think of. I admire her as a professor, a researcher, a leader, a mentor, a wife, a mother, and a friend, and a human being. In fact, in the past four years, she made IGL a place like home, a haven where I could lay bear my fear, sorrow, and frustration. Olga, thank you for showing me what is possible.

During my thesis, I had the opportunity to collaborate and work with many people from different organizations, including Disney Research, Beijing Film Academy, Shenzhen University, and Adobe Research. Each experience helped to shape my Ph.D. journey profoundly. I want to particularly thank Dr. Shihao Wu, who taught me how to do research, for his unparallel determination and optimism in the deadline frenzy. I owe a lot to Dr. Alexander Sorkine-Hornung and Dr. Christopher Schroers, who motivated and set off my Ph.D. journey; to Professor Daniel Cohen-Or, who enlightened me with his unceasing passion and curiosity in research; also to Dr. Vladimir Kim, who gave me an unforgettable summer in Seattle and inspired me deeply with the depth and breadth of his expertise.

The 8 years in Zurich have been an absolute highlight of my life. It is fair to say that I have truly evolved as a person in this small but beautiful city, all thanks to my dear, brilliant and crazy friends. I'm proud to be a friend of two beautiful, talented, hardworking, and courageous girls, Ana and Milica. I adore my clever, charming, and creative friend and flatmate, An-phi, and my extremely inspiring yet humble friend, Wookie. I cherish the beautiful bond and sleepless nights shared by the members of the infamous Langstrassegang. I'm also thankful to Federico and his family for their support and love.

I want to thank the current and past of the members of the visual computing institutes. To Danielle, and also Marianna, you are part of the reason why IGL feels like home. To Katja, Floor, Shihao, Philipp, Oliver, Christian, Michael, Alexandre, Peizhuo, it’s my greatest pleasure to not only work with you but also get to know your unique and colorful personalities and jaw-dropping talents.

I’m very lucky to be the “little friend” of Adrien. Your generosity, positivity, sense of responsibility amaze me every day. Thank you for being who you are. I’m also extremely honored to get know the members of the Kerroux family, each one of them a brilliant superstar with fascinating talents and characters.

Finally, my special thanks go to Anaïs, who drew the most beautiful dissertation cover in the history of science.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Sweeping digitization in our everyday life has been fundamentally reshaping the way we experience and interact with the world – tangible cultural heritage can be accessed and studied from the opposite side of the globe in immersive digital formats [Uni21]; personal avatars capture increasingly subtle motions to establish seamless human interactions despite geographical distances [AG21; Inc19]; 3D models of organs can be created and modeled for telemedicine [Est+21]; and immersive labs are created to promote equal access to science [Lon; LB20]. Wherever we look, it is fair to say that we are living amid a transition where the boundary between the real (physical) and virtual (digital) world is becoming blurrier. Recently, the global pandemic and the rising environmental threats not only bestowed new meanings to this digital transition but also gave it hightened urgency.

As we humans perceive the surrounding environment in three dimensions, weaving virtual threads into the fabric of reality calls for large amount of high-quality 3D assets. Needless to say, the fidelity of these artifical 3D content compared to their real-life counterparts have a profound impact on the usability of the application in question. In particular, accurately representing and reconstructing detailed geometry features is not only key to creating realistic 3D scenes in AR/VR applications, but also critical in robotics for analyzing the physical properties of the subjects in order to correctly model their interactions with the real world.

However, constructing geometric details is very difficult. Creating them from scratch is a tremendously demanding manual work and requires highly specialized skills. Even though many sophisticated softwares and intuitive modeling tools [Gla+16; Jac+14; VSH19] have been developed to assist the artistic creation, scalability issues still prohibit large-scale general-purpose deployment. Alternatively, geometric details can be captured from the real world. But since acquisition technologies are inevitably subject to hardware limitations and external interferences, the captured geometry is prone to various artifacts including noise and incompleteness. Consequently, post-processing algorithms must be applied to reconstruct and enhance geometric features.

At the same time, once the geometric details are constructed, preserving them during further manipulation and editing is also challenging. While numerous prior works have proposed outstanding solutions regarding this issue in the context of surface parameterization, meshing and deformation, many of them are designed for interactive shape modeling and thus require user guidance to achieve the desired outputs. There are also exemplar-based shape manipulation, where the necessary information for a desired output is extrapolated from user-provided exemplars instead of extensive fine-grained user inputs. While these approaches have potential to scale up for an automated pipeline, the quality of outputs is at the mercy of the compatibility of the exemplars. In order to produce plausible outputs even from suboptimal exemplars, prior knowledge must be injected, which typically relies on direct control inputs from the user or highly involved parameter tuning.

In face of these challenges, the incredible demand to push forward the digital transition calls for new algorithms that can efficiently capture, reconstruct, create and manipulate detailed 3D geometries. In order to account for underconstrained working environment in everyday use, particularly sought after are algorithms that exhibit good generalization properties and minimal dependency on user interventions.

Deep learning and neural networks.

Deep learning is a subset of machine learning and lies in the center of artificial intelligence. It is designed to enable computers to learn complex concepts by hierarchically extracting knowledge from observations of the world. The hierarchy is typically very deep, hence the name "deep" learning.

The most important machinery in deep learning is neural networks. Loosely modeled on a human brain, neural networks comprise many computation layers, each of which consists of many densely connected computation nodes, assembling neurons in a human brain. The actual functionality of the neural networks is governed by the millions of values stored in the neurons, called "weights". Even though each neuron only performs simple binary operations, by the universal approximation thereom [Cyb89; Hor91; Has+95; HL94], the collection of these simple neurons can approximate arbitrarily complex functions when given appropriate weights. In other words, input signals from raw data can be transformed to abstract and complex concepts, with each layer of the neural network distilling knowledge gradually from simple and local to abstract and global. The term "learning" refers to the optimization of the neural networks' weights such that given some input values the output of the neural network fits an expected reference. For instance, in image classification the input is RGB values of the image pixels, and the reference can be the probability of the image belonging to different classes; or in image super-resolution, the input can be pixel values of a downsampled image and the reference is pixel values of the original image.

There are two reasons that make deep learning the promising instrument to address the needs for efficient and robust geometry processing. First, as the hallmark of deep learning, it is able to not only learn the mapping from certain input representation to output, but also the representation itself. As a result, there's no longer need to craft hard-coded rules in order to extract useful information from the raw data. Instead, neural networks learn to transform the raw data to the suitable representations for the task at hand. Compared to hand-crafted representations, these learned representations can be more generalizable and robust, as they can capture statistical characteristics across large dataset, *e.g.* implicitly learn the consistencies within the training data and automatically account for noise and other variances of the input data. Second, neural networks requires little to none human intervention during the deployment, also known as the inference phase. This is because once the weights of the network have been optimized in the training phase, the inference typically only requires a feed-forward pass, *i.e.* the input data moves through the fixed layers without additional tuning or adjustment from the user. Furthermore, thanks to contemporary specialized hardware and recent network optimization techniques, this feed-forward pass can be very fast and efficient, ergo making large-scale online applications possible.

Due to these attractive properties, neural networks have become the Swiss army knife for many long-standing problems in computer vision and natural language processing. Particularly in image processing, where details are equally important as in geometry processing, deep learning has become the state-of-the-art approach for most tasks.

However, compared to image and natural language processing, 3D content poses new challenges for neural networks. First, 3D data is presented in a wide range of heterogeneous forms catering for different applications and needs. For instance, as the primary outputs of acquisition devices, point clouds are the most common form in 3D understanding and reconstruction tasks, while meshes, which can be divided further into surface and volumetric meshes, are mainly used for geometry manipulation for their compactness and explicitness w.r.t. shape topology. Irregularity is another problem related to the data form. Unlike word embedding and 2D pixels, most common 3D data forms are unstructured or have irregular connectivities. Consequently workarounds must be created to allow the neural networks to traverse through the data efficiently, and more importantly, some backbone operations such as convolution, which is the key contributor to the success of deep learning in the 2D domain and is universally used in all image processing neural networks, must be redesigned to handle the irregular input forms. Last but not the least, due to higher acquisition requirements, the amount and variety of 3D data for training neural networks is significantly insufficient. For a method that excels by exploiting vast amount of data, data scarcity creates a serious hurdle for deep learning to advance rapidly in the 3D domain.

During the course of this thesis, extensive effort and resources from different corners of the community have been dedicated to address the aforementioned challenges in hope to advance the state-of-the-art geometry processing with deep learning. The work included in this thesis is part of this joint effort.

1.1. Topics in this thesis

A typical geometric processing pipeline comprises the following components:

1. *data acquisition*, which concerns the capture of point clouds or volumetric data representing the surface or volume of the object of interest;
2. *data processing and enhancement*, which includes the post-acquisition treatment of defects in the acquired raw data, such as noise, distortions and topological errors, as well as the enhancement of geometric features, such as sharp edges, *etc.*;
3. *surface reconstruction*, which focuses on the approximation of surfaces, most commonly in form of triangle meshes, from the (processed) acquisition data;
4. *shape manipulation*, which covers various low-level surface processing tasks such as surface smoothing, parameterization, remeshing, and high-level shape modeling tasks such as deformation and stylization.

This thesis touches upon all the aspects of a geometry processing pipeline after data acquisition, namely data processing and enhancement, surface reconstruction and shape manipulation. In all these aspects, we focus on elevating the geometric details in the outputs, while at the same time reduce the reliance on human intervention and optimize for efficiency and generalizability.

1.1.1. Raw data processing and enhancement

Defects are omnipresent in raw data [BW10]. They arise from hardware limitations and environmental constraints, but also from problematic surface properties such as high reflectivity and self-occlusion. The most common defects include holes, noise and distortions. The existence of these distortions can gravely affect the outcome of surface reconstruction as well as other downstream tasks. Moreover, due to inherent undersampling problem at edge singularities [Hua+13a], reconstructing surfaces directly from raw data, even if noise-free, often leads to blurred sharp features and oversmoothed geometric details. Hence approporiate treatment is required to repair and enhance the acquired raw data for accurate detail representation in the subsequent processing steps such as surface reconstruction or rendering.

Existing data acquisition methods either yield point clouds or volumetric data such as depth map and voxels. In this thesis we focus on point cloud processing and enhancement, a process also known as point cloud consolidation. In traditional point-based geometry processing, point cloud consolidation is formulated as a projection problem. The acquired 3D points, treated as noisy samples of an underlying surface, are reprojected onto the surface by minimizing an error metric that essentially measures point-to-surface distances. In essense, these methods rely on the fitting of local geometry, e.g., normal estimation, using local point distribution. In order to resolve ambiguity in the solution space, hand-crafted priors such as smoothness assumptions *etc*. must be injected into the formulation *e.g.* as regularizers. As a result, the balance between sufficient outlier removal and successful feature recovery (such as corners and edges) requires careful tuning of these regularizer terms in the optimization objective based on subjective assessment. Consequently, these methods struggle with robust multiscale structure preservation under sparse sampling conditions and strong noise.

In this thesis, we take on a data-driven approach with the aim to learn the mapping between the flawed and correct point distribution by observing examples from data. Instead of addressing the ambiguity with hand-crafted priors (consequently leading to algorithm bias), we use neural networks to learn more generalizable priors from large amount of data, thus achieving improved robustness particularly under challenging sampling conditions. The focus of our approach is to overcome the shortcomings of neural networks in operating with unstructured data. Two methods are presented in this thesis, which pursue this mission from two distinctive angles, nonetheless both succeed by exploiting the advances of deep learning in the image processing domain.

1.1.2. Surface reconstruction

Surface reconstruction refers to the procedure to convert point clouds and volumetric data to polygon meshes. Polygon meshes are efficient to store and render, the well-defined connectivity facilitates efficient evaluation of intrinsic geometric properties, such as topology, geodesic distance, *etc*. Therefore, polygon meshes are regarded as the standard representation for most common geometry processing and shape modeling algorithms. Consequently, surface reconstruction is an essential step in the general geometry processing pipeline.

Existing surface reconstruction methods can be categorized into explicit and implicit reconstructions. The explicit approaches either directly establish connectivity from the point samples (in case of point clouds data) [TL94], or determine the interior/exterior separation by casting the problem as 3D segmentation task (in case of volumetric data) [SD99; FLBMB90; BBH08; Vog+07; GCS06]. The implicit approaches [Hop+92a; Car+01; SOS04; KBH06; KH13a; Kaz+20], on the other hand, use scalar functions (*e.g.* signed distance functions) as an intermediator, from which the surface can be extracted as a levelset of the implicit functions, *i.e.* $\{\mathbf{x} | f(\mathbf{x}) = C\}$. Implicit approaches are less sensitive to noisy and misaligned data, and more advantageous for detail reconstruction since they are free from resolution constraints during discretization. However, an additional step is required to extract the final polygon meshes from the fitted implicit function. Marching Cubes [LC87] is the most widely used method for this purpose, which determines the mesh vertices (and their connectivity) by approximating the roots of the implicit function from grid samples.

Given that the nature of implicit surface reconstruction is a function approximation problem, neural network's potential as a universal approximater [Cyb89; Hor91; Has+95; HL94] makes it a perfectly viable alternative solution for surface reconstruction. Several works [Par+19; Mes+19; CZ19] concurrently demonstrated this proposition by parameterizing the implicit function with neural networks. Given 3D coordinates as inputs, the network is trained to output the corresponding signed distance value [Par+19] or occupancy probability [Mes+19; CZ19], therefore this type of networks are also referred to as coordinate-based networks. Compared to previous works, neural networks can potentially represent a larger range of functions than a mixture of polynomials or Gaussians (as is previously the case). More importantly, by training from collections of shapes, the networks can capture useful shape priors from the consistencies within the training dataset. As a result, neural implicit functions can fill large missing area with plausible geometries. Building upon this idea, some recent works extended the neural network to approximate not only the implicit surface function but also the surface texture and even view-dependent appearance in a similar manner. These methods paved the way for simultaneous reconstruction of surface geometry and appearance from 2D observations – one of the most studied tasks in computer vision, commonly referred to as multiview surface reconstruction (MVS). By considering geometry and appearance jointly, these neural networks can achieve superior reconstruction quality even under complex lighting and challenging surface material, where the traditional MVS methods struggle due to the difficulty to establish consistencies between different views.

As neural implicit surface representation is emerging as a prominent basis for many long-standing tasks in computer graphics and vision alike, the research community is dedicating significant attention to improve the generalizability, robustness, efficiency as well as the representational power of neural implicit surfaces. This thesis contributes to this research direction with two methods, which address respectively the training efficiency and noise tolerance, and the efficacy in regard to geometric detail representation.

1.1.3. Shape manipulation

There are a plentiful of shape manipulation operations. Remeshing, mesh simplification and parameterization are a few examples of low-level shape manipulation operations, which serve as the foundation to efficiently perform further high-level shape manipulation tasks, such as shape deformation, animation, stylization *etc*.

In this thesis, we focus on shape deformation, which refers to the task to transform a given shape to a match specific pose or another aligned shape without altering the topology (which prescribes a continuous and globally bijective transformation) and geometric features (which requires the transformation to be smooth). It is one of the most important shape modeling tasks, and has been actively researched in the last decade. Automating shape deformation provides an alternative way to generate new shapes efficiently by deforming existing shapes. It has many appealing applications including 3D stock amplification, automatic design and 3D character posing.

The main research focus for shape deformation has been about improving the algorithm efficiency and reducing geometric distortion, so as to achieve highly realistic real-time deformation for interactive shape modeling. It is typically assumed that the deformation target is provided by the user either directly or via an intermediary shape, in which case the deformation is driven by sparse correspondences that can be specified by the user. Correspondingly, whether the deformation is contextually correct and semantically feasible lies in the user's responsibility. The need for human guidance makes designing an automated deformation paradigm very challenging, since such paradigms must be able to implement abstract semantic knowledge.

To this end, several recent works started deploying deep learning for semantic shape understanding, such as part segmentation and correspondence estimation [GEM18; Mo+19; Wu+19; Gro+18b]. The learned high-level semantic knowledge can be injected to deformation tasks. In essence, deep neural nets are used in these approaches to learn a feasible deformation space of certain shape categories by extracting data priors from compatible shape pairs. While these learning-based methods have accomplished impressive progress in correspondence matching, they seem to underperform in terms of distortion minimization and geometric feature preservation - a caveat, which may be ascribed to suboptimal network architecture and the difficulty to come up with computationally efficient loss terms that are equivalent to the instance-specific distortion energy used in classic non-data-driven approaches.

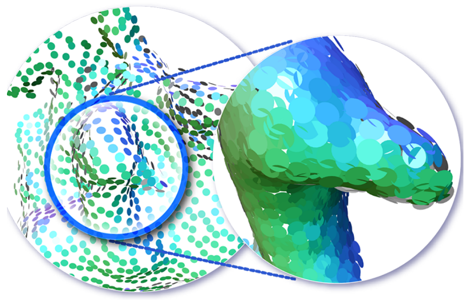
In this thesis, we propose a novel shape deformation method, which on one hand leverages priors learned from large dataset to generate plausible deformations of a given shape category, on the other hand considerably improves the quality of the deformation w.r.t. geometric detail preservation via a novel network architecture that is feature-preserving by construction.

1.2. Contributions of this thesis

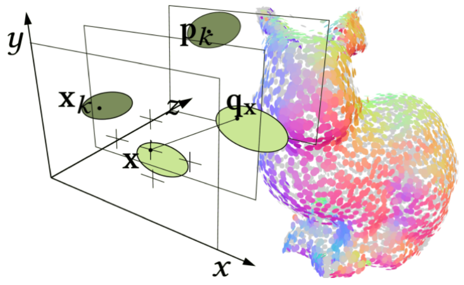
This thesis contributes to the instigation of *deep* geometric processing and advances the general geometry processing pipeline in all its key steps with five algorithms. The specific contributions are outlined as follows.

1.2.1. Point cloud processing and enhancement

We propose two algorithms for point cloud processing and enhancement. In the first one, we focus on point cloud upsampling and propose a novel neural network to upsample point clouds even for large upscaling ratios (see Fig. 3.1). In the second one, we focus on more general point cloud processing, where we propose to manipulate the attributes of a given point set, *e.g*. the points' positions, normal directions and colors, using 2D inputs via a differentiable point renderer.



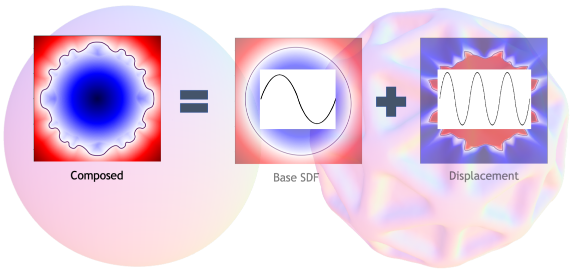
Both works focus on lifting neural network's weakness in regards to processing unstructured data. The former does so by transferring and adapting mature network designs and training strategies deployed successfully in image super-resolution. Particular care is put into the architectural design such that geometric structures across multiple scales can be simultaneously attended to, and at the same time information from a more global context flows efficiently to guide the output of the local reconstruction. Correspondingly, the proposed method demonstrates superior performance in terms of reconstruction accuracy for large upscaling ratio compared to previous state-of-the-art point upsampling methods, data-driven or not; and it is able to produce plausible structures robustly even for severely undersampled point clouds.



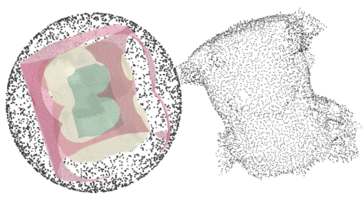
In the second work, we directly take advantage of abundant image filtering techniques by designing a novel point differentiable renderer, which propagates edits from the 2D renderings to point positions in the 3D domain. It's the first fully differentiable point cloud renderer, which is differentiable w.r.t. both the point positions and the point normals. The main technical contribution of this work lies in 1) defining a surrogate gradient for the rasterization function, which is discontinuous and thus non-differentiable per se; 2) proposing two regularization terms to address optimization local minima, mitigating non-surfacial point distribution and point clustering artifacts. We show that the proposed differentiable point renderer can be successfully applied to perform both large-scale deformations as well as more fine-grained surface filtering. More importantly, it can be seamlessly integrated with any neural networks in the 2D domain, thus opening new possibilities for creative point cloud processing.

1.2.2. Implicit surface reconstruction

We focus on the recent line of surface reconstruction approaches, which represent surfaces with implicit functions that are parameterized with neural networks. Two algorithms are proposed.



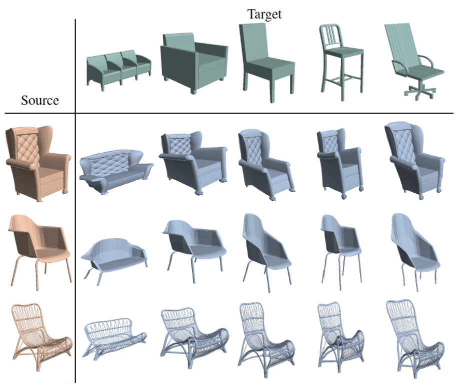
The first one focuses on the representation itself. More concretely, It tackles the well-known limitation of neural networks in handling high-frequency signals, termed as spectral bias [Rah+19], which inevitably leads to insufficient detail reconstruction. The key idea is a novel factorization of the neural signed distance function, which is inspired by displacement mapping - a classic technique to model surface details in computer graphics. The main technical contribution lies in extending the classic displacement mapping, which is discrete and lies only on the base surface, to a continuous function in the $\mathbb{R}^3$ domain and incorporating it into contemporary neural implicit representations. The resulting factorization, which we call *implicit displacement field*, is an extremely compact surface representation and has a significantly more stable convergence performance during the fitting phase. More importantly, it demonstrates excellent representational power for high-frequency geometric features while showing superior memory efficiency.



The second algorithm focuses on the training efficiency and noise tolerance of neural implicit surfaces. We propose to use "iso-points" as an explicit complementary representation to a neural implicit function, which allows us to impose geometry-aware sampling and regularization that can significantly improve the fidelity of reconstructions. The main technical contribution of this work is a pipeline, where the iso-points can be computed and updated on-the-fly during training to capture important geometric features and impose geometric constraints on the optimization. We demonstrate that our method can be adopted to improve state-of-the-art techniques for reconstructing neural implicit surfaces from multi-view images or point clouds.

Quantitative and qualitative evaluations show that, compared with existing sampling and optimization methods, our approach allows faster convergence, better generalization, and accurate recovery of details and topology.

1.2.3. Shape deformation



We propose one algorithm for shape deformation. In this work, we propose a neural network to warp a source shape to match a target shape, which can be topologically and geometrically very dissimilar, without dense correspondence estimation. The key contribution of our approach is to address the feature distortion issue of existing learning-based deformation methods, which requires simultaneously optimizing two competing objectives: 1. close alignment with the target, 2. preservation of local geometric features of the source. To this end, our method extends a traditional cage-based deformation technique, where the source shape is enclosed by a coarse control mesh, termed cage, and translations prescribed on the cage vertices are interpolated to any point on the source mesh via special weight functions. The use of this sparse cage scaffolding enables preserving surface details regardless of the shape's intricacy and topology. The proposed method succeeds in generating feature-preserving deformations for synthesizing shape variations and deformation transfer, and better preserves salient geometric features than competing methods.

1.3. Thesis outline

The dissertation is divided in 6 chapters. The remaining chapters are organized as follows.

Chapter 2 presents an overview of previous works in the relevant topics of the general geometry processing pipeline (namely point cloud processing, surface reconstruction and shape deformation) and provides a more detailed discussion about research that is closely related to our work.

Chapter 3 presents two algorithms for point cloud processing and enhancement – one point cloud upsampling method using neural networks and one point cloud transformation method using inverse rendering. Accordingly, the chapter is divided in two parts. In the first part, we describe the design and training of a multi-scale progressive point upsampling neural network and showcase the upsampling results for various challenging point cloud inputs. The second part introduces the definition of point renderer and identifies the problem of non-differentiability introduced during rasterization. It then describes our solution, including the definition of a surrogate gradient where the rendering function is non-differentiable and two optimization regularizers to improve the point distribution on surfaces. The effectiveness of this differentiable point renderer is then demonstrated in multi-view surface reconstruction, image-based point cloud filtering and denoising.

Chapter 4 presents two algorithms for surface reconstruction. A reparameterization for neural implicit surface is introduced in the first section. Implicit displacement field, a novel neural surface representation, is formally defined by extending displacement mapping. Then we outline the network architecture and training scheme designed specifically for implicit displacement field and demonstrate the advantage of this new representation over competitive methods in surface reconstruction task. Finally we introduce transferability to implicit displacement field and showcase its application in detail transfer. In the second section, we focus on surface reconstruction from noisy and incomplete inputs. A hybrid neural surface representation is developed, which uses iso-points as an explicit representation for a neural implicit function. We describe the mechanism to efficiently extract iso-points during network optimization and demonstrate the utility of iso-points for reconstructing neural implicit surfaces from multi-view images or noisy point clouds

Chapter 5 presents one algorithm for automatic shape deformation, which utilizes cage deformation in interactive shape modeling to improve the preservation of geometric details. It starts with a brief overview of the principles of cage-based deformation, and then outlines the necessary steps to incorporate these principles in neural networks so as to learn cage-based deformations from collections of shapes. The utility of the novel deformation method is demonstrated in two applications: 1. we generate shape variations by deforming a 3D model using other shapes as well as images as targets; 2. we also use our method to pose a human according to a target humanoid character, and, given a few sparse correspondences, perform deformation transfer and pose an arbitrary novel humanoid.

Chapter 6 summarizes our contributions and reflects on potential avenues for future work.

1.4. Publications

In the context of this thesis, the following work has been published:

[Yif+19a]	Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. "Differentiable Surface Splatting for Point-based Geometry Processing". In: *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA)* 38.6 (2019).
[Yif+19b]	Wang Yifan, Shihao Wu, Hui Huang, Daniel Cohen-Or, and Olga Sorkine-Hornung. "Patch-based Progressive 3D Point Set Upsampling". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5958–5967 (cit. on pp. 16, 90, 91).
[Yif+20a]	Wang Yifan, Noam Aigerman, Vladimir G. Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. "Neural Cages for Detail-Preserving 3D Deformations". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.
[Yif+21]	Wang Yifan, Shihao Wu, Cengiz Oztireli, and Olga Sorkine-Hornung. "Iso-Points: Optimizing Neural Implicit Surfaces With Hybrid Representations". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 374–383.
[YRSH21]	Wang Yifan, Lukas Rahmann, and Olga Sorkine-Hornung. *Geometry-Consistent Neural Shape Representation with Implicit Displacement Fields*. 2021. arXiv: `2106.05187 [cs.CV]`.

During the course of this thesis, the following peer-reviewed papers were also published:

[Cor+19]	Victor Cornillere, Abdelaziz Djelouah, Wang Yifan, Olga Sorkine-Hornung, and Christopher Schroers. "Blind image super-resolution with spatially variant degradations". In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), pp. 1–13.
[Wan+18d]	Yifan Wang, Federico Perazzi, Brian McWilliams, Alexander Sorkine-Hornung, Olga Sorkine-Hornung, and Christopher Schroers. "A fully progressive approach to single-image super-resolution". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018, pp. 864–873 (cit. on pp. 19, 31, 34).

CHAPTER 2

Related Work

In this chapter, we first provide a broader overview of the state-of-the-art development in neural geometry processing (Sec. 2.1), then in Sec. 2.2-2.4 a more in-depth review is provided over relevant prior works related to the topics discussed in this thesis. [1].

2.1. Neural geometry processing

Neural geometry processing is evolving at an astounding speed. The ever-expanding landscape may be charted roughly based on the choice of geometry representations. To date, the most commonly used representations in neural geometry processing are voxels, point clouds, meshes and implicit surfaces.

As the pixel-equivalent in R^3, voxels are the earliest representation adopted in neural geometry processing literatures. ShapeNets [Wu+15b] proposed the first networks for shape understanding by directly replacing 2D convolutions from neural image processing with 3D convolutions. After that, many 3D-convolutional networks have been proposed for various applications, including completion [Wan+17b; DRQN17; Dai+18; Han+17; HTM17], multi-view surface reconstruction [Pas+18; Tul+17], novel view synthesis [Sit+18; Lom+19] and detailization [Che+21]. However, 3D convolutions on dense

[1]With exceptionally active development in neural geometry processing, there are excellent publications from fellow researchers extending the algorithms proposed in this thesis. We will refer to the most relevant ones in the concluding sections of each chapter.

voxels incur large memory and computation cost, which hinders the voxel representation from being adopted for detail-centric geometry processing tasks. Sparse 3D-convolution on structured grid, such as octree, was proposed in [Wan+17a; ROUG17], yet since these compact data structures cannot be easily updated on-the-fly, these methods are mainly limited to applications, in which the shape is known a priori, such as classification and segmentation of rigid shapes.

Points are a sparse representation of 3D shapes. Their unstructured and unordered nature sparked active search for new types of neural networks, since the popular convolutional neural network only works on structured data. PointNet [Qi+17a] pioneered in this domain, proposing an order-invariant operation composed of a point-wise non-linear mapping and a pooling layer. Since then, many powerful improvements have been introduced in various aspects of the algorithm, such as enhancing the "convolution" kernel [Li+18b; AML18a; WQF19; Xu+18], enforcing rotational-invariance [Che+19; Her+20b] and introducing hierarchical structures [Wan+18e; Qi+17b; Zha+19]. While the above works focused on the learning discriminative features from point clouds, some other works searched for the suitable network design for point cloud generations [FSG17; Ach+18]. FoldingNet [Yan+18] and AtlasNet [Gro+18a] replace direct generation [FSG17] by mapping ("folding") 2D samples to the 3D space; PointGrow [Sun+20], on the other hand, models the generation process as a distribution transformation via continuous normalizing flows. Thanks to these works, neural point processing has seen success in a variety of applications, including object detection from large lidar scans [SWL19; SR20], point cloud enhancement such as denoising and upsampling [Rak+19; HRR19; Yu+18b; Yif+19b; Yu+18a; Li+19], object-level shape completion and generation [Yua+18; WAJL20; Sun+20; Wen+20], as well as deformation [Gro+18b; Gro+19; Wan+19].

Meshes are another sparse representation for shapes. Compared to points, the known connectivity on the vertices allows information to traverse in geodesic path. The initiation of neural mesh processing can be credited to the study of deep learning on graphs [Est+14; HBL15; Mas+15; DBV16; Mon+17; XDZ17; Bro+21], which generalizes euclidean 2D convolutions for non-euclidean data either from the spectral domain or in the local spatial neighborhood. A comprehensive survey on this line of works is provided by Bronstein et al. [Bro+17]. When applied to mesh, these methods can extract intrinsic geometric features and are successfully applied to find correspondences between deformable 3D shapes. At the same time, several methods used parameterization to map the surface meshes to another domain, such as planar flat-torus [Mar+17; Hai+19] or icosahedral spheres [Jia+19] where convolutions can be carried out easily. However, these approaches can only be applied to a specific topology. Perhaps the real breakthrough of neural mesh processing came with MeshCNN [Han+19]. It exploited the unique structures and properties of triangle meshes and defines an efficient equivalence of convolution and pooling operations directly on triangle edges. While MeshCNN was original proposed for discriminative tasks such as classification and segmentation, it has been adopted as the basis for many other applications such as surface reconstruction [Han+20], learning articulations [Li+21a], mesh subdivision [Liu+20a] and texture synthesis [Her+20a].

Implicit surfaces are a relatively new representation in neural geometry processing, yet thanks to its unique properties it has been gaining widespread popularity especially in generative tasks such as surface reconstruction and novel view synthesis. Implicit surfaces differ from other aforementioned representations in that they are a continuous representation, thus they can potentially represent geometries at infinite resolution. The first neural implicit representations were proposed concurrently in [Par+19; CZ19; Mes+19] for shape generation, where the implicit surface function is approximated by a neural network. Soon after, many improvements were rapidly proposed to achieve better surface reconstruction accuracy, *e.g.* by improving the training schemes [Xu+20; Dua+20], or by leveraging global-local context [Xu+19; Erl+20] or by adopting specific parameterizations [Gen+19; Den+20; CTZ20], or introducing spatial partitions [Gen+20; Tre+20; Cha+20a; Mar+21]. At the same time, neural implicit representations have been extended to encode signals other than geometries, such as surface textures [Oec+19], deformations [Nie+19], surface light fields [Oec+20] and volumetric radiance fields [Mil+20]. They enabled a holistic scene representation that unifies the geometry, appearance and dynamics in a common framework. With the recent development in differentiable renderers for neural implicit surfaces [SZW19; Nie+20; Yar+20; Mil+20], these aforementioned neural implicit fields can be effectively learned using only 2D observations, which led to exciting progresses in image-based 3D scene reconstruction [Yu+21; MB+21; Sai+19; Sai+20; Pum+21] and geometry consistent view synthesis [Sch+20; Cha+20b] and scene generations [NG21; Kos+21].

Each of the representations have their individual strengths and weaknesses. For instance, while implicit functions are great at generative tasks, they are a lesser choice for shape analysis compared to other explicit representations. Meshes are uniquely positioned to encode intrinsic geometric features, but they are not well suited for shape transformations or generations where the change of topology is involved. Points are suitable for both analysis and generations, but the ambiguity of neighborhood can lead to wrong topology and distorted features. Voxels are also a valid choice for both shape analysis and generation, but the resolution and performance is constrained by the computational power and network capacity. Therefore it is crucial to choose the suitable representation depending on the requirements of applications. In this thesis, two of the five introduced algorithms focus on points, two other on implicit surfaces and one can be applied to both points and meshes.

2.2. Point cloud processing and enhancement

2.2.1. Point clouds upsampling

Optimization-based approaches. Early optimization-based point set upsampling methods rely on shape priors. Assuming a smooth underlying surface, Alexa et al. [Ale+03] introduced the moving least squares (MLS) surface model, and applied it for point clouds upsampling by inserting new points at the vertices of the Voronoi diagram. Aiming to preserve sharp edges, Öztireli et al. [ÖGG09] proposed the robust implicit moving least squares (RIMLS) surface model, which iteratively optimizes the local implicit surface function considering the point normal directions. Huang et al. [Hua+13a] employed an anisotropic locally optimal projection operator [Lip+07; Hua+09] to consolidate and push points away from the edges, which is followed by a progressive edge-aware upsampling procedure. Wu et al. [Wu+15a] filled points in large areas of missing data by jointly optimizing both the surface and the inner points, using the extracted meso-skeleton to guide the surface point set resampling. These methods fit local geometry, e.g., normal estimation, and struggle with multiscale structure preservation.

Deep learning approaches. Zhang et al. [Zha+18a] extended a PointNet-based point generation model [Ach+18] to point set upsampling. Extensive experiments showed its generalization to different categories of shapes. However, note that [Ach+18] is trained on the entire object, which limits its application to low-resolution input. PU-Net [Yu+18b], on the other hand, operates on patch level and thus it can handle high-resolution input, but the upsampling results lack fine-grained geometry structures. Its follow-up work, the EC-Net [Yu+18a], improves restoration of sharp features by minimizing a point-to-edge distance but requires a rather expensive edge annotation for training. We propose in this thesis a multi-step, patch-based neural network architecture to channel the attention of the network to both global and local features. Our method also differ from the PU-Net and EC-Net in feature extraction, expansion, and loss computation.

method	objective	position	depth	normal	occlusion	silhouette	topology
OpenDR	mesh	✓	✗	via position	✗	✓	✗
NMR	mesh	✓	✗	via position	✗	✓	✗
Paparazzi	mesh	limited	limited	via position	✗	✗	✗
Soft Rasterizer	mesh	✓	✓	via position	✓	✓	✗
Pix2Vex	mesh	✓	✓	via position	✓	✓	✗
Ours	points	✓	✓	✓	✓	✓	✓
works released after the publication of our method							
SynSin	sphere	✓	✓	✗	✓	✓	✓
Pulsar	sphere	✓	✓	via extra channel	✓	✓	✓

Table 2.1.: Comparison of generic differential renderers. By design, OpenDR [LB14] and NMR [KUH18] do not propagate gradients to depth; Paparazzi [LTJ18] has limitation in updating the vertex positions in directions orthogonal their face normals, thus can not alter the silhouette of shapes; Soft Rasterizer [Liu+19a] and Pix2Vex [Pet+19] can pass gradient to occluded vertices, through blurred edges and transparent faces. All mesh renderers do not consider the normal field directly and cannot modify mesh topology. Our method uses a point cloud representation, updates point position and normals jointly, considers the occluded points and visibility changes and enables large deformation including topology changes. SynSin [Wil+20] and Pulsar [LZ21a] are published after our method, they use opaque spheres as the representation and do not yield surfaces points.

Upsampling in deep learning. Modern deep convolutional neural networks (CNN) [KSH12] process multiscale information using skip-connections between different layers, e.g. U-Net [RFB15], ResNet [He+16] and DenseNet [Hua+17]. In image super-resolution, state-of-the-art methods such as LapSRN [Lai+17] and ProSR [Wan+18d] gained substantial improvement by carefully designing layer connections with progressive learning schemes [Kar+18; Wan+18c], which usually contribute to faster convergence and better preservation of all levels of detail. Intuitively, such multiscale skip-connections are useful for point-based deep learning as well. A few recent works have exploited the power of multiscale representation [KL17; Wan+18b; GWM18; JWL18; Liu+18] and skip-connection [DBI18; Ret+18] in 3D learning. In our method, we focus on point cloud upsampling and propose intra-level and inter-level point-based skip-connections.

2.2.2. Point processing via inverse rendering.

General-purpose differentiable renderer. Loper and Black [LB14] developped a differentiable renderer framework called OpenDR that approximates a primary renderer and computes the gradients via automatic differentiation. Neural mesh renderer (NMR) [KUH18] approximates the backward gradient for the rasterization operation using a handcrafted function for visibility changes. Liu et al. [LTJ18] proposed Paparazzi, an analytic differentiable renderer for mesh geometry processing using image filters. In concurrent work, Petersen et al. [Pet+19] presented *Pix2Vex*, a C^∞ differentiable renderer via soft blending schemes of nearby triangles, and Liu et al. [Liu+19a] introduced *Soft Rasterizer*, which renders and aggregates the probabilistic maps of mesh triangles, allowing flowing gradients from the rendered pixels to the occluded and far-range vertices. Li et al. [Li+18a] and Azinovic et al. [Azi+19] introduced a differentiable ray tracer to implement the differentiability of physics-based rendering effects, handling camera position, lighting and texture. All these generic DR frameworks rely on mesh representation of the scene geometry. We summarize the properties of these renderers in Table 2.1.

Differentiable rendering in neural networks. Numerous works have employed differentiable renderers in the neural network to infer 3D shapes from 2D images, such as in single view image reconstruction [Yan+16; Pon+17; Zhu+17], face reconstruction [Ric+17], shape completion [Hu+19], and image synthesis [Sit+18]. In comparison to these methods, the differentiable renderer introduced in this thesis is a general purpose differentiable renderer not tied to a pretrained network. It could be extended and adapted to the above applications, but also it can be also as an independent module for general shape editing, filtering, and reconstruction.

Differentiable point renderers. A number of works render depth maps of point sets [LKL18; ID18; Rov+18b] for point cloud classification or generation. These renderers do not define proper gradients for updating point positions or normals, thus they are commonly applied as an add-on layer behind a point processing network, to provide 2D supervision. Typically, their gradients are defined either only for depth values [LKL18], or within a small local neighborhood around each point. Such gradients are not sufficient to alter the shape of a point cloud, as we show in a pseudo point renderer in Fig. 3.24.

Surface splatting. Surface splatting is fundamental to our method. Splatting has been developed for simple and efficient point set rendering and processing in the early seminal point based works [Pfi+00; Zwi+01; Zwi+02; Zwi+04]. Recently, point based techniques have gained much attention for their superior potential in geometric learning. To the best of our knowledge, we are the first to implement high-fidelity differentiable surface splatting.

2.3. Implicit surface reconstruction

2.3.1. Detail-driven implicit surface representation.

Implicit surface representations. Implicit functions are a flexible representation for surfaces in 3D. Traditionally, implicit surfaces are represented globally or locally with radial basis functions (RBF) [Car+01], moving least squares (MLS) [Lev98], volumetric representation over uniform grids [CL96], or adaptive octrees [KBH06]. Recent works investigated neural implicit surface representations, i.e., using deep neural networks to encode implicit function [Par+19; SZW19], which achieved promising results in reconstructing surfaces from 3D point clouds [AL20a; Sit+20; Erl+20] or images [LWL20; Yar+20; Nie+20].

Compared with simple polynomial or Gaussian kernels, implicit functions defined by nested activation functions, e.g., MLPs [CZ19] or SIREN [SZW19], have more capability in representing complex structures. However, fitting neural implicit functions requires clean points for supervision [Xu+20] and careful optimization to prevent either overfitting to noise or underfitting to details and structure.

Hierachical neural implicit shape representation. Neural implicit shape representation was initially proposed by several works concurrently [Par+19; CZ19; Mes+19], and since then many works have sought to introduce hierarchical structures into the neural representation for better expressiveness and generalizability. The majority of these methods focus on spatial structures. DLS [Cha+20a] and PiFU [Sai+19; Sai+20] use sparse regular voxels and dense 2D grid, respectively, to improve detail reconstruction. In the spirit of classic approaches [Fri+00; Oht+03], NSVF [Liu+20b], NGLOD [Tak+21] and ACORN [Mar+21] leveraged shape-adaptive structured grids, leading to significantly higher reconstruction quality and increased rendering speed. A common disadvantage of these methods is that the memory use and model complexity are directly tied to the desired geometric resolution. In parallel, other proposed methods learn the spatial partition. Some of these methods decompose the given shape using parameterized templates, such as anisotropic Gaussians [Gen+19], convex shape CVXNet [Den+20; CTZ20] or simple primitives [Hao+20], while others represent local shapes with small neural networks and combine them together either using Gaussians [Gen+20] or surface patches [Tre+20]. Due to limitations of template functions and delicate spatial blending issues, these methods can only handle very coarse geometries.

High-frequency representation in neural networks As formally explained in [Rah+19; Bas+20], neural networks have a tendency to learn low-frequency functions. To combat this issue, Mildenhall et al. [Mil+20] incorporated "positional encoding" for neural rendering and demonstrate remarkable progress in terms of detail reconstruction, which is a sinusoidal mapping for the input signal, a practice later theoretically justified by Tancik et al. [Tan+20]. Alternatively, SIREN also shows impressive advances in detail representation by replacing ReLU activation with sin functions. With these new networks gaining popularity, a few works delve deeper and apply a coarse-to-fine frequency hierarchy in the training process for deformable shape representation [Par+20] and meshing [Her+21]. In our method, we also create a frequency hierarchy by leveraging this new form of networks – not only in the training scheme but also explicitly in the construction of the networks to reflect our geometry-motivated design principles.

Detail transfer using disentangled implicit functions. Detail transfer refers to transplanting the disentangled geometric details from a source shape onto a target object with high fidelity and plausibility. Classic detail transfer methods represent surface details as normal displacements [Bot+10; Zho+07; SB09]. The majority of them are parametric [Yin+01; Bie+02; Sor+04; Zho+06; Tak+11], relying on a consistent surface parameterization between the source and the target shape. Non-parametric approaches [Che+12; Ber+17], on the other hand, find best-matching surface patches between the source and target, and copy the details iteratively from coarse to fine. These classic approaches produce high quality results, but often require a pre-defined base surface or abundant user inputs. In the "deep" realm, DeepCage [Yif+20b] proposed a neural deformation method that maps solely the coarse geometry, hence allowing detail transfer without tackling detail disentanglement. Hertz et al. [Her+20a] learn the coarse-to-detail correspondence iteratively from multi-scale training data, while DecorGAN [Che+21] synthesizes details by upsampling a coarse voxel shape according to a style code of another shape using GANs. All of these approaches use explicit representations, hence they are subject to self-intersection and resolution limitations. D^{2}IM-Net [LZ21b] uses two planar displacement maps to transfer surface details by mapping the coordinates of the source and target shapes using part segementation, thus limiting the application to man-made rigid shapes. In comparison, our method does not require any correspondence mapping.

2.3.2. Optimizing neural implicit surfaces.

Optimizing neural implicit surfaces with partial observations. Given raw 3D data, Atzmon and Lipman [AL20a; AL20b] use sign agnostic regression to learn neural implicit surfaces without using a ground truth implicit function for supervision. Gropp et al. [Gro+20] use the Eikonal term for implicit geometric regularization and provide a theoretical analysis of the plane reproduction property possessed by the neural zero level set surfaces. Erler et al. [Erl+20] proposed a patch-based framework that learns both the local geometry and the global inside/outside information, which outperforms existing data-driven methods. None of these methods exploit an explicit sampling of the implicit function to improve the optimization. Poursaeed et al. [Pou+20] use two different encoder-decoder networks to simultaneously predict both an explicit atlas [Gro+18a] and an implicit function. In this thesis, we propose a hybrid representation using a single network.

When the input observations are in the form of 2D images, differentiable rendering allows us to use 2D pixels to supervise the learning of 3D implicit surfaces through automatic differentiation and approximate gradients [Kat+20; Tew+20]. The main challenge is to render the implicit surface and compute reliable gradients at every optimization step efficiently. Liu et al. [Liu+20c] accelerate the ray tracing process via a coarse-to-fine sphere tracing algorithm [Har96], and use an approximate gradient in back propagation. In [Liu+19c], a ray-based field probing and an importance sampling technique are proposed for efficient sampling of the object space. Although these methods greatly improve rendering efficiency, the sampling of ray-based algorithms, i.e., the intersection between the ray and the iso-surface, are intrinsically irregular and inefficient. Most of the above differentiable renderers use ray casting to generate the supervision points. Our method propose another type of supervision points by sampling the implicit surface in-place.

Sampling implicit surfaces. In 1992, Figueiredo et al. [Fig+92] proposed a powerful way to sample implicit surfaces using dynamic particle systems that include attraction and repulsion forces. Witkin and Heckbert [WH94] further developed this concept by formulating an adaptive repulsion force. While the physical relaxation process is expensive, better initialization techniques have been proposed, such as using seed flooding on the partitioned space [LGS06] or the octree cells [PJS07]. Huang et al. [Hua+13b] resample point set surfaces to preserve sharp features by pushing points away from sharp edges before upsampling. When sampling a neural implicit surface, existing works such as Atzmon et al. [Atz+19] project randomly generated 3D points onto the iso-surface along with the gradient of the neural level set. However, such sampled points are unevenly distributed, and may leave parts of the surface under-sampled or over-sampled.

2.4. Shape deformation

Learning 3D deformations. Many recent works in learning 3D geometry have focused on generative tasks, such as synthesis [Gro+18a; Mes+19] and editing [Zhu+18] of unstructured geometric data. These tasks are especially challenging if one desires high-fidelity content with intricate details. A common approach to producing intricate shapes is to deform an existing generic [Wan+18a] or category-specific [Gro+18b] template. Early approaches represented deformations as a single vector of vertex positions of a template [Tan+18], which limited their output to shapes constructable by deforming the specific template, and also made the architecture sensitive to the template tessellation. An alternative is to predict a freeform deformation field over 3D voxels [Jac+18; Han+18; YM16]; however, this makes the deformation's resolution dependent on the voxel resolution, and thus has limited capability to adapt to a specific shape categories and source shapes.

Alternatively, some architectures learn to map a single point at a time, conditioned on some global descriptor of the target shape [Gro+18b]. These architectures can also work for novel sources by conditioning the deformation field on features of both source and target [Gro+19; Wan+19]. Unfortunately, due to network capacity limits, these techniques struggle to represent intricate details and tend to blur high-frequency features.

Traditional methods for mesh deformation. Research on detail-preserving deformations in the geometry processing community spans several decades and has contributed various formulations and optimization techniques [SB09]. These methods usually rely on a sparse set of control points whose transformations are interpolated to all remaining points of the shape; the challenge lies in defining this interpolation in a way that preserves details. This can be achieved by solving an optimization problem to reduce the distortion of the deformation such as [SA07]. However, defining the output deformation as the solution to an intricate non-convex optimization problem significantly limits the ability of a network to learn this deformation space.

Instead, we use cage-based deformations as our representation, where the source shape is enclosed by a coarse *cage* mesh, and all surface points are written as linear combinations of the cage vertices, i.e., generalized barycentric coordinates. Many designs have been proposed for these coordinate functions such that shape structure and details are preserved under interpolations [JSW05; LLCO08; Jos+07; CB17; TTB12; SVJ15; XLG12].

Shape synthesis and deformation transfer. Automatically aligning a source shape to a target shape while preserving details is a common task, used to synthesize variations of shapes for amplification of stock datasets [HWK15] or for transferring a given deformation to a new model, targeting animation synthesis [SP04]. To infer the deformation, correspondence between the two shapes needs to be accounted for, either by explicitly inferring corresponding points [LSP08; Li+12; Hua+08], or by implicitly conditioning the deformation fields on the latent code of the target shape [Gro+19; Wan+19; Han+18]. Our work builds upon the latter learning-based framework, but uses cages to parameterize the space of deformations.

Gao et al. [Gao+18] automate the deformation transfer for unpaired shapes using cycled generative adversarial networks, thus the trained method cannot be easily adapted for new shape targets. Some prior techniques focus on transferring and interpolating attributes between various latent spaces trained for shape generation [Yin+19; Gao+19]. These generative models are not capable of fully preserving local geometric features, especially if the source is not pre-segmented into simpler primitives (as assumed by [Gao+19]). In general, such methods are only expected to perform well if the input shapes are relatively similar to those observed at training time.

CHAPTER 3

Point Cloud Processing and Enhancement

Point clouds are the most common input for the general geometric processing pipeline, their quality has a substantial impact on the performance of all the downstream applications. In reality, point clouds are often the result of a 3D acquisition procedure and are typically sparse, noisy, and incomplete due to hardware limitations and environmental constraints. In this chapter, we introduce two algorithms, which enhance the quality of point clouds from different perspectives.

3.1. Point cloud super-resolution

Undersampling is one of the most common defections found in point clouds captured by typical 3D acquisition systems. It can be caused by a variety of reasons, such as insufficient processing bandwidth, over-extended scanning distance *etc*. If not treated, undersampled point clouds can lead to distortions in the reconstructed surface varying from skewed geometric features to false topological structures.

The success of neural super-resolution techniques in image space encourages the development of upsampling methods for 3D point sets. A recent plethora of deep learning super-resolution techniques have achieved significant improvement in single image super-resolution performance [Don+16;

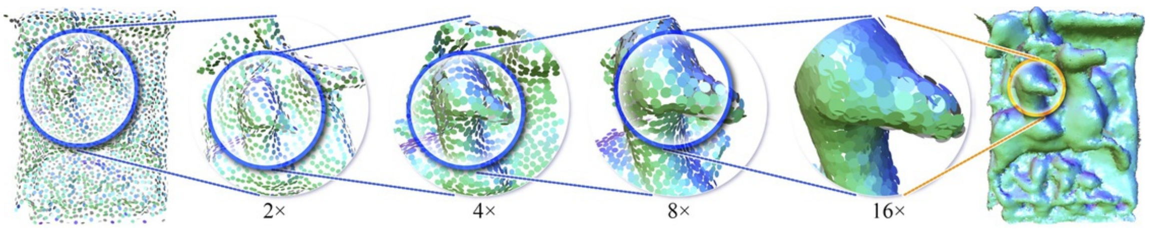


Figure 3.1.: Illustrative overview of our point upsampling algorithm, MPU. Intuitively, our network learns different levels of detail in multiple steps, where each step focuses on a local patch from the output of the previous step. By progressively training our patch-based network end-to-end, we successfully upsample a sparse set of input points, step by step, to a dense point set with rich geometric details. Here we use discs for points rendering, which are color-coded by point normals.

KKLML16; Shi+16; Led+17]; in particular, multi-step methods have been shown to excel in their performance [Lai+17; Fan+17; Zha+18b].

Dealing with 3D point sets, however, is challenging since, unlike images, the data is unstructured and irregular [Li+18b; HTY18; Xu+18; Her+18; AML18b]. Moreover, point sets are often a result of customer-level scanning devices, and they are typically sparse, noisy and incomplete. Thus, upsampling techniques are particularly important, and yet the adaption of image-space techniques to point sets is far from straightforward.

Neural point processing is pioneered by PointNet [Qi+17a] and PointNet++ [Qi+17b], where the problem of irregularity and the lack of structure is addressed by applying *shared* multilayer perceptrons (MLPs) for the feature transformation of individual points, as well as a symmetric function, e.g., max pooling, for global feature extraction. Yu et al. [Yu+18b] introduced the first end-to-end point set upsampling network, PU-Net, where both the input and the output are the 3D coordinates of a point set. PU-Net extracts multiscale features based on PointNet++ [Qi+17b] and concatenates them to obtain aggregated multi-scale features on each input point. These features are expanded by replication, then transformed to an upsampled point set that is located and uniformly distributed on the underlying surface. Although multiscale features are gathered, the level of detail available in the input patch is fixed, and thus both high-level and low-level geometric structures are ignored. The method consequently struggles with input points representing large-scale or fine-scale structures, as shown in Figures 3.7 and 3.8.

In this work, we present a patch-based progressive upsampling network for point sets. The concept is illustrated in Figure 3.2 and 3.1. The multi-step upsampling breaks a, say, 16×-upsampling network, into four 2× subnets, where each subnet focuses on a different level of detail. To avoid exponential growth in points and enable end-to-end training for large upsampling ratios and dense outputs, all subnets are fully patch-based, and the input patch

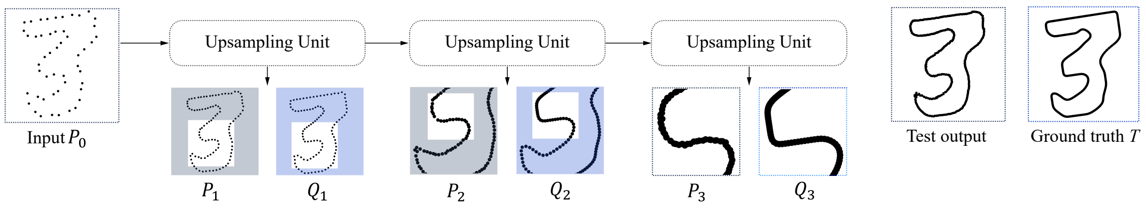


Figure 3.2.: Overview of our multi-step patch-based point set upsampling network with 3 levels of detail. Given a sparse point set as input, our network predicts a high-resolution set of points that agree with the ground truth. Instead of training an 8×-upsampling network, we break it into three 2× steps. In each training step, our network randomly selects a local patch as input, upsamples the patch under the guidance of ground truth, and passes the prediction to the next step. During testing, we upsample multiple patches in each step independently, then merge the upsampled results to the next step.

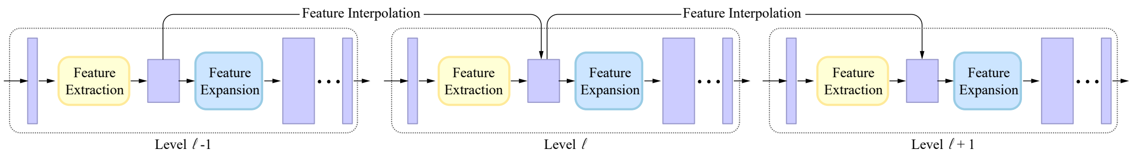


Figure 3.3.: Illustration of three upsampling network units. Each unit has the same structure but applied on different levels.

size is adaptive with respect to the present level of detail. Last but not least, we propose a series of architectural improvements, including novel dense connections for point-wise feature extraction, code assignment for feature expansion, as well as bilateral feature interpolation for inter-level feature propagation. These improvements contribute to further performance boost and significantly improved parameter efficiency.

We show that our model is robust under noise and sparse inputs. It compares favorably against existing state-of-the-art methods in all quantitative measures and, most importantly, restores fine-grained geometric details.

3.1.1. Method

Given an unordered set of 3D points, our network generates a denser point set that lies on the underlying surface. This problem is particularly challenging when the point set is relatively sparse, or when the underlying surface has complex geometric and topological structures. In this work, we propose an end-to-end progressive learning technique for point set upsampling. Intuitively, we train a multi-step patch-based network to learn the information from different levels of detail. As shown in Figures 3.2 and 3.3, our model consists of a sequence of upsampling network units. Each unit has the same structure, but we employ it on different levels of detail. The information of all levels is shared via our intra-level and inter-level connections inside and between the units. By progressively training all network units end-to-end, we achieve significant improvements over previous works. We first present the global design of our network and then elaborate on the upsampling units.

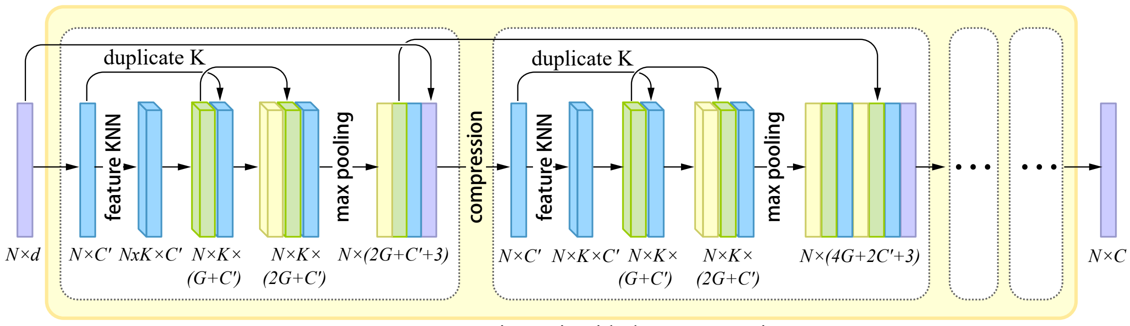
Multi-step upsampling network

Multi-step supervision is common practice in neural image super-resolution [Lai+17; Fan+17; Zha+18b]. In this section, we first discuss the difficulties in adapting multi-step learning to point set upsampling, which motivates the design of our multi-step *patch-based* supervision method. Next, we illustrate the end-to-end training procedure for a cascade of upsampling network units for large upsampling ratios and high-resolution outputs.

Multi-step patch-based receptive field. Ideally, a point set upsampling network should span the receptive field adaptively for various scales of details to learn geometric information from multiple scales. However, it is challenging to apply a multi-scope receptive field on a dense irregular point set due to practical constraints. In contrast to images, point sets do not have the regular structure, and the neighborhoods of points are not fixed sets. Neighborhood information must be *collected* by, e.g., k-nearest neighbors (kNN) search. This per-layer and per-point computation is rather expensive, prohibiting a naive implementation of a multi-step upsampling network to reach large upsampling ratios and dense outputs. Therefore, it is necessary to optimize the network architecture, such that it is scalable to a high-resolution point set.

Our key idea is to use a multi-step patch-based network, and the patch size should be adaptive to the scope of receptive fields at the present step. Note that in neural point processing, the scope of a receptive field is usually defined by the kNN size used in the feature extraction layers. Hence, if the neighborhood size is fixed, the receptive field becomes narrower as the point set grows denser. This observation suggests that it is unnecessary for a network to process all the points when the receptive field is relatively narrow. As shown in Fig. 3.2, our network *recursively* upsamples a point set while at the same time reduces its spatial span. This multi-step patch-based supervision technique allows for a significant upsampling ratio.

Multi-step end-to-end training. Our network takes L steps to upsample a set of points by a factor of 2^L. For L levels of detail, we train a set of



Feature extraction unit with dense connection

Figure 3.4.: Illustration of the feature extraction unit with dense connections.

subnet units $\{U_1, U_2, \ldots, U_L\}$. We train such a sequence of upsampling units by progressively activating the training of units; it has been used in many multiscale neural image processing works [Wan+18d; Kar+18].

More specifically, our entire training process has $2L - 1$ stages, i.e., every upsampling unit has two stages except the first one. We denote the currently targeted level of detail by $\hat{L}$. In the first stage of $U_{\hat{L}}$ we fix the network parameters of units U_1 to $U_{\hat{L}-1}$ and start the training of unit $U_{\hat{L}}$. In the second stage, we unleash the fixed units and train all the units simultaneously. This progressive training method is helpful because an immature unit can impose destructive gradient turbulence on the previous units [Kar+18].

We denote the ground truth model, prediction patch and reference patch with T, P and Q respectively and use $\hat{L}$ and ℓ to denote the targeted level of detail and an intermediate level, as illustrated in Fig. 3.2 and 3.6. In practice, we recursively shrink the spatial scope by confining the input patch to a fixed number of points (N). For more technical detail about extracting such input patches on-the-fly and updating the reference patches accurately, please refer to Sec. 3.1.1.

Upsampling Network Unit

Let us now take a closer look at an upsampling network unit U_ℓ. It takes a patch from $P_{\ell-1}$ as input, extracts deep feature, expands the number of features, compresses the feature channels to d-dimensional coordinates P_ℓ. In the following, we explain each component in greater detail.

Feature extraction via intra-level dense connections. We strive for extracting structure-aware features ($N \times C$) from an input point set ($N \times d$). In neural image processing, skip-connection is a powerful tool to leverage features extracted across different layers of the network [He+16; Hua+16; Hua+17;

Lin+18]. Following PointNet++ [Qi+17b], most existing point-based networks extract multiple scales of information by hierarchically downsampling the input point sets [LCL18; Yu+18b]. Skip-connections have been used to combine multiple levels of features. However, a costly point correspondence search must be applied prior to skip-connections, due to the varying point locations caused by the downsampling step.

We propose an architecture that facilitates *efficient* dense connections on point sets. Inspired by the dynamic graph convolution [Wan+18e; She+18], we define our local neighborhood in feature space. The point features are extracted from a local neighborhood that is computed dynamically via kNN search based on feature similarity. As a result, our network obtains long-range and nonlocal information without point set subsampling.

As shown in Fig. 3.4, our feature extraction unit is composed of a sequence of dense blocks. In each dense block, we convert the input to a fixed number (C') of features, group the features using feature-based KNN, refine each grouped feature via a chain of densely connected MLPs, and finally compute an order-invariant point feature via max-pooling.

We introduce dense connections both within and between the dense blocks. Within the dense blocks, each MLP's output, i.e., a fixed number (G) of features, is passed to *all* subsequent MLPs; between the blocks, the point features produced by each block are fed as input to *all* following blocks. All these skip-connections enable explicit information re-use, which improves the reconstruction accuracy while significantly reducing the model size, as demonstrated in Sec. 3.1.2. Overall, our $16\times$-upsampling network with four $2\times$-upsampling units has much fewer network parameters than a $4\times$-upsampling PU-Net [Yu+18b]: 304K vs. 825K.

Feature expansion via code assignment. In the feature expansion unit, we aim to transform the extracted features ($N \times C$) to an upsampled set of coordinates ($2N \times d$).

PU-Net [Yu+18b] replicates the per-point features and then processes each replicant independently by an individual set of MLPs. This approach may lead to clustered points around the original points positions, which is alleviated by introducing a repulsion loss. Instead of training the network to disentangle the replicated features in-place, we explicitly offer the network the information about the position variation.

In conditional image generation models [MO14], a category-variable is usually concatenated to a latent code to generate images of different categories.

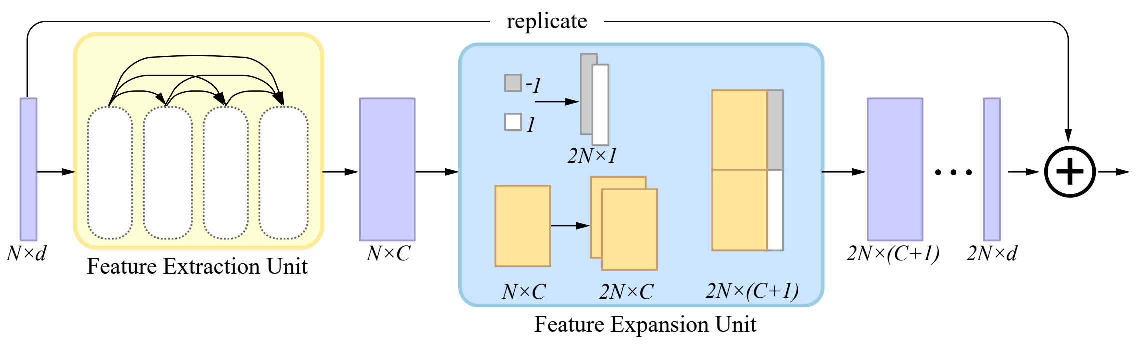


Figure 3.5.: Illustration of one upsampling network unit.

Similarly, we assign a 1D code, with value -1 and 1, to each of those duplicated features to transform them to different locations, as shown in Fig. 3.5. Next, we use a set of MLPs to compress the $2N \times (C+1)$ features to $2N \times d$ residuals, which we add to the input coordinates to generate the output points.

Our experiments show that the proposed feature expansion method results in a well distributed point set without using an additional loss. Also, the number of network parameters is independent of the upsampling ratio, since all expanded features share the consecutive MLPs.

Our feature expansion method is also related to recent point cloud generative models FoldingNet [Yan+18] and AtlasNet [Gro+18a], where the coordinates of a 2D point are attached to the learned features for point generation. Here, we show that the choice of an attached variable can be as simple as a 1D variable.

Inter-level skip connection via bilateral feature interpolation. We introduce inter-level skip-connections to enhance the communication between the upsampling units, which serves as bridges for features extracted with different scopes of the receptive fields, as shown in Fig. 3.3.

To pass features from previous levels the current level, the key is a feature interpolation technique that constructs corresponding features from the previous upsampling unit, as the upsampling and patch extraction operations change the point correspondence. Specifically, we use bilateral interpolation. For the current level ℓ, we denote by p_i and f_i the coordinates of the i-th point and its features generated by the feature extraction unit respectively, and $\mathcal{N}_i'$ denotes the spatial kNN of p_i from level ℓ'. the interpolated feature for $\tilde{f}_i$ can be written as:

$$\tilde{f}_i = \frac{\sum_{i' \in \mathcal{N}_i'} \theta(p_i, p_{i'}) \psi(f_i, f_{i'}) f_{i'}}{\sum_{i' \in \mathcal{N}_i'} \theta(p_i, p_{i'}) \psi(f_i, f_{i'})}, \tag{3.1}$$

with the joint weighting functions: $\theta(p_1, p_2) = e^{-\left(\frac{\|p_1-p_2\|}{r}\right)^2}$, $\psi(f_1, f_2) = e^{-\left(\frac{\|f_1-f_2\|}{h}\right)^2}$. The width parameters r and h are computed using average distance to the closest neighbor.

One way to implement the inter-level connection is to interpolate and concatenate $\tilde{f}_i$ from *all* previous layers, i.e., use dense links the same as those within the feature extraction units. However, doing so would result in a very wide network, with ℓC features in level ℓ (typically $C = 216$), causing scalability issues and optimization difficulties [Wan+18d]. Instead, we apply residual skip-connections, i.e., $f_i = \tilde{f}_i + f_i$. By applying such residual links per-level, contextual information from coarser scales can be propagated through the entire network and incorporated for the restoration of finer structures. We learn through experiments that both dense links and residual links contribute positively to the upsampling result, but the latter has better performance in terms of memory efficiency, training stability and reconstruction accuracy.

Implementation details

Iterative patch extraction. In each training step, the target resolution $\hat{L}$ is fixed. $P_{\hat{L}}$ and $Q_{\hat{L}}$ denote the prediction and reference patch in $\hat{L}$, whereas $T_{\hat{L}}$ denotes the *entire* reference shape in this resolution. We compute $P_{\hat{L}}$ and $Q_{\hat{L}}$ iteratively from a series of intermediate predictions and references, denoted as P_ℓ and $\tilde{Q}_\ell$ where $\ell = 1 \ldots \hat{L} - 1$. More specifically, the input to level ℓ is obtained using kNN ($k = N$) around a random point $p^*_{\ell-1}$ in $P_{\ell-1}$. $\tilde{Q}_\ell$ should matche the spatial extent of P_ℓ but has a higher resolution, hence it can be extracted by kNN search in $\tilde{Q}_{\ell-1}$ using the same query point $p^*_{\ell-1}$, whereas $k = 2^{\hat{L}-l+1}N$. Note that we normalize the patches to a unit cube to improve the computational stability. In Fig. 3.6 we illustrate the described procedure for $\hat{L} = 3$.

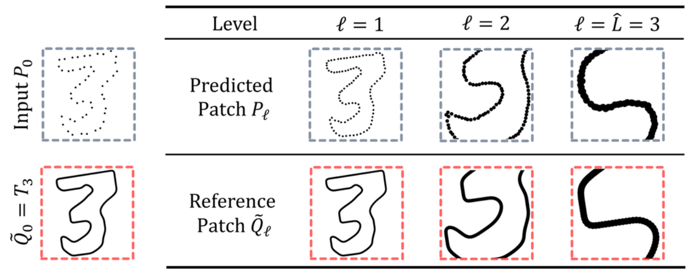


Figure 3.6.: Extraction of patches for $\hat{L} = 3$ during training. In this example, since there are only a small number of input points in 2D data, the first level contains the whole input shape ($N = |P_0|$).

For inference, the procedure differs from above in two points: 1. In each level, we extract H overlapping input patches to ensure coverage of the entire input point set, the query points are sampled with farthest sampling; 2. We obtain P_ℓ by first merging the H overlapping partial outputs and then resampling with farthest sampling such that $|P_\ell| = 2|P_{\ell-1}|$. The resampling leads to uniform point distribution despite overlapping regions.

Using a small N could theoretically restrict the contextual information, while a larger N could unnecessarily increase the input complexity thus training difficulty. In our experiments, the choice of the input patch size N is not that critical for the upsampling quality.

Loss function. We use Euclidean distance for patch extraction for its speed and flexibility. This implies that the patch pairs P_ℓ and Q_ℓ might have misalignment problems on their borders. We observe that the loss computed on those unmatched points adds noise and outliers in the result. Thus, we propose a modified Chamfer distance:

$$\mathcal{L}(P,Q)=\frac{1}{|P|}\sum_{p\in P}\xi\left(\min_{q\in Q}\|p-q\|^2\right)+\frac{1}{|Q|}\sum_{q\in Q}\xi\left(\min_{p\in P}\|p-q\|^2\right), \tag{3.2}$$

where the function ξ filters outliers above a threshold δ:

$$\xi(d) = \begin{cases} d, & d \le \delta \\ 0, & \text{otherwise} \end{cases}.$$

We set δ to be a multiple of the average nearest neighbor distance so as to dynamically adjust to patches of different scales.

3.1.2. Results

In this section, we compare our method quantitatively and qualitatively with state-of-the-art point upsampling methods, and evaluate various aspects of our model. Please refer to the supplementary for further implementation details and extended experiments.

The metrics used for evaluation are (i) Chamfer distance, (ii) Hausdorff distance [Ber+13] and (iii) point-to-surface distance computed against the ground truth mesh.

Training and testing data. We generate two datasets for our experiments: MNIST-CP, Sketchfab and ModelNet10[Wu+15b]. MNIST-CP consists of 50K and 10K training and testing examples of 2D *contour points* extracted from the MNIST dataset [LC10]. Given a set of 2D pixel points, we apply Delaunay triangulation [Boi+02], Loop surface subdivision [Loo87], boundary edge extraction, and WLOP [Hua+09] to generate a uniformly distributed point set lying on the contour curve of the image. The number of points in input P and ground truth point sets T_1, T_2 and T_3 are 50, 100, 200 and 800, respectively. Sketchfab consists of 90 and 13 highly detailed 3D models downloaded from SketchFab [Skea] for training and testing, respectively. ModelNet10 is comprised of 10 categories, containing 3991 and 908 CAD meshes for training and testing, respectively. We use the Poisson-disk sampling [CCS12] implemented in Meshlab [Cig+08] to sample input and ground truth point sets with the number of points ranging from 625 to 80000. Our data augmentation includes random rotation, scaling and point perturbation with gaussian noise.

Comparison. We compare our method on relatively sparse (625 points) and dense (5000 points) inputs with three state-of-the-art point set upsampling methods: EAR [Hua+13a], PU-Net [Yu+18b] and EC-Net [Yu+18a].

The code of these methods is publicly available. For EAR, we set the parameter $\sigma_n = 35^\circ$ to favor sharp feature preservation. For PU-Net and EC-Net, we obtain 16× results by iteratively applying their 4×-upsampling model twice, as advised by the authors. As for comparison, we train a four-step 16× model using our method, where the initial patch size falls into a similar level of detail as PU-Net. For all experiments, we add to the input Gaussian noise with 0.25% magnitude of the model dimensions.

Method	Sparse input			Dense input			# Param.
	CD	HD	P2F	CD	HD	P2F	
EAR	0.67	7.75	5.25	0.09	1.82	1.88	-
PU	0.72	9.24	6.82	0.41	5.45	3.39	814K
EC	0.91	13.4	6.42	0.24	4.21	2.64	823K
Ours	**0.54**	**6.92**	**3.32**	**0.06**	**1.31**	**1.11**	304K

Table 3.1.: Quantitative comparison with state-of-the-art approaches for 16× upsampling from 625 and 5000 input points tested on Sketchfab dataset.

Table 3.1 and 3.2 summarizes the quantitative comparison conducted using Sketchfab and ModelNet10. Note that because many models in ModelNet10 are not watertight, we omit the point-to-surface distance in Table 3.2. Examples of the upsampling results are provided in Figures 3.7 and 3.8 for visual comparison, where we apply surface reconstruction to the upsampled point sets using PCA normal estimation (neighborhood number = 25) [Hop+92b] and screened Poisson reconstruction (depth = 9) [KH13b].

As seen in Figures 3.7 and 3.8, EAR generates competitive results for denser inputs but struggles with sparse inputs. As shown in Table 3.1, the performance of PU-Net on sparse and dense inputs is similar, revealing its limitation for high levels of detail. For denser inputs, EC-Net produces clean and more well defined outputs than PU-Net, but also shows signs of over-sharpening.

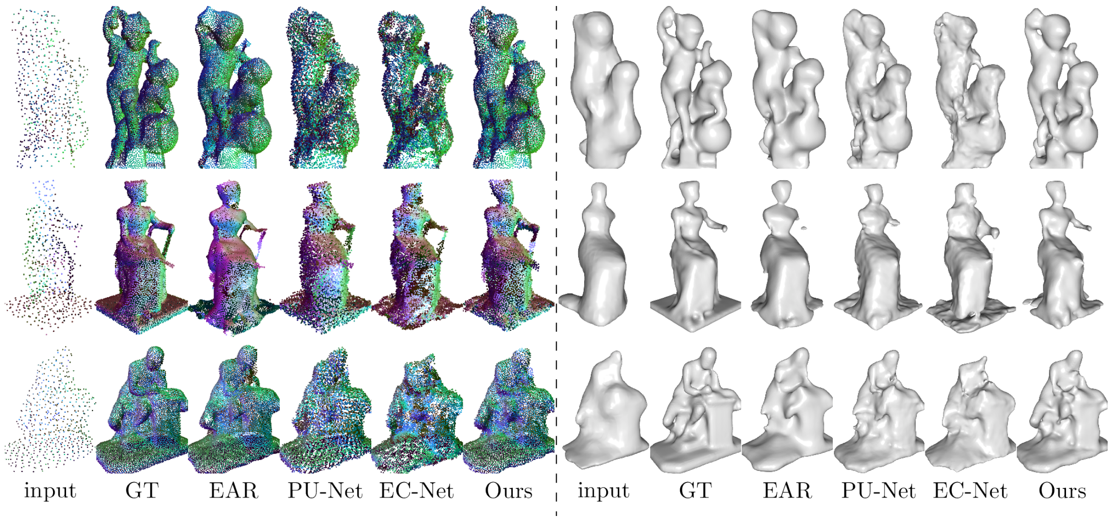


Figure 3.7.: 16× upsampling results from 625 input points (left) and reconstructed mesh (right).

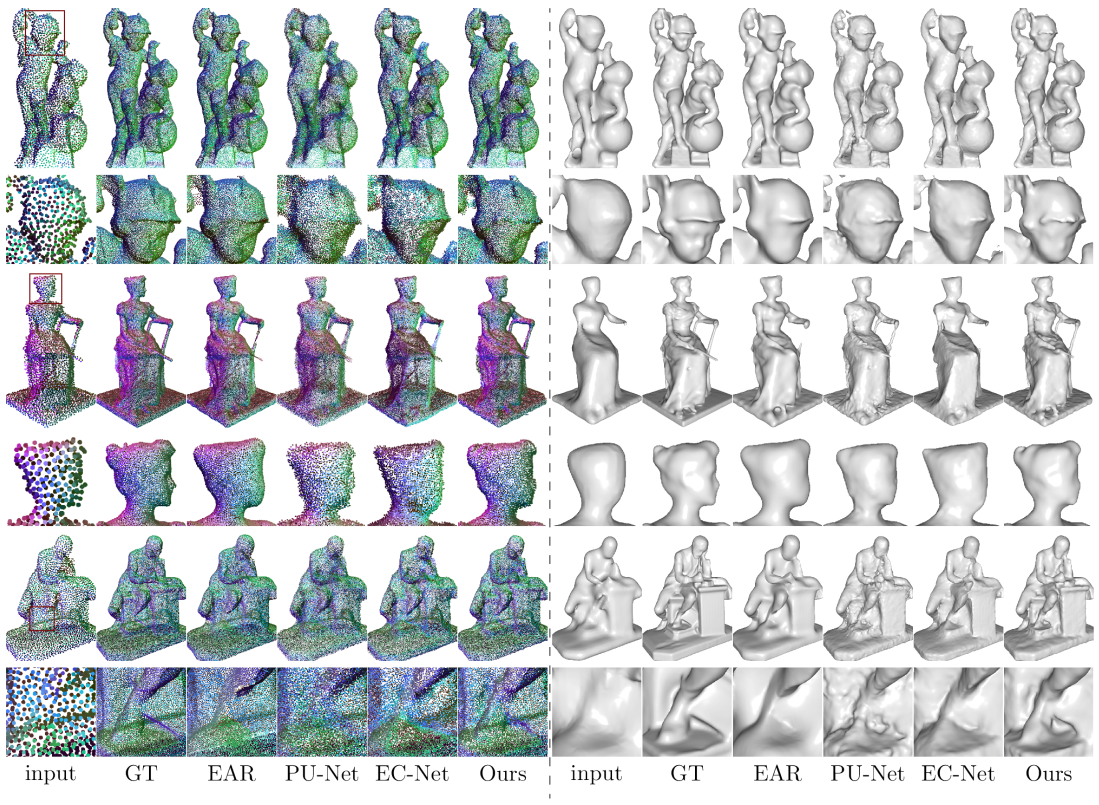


Figure 3.8.: 16× upsampling results from 5000 input points (left) and reconstructed mesh (right).

	10^{-3}	bathtub	bed	chair	desk	dresser	monitor	n. stand	sofa	table	toilet
CD	PU	1.01	1.12	**0.82**	1.22	1.55	1.19	1.77	1.13	0.69	1.39
	EC	1.43	1.81	1.8	1.30	1.43	2.04	1.88	1.79	1.00	1.72
	ours	**0.70**	**0.77**	0.90	**0.96**	**1.13**	**0.83**	**1.37**	**0.67**	**0.58**	**1.02**
HD	PU	10.77	12.39	10.38	13.29	14.08	14.01	16.21	11.66	9.7	14.74
	EC	15.71	23.17	18.65	16.12	16.37	30.48	20.29	19.97	12.42	18.58
	ours	**7.76**	**9.36**	**9.70**	**9.19**	**11.33**	**9.90**	**13.52**	**8.37**	**5.87**	**10.95**

Table 3.2.: Quantitative comparison with state-of-the-art approaches on ModelNet10 dataset for 16× upsampling from 625 input points.

Removed/Replaced component	CD 10^{-3}	HD 10^{-3}	P2F 10^{-3}	Param.
1. Multi-stage architecture	0.69	9.98	4.07	65K
2. End-to-end training	0.73	9.91	3.34	263K
3. Progressive end-to-end training	0.55	7.46	3.49	304K
4. Dense feature extraction	0.61	9.17	4.17	2855K
5. Feature expansion	0.73	9.83	5.30	1642K
6. Inter-level skip-connections	0.61	7.65	3.38	263K
Our full model	**0.54**	**6.92**	**3.32**	304K

Table 3.3.: Ablation study with 16×-upsampling factor tested on the Sketchfab dataset using 625 points as input. We evaluate the contribution of each proposed component quantitatively with Chamfer distance (CD), Hausdorff distance (HD) and mean point-to-surface distance (P2F), and also report the number of parameters in the rightmost column.

For sparse input though, EC-Net produces more artifacts, possibly because the geodesic KNN, which EC-Net is built upon, becomes unreliable under sparse inputs. In comparison, our method outperforms all these methods quantitatively by a large margin. Qualitatively, our results are less noisy and contain notably more details.

Ablation study. An ablation study quantitatively evaluates the contribution of each of our proposed components:

1. Multi-stage architecture: we train a 2×-upsampling model for all levels of detail and test by iteratively applying the model 4 times.
2. Progressive training: instead of progressively activating the training of each upsampling unit as described in Sec. 3.1.1, we train all units simultaneously.

4-6. Dense feature extraction, expansion, and inter-level skip-connections: we either remove or replace each of these modules with their counterpart in PU-Net.

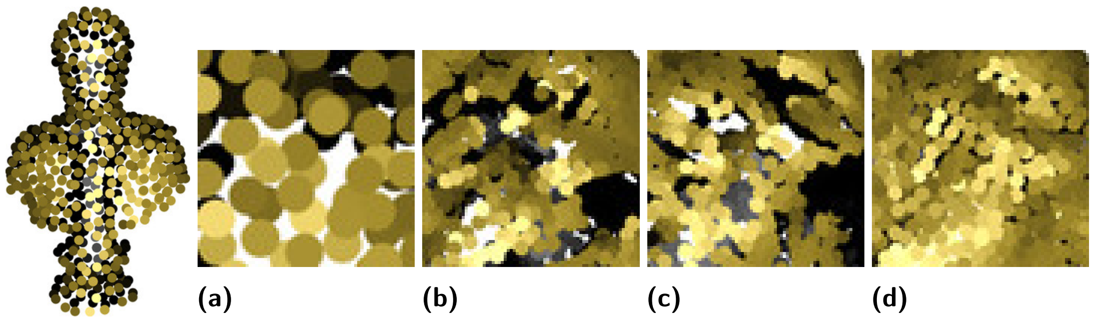


Figure 3.9.: Visual comparison for ablation study. We perform 16×-upsampling from 625 points (left). (a)-(d) show a point patch of the input and the results from the single-stage model, separately trained model and our full model.

As Table 3.3 shows, all components contributes positively to the full model. In particular, removing multi-stage architecture significantly increased the difficulty of the task, resulting in artifacts shown in Fig. 3.9b. We observe similar artifacts when the upsampling units are trained separately (Fig. 3.9c), as the networks cannot counteract the mistakes made in previous stages. The proposed dense feature extraction, feature expansion, and inter-level skip-connections considerably improve the upsampling results. Moreover, the feature extraction and expansion unit contribute to significant parameter reduction.

Study of patch-based progressive upsampling. We evaluate the effect of our core idea, patch-based progressive upsampling, in greater detail. For this purpose, we start from the architecture proposed by PU-Net and add the techniques introduced in Sec. 3.1.1 one by one. Specifically, we conduct the following experiments on MNIST-CP dataset: (i) train a PU-Net with direct 16× upsampling, (ii) train one 2× PU-Net using training examples sampled with all available patch densities and then apply it iteratively 4 times, (iii) train a network for each level of detail separately, (iv) progressively train all networks but omit the per-stage patch extraction technique introduced in Sec. 3.1.1, and finally (v) progressively train all networks with patch extraction.

The results are shown in Fig. 3.10. Both direct upsampling and single-stage model ((i) and (ii)) are unable to reconstruct faithful geometry in curvy regions, suggesting that a multi-stage architecture is necessary for capturing high levels of detail. The multi-stage PU-Net (iii) notably improves the result but shows more artifacts compared with an end-to-end multi-stage model (iv), since the network has a chance to correct the mistakes introduced in earlier stages. Finally, applying adaptive patch extraction (v) further refines

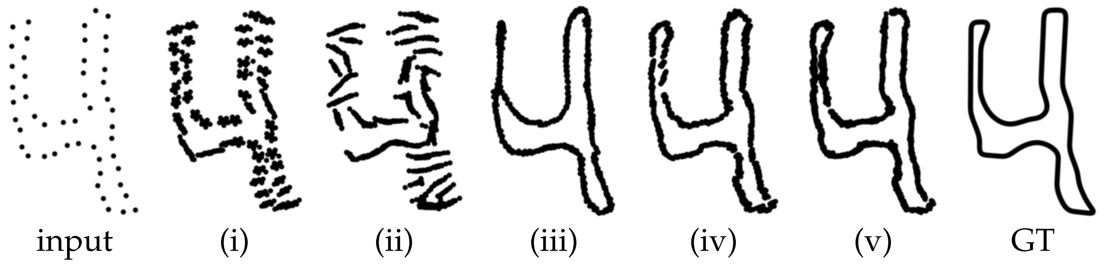


Figure 3.10.: Study of patch-based progressive upsampling. From left to right: input with 50 points, (i) direct 16× upsampling, (ii) iterative 2× upsampling trained with augmented data, (iii) multi-stage network trained separately, (iv) multi-stage network trained progressively, (v) patch-based multi-stage network trained progressively, and ground truth.

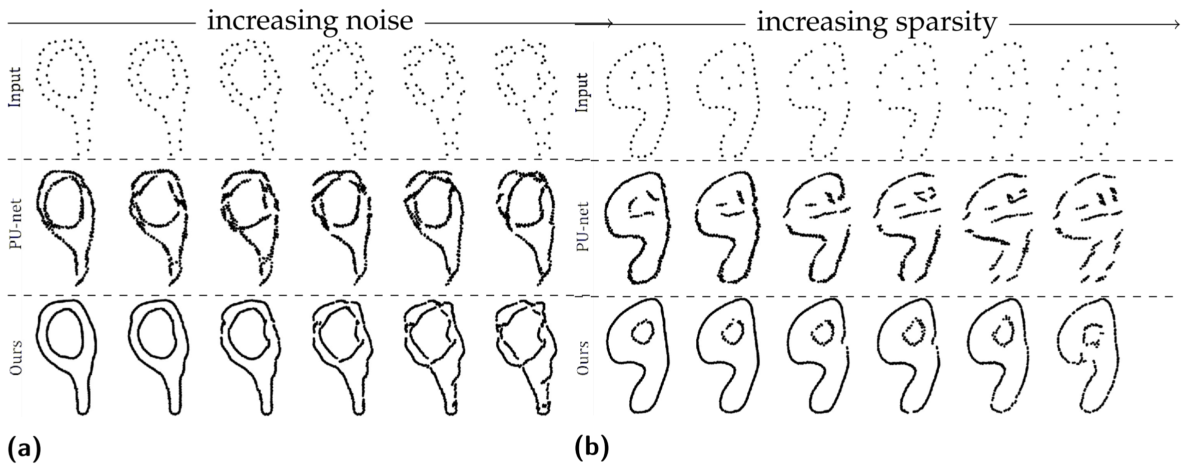


Figure 3.11.: Stress test with increasing noise (a) and sparsity (b). The model is trained using 50 input points and Gaussian noise of 0.25% magnitude of the point set dimensions. In (a) we test with noise level of 0, 0.25%, 0.5%, 1%, 1.5% and 2%; in (b) we test with 50, 45, 40, 35, 30, and 25 input points.

the local geometry, indicating that it helps the network to focus on local details by adapting the spatial span of input to the scope of receptive fields.

Stress test. To test the robustness to noise and sparsity, we subject an input point set to different noise levels ranging from 0% to 2%, and for sparsity we randomly remove 10% to 50% of the points from the input.

The corresponding results from MNIST-CP datasets are shown in Figures 3.11a and 3.11b. Compared to PU-Net, our model is more robust against noise and sparsity.

Upsampling of scanned data. We test our method for real world data in two settings.

In the first one, we diretly apply the trained model (trained using uniformly sampled point clouds) for scanned data. The input scanned data is acquired using a hand-held 3D scanner Intel RealSense SR300. Albeit dense, such data is severely ridden with noise and outliers. Therefore, we first employ WLOP [Hua+09], a point set denoising tool known to be robust against noise and outliers, to consolidate and simplify the point set. We then apply our model to the resulting, denoised yet sparse point set and obtain a dense and clean output, as shown in Fig. 3.12c.

In the second setting, we retrain our model using synthetic scans. To this end, we implement a virtual scanner to scan the training set of the Sketchfab dataset from a random viewpoint using different camera resolutions. By gradually increasing the camera resolution, we obtain a list of virtual scans with increasing density.

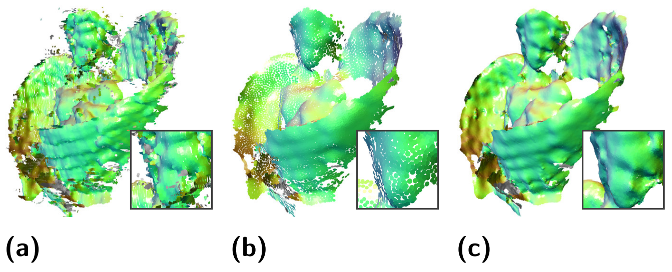
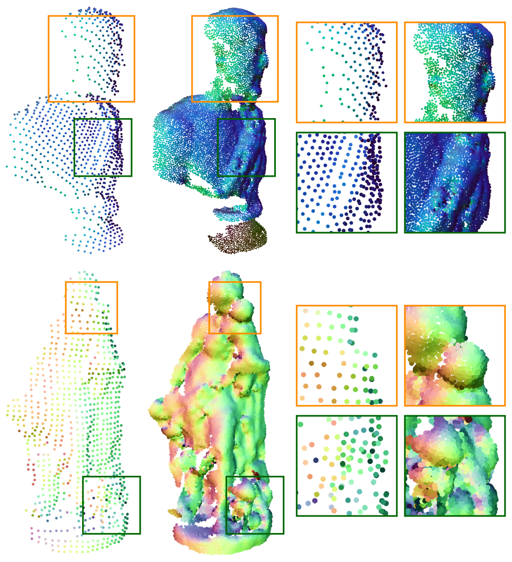
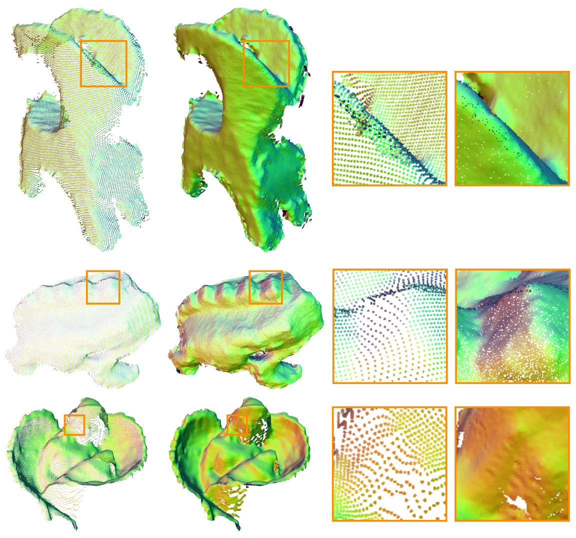


Figure 3.12.: 16× upsampling results using a real scan as input. Given a noisy input (a), we use WLOP [Hua+09] to obtain a consolidated point set (b), to which we apply our upsampling network (c).

We test the trained model on both virtual and real scans. The results are shown in Fig. 3.13a and 3.13b. The virtual scans are generated from the testing set of the Sketchfab dataset, while the real scans are acquired via the Intel RealSense SR300 scanner. As an alternative to the WLOP consolidation, we use the scanner's built-in filter to obtain a relatively clean input, which typically smears the fine-grained details and reduces the frame rate. As shown in Fig. 3.13a, our model is able to reveal fine-grained geometry from very sparse inputs. In addition, it can fill holes and preserve sharp features, as shown in Fig. 3.13b.

(a) 16× upsampling results on virtual scans. **(b)** 16× upsampling results on real scans.

Figure 3.13.: Upsampling results using models trained with virtual scanning data.

3.2. Point processing via inverse rendering

In this section, we introduce different approach for point cloud processing. Instead of directly operating in the 3D domain, we administer changes solely in the 2D renderings the 3D scene. Then, in a process called *inverse rendering*, the changes from 2D domain are propagated back and applied to the 3D scene. This seemingly deviant approach is often favored over the direct approach, given that 2D data is substantially easier to acquire and has a larger range of processing tools including especially the contemporary neural image processing.

The key to inverse rendering is differentiable renderer (DR). A differentiable renderer $\mathcal{R}$ takes scene-level information θ such as 3D scene geometry, lighting, material and camera position as input, and outputs a synthesized image $\mathbb{I} = \mathcal{R}(\theta)$. Any changes in the image $\mathbb{I}$ can thus be propagated to the parameters θ, allowing for image-based manipulation of the scene. Assuming a differentiable loss function $\mathcal{L}(\mathbb{I}) = \mathcal{L}(\mathcal{R}(\theta))$ on a rendered image $\mathbb{I}$, we can update the parameters θ with the gradient $\frac{\partial \mathcal{L}}{\partial \mathbb{I}} \frac{\partial \mathbb{I}}{\partial \theta}$. This view provides a generic and powerful shape-from-rendering framework where we can exploit vast image datasets available, deep learning architectures and computational frameworks, as well as pre-trained models. The challenge, however, is being able to compute the gradient $\partial \mathbb{I} / \partial \theta$ in the renderer.

Existing DR methods can be classified into three categories based on their geometric representation: voxel-based [NP+18; Tul+17; Liu+17], mesh-based [LB14; KUH18; LTJ18], and point-based [ID18; LKL18; Rov+18a; Raj+18]. Voxel-based methods work on volumetric data and thus come with high memory requirements even for relatively coarse geometries. Mesh-based DRs solve this problem by exploiting the sparseness of the underlying geometry in the 3D space. However, they are bound by the mesh structure with limited room for global and topological changes, as connectivity is not differentiable. Equally importantly, acquired 3D data typically comes in an unstructured representation that needs to be converted into a mesh form, which is itself a challenging and error-prone operation. *Point-based* DRs circumvent these problems by directly operating on point samples of the geometry, leading to flexible and efficient processing. However, existing point-based DRs use simple rasterization techniques such as forward-projection or depth maps, and thus come with well-known deficiencies in point cloud processing when capturing fine geometric details, dealing with gaps and occlusions between near-by points, and forming a continuous surface.

In this work, we introduce Differentiable Surface Splatting (DSS), the first *high fidelity point based differentiable renderer*. We utilize ideas from surface splatting [Zwi+01], where each point is represented as a disk or ellipse in the object space, which is projected onto the screen space to form a splat. The splats are then interpolated to encourage hole-free and antialiased renderings. For inverse rendering, we carefully design gradients with respect to point locations and normals by taking each forward operation apart and utilizing domain knowledge. In particular, we introduce regularization terms for the gradients to carefully drive the algorithms towards the most plausible point configuration. There are infinitely many ways splats can form a given image due to the high degree of freedom of point locations and normals. Our inverse pass ensures that points stay on local geometric structures with uniform distribution.

We apply DSS to render multi-view color images as well as auxiliary maps from a given scene. We process the rendered images with state-of-the-art techniques and show that this leads to high-quality geometries when propagated utilizing DSS. Experiments show that DSS yields significantly better results compared to previous DR methods, especially for substantial topological changes and geometric detail preservation. We focus on the particularly important application of point cloud denoising.

3.2.1. Method

In essence, a differentiable renderer $\mathcal{R}$ is designed to propagate image-level changes to scene-level parameters θ. This information can be used to optimize the parameters so that the rendered image $\mathbb{I} = \mathcal{R}(\theta)$ matches a reference image $\mathbb{I}^*$. Typically, θ includes the coordinates, normals and colors of the points, camera position and orientation, as well as lighting. Formally, this can be formulated as an optimization problem

$$\theta^* = \arg\min_{\theta} \mathcal{L}\left(\mathcal{R}(\theta), \mathbb{I}^*\right), \tag{3.3}$$

where $\mathcal{L}$ is the image loss, measuring the distance between the rendered and reference images.

Methods to solve the optimization problem (3.3) are commonly based on gradient descent which requires $\mathcal{R}$ to be differentiable with respect to θ. However, gradients w.r.t. point coordinates $\mathbf{p}$ and normals $\mathbf{n}$, i.e., $\frac{d\mathbb{I}}{d\mathbf{p}}$ and $\frac{d\mathbb{I}}{d\mathbf{n}}$, are not defined everywhere, since $\mathcal{R}$ is a discontinuous function due to occlusion events and edges.

The key to our method is two-fold. First, we define a gradient $\frac{d\mathbb{I}}{d\mathbf{p}}$ and $\frac{d\mathbb{I}}{d\mathbf{n}}$ which enables information propagation from long-range pixels without additional hyper-parameters. Second, to address the optimization difficulty that

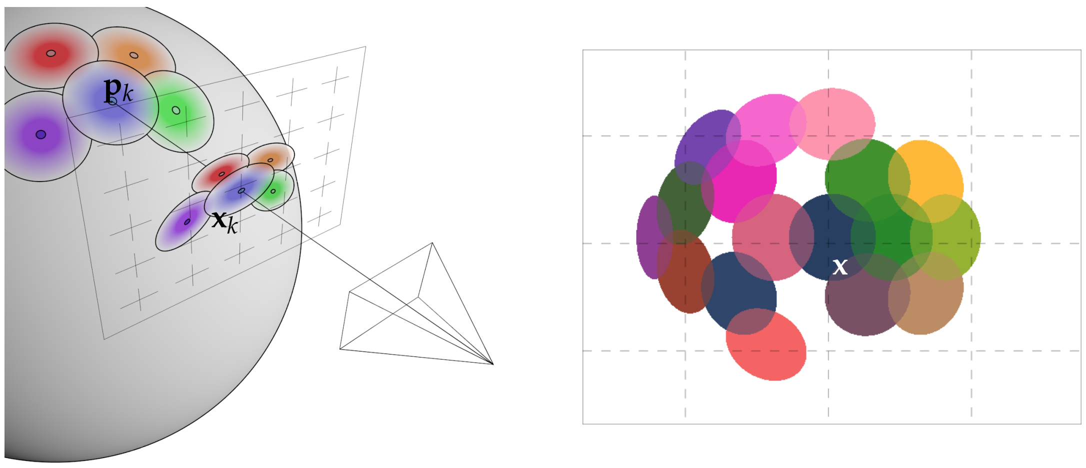


Figure 3.14.: Illustration of forward splatting using EWA [Zwi+01]. A point in space $\mathbf{p}_k$ is rendered as an anisotropic ellipse centered at the projection point $\mathbf{x}_k$. The final pixel value $\mathbb{I}_\mathbf{x}$ at a pixel $\mathbf{x}$ in the image (shown on the right) is the normalized sum of all such ellipses overlapping at $\mathbf{x}$.

arises from the significant number of degrees of freedom due to the unstructured nature of points, we introduce regularization terms that contribute to obtaining clean and smooth surface points.

In this section, we first review screen space EWA (elliptical weighted average) [Zwi+01; Hec89], which we adopt to efficiently render high-quality realistic images from point clouds. Then we propose an occlusion-aware gradient definition for the rasterization step, which, unlike previously proposed differential mesh renderers, propagates gradients to depth and allows large deformation. Lastly, we introduce two novel regularization terms for generating clean surface points.

Forward pass

The forward pass refers to the generation of a 2D image from 3D scene-level information, $\mathbb{I} = \mathcal{R}(\theta)$. Our forward pass closely follows the screen space elliptical weighted average (EWA) filtering described in [Zwi+01]. In the following, we briefly review the derivation of EWA filters.

In a nutshell, the idea of screen space EWA is to apply an isotropic Gaussian filter to the attribute $\mathit{\Phi}$ of a point in the tangent plane (defined by the normal at that point). The projection onto the image plane defines elliptical Gaussians, which, after truncation to bounded support, form a disk, or splat, as shown in Fig. 3.14. For a point $\mathbf{p}_k$, we write the filter weight of the isotropic Gaussian at position $\mathbf{p}$ as

$$\mathcal{G}_{\mathbf{p}_k,\mathbf{V}_k}(\mathbf{p}) = \frac{1}{2\pi|\mathbf{V}_k|^{\frac{1}{2}}} e^{(\mathbf{p}-\mathbf{p}_k)^\top \mathbf{V}_k^{-1}(\mathbf{p}-\mathbf{p}_k)}, \quad \mathbf{V}_k = \sigma_k^2\mathbf{I}, \tag{3.4}$$

where σ_k is the standard deviation and $\mathbf{I}$ is the identity matrix.

Now we consider the projected Gaussian in screen space. Points $\mathbf{p}_k$ and $\mathbf{p}$ are projected to $\mathbf{x}_k$ and $\mathbf{x}$, respectively. We write the Jacobian of this projection from the tangent plane to the image plane as $\mathbf{J}_k$; we refer the reader to the original surface splatting paper [Zwi+01] for the derivation of $\mathbf{J}_k$. Then at $\mathbf{x}$, the screen space elliptical Gaussian weight is

$$\begin{aligned} r_k(\mathbf{x}) &= \mathcal{G}_{\mathbf{V}_k}\left(\mathbf{J}_k^{-1}(\mathbf{x}-\mathbf{x}_k)\right) \\ &= \frac{1}{\left|\mathbf{J}_k^{-1}\right|}\mathcal{G}_{\mathbf{J}_k\mathbf{V}_k\mathbf{J}_k^\top}(\mathbf{x}-\mathbf{x}_k). \end{aligned} \tag{3.5}$$

Note that r_k is determined by the point position $\mathbf{p}_k$ and the normal $\mathbf{n}_k$, because $\mathbf{J}_k$ is determined by $\mathbf{p}_k$ and $\mathbf{n}_k$.

Next, a low-pass Gaussian filter with variance $\mathbf{I}$ is convolved with Eq. (3.5) in screen space. Thus the final elliptical Gaussian is

$$\bar{\rho}_k(\mathbf{x}) = \frac{1}{\left|\mathbf{J}_k^{-1}\right|}\mathcal{G}_{\mathbf{J}_k\mathbf{V}_k\mathbf{J}_k^\top+\mathbf{I}}(\mathbf{x}-\mathbf{x}_k). \tag{3.6}$$

In the final step, two sources of discontinuity are introduced to the fully differentiable $\bar{\rho}$. First, for computational reasons, we limit the elliptical Gaussians to a limited support in the image plane for all $\mathbf{x}$ outside a cutoff radius $\mathcal{C}$, i.e., $\frac{1}{2}\mathbf{x}^\top\left(\mathbf{J}\mathbf{V}_k\mathbf{J}^\top+\mathbf{I}\right)\mathbf{x} > \mathcal{C}$. Second, we set the Gaussian weights for occluded points to zero. Specifically, we keep a list of the maximum K (we choose $K=5$) closest points at each pixel position, and compute their depth difference to the front-most point, and then set the Gaussian weights to zero for points that are behind the front-most point by more than a threshold $\mathcal{T}$ (we set $\mathcal{T} = 1\%$ of the bounding box diagonal length). These K points are cached for gradient evaluation in backward pass, as will be explained in Sec. 3.2.1.

The resulting truncated Gaussian weight, denoted as ρ_k, can be formally defined as

$$\rho_k(\mathbf{x}) = \begin{cases} 0, & \text{if } \frac{1}{2}\mathbf{x}^\top\left(\mathbf{J}\mathbf{V}_k\mathbf{J}^\top+\mathbf{I}\right)\mathbf{x} > \mathcal{C}, \\ 0, & \text{if } \mathbf{p}_k \text{ is occluded,} \\ \bar{\rho}_k, & \text{otherwise.} \end{cases} \tag{3.7}$$

The final pixel value is simply the normalized sum of all filtered point attributes $\{\mathbf{w}_k\}_{k=0}^N$ evaluated at the center of pixels, i.e.,

$$\mathbb{I}_\mathbf{x} = \frac{\sum_{k=0}^{N-1}\rho_k(\mathbf{x})\ \mathbf{w}_k}{\sum_{k=0}^{N-1}\rho_k(\mathbf{x})}. \tag{3.8}$$

In practice, this summation can be greatly optimized by computing the

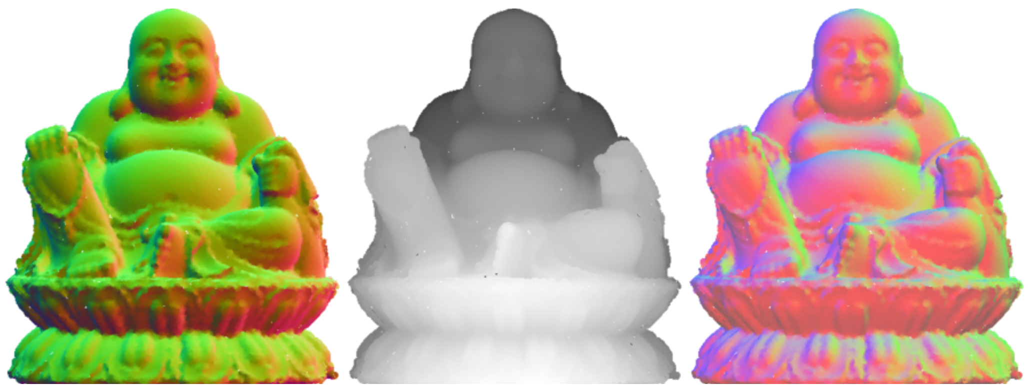


Figure 3.15.: Examples of images rendered using DSS. From left to right, we render the normals, inverse depth values and diffuse shading with three RGB-colored sun light sources.

bounding box of each ellipse and only considering points whose elliptical support covers the pixel $\mathbf{x}$.

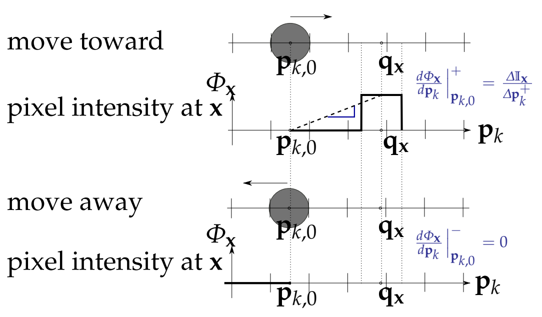
The point value Φ can be any point attribute, e.g., albedo color, shading, depth value, normal vector, etc. In most of our experiments, we use diffuse shading under three orthogonally positioned RGB-colored sun lights. This way, Φ carries strong information about point normals, and at the same time it is independent of point position (unlike with point lights), which greatly simplifies the factorization for gradient computation, as explained in Sec. 3.2.1.

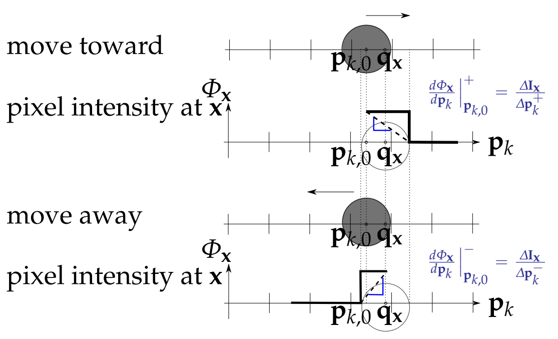
Fig. 3.15 shows some examples of rendered images. Unlike many pseudo renderers which achieve differentiability by blurring edges and transparent surfaces, our rendered images faithfully depict the actual geometry in the scene.

Backward pass

The backward pass refers to the information flow from the rendered image $\mathbb{I} = \mathcal{R}(\theta)$ to the scene parameters θ based on approximating the gradient $\frac{d\mathbb{I}}{d\theta}$. As discussed, the key to address the discontinuity of $\mathcal{R}$ lies in the approximation of the gradient $\frac{d\mathbb{I}}{d\mathbf{p}}$ and $\frac{d\mathbb{I}}{d\mathbf{n}}$.

The discontinuity of $\mathcal{R}$ is encapsulated in the truncated Gaussian weights ρ_k as described Eq. (3.7). We can factorize the discontinuous ρ_k into the fully differentiable term $\bar{\rho}_k$ and a discontinuous visibility term $h_{\mathbf{x}} \in \{0, 1\}$, i.e.,


move toward
pixel intensity at x
$\Phi_{\mathbf{x}}$
$\mathbf{p}_{k,0}$
$\mathbf{q}_{\mathbf{x}}$
$\mathbf{p}_k$
$\frac{d\Phi_{\mathbf{x}}}{d\mathbf{p}_k}\Big|^{+}_{\mathbf{p}_{k,0}} = \frac{\Delta\mathbb{I}_{\mathbf{x}}}{\Delta\mathbf{p}_k^{+}}$
move away
pixel intensity at x
$\frac{d\Phi_{\mathbf{x}}}{d\mathbf{p}_k}\Big|^{-}_{\mathbf{p}_{k,0}} = 0$


move toward
pixel intensity at x
$\Phi_{\mathbf{x}}$
$\mathbf{p}_{k,0}$
$\mathbf{q}_{\mathbf{x}}$
$\mathbf{p}_k$
$\frac{d\Phi_{\mathbf{x}}}{d\mathbf{p}_k}\Big|^{+}_{\mathbf{p}_{k,0}} = \frac{\Delta\mathbb{I}_{\mathbf{x}}}{\Delta\mathbf{p}_k^{+}}$
move away
pixel intensity at x
$\frac{d\Phi_{\mathbf{x}}}{d\mathbf{p}_k}\Big|^{-}_{\mathbf{p}_{k,0}} = \frac{\Delta\mathbb{I}_{\mathbf{x}}}{\Delta\mathbf{p}_k^{-}}$

(a) The ellipse centered at $\mathbf{p}_{k,0}$ is not visible at $\mathbf{x}$. **(b)** The ellipse centered at $\mathbf{p}_{k,0}$ is visible at $\mathbf{x}$.

Figure 3.16.: An illustration of the artificial gradient in two 1D scenarios: the ellipse centered at $\mathbf{p}_{k,0}$ is invisible (Fig. 3.16a) and visible (Fig. 3.16b) at pixel $\mathbf{x}$. $\Phi_{\mathbf{x},k}$ is the pixel intensity $\mathbb{I}_{\mathbf{x}}$ as a function of point position $\mathbf{p}_k$, $\mathbf{q}_{\mathbf{x}}$ is the coordinates of the pixel $\mathbf{x}$ back-projected to world coordinates. Notice the ellipse has constant pixel intensity after normalization (Eq. (3.8)). We approximate the discontinuous $\Phi_{\mathbf{x},k}$ as a linear function defined by the change of pixel intensity $\Delta\mathbb{I}_{\mathbf{x}}$ and the movement of the $\Delta\mathbf{p}_k$ during a visibility switch. As $\mathbf{p}_k$ moves toward ($\Delta\mathbf{p}_k^{+}$) or away ($\Delta\mathbf{p}_k^{-}$) from the pixel, we obtain two different gradient values. We define the final gradient as their sum.

$\rho_k = h_{\mathbf{x}}\bar{\rho}_k$, where $h_{\mathbf{x}}$ is defined as

$$h_{\mathbf{x}}(\mathbf{p}_k) = \begin{cases} 0, & \text{if} \frac{1}{2}\mathbf{x}^\top \left(\mathbf{J}\mathbf{V}_k\mathbf{J}^\top + \mathbf{I}\right)\mathbf{x} > \mathcal{C}, \\ 0, & \text{if } \mathbf{p}_k \text{ is occluded}, \\ 1, & \text{otherwise.} \end{cases} \tag{3.9}$$

Note that even though $h_{\mathbf{x}}$ is indirectly influenced by $\mathbf{n}_k$ through $\mathbf{J}$, since this only impacts the visibility of a small set of pixels around the ellipse, we omit this $\mathbf{n}_k$ in this formulation. Therefore, if we write $\mathbb{I}_{\mathbf{x}}$ as a function of $\mathbf{w}_k$, $\bar{\rho}_k$ and h_k, then by the chain rule we have

$$\frac{d\mathbb{I}_{\mathbf{x}}(\mathbf{w}_k, \bar{\rho}_k, h_{\mathbf{x}})}{d\mathbf{n}_k} = \frac{\partial\mathbb{I}_{\mathbf{x}}}{\partial\mathbf{w}_k}\frac{\partial\mathbf{w}_k}{\partial\mathbf{n}_k} + \frac{\partial\mathbb{I}_{\mathbf{x}}}{\partial\bar{\rho}_k}\frac{\partial\bar{\rho}_k}{\partial\mathbf{n}_k}, \tag{3.10}$$

$$\frac{d\mathbb{I}_{\mathbf{x}}(\mathbf{w}_k, \bar{\rho}_k, h_{\mathbf{x}})}{d\mathbf{p}_k} = \frac{\partial\mathbb{I}_{\mathbf{x}}}{\partial\mathbf{w}_k}\frac{\partial\mathbf{w}_k}{\partial\mathbf{p}_k} + \frac{\partial\mathbb{I}_{\mathbf{x}}}{\partial\bar{\rho}_k}\frac{\partial\bar{\rho}_k}{\partial\mathbf{p}_k} + \frac{\partial\mathbb{I}_{\mathbf{x}}}{\partial h_{\mathbf{x}}}\frac{\partial h_{\mathbf{x}}}{\partial\mathbf{p}_k}, \tag{3.11}$$

where Eq. (3.10) is fully differentiable but Eq. (3.11) is not, as $\frac{\partial h_{\mathbf{x}}}{\partial\mathbf{p}_k}$ is undefined at the edge of ellipses.

We focus on the partial gradient $\frac{\partial\mathbb{I}_{\mathbf{x}}}{\partial h_{\mathbf{x}}}\frac{\partial h_{\mathbf{x}}}{\partial\mathbf{p}_k}$. Denoting $\Phi_{\mathbf{x}}(\mathbf{p}_k) = \mathbb{I}_{\mathbf{x}}(h_{\mathbf{x}}(\mathbf{p}_k))$, this gradient can be written as $\frac{d\Phi_{\mathbf{x}}}{d\mathbf{p}_k}$, which describes the change of a pixel intensity $\mathbb{I}_{\mathbf{x}}$ due to the visibility change of a point caused by its varying position $\mathbf{p}_k$.

1D scenario. Let us first consider a simplified scenario where a single point only moves in 1D space. As shown in Fig. 3.16, $\Phi_{\mathbf{x}}$ is generally discontinuous; it is zero almost everywhere except in a small region around $\mathbf{q}_{\mathbf{x}}$, the coordinates of the pixel $\mathbf{x}$ back-projected to world coordinates. Similar to NMR

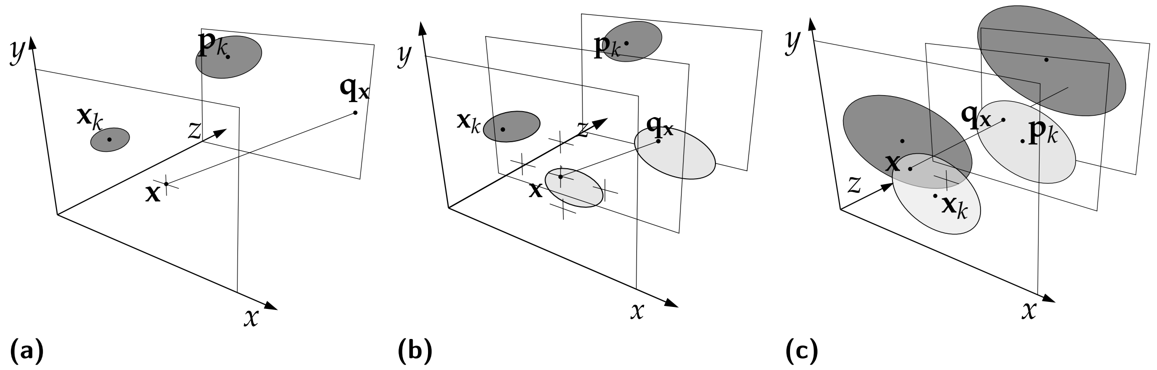


Figure 3.17.: Illustration of the 3 cases for evaluating Eq. (3.12) for 3D point clouds.

[KUH18], we approximate $\Phi_\mathbf{x}$ as a linear function defined by the change of point position $\Delta\mathbf{p}_k$ and the pixel intensity $\Delta\mathbb{I}$ before and after the visibility change.

As $\mathbf{p}_k$ moves toward or away from $\mathbf{q_x}$, we obtain two different linear functions with gradients $\frac{d\Phi_\mathbf{x}}{d\mathbf{p}_k}\Big|^{+}_{\mathbf{p}_{k,0}}$ and $\frac{d\Phi_\mathbf{x}}{d\mathbf{p}_k}\Big|^{-}_{\mathbf{p}_{k,0}}$, respectively. Specifically, when $\mathbf{p}_k$ is invisible at $\mathbf{x}$ (Fig. 3.16a), moving away will always induce zero gradient, while when $\mathbf{p}_k$ is visible, we obtain two gradients with opposite signs (Fig. 3.16b). The final gradient is defined as the sum of both, i.e.,

$$\frac{d\Phi_x}{d\mathbf{p}_k}\Big|_{\mathbf{p}_{k,0}} = \begin{cases} \frac{\Delta\mathbb{I}_\mathbf{x}}{\|\Delta\mathbf{p}_k^+\|^2+\epsilon}\Delta\mathbf{p}_k^+, & \mathbf{p}_k \text{ invisible at } \mathbf{x} \\ \frac{\Delta\mathbb{I}_\mathbf{x}}{\|\Delta\mathbf{p}_k^-\|^2+\epsilon}\Delta\mathbf{p}_k^- + \frac{\Delta\mathbb{I}_\mathbf{x}}{\|\Delta\mathbf{p}_k^+\|+\epsilon}\Delta\mathbf{p}_k^+, & \text{otherwise,} \end{cases} \tag{3.12}$$

where $\Delta\mathbf{p}_k^-$ and $\Delta\mathbf{p}_k^+$ denote the point movement toward and away from $\mathbf{x}$, starting from the current position $\mathbf{p}_{k,0}$. The value ϵ is a small constant (we set $\epsilon = 10^{-5}$). It prevents the gradient from becoming extremely large when $\mathbf{p}_k$ is close $\mathbf{q_x}$, which would lead to overshooting, oscillation and other convergence problems.

3D cases. Extending the single point 1D-scenario to a point cloud in 3D requires evaluating $\Delta\mathbb{I}$ and $\Delta\mathbf{p}$ with care. As depicted in Fig. 3.17, the following cases are considered: (a) $\mathbf{p}_k$ is not visible at $\mathbf{x}$ and $\mathbf{x}$ is not rendered by any other ellipses *in front of* $\mathbf{p}_k$; (b) $\mathbf{p}_k$ is not visible at $\mathbf{x}$ and $\mathbf{x}$ is rendered by other ellipses *in front of* $\mathbf{p}_k$; (c) $\mathbf{p}_k$ is visible at $\mathbf{x}$.

For (a) and (c), we only need to compute the gradient in screen space, whereas for (b), $\mathbf{p}_k$ must move forward in order to become visible, resulting in a negative depth gradient. Furthermore, for (a) and (b) we evaluate the new $\mathbb{I}_\mathbf{x}$ using Eq. (3.8), *adding* the contribution from $\mathbf{p}_k$, while for (c) we need to *subtract* the contribution of $\mathbf{p}_k$, which may include previously occluded ellipses into Eq. (3.8). For this purpose, as mentioned in 3.2.1, we cache an ordered list of the top-K (we choose K=5) closest ellipses that can be projected onto each pixel and save their ρ, Φ and depth values during the forward pass. The value of K is related to the merging threshold $\mathcal{T}$, and as $\mathcal{T}$ is typically small, we find $K = 5$ is sufficient even for dense point clouds. Finally, similar to NMR, when evaluating Eq. (3.12) for the optimization problem Eq. (3.3), we set the gradient to zero if the change of pixel intensity cannot reduce the image loss $\mathcal{L}$, i.e.,

$$\left.\frac{d\Phi_x}{d\mathbf{p}_k}\right|_{\mathbf{p}_k=\mathbf{p}_{k,0}} = 0 \quad \text{if} \quad \frac{d\mathcal{L}}{d\mathbb{I}_\mathbf{x}}\Delta\mathbf{I}_\mathbf{x} >= 0. \tag{3.13}$$

Comparison to other differentiable renderers A few differential renderers have been proposed for meshes. In Paparazzi [LTJ18], the rendering function is simplified enough such that the gradients can be computed analytically, which is prohibitive for silhouette change where handling significant occlusion events is required. OpenDR [LB14] computes gradients only in screen space from a small set of pixels near the boundary, which is conceptually less accurate than our definition. SoftRasterizer [Liu+19a] alters the forward rendering to make the rasterization step inherently differentiable; this leads to impeded rendering quality and relies on hyper-parameters to control the differentiability (i.e., support of non-zero gradient). The work related most closely to our approach in terms of gradient definition is the neural mesh renderer (NMR) [KUH18]. We both construct $\Phi_\mathbf{x}$ depending on the change of pixel $\Delta\mathbb{I}_\mathbf{x}$, but our method differs from NMR in the following aspects: 1. we consider the movement of $\mathbf{p}_k$ in 3D space, while NMR only considers movement in the image plane, hence neglecting the gradient in z-dimension. 2. we define the gradient for all dimensions of $\mathbf{p}$ jointly. In contrast, NMR evaluates the 1D gradients separately and consequently considers only pixels in the same row and column; 3. we consider a set of occluded and occluding ellipses projected to pixel $\mathbf{x}$. This not only leads to more accurate gradient values, but also encourages noisy points inside the model to move onto the surface, to a position with matching pixel color.

Comparison to filter-based gradient approximation. Alternatively, related to SoftRasterizer [Liu+19a] and Pix2Vex [Pet+19], one can define the gradient of the discontinuous function $\Phi_{\mathbf{x},k}$ by replacing it with a C^∞ function, e.g., a radial basis function (RBF). This is a seemingly natural choice for EWA-based point rendering, since each point is represented as a RBF in the forward pass. We compare the RBF-derived gradient with our approximation in a single point 1D scenario, and evaluate the gradient value and convergence

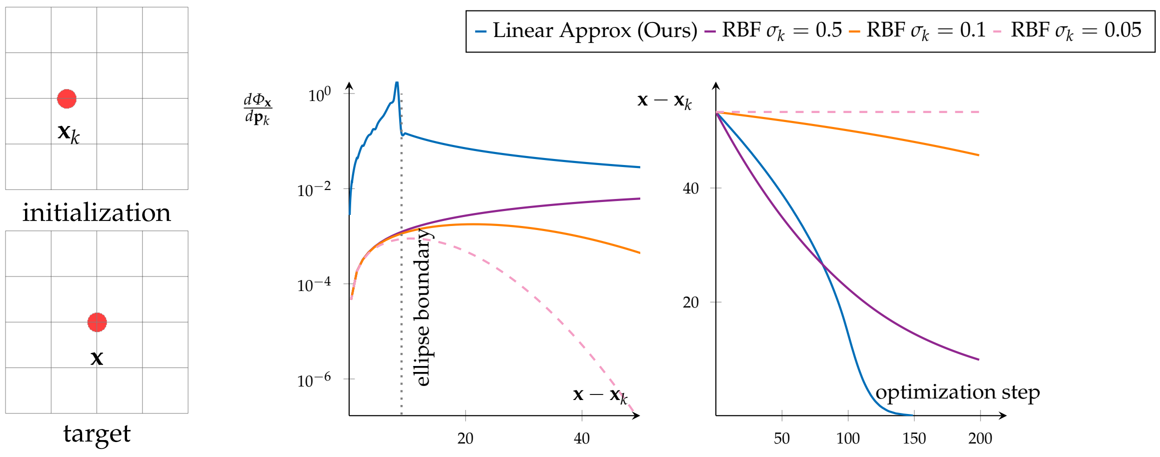


Figure 3.18.: Comparison between RBF-based gradient and our gradient approximation in terms of the gradient value at pixel $\mathbf{x}$ and residual in image space $\mathbf{x} - \mathbf{x}_k$ as we optimize the point position $\mathbf{p}_k$ in the initial rendered image to match the target image. While our approximation (blue) is invariant under the choice of the hyper-parameter σ_k, the RBF-based gradient (purple, orange and the dashed pink curves) is highly sensitive to its value. Small variations of σ_k can severely impact the convergence rate.

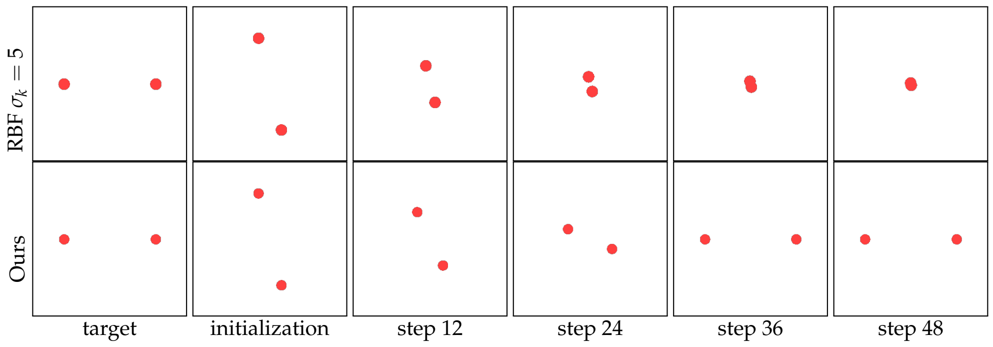


Figure 3.19.: Optimization progress using our gradient approximation and RBF-derived gradient. The RBF-derived gradient is prone to local minima when optimizing for multiple points.

rate. As shown in Fig. 3.18, the RBF-derived gradient is highly sensitive to the Gaussian filter's standard deviation σ_k. A small σ_k leads to diminishing gradient for distant pixels, causing convergence issues, as demonstrated with the dashed plot. For a large σ_k, $\|\frac{d\Phi_\mathbf{x}}{d\mathbf{p}_k}\|$ can increase with $\mathbf{x} - \mathbf{x}_k$ when the pixel is outside the ellipse boundary; as a result, the optimization is prone to fall into a local minima in multi-point scenario as shown in Fig. 3.19. Lastly, it is not obvious how to extend the RBF derivation for the depth dimension, while the linear approximation naturally applies to all dimensions.

Surface regularization

The lack of structure in point clouds, while providing freedom of massive topology changes, can pose a significant challenge for optimization. First, the gradient derivation is entirely paralleled; as a result, points move irrespective of each other. Secondly, as the movement of points will only induce small and sparse changes in the rendered image, gradients on each point are less structured compared to corresponding gradients for meshes. Without proper regularization, one can quickly end up in local minima.

Inspired by [Hua+09; ÖGG09], we propose regularization to address this problem based on two parts: a repulsion and a projection term. The repulsion term is aimed at generating uniform point distributions by maximizing the distances between its neighbors on a local projection plane, while the projection term preserves clean surfaces by minimizing the distance from the point to the surface tangent plane.

Obviously, both terms require finding a reliable surface tangent plane. However, this can be challenging, since during optimization, especially in the case of multi-view joint optimization, intermediate point clouds can be very noisy and contain many occluded points inside the model, hence we propose a weighted PCA to penalize the occluded inner points. In addition to the commonly used bilateral weights which considers both the point-to-point euclidean distance and the normal similarity, we propose a visibility weight, which penalizes occluded points, since they are more likely to be outliers inside the model.

Let $\mathbf{p}_i$ denote a point in question and $\mathbf{p}_k$ denote one point in its neighborhood, $\mathbf{p}_k \in \{\mathbf{p} | \, \|\mathbf{p} - \mathbf{p}_i\| \leq \mathcal{D}\}$, we propose computing a weighted PCA using the following weights

$$\psi_{ik} = \exp\left(-\frac{\|\mathbf{p}_i - \mathbf{p}_k\|^2}{\mathcal{D}^2}\right) \tag{3.14}$$

$$\theta_{ik} = \exp\left(-\frac{\left(1 - \mathbf{n}_k^\top \mathbf{n}_i\right)^2}{\max\left(1e^{-5}, 1 - \cos\left(\Theta\right)\right)}\right) \tag{3.15}$$

$$\phi_{ik} = \frac{1}{o_k + 1}, \tag{3.16}$$

where ψ_{ik} and θ_{ik} are bilateral weights which favor neighboring points that are spatially close and have similar normal orientation respectively, and ϕ_{ik} is the proposed visibility weight which is defined using an occlusion counter o_k that counts the number of times $\mathbf{p}_k$ is occluded in all camera views. Then a

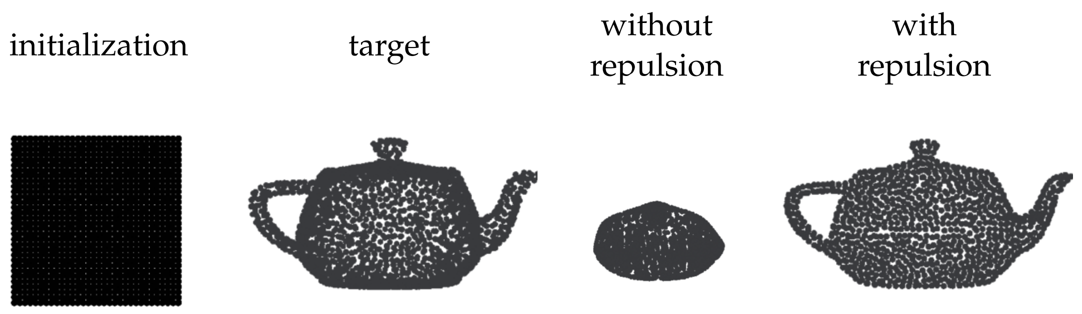


Figure 3.20.: The effect of repulsion regularization. We deform a 2D grid to the teapot. Without the repulsion term, points cluster in the center of the target shape. The repulsion term penalizes this type of local minima and encourages a uniform point distribution.

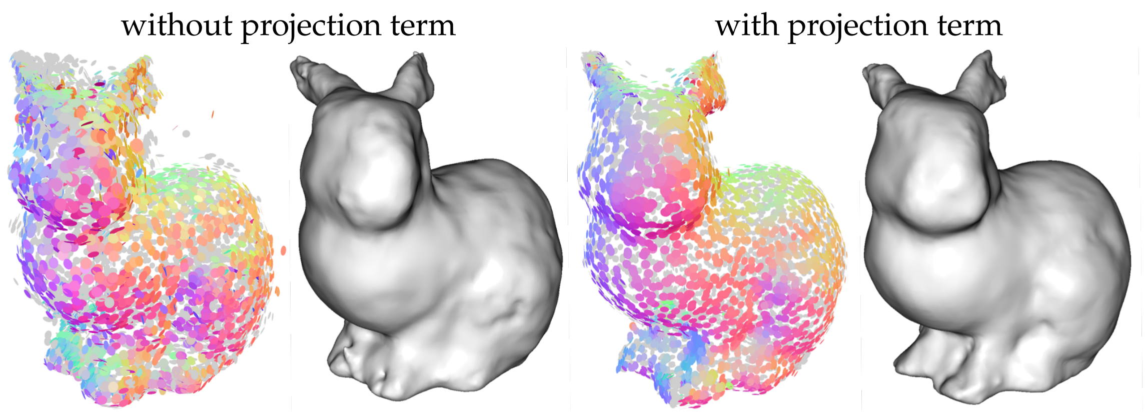


Figure 3.21.: The effect of projection regularization. The projection term effectively enforces points to form a local manifold. For a better visualization of outliers inside and outside of the object, we use a small disk radius and render the backside of the disks using light gray color.

reliable projection plane can be obtained using singular value decomposition from weighted vectors $w_{ik}\left(\mathbf{p}_i - \sum_{k=0}^{K} w_{ik}\mathbf{p}_k\right)$, where $w_{ik} = \frac{\psi_{ik}\theta_{ik}\phi_{ik}}{\sum_{i=0}^{K}\psi_{ik}\theta_{ik}\phi_{ik}}$.

For the repulsion term, the projected point-to-point distance is obtained via $d_{ik} = \tilde{\mathbf{V}}\tilde{\mathbf{V}}^{\top}\left(\mathbf{p}_i - \mathbf{p}_k\right)$, where $\tilde{\mathbf{V}}$ contains the first 2 principal components. We define the repulsion loss as follows and minimize it together with the per-pixel image loss

$$\mathcal{L}_r = \frac{1}{N}\sum_{N}\sum_{K}\frac{\psi_{ik}}{d_{ik}^2 + 10^{-4}}. \tag{3.17}$$

For the projection term, we minimize the point-to-plane distance via $d_{ik} = \mathbf{V}_n\mathbf{V}^{\top}\left(\mathbf{p}_i - \mathbf{p}_k\right)$, where $\mathbf{V}_n$ is the last components. Correspondingly, the projection loss is defined as

$$\mathcal{L}_p = \frac{1}{N}\sum_{N}\sum_{K} w_{ik} d_{ik}^2. \tag{3.18}$$

The effect of repulsion and projection terms are clearly demonstrated in Fig. 3.20 and Fig. 3.21. In Fig. 3.20, we aim to move points lying on a 2D grid to match the silhouette of a 3D teapot. Without the repulsion term, points quickly shrink to the center of the reference shape, which is a common local minima since the gradient coming from surrounding pixels cancel each other out. With the repulsion term, the points can escape such local minima and distribute evenly inside the silhouette. In Fig. 3.21 we deform a sphere to bunny from 12 views. Without projection regularization, points are scattered within and outside the surface. In contrast, when the projection term is applied, we can obtain a clean and smooth surface.

Implementation details.

Optimization objective. We choose Symmetric Mean Absolute Percentage Error (SMAPE) as the image loss $\mathcal{L}_{\mathbb{I}}$. SMAPE is designed for high dynamic range images such as rendered images therefore it behaves more stable for unbounded values [Vog+18]. It is defined as

$$\mathcal{L}_{\mathbb{I}} = \frac{1}{HW} \sum_{\mathbf{x} \in \mathbb{I}} \sum_{c}^{C} \frac{|\mathbb{I}_{\mathbf{x},c} - \mathbb{I}^*_{\mathbf{x},c}|}{|\mathbb{I}_{\mathbf{x},c}| + |\mathbb{I}^*_{\mathbf{x},c}| + \epsilon}, \tag{3.19}$$

where H and W are the dimensions of the image, the value of ϵ is typically chosen as 10^{-5}.

The total optimization objective corresponding to Eq. (3.3) for a set of views V amounts to

$$\sum_{v=0}^{V} \mathcal{L}\left(\mathbb{I}_v, \mathbb{I}^*_v\right) = \sum_{v=0}^{V} \mathcal{L}_{\mathbb{I}}\left(\mathbb{I}_v, \mathbb{I}^*_v\right) + \gamma_p \mathcal{L}_p + \gamma_r \mathcal{L}_r. \tag{3.20}$$

Loss weights γ_p and γ_r are typically chosen to be $0.02, 0.05$ respectively.

Alternating normal and point update For meshes, the face normals are determined by point positions. For points, though, normals and point positions can be treated as independent entities thus optimized individually. Our pixel value factorization in Eq. (3.11) and Eq. (3.10) means that, the gradient on point positions $\mathbf{p}$ mainly stems from the visibility term, while gradients on normals $\mathbf{n}$ can be derived from $\mathbf{w}_k$ and ρ_k. Because the gradient w.r.t. $\mathbf{n}$ and $\mathbf{p}$ assumes the other stays fixed, we apply the update of $\mathbf{n}$ and $\mathbf{p}$ in an alternating fashion. Specifically, we start with normals, execute optimization for $T_{\mathbf{n}}$ times then we optimize point positions for $T_{\mathbf{p}}$ times.

As observed in many point denoising works [ÖGG09; Hua+09; Gue+18], finding the right normal is the key for obtaining clean surfaces. Hence we efficiently utilize the improved normals even if the point positions are not being updated, in that we directly update the point positions using the gradient from the regularization terms $\frac{\partial \mathcal{L}_p}{\partial \mathbf{p}_k}$ and $\frac{\partial \mathcal{L}_r}{\partial \mathbf{p}_k}$. In fact, for local shape surface modification, this simple strategy consistently yields satisfying results.

Error-aware view sampling

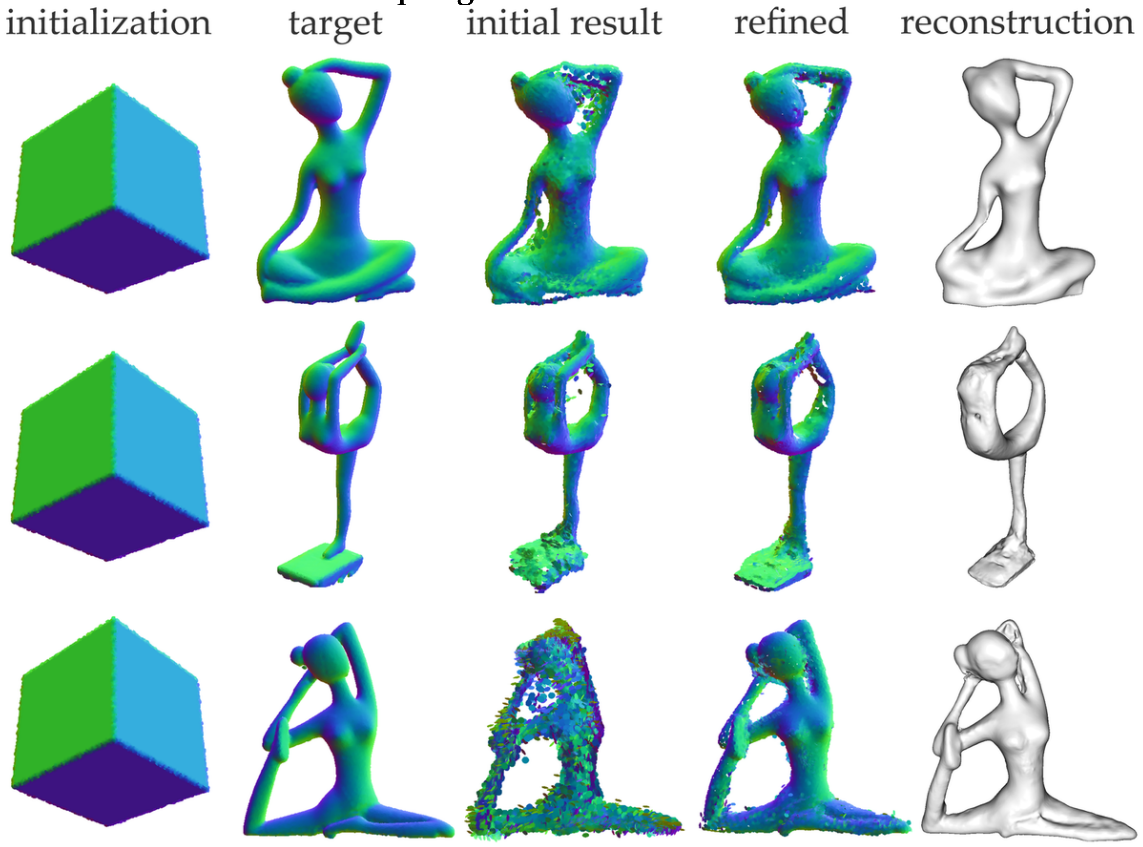


Figure 3.22.: DSS deforms a cube to three different Yoga models. Noisy points may occur when camera views are under-sampled or occluded (as shown in the initial result). We apply an additional refinement step improving the view sampling as described in Sec. 3.2.1.

View selection is very important for quick convergence. In our experiments, we aim to cover all possible angles by sampling camera positions from a hulling sphere using farthest point sampling. Then we randomly perturb the sampled position and set the camera to look at the center of the object. The sampling process is repeated periodically to further improve optimization.

However, for shapes with complex topology, such a sampling scheme is not enough. We propose an error-aware view sampling scheme which chooses the new camera positions based on the current image loss.

Specifically, we downsample the reference image and the rendered result, then compute the pixel position with the largest image error. Then we find K points whose projection is closest to the found pixel. The mean 3D position of these points will be the center of focus. Finally, we sample camera positions on a sphere around this focal point with a relatively small distance. Such techniques help us to improve point positions in small holes during large shape deformation. An example of applying this sampling technique is shown in Fig. 3.22.

3.2.2. Results

We evaluate the performance of DSS by comparing it to state-of-the-art DRs, and demonstrate its applications in point-based geometry editing and filtering.

Our method is implemented in Pytorch [Pas+17], we use stochastic gradient descent with Nesterov momentum [Sut+13] for optimization. A learning rate of 5 and 5000 is used for points and normals, respectively. In all experiments, we render in back-face culling mode with 256×256 resolution and diffuse shading, using RGB sun lights fixed relative to the camera position.

Unless otherwise stated, we optimize for up to 16 cycles of $T_{\mathbf{n}}$ and $T_{\mathbf{p}}$ optimization steps for point normal and position (for large deformation $T_{\mathbf{p}} = 25$ and $T_{\mathbf{n}} = 15$; for local surface editing $T_{\mathbf{n}} = 19$ and $T_{\mathbf{p}} = 1$). In each cycle, 12 randomly sampled views are used simultaneously for an optimization step. To test our algorithms for noise resilience, we use random white Gaussian noise with a standard deviation measured relative to the diagonal length of the bounding box of the input model.

Comparison of different DRs.

We compare DSS in terms of large geometry deformation to the state-of-the-art mesh-based DRs, i.e., OpenDR [LB14], NMR [KUH18] and Paparazzi [LTJ18]. For the mesh DRs, we use the publicly available code provided by the authors and report the best results among experiments using different parameters (e.g., number of cameras and learning rate). All methods use the same initial and target shape, and similar camera positions.

The results are shown in Fig. 3.23. Among the mesh-based methods, OpenDR can best deform an input sphere to match the silhouette of a target teapot. However, none of these methods can handle topology changes (see the handle) and struggle with large deformation (see the spout). In comparison, DSS recovers these geometry structures with high fidelity and at the same time

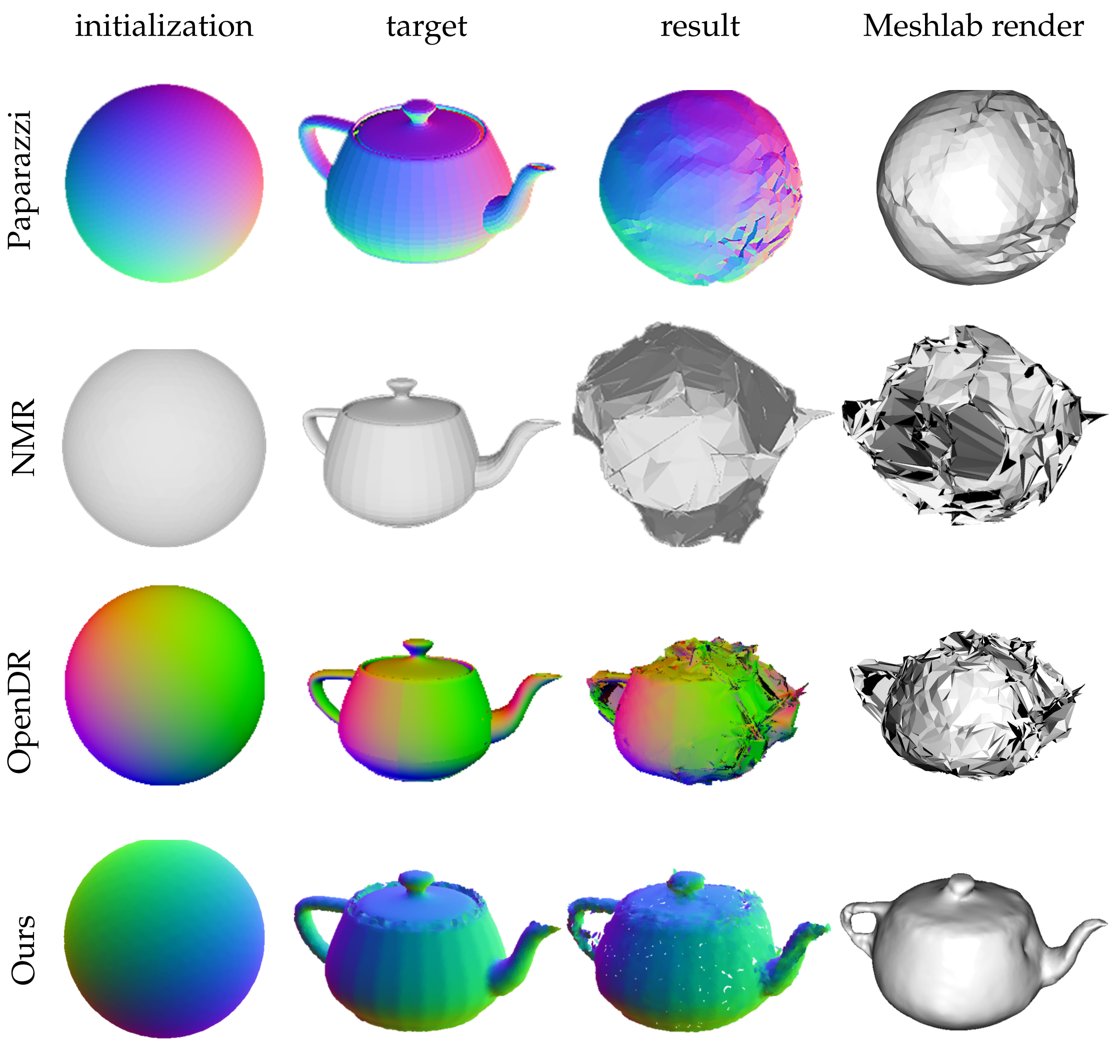


Figure 3.23.: Large shape deformation with topological changes, compared with three mesh-based DRs, namely Paparazzi [LTJ18], OpenDR [LB14] and Neural Mesh Renderer [KUH18]. Compared to the mesh-based approaches, DSS faithfully recovers the handle and cover of the teapot thanks to the flexibility of the point-based representation.

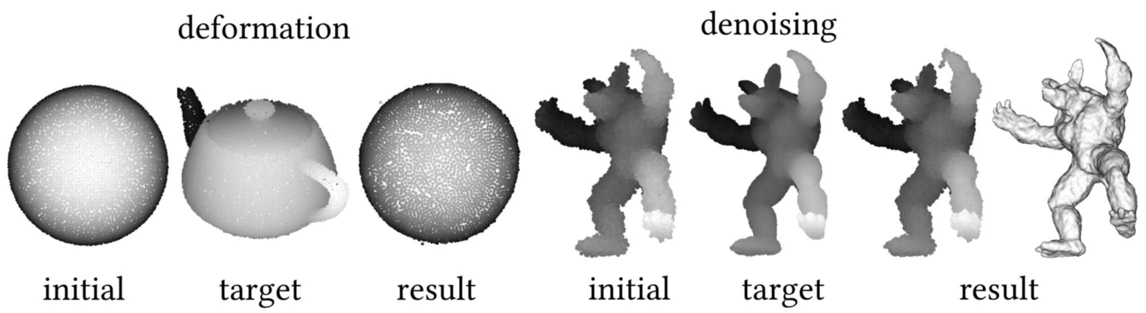


Figure 3.24.: A simple projection-based point renderer which renders depth values fails in deformation and denoising tasks.

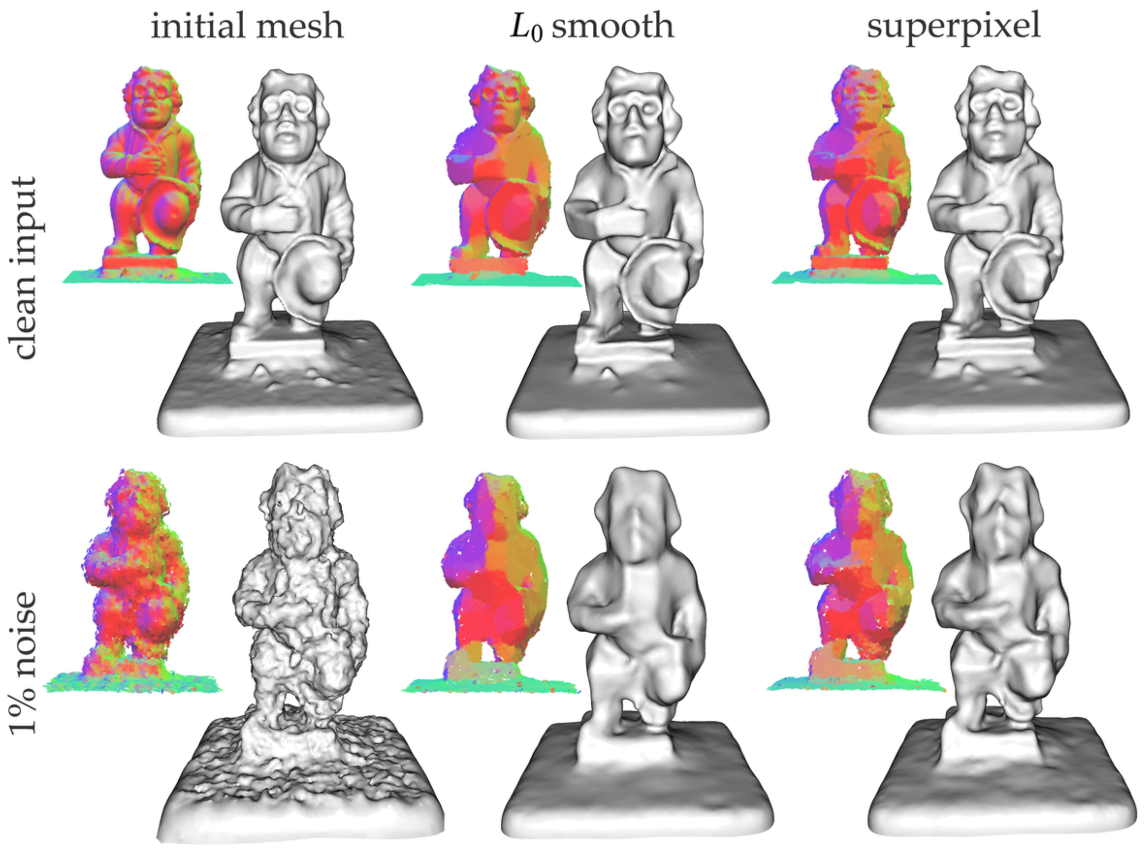


Figure 3.25.: Examples of DSS-based geometry filtering. We apply image filters on the DSS rendered multi-view images and propagate the changes of pixel values to point positions and normals. From left to right are the Poisson reconstruction of input points, points filtered by L_0-smoothing, and superpixel segmentation. In the first row, a clean point cloud is used as input, while in the second row, we add 1% white Gaussian noise. In both cases, DSS can update the geometry accordingly to match the changes in the image domain.

produces more elaborate surface details (see the pattern on the body of the teapot).

Finally, we compare with a naive point DR based on [Rov+18a; Rov+18b; ID18], where the pixel intensities are represented by the sum of smoothed depth values. As shown in Fig. 3.24, such a naive implementation of point-based DR cannot handle large-scale shape deformation nor fine-scale denoising, because position gradient is confined locally restricting long-range movement and normal information is not utilized to fine-grained geometry update.

Application: shape editing via image filter

As demonstrated in Paparazzi, one important application of DR is shape editing using existing image filters. It allows many kinds of geometric filtering

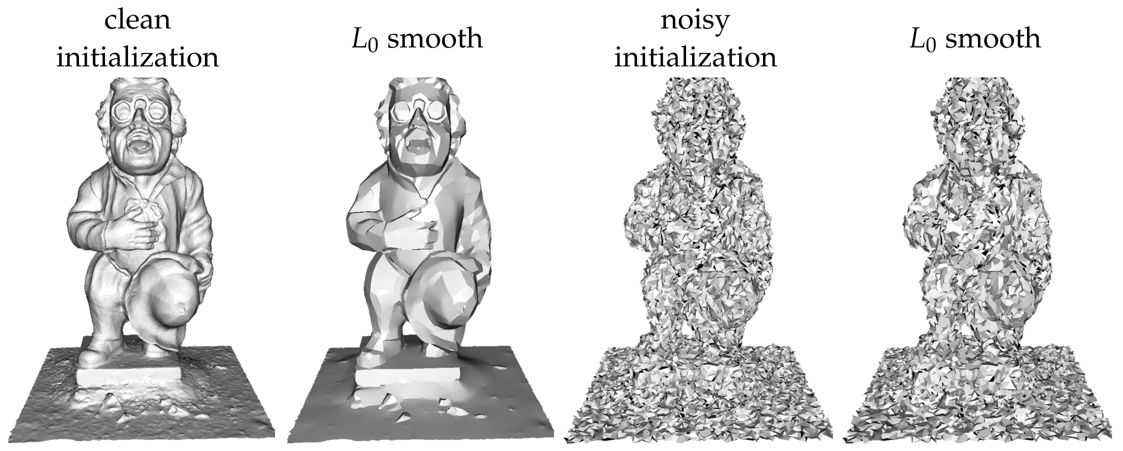


Figure 3.26.: Paparazzi [LTJ18] successfully applies a L_0 image filter to a clean mesh (Left) but fails on an input containing 0.5 % noise (Right).

and style transfer, which would have been challenging to define purely in the geometry domain. This benefit also applies to DSS.

We experimented with two types of image filters, L0 smoothing [Xu+11] and superpixel segmentation [Ach+12]. These filters are applied to the original rendered images to create references. Like Paparazzi, we keep the silhouette of the shape and change the local surface geometry by updating point normals, then the projection and repulsion regularization are applied to correct the point positions.

As shown in Fig. 3.25, DSS successfully transfers image-level changes to geometry. Even under 1% noise, DSS continues to produce reasonable results. In contrast, mesh-based DRs are sensitive to input noise, because it leads to small piecewise structures and flipped faces in image space (see Fig. 3.26), which are troublesome for the computation of gradients. In comparison, points are free of any structural constraints; thus, DSS can update normals and positions independently, which makes it robust under noise.

Application: point cloud denoising

One of the benefits of the shape-from-rendering framework is the possibility to leverage powerful neural networks and vast 2D data. We demonstrate this advantage in a point cloud denoising task, which is known to be an ill-posed problem where handcrafted priors struggle with recovering all levels of smooth and sharp features.

First, we train an image denoising network based on the Pix2Pix [Iso+17] framework, which utilizes the generative adversarial network [Goo+14] to

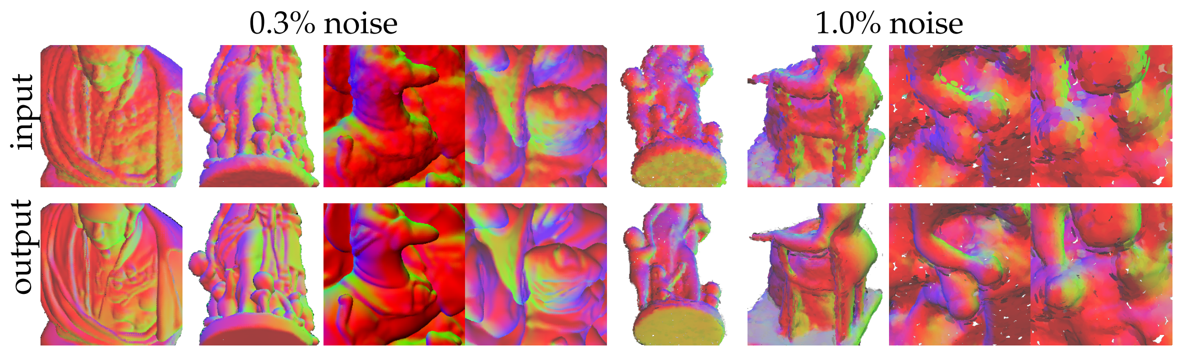


Figure 3.27.: Examples of the input and output of the Pix2Pix denoising network. We train two models to target two different noise levels (0.3% and 1.0%). In both cases, the network is able to recover smoothly detailed geometry, while the 0.3% noise variant generates more fine-grained details.

add plausible details for improved visual quality (we refer readers to Appendix for further details on the training data preparation as well as the adapted network architecture). During test time, we render images of the noisy point cloud from different views and use the trained Pix2Pix network to reconstruct geometric structure from the noisy images. Finally, we update the point cloud using DSS with the denoised images as reference.

To maximize the amount of hallucinated details, we train two models for 1.0% and 0.3% noise respectively. Fig. 3.27 shows some examples of the input and output of the network. Hallucinated delicate structures can be observed clearly in both noise levels. Furthermore, even though our Pix2Pix model is not trained with view-consistency constraints, the hallucinated details remain mostly consistent across views. In case small inconsistencies appear in regions where a large amount of high-frequency details are created, DSS is still able to transfer plausible details from the 2D to the 3D domain without visible artefacts, as shown in Fig. 3.30, thanks to simultaneous multi-view optimization.

Evaluation of DSS denoising. We perform quantitative and qualitative comparison with state-of-the-art optimization-based methods WLOP [Hua+09], EAR [Hua+13a], RIMLS [ÖGG09] and GPF [Lu+18], as well as a learning-based method, PointCleanNet [Rak+19], using the code provided by the authors. For quantitative comparison, we compute Chamfer distance (CD) and Hausdorff distance (HD) between the reconstructed and ground truth surface.

First, we compare the denoising performance on a relatively noisy (1% noise) and sparse (20K points) input data, as shown in Fig. 3.29. Optimization-based methods can reconstruct a smooth surface but also smear the low-level details. The learning-based PointCleanNet can preserve some detailed

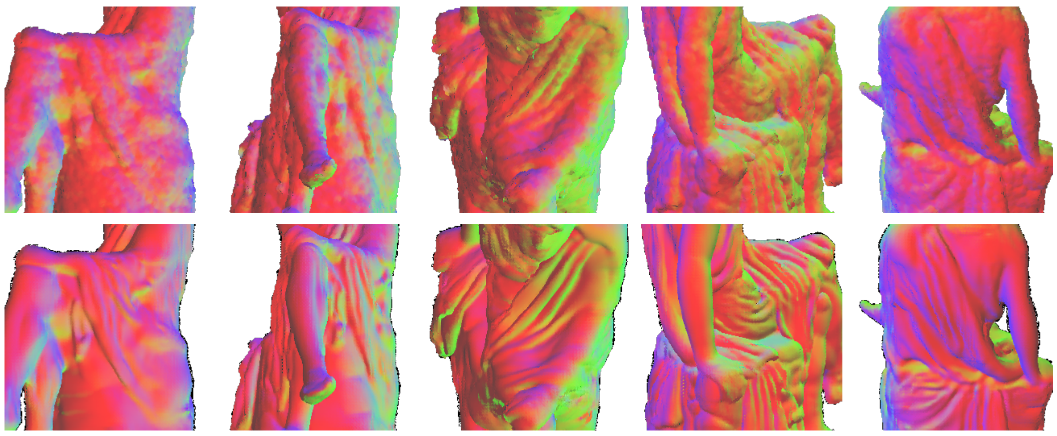


Figure 3.28.: Examples of multi-view Pix2Pix denoising on the same 3D model. As our Pix2Pix model processes each view independently, inconsistencies across different views might occur in the generated high-frequency details. In spite of that, DSS recovers plausible structures in the 3D shape (see Fig. 3.30) thanks to our simultaneous multi-view optimization.

structure, like the fingers of armadillo, but cannot remove all high-frequency noise. We test DSS with two image filters, i.e., the L_0 smoothing and the Pix2Pix model trained on data with 20K points and 1% noise. L_0-DSS has a similar performance with the optimization-based method. Pix2Pix-DSS outperforms the other compared methods quantitatively and qualitatively.

Second, we evaluate on a relatively smooth (0.3% noise) and dense (100K points) input data, as shown in Fig. 3.30. Optimization-based methods and L_0-DSS produce high-accuracy reconstruction. PointCleanNet's result deteriorates significantly, due to generalizability issues which is common for direct learning-based methods. In contrast, the proposed image-to-geometry denoising method is inherently less sensitive to the characteristic of points sampling. As a result, even though our Pix2Pix model is trained with 20K points, Pix2Pix-DSS reconstructs a clean surface, and at the same time shows abundant hallucinated details.

Finally, we evaluate Pix2Pix-DSS using real scanned data.

We acquire a 3D scan of a dragon model by ourselves using a hand-held scanner and resample 50K points as input. We compare the point cloud cleaning performance of EAR, RIMLS, PointCleanNet and Ours as shown in Fig. 3.31. EAR outputs clean and smooth surfaces but tends to produce underwhelming geometry details. RIMLS preserves sharp geometry features, but compared to our method, its result contains more low-frequency noise. The output of PointCleanNet is notably noisier than other methods, while its reconstructed model falls between EAR and RIMLS in terms of detail preservation and surface smoothness. In comparison, our method yields clean and smooth surfaces with rich geometry details.

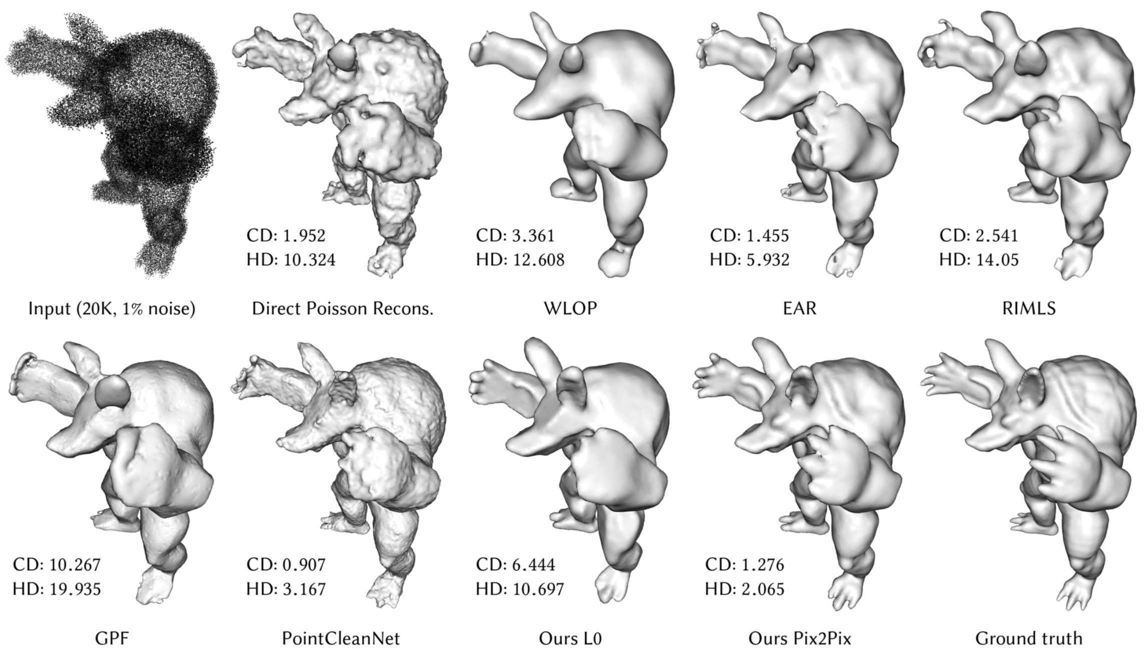


Figure 3.29.: Quantitative and qualitative comparison of point cloud denoising. The Chamfer distance (CD) and Hausdorff distance (HD) scaled by 10^{-4} and 10^{-3}. With respect to HD, our method outperforms its competitors, for CD only PointCleanNet can generate better, albeit noisy, results.

Performance

Our forward and backward rasterization passes are implemented in CUDA. We benchmark the runtime using an NVIDIA 1080Ti GPU with CUDA 10.0 and summarize the runtime as well as memory demand for all of the applications mentioned above on one exemplary model in Table 3.4. As before, models are rendered with 256×256 resolution and 12 views are used per optimization step.

As a reference, for the teapot example, one optimization step in Paparazzi and Neural Mesh Renderer takes about 50ms and 160ms respectively, whereas it

	Fig. 3.23	Fig. 3.25	Fig. 3.30
number of points	8003	20000	100000
total opt. steps for position	200	8	8
total opt. steps for normal	120	152	152
avg. forward time (ms)	19.3	42.8	258.1
avg. backward time (ms)	79.9	164.6	680.2
total time (s)	336	665	1951
GPU memory (MB)	1.7MB	1.8MB	2.3MB

Table 3.4.: Runtime and GPU memory demand for exemplar models in different applications. The images are rendered with 256×256 resolution and 12 views are used per optimization step.

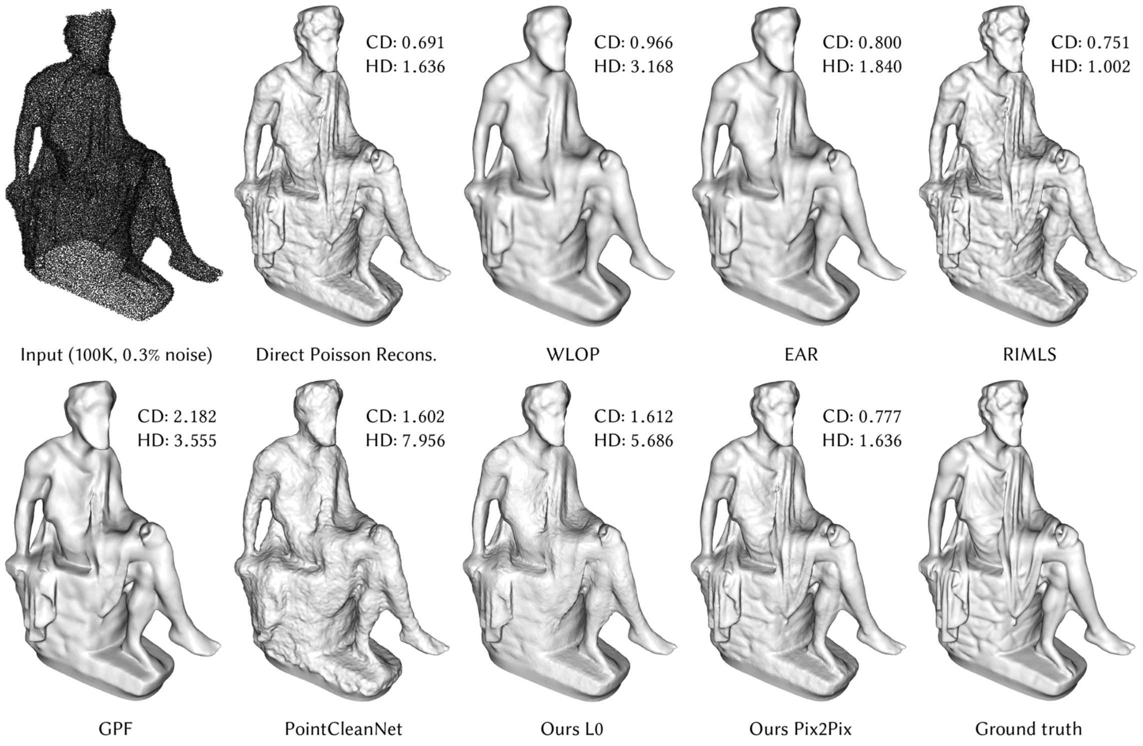


Figure 3.30.: Quantitative and qualitative comparison of point cloud denoising with 0.3% noise. We report CD and HD scaled by 10^{-4} and 10^{-3}. Despite some methods performing better with respect to quantitative evaluation, our result matches the ground truth closely in contrast to other methods.

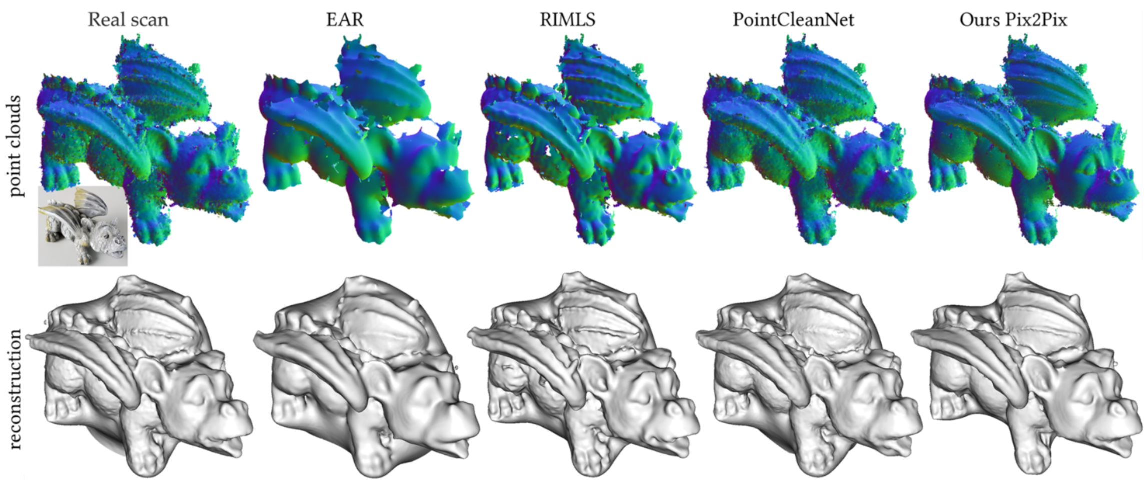


Figure 3.31.: Qualitative comparison of point cloud denoising on a dragon model acquired using a hand-held scanner (without intermediate mesh representation). Our Pix2Pix-DSS outperforms the compared methods.

takes us 100ms (see the second row in Table 3.4). However, since Paparazzi does not jointly optimize multiple-views, it requires more iterations for convergence. In the L0-Smoothing example (see Fig. 3.26), it takes 30 minutes and 30000 optimization steps to obtain the final result, whereas DSS needs 160 steps and 11 minutes for a similar result (see the third row in Table 3.4).

3.3. Concluding remarks

Contributions. In this chapter, we presented two algorithms for point cloud processing and enhancement.

First, we proposed a multi-scale progressive point set upsampling network (MPU) that reveals detailed geometric structures from sparse and noisy inputs. Specifically, we introduced a) an end-to-end progressive multi-scale neural network architecture, designed to attend to different levels of detail for high-resolution point upsampling; b) a patch-based architecture, which adjust the spatial span depending on the scope of the receptive field; c) a series of architectural improvements, which contribute to higher efficiency and better performance, including dense connections for feature extraction, code assignment for efficient feature expansion, as well as bilateral feature interpolation for interlinks across the steps. Extensive experiments and studies demonstrate the superiority of our method compared with previous state-of-the-art techniques.

In the second work, we developed a high-quality differentiable point renderer based on surface splatting, called Differentiable Surface Splatting (DSS). DSS inherits the flexibility of point-based representations and can propagate gradients to point positions and normals to produce accurate geometries and topologies. We showcased a few applications of how such a renderer can be utilized for image-based geometry processing. In particular, combining DSS with contemporary deep neural network architectures yielded state-of-the-art results.

Limitations. Both MPU and DSS are one of the first achievements in their respective research area of point cloud processing. There exist several drawbacks that can be or have been revamped since their publications.

Trained with point clouds obtained either from uniform sampling or virtual partial scans, MPU assumes fixed homogeneous point distribution. Although MPU performs reasonably robust under mild violation of this assumption – as it can fill holes in the real scanned data – its quality degenerates when the test data contains large missing areas or has a distribution pattern different from the one observed in the training data. One solution to this problem is to use another loss function in place of (or in addition to) the proposed Chamfer Distance to reduce the overfitting to the training data and encourage the discovery of "natural" point distribution. PU-GAN [Li+19] explored in the direction by using an adversarial training loss. To further improve uniformity in the upsampled point clouds, a two-stage approach, such as the one proposed by Li et al. [Li+21b], can be adopted to first densify the points then refine them to improve uniformity. In the similar spirit, one can address the larger holes in the input data by first reconstructing the missing regions then upsampling the entire point cloud to obtain the dense output. Wang et al. [WAJL20] combined this idea with the previously mentioned adversarial loss for point cloud completion.

The first limitation of DSS is the performance. This is mainly due to the heavy computation during the gradient approximation, since one must re-evaluate the color of each pixel for every splat, resulting in an extra forward pass during the backward pass. Essentially, this approach assumes that during the optimization of the point positions the colors and normals of the points stay constant. Correspondingly, it works well with alternating optimization strategy as described in Sec. 3.2.1. While our initial implementation closely follows this approach, we considerably simplified the formulation in our revised implementation by dropping the dependency of color information, *i.e.* replacing the change of pixel color in Eq. (3.12) with the change of silhouette. Together with other GPU-specific tricks [Rav+20], this revised implementation yielded a 20$\times$ performance gain. Another limitation of DSS lies in the locally defined surface regularization terms. Since both the repulsion and projection terms rely on local plane fitting using PCA, they are not robust under the presence of strong outliers. Therefore, a more global geometric prior is desirable. Alternatively, as proposed by Lassner and Zollhofer [LZ21a], a very different approach would be to consider volumetric splatting, where each point additionally carries an opacity attribute. When the extraction of the surface is not strictly required, the volumetric approach bypasses surface regularizations and the need for a known 2D silhouette.

CHAPTER 4

Implicit Surface Reconstruction

Implicit surface representations are a family of continuous surface representations. They represent a surface $\mathcal{S}$ as the levelset of some parameterized scalar functions $f : \mathbb{R}^3 \mapsto \mathbb{R}$, *i.e.* $\mathcal{S} = \left\{\mathbf{x} \in \mathbb{R}^3 | f\left(\mathbf{x}\right) = C\right\}$, where C is a scalar constant. Implicit surface representations are a powerful representation for surfaces because they are not tied to a specific resolution, which makes them a favorable choice for representing highly detailed surfaces.

Obviously, the choice of f determines the surface properties. Traditionally, f is commonly modeled as piecewise polynomial functions to favor arbitrary smoothness. Recently, neural networks have emerged as a powerful alternative. In this case, the weights of the networks are trained to "overfit"[1] (samples of) the implicit function. Often, this results in an efficient representation that can be conveniently deployed for further applications such as neural rendering [Yar+20; Mil+20; Nie+20] and simulation [Den+20].

In this chapter, we enhance neural implicit representations with two algorithms. The first one introduces a novel formulation of signed distance functions, which leads to a corresponding novel network architecture that boosts the representational power of neural networks in approximating highly detailed surfaces; The second one proposes a technique that complements the optimization of neural implicit functions and enhances the quality and performance of implicit surface reconstruction under imperfect data.

[1]"overfit" underlines the difference to the typical use of neural nets in machine learning frameworks, where the neural networks are trained and deployed in two distinct data sets.

4.1. Detail-driven implicit surface representation

While neural implicits can theoretically model geometry with infinite resolution, in practice the output resolution is dependent on the representational power of neural nets. So far, the research community approaches the problem from two main directions. The first is to partition the implicit function using spatial structures [Cha+20a; Jia+20; Liu+20b; Tak+21], thus making the memory and computation demands dependent on the geometric complexity. The other direction focuses on improving networks' ability to represent high-frequency signals, either in a preprocessing step (referred to as positional encoding) [Mil+20] or by using sinusoidal representation networks (SIREN) [Sit+20]. However, training these networks is very challenging, as they are prone to overfitting and optimization local minima.

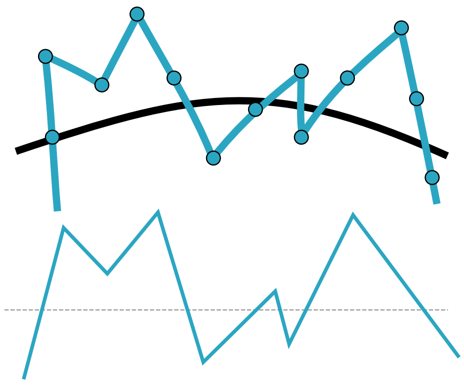


Figure 4.1.: Displacement mapping in 1D. The detailed surface (upper blue) is created by offsetting samples of the base surface (upper black) using the height map shown below.

Inspired by the classic computer graphics technique, displacement mapping [Coo84; CCC87], we propose a novel parameterization of neural implicit functions, *implicit displacement field*, abbreviated as IDF, to circumvent the above issues. Our method automatically disentangles a given detailed shape into a coarse base shape represented as a continuous, low-frequency signed distance function and a continuous high-frequency implicit displacement field, which offsets the base iso-surface along the normal direction.

The key novelty of our approach lies in extending the classic displacement mapping, which is discrete and lies only on the base surface, to a continuous function in the $\mathbb{R}^3$ domain and incorporating it into contemporary neural implicit representations, ergo achieving a disentanglement of geometric details in an unsupervised manner.

Our main technical contribution includes

1. a principled and theoretically grounded extension of explicit discrete displacement mapping to the implicit formulation,
2. a neural architecture that creates a geometrically interpretable frequency hierarchy in the neural implicit shape representation by exploiting the inductive bias of SIRENs, and

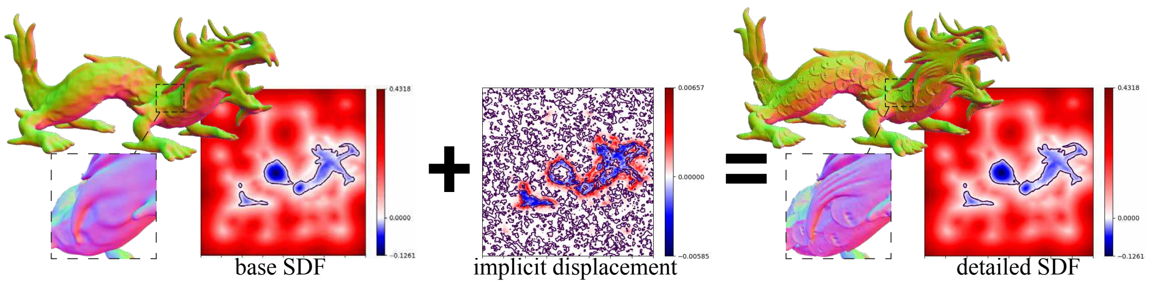


Figure 4.2.: Method overview. We represent detailed geometries as a sum of a coarse base shape represented as low-frequency signed distance function and a high-frequency implicit displacement field, which offsets the base iso-surface along the base's normal directions.

3. introducing transferable implicit displacement fields by replacing the common coordinates input with carefully constructed transferrable features, thus opening up new opportunities for implicit geometry manipulation and shape modeling.

Systematic evaluations show that our approach is significantly more powerful in representing geometric details, while being lightweight and highly stable in training.

4.1.1. Method

We represent a shape with fine geometric details using two SIREN networks of different frequencies in the activation functions. The SIREN with lower frequency describes a smooth base surface; the SIREN with higher frequency adds microstructure to the base iso-surface by producing an implicit displacement field along the base's normal direction (see Fig. 4.2).

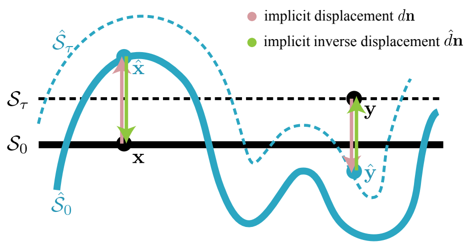


Figure 4.3.: An implicit displacement field for a 1D-curve. The displacement is defined not only on the zero-isosurface S_0 but also on arbitrary isosurfaces S_τ

In this section, we first formally define implicit displacement field by generalizing the classic explicit and discrete displacement mapping in Sec. 4.1.1, then in Sec. 4.1.1 we introduce the network architectures and training strategies that are tailored to this definition, finaly in Sec. 4.1.1 we extend the implicit displacement to address transferability.

Implicit displacement fields

In classic displacement mapping as shown in Fig. 4.1, high-frequency geometric details are obtained on a smooth base surface by taking samples from the base surface and offsetting them along their normal directions by a distance obtained (with interpolation) from a discrete height map. Two elements in this setting impede a direct adaptation for implicit shape representation: 1. the displacement mapping is defined discretely and only on the base surface, whereas implicit surface functions are typically defined continuously on the $\mathbb{R}^3$ domain; 2. the base surface is known and fixed, whereas our goal is to learn the base surface and the displacement jointly on-the-fly.

Addressing the above challenges, we first define *implicit displacement fields* (IDF), which are continuous analog to height maps that extend displacement mapping to the $\mathbb{R}^3$ domain.

Definition 1. *Given two signed distance functions f and $\hat{f}$ and their respective iso-surfaces at a given value $\tau \in \mathbb{R}$, $\mathcal{S}_\tau = \left\{\mathbf{x} \in \mathbb{R}^3 | f(\mathbf{x}) = \tau\right\}$ and $\hat{\mathcal{S}}_\tau = \left\{\mathbf{x} \in \mathbb{R}^3 | \hat{f}(\mathbf{x}) = \tau\right\}$, an* implicit displacement field *d: $\mathbb{R}^3 \to \mathbb{R}$ defines the deformation from $\mathcal{S}_\tau$ to $\hat{\mathcal{S}}_\tau$ such that*

$$f\left(\mathbf{x}\right) = \hat{f}\left(\mathbf{x} + d\left(\mathbf{x}\right)\mathbf{n}\right), \text{ where } \mathbf{n} = \frac{\nabla f\left(\mathbf{x}\right)}{\|\nabla f\left(\mathbf{x}\right)\|}. \tag{4.1}$$

This definition is schematically illustrated in Fig. 4.3, where the iso-surface $\mathcal{S}_0$ and $\mathcal{S}_\tau$ are mapped to $\hat{\mathcal{S}}_0$ and $\hat{\mathcal{S}}_\tau$ with the same implicit displacement field d. Notably, the height map in classic displacement mapping is a discrete sampling of IDF for the limited case $\tau = 0$.

In the context of surface decomposition, our goal is to estimate the base surface f and the displacement d given an explicitly represented detailed surface $\hat{\mathcal{S}}_0$. Following Eq. (1), we can do so by minimizing the difference between the base and the ground truth signed distance at query points $\mathbf{x} \in \mathbb{R}^3$ and their displaced position $\hat{\mathbf{x}} = \mathbf{x} + d\left(\mathbf{x}\right)\mathbf{n}$, *i.e.*, $\min \left|f\left(\mathbf{x}\right) - \hat{f}_{\mathrm{GT}}\left(\hat{\mathbf{x}}\right)\right|$.

However, this solution requires evaluating $\hat{f}_{\mathrm{GT}}\left(\hat{\mathbf{x}}\right)$ dynamically at variable positions $\hat{\mathbf{x}}$, which is a costly operation as the detailed shapes are typically given in explicit form, *e.g.*, as point clouds or meshes. Hence, we consider the inverse implicit displacement field $\hat{d}$, which defines a mapping from $\hat{\mathcal{S}}_\tau$ to $\mathcal{S}_\tau$, $f\left(\hat{\mathbf{x}} + \hat{d}\left(\hat{\mathbf{x}}\right)\mathbf{n}\right) = \hat{f}\left(\hat{\mathbf{x}}\right)$, as depicted in Fig. 4.3.

Assuming the displacement distance is small, we can approximate $\mathbf{n}$, the normal after inverse displacement, with that before the inverse displacement, *i.e.*

$$f\left(\hat{\mathbf{x}}+\hat{d}\left(\hat{\mathbf{x}}\right)\hat{\mathbf{n}}\right)=\hat{f}\left(\hat{\mathbf{x}}\right),\text{where }\hat{\mathbf{n}}=\frac{\nabla f\left(\hat{\mathbf{x}}\right)}{\left\|\nabla f\left(\hat{\mathbf{x}}\right)\right\|}. \tag{4.2}$$

This is justified by the following theorem and corollary, which we prove in the Appendix A.1.

Theorem 1. *If function $f:\mathbb{R}^n\to\mathbb{R}$ is differentiable, Lipschitz-continuous with constant L and Lipschitz-smooth with constant M, then $\left\|\nabla f\left(\mathbf{x}+\delta\,\nabla f\left(\mathbf{x}\right)\right)-\nabla f\left(\mathbf{x}\right)\right\|\leq|\delta|LM$.*

Corollary 1. *If a signed distance function f satisfying the eikonal equation up to error $\epsilon>0$, $|\|\nabla f\|-1|<\epsilon$, is Lipschitz-smooth with constant M, then $\left\|\nabla f\left(\mathbf{x}+\delta\,\nabla f\left(\mathbf{x}\right)\right)-\nabla f\left(\mathbf{x}\right)\right\|<(1+\epsilon)|\delta|M$.*

Given $\mathbf{n}=\frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}$, $\hat{\mathbf{n}}=\frac{\nabla f(\hat{\mathbf{x}})}{\|\nabla f(\hat{\mathbf{x}})\|}$, and $\hat{\mathbf{x}}=\mathbf{x}+d\left(\mathbf{x}\right)\mathbf{n}$, let $\delta=\frac{d(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}$, we can show $\|\hat{\mathbf{n}}-\mathbf{n}\|\leq\dfrac{1+\epsilon}{1-\epsilon}\left|\delta\right|M$ (*c.f.* Appendix A.1). In other words, the difference of $\hat{\mathbf{n}}$ and $\mathbf{n}$ is bounded by a small constant. Thus we obtain the approximation in Eq. (4.2), which allows us to presample training samples $\{\hat{\mathbf{x}}\}$ and use precomputed $\hat{f}\left(\hat{\mathbf{x}}\right)$ or its derivatives (see Sec 4.1.1) for supervision.

Network design and training

The formulation of (inverse) implicit field in the previous section is based on three assumptions: (i) f is smooth, (ii) d is small, (iii) f satisfies the eikonal constraint up to an error bound. In this section, we describe our network architecture and training technique, with emphasis on meeting these requirements.

Network architecture. We propose to model f and $\hat{d}$ with two SIRENs denoted as $\mathcal{N}^{\omega_B}$ and $\mathcal{N}^{\omega_D}$, where ω_B and ω_D refer to the frequency hyperparameter in the sine activation functions $\mathbf{x}\mapsto\sin\left(\omega\mathbf{x}\right)$. Evidently, as shown in Fig. 4.4, ω dictates an upper bound on the frequencies the network is capable of representing, thereby it also determines the network's inductive bias for smoothness. Correspondingly, we enforce the smoothness of f and detail-representing capacity of $\hat{d}$ using a smaller ω_B and a larger ω_D. Empirically, we find that $\omega_B=15$ and $\omega_D=60$ fits our needs in most cases. Moreover, we add a scaled tanh activation to the last linear layer of $\mathcal{N}^{\omega_D}$, *i.e.* $\alpha\tanh\left(\cdot\right)$, which ensures that the displacement distance is smaller than the constant α.

Networks containing high-frequency signals, *e.g.* $\mathcal{N}^{\omega_D}$, require large amounts of accurate ground truth data for supervision to avoid running into optimization local minima [Par+20]. Consequently, when dense and accurate

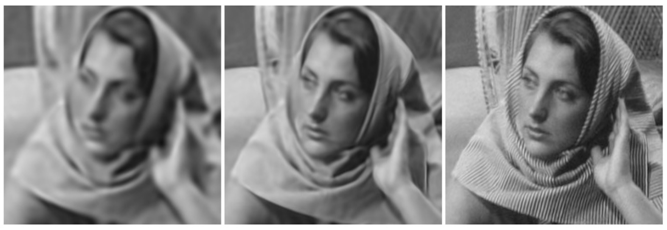


Figure 4.4.: Smoothness control via SIREN's frequency hyperparameter ω. Overfitting SIREN to the first image with $\omega = 30$ (middle) and $\omega = 60$ (right) shows that smaller ω leads to a smoother result.

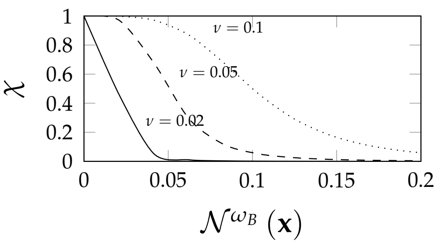


Figure 4.5.: Attenuation as a function of base SDF.

ground truth SDF values are not available, high-frequency signals often create artifacts. This is often the case in void regions when learning from point clouds, as only implicit regularization and fuzzy supervision is applied (see the first and last terms of Eq. (4.4)). Hence, we apply an attenuation function $\chi\left(\mathcal{N}^{\omega_B}\right) = \frac{1}{1+\left(\mathcal{N}^{\omega_B}(\mathbf{x})/\nu\right)^4}$ to subdue $\mathcal{N}^{\omega_D}$ far from the base surface, where ν determines the speed of attenuation as depicted in Fig. 4.5.

Combining the aforementioned components, we can compute the signed distance of the detailed shape at query point $\mathbf{x}$ in two steps:

$$f(\mathbf{x}) = \mathcal{N}^{\omega_B}(\mathbf{x}), \quad \hat{f}(\mathbf{x}) = \mathcal{N}^{\omega_B}\left(\mathbf{x} + \chi\left(f(\mathbf{x})\right)\mathcal{N}^{\omega_D}(\mathbf{x})\frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}\right). \tag{4.3}$$

Training. We adopt the loss from SIREN, which is constructed to learn SDFs directly from oriented point clouds by solving the eikonal equation with boundary constraint at the on-surface points. Denoting the input domain as Ω (by default set to $[-1,1]^3$) and the ground truth point cloud as $\mathcal{P} = \{(\mathbf{p}_i, \mathbf{n}_i)\}$, the loss computed as in Eq. (4.4), where $\lambda_{\{0,1,2,3\}}$ denote loss weights:

$$\begin{aligned}\mathcal{L}_{\hat{f}} = \textstyle\sum_{\mathbf{x}\in\Omega} \lambda_0 \left| \|\nabla \hat{f}(\mathbf{x})\| - 1 \right| &+ \sum_{(\mathbf{p},\mathbf{n})\in\mathcal{P}} \left(\lambda_1 |\hat{f}(\mathbf{p})| + \lambda_2 \left(1 - \left\langle \nabla \hat{f}(\mathbf{p}), \mathbf{n} \right\rangle\right)\right) \\ &+ \sum_{\mathbf{x}\in\Omega\setminus\mathcal{P}} \lambda_3 \exp\left(-100 \hat{f}(\mathbf{x})\right). \end{aligned} \tag{4.4}$$

As the displacement and the attenuation functions depend on the base network, it is beneficial to have a well-behaving base network when training the displacement (see Sec. 4.1.2). Therefore, we adopt a progressive learning scheme, which first trains $\mathcal{N}^{\omega_B}$, and then gradually increase the impact of $\mathcal{N}^{\omega_D}$. Notably, similar frequency-based coarse-to-fine training techniques are shown to improve the optimization result in recent works [Par+20; Her+21].

We implement the progressive training via symmetrically diminishing/increasing learning rates and loss weights for the base/displacement networks. For brevity, we describe the procedure for loss weights only, and

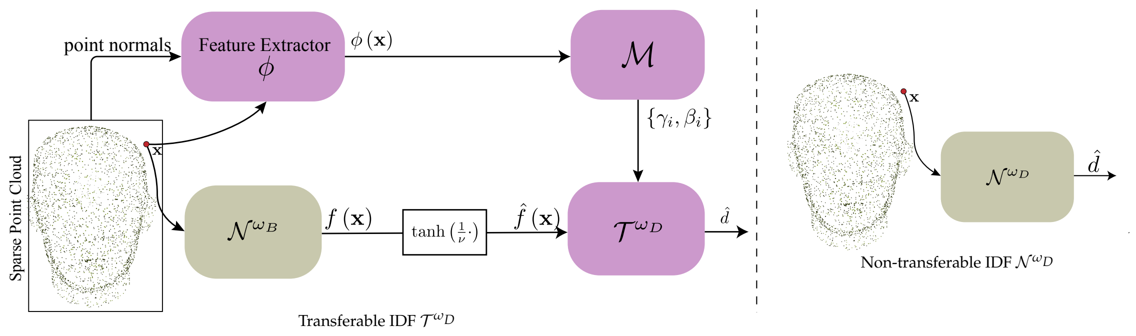


Figure 4.6.: Illustrations for transferable and non-transferable implicit fields. The transferable modules are in pink and the shape-specific modules are in yellow. Instead of consuming the euclidean coordinates, the transferable displacement network takes a scale-and-translation invariant feature as inputs, which describes the relative position of the query point to the base shape.

we apply the same to the learning rates in our implementation. First, we train $\mathcal{N}^{\omega_B}$ by substituting $\hat{f}$ in the loss Eq. (4.4) with f, resulting a base-only loss denoted $\mathcal{L}_f$. Then, starting from a training percentile $T_m \in [0,1]$, we combine $\mathcal{L}_f$ and $\mathcal{L}_{\hat{f}}$ via $\kappa\,\mathcal{L}_f + (1-\kappa)\,\mathcal{L}_{\hat{f}}$ with $\kappa = \frac{1}{2}\left(1+\cos\left(\pi\frac{(t-T_m)}{(1-T_m)}\right)\right)$, where $t \in [T_m,\ 1]$ denotes the current training progress.

Transferable implicit displacement field.

In classic displacement mapping, the displacement is queried by the UV-coordinates from surface parameterization, which makes the displacement independent of deformations of the base surface. We can achieve similar effect *without parameterization* by learning *query features*, which emulate the UV-coordinates to describe the location of the 3D query points w.r.t. the base surface.

We construct the query features using two pieces of information: (i) a global context descriptor, $\phi(\mathbf{x})$, describing the location of the query point in relation to the base surface in a semantically meaningful way, (ii) the base signed distance value $f(\mathbf{x})$, which gives more precise relative location with respect to the base surface. Since both are differentiable w.r.t. the euclidean coordinates of the query point, we can still train $\mathcal{N}^{\omega_D}$ using derivatives as in Eq. (4.4).

Our global context descriptor is inspired by Convolutional Occupancy Networks [Pen+20]. Specifically, we project the sparse on-surface point features obtained using a conventional point cloud encoder onto a regular 3D (or 2D, *c.f.* Sec. 4.1.2) grid, then use a convolutional module to propagate sparse on-surface point features to the off-surface area, finally obtain the query feature using bilinear interpolation. We use normals instead of point positions as the input to the point cloud encoder, making the features scale-invariant and translation-invariant. Note that ideally the features should also be rotation-invariant. Nevertheless, as we empirically show later, normal features can in fact generalize under small local rotational deformations, which is sufficient for transferring displacements between two roughly aligned shapes. We leave further explorations in this direction for future work.

$\mathcal{N}^{\omega_D}$ is tasked to predict the displacement conditioning on $\phi(\mathbf{x})$ and $f(\mathbf{x})$. However, empirical studies [Cha+20b] suggest that SIREN does not handle high-dimensional inputs well. Hence, we adopt the FiLM conditioning [Per+18; Dum+18] as suggested by Chan et al. [Cha+20b], which feeds the conditioning latent vector as an affine transformation to the features of each layer. Specifically, a mapping network $\mathcal{M}$ converts $\phi(\mathbf{x})$ to a set of C-dimensional frequency modulators and phase shifters $\{\gamma_i, \boldsymbol{\beta}_i\}$, which transform the i-th linear layer to $\left(\mathbf{1} + \frac{1}{2}\gamma_i\right) \circ (\mathbf{W}_i \mathbf{x} + \mathbf{b}_i) + \boldsymbol{\beta}_i$, where $\mathbf{W}_i$ and $\mathbf{b}_i$ are the parameters in the linear layer and $\circ$ denotes element-wise multiplication. Finally, since SIREN assumes inputs in range $(-1, 1)$, we scale f using $\bar{f}(\mathbf{x}) = \tanh\left(\frac{1}{\nu} f(\mathbf{x})\right)$ to capture the variation close to the surface area, where ν is the attenuation parameter described in Sec. 4.1.1.

Fig. 4.6 summarizes the difference between transferable and non-transferable displacement fields. Formally, the signed distance function of the detailed shape in Eq. (4.3) can be rewritten as

$$\hat{f}(\mathbf{x}) = \mathcal{N}^{\omega_B}\left(\mathbf{x} + \chi(f(\mathbf{x}))\, \mathcal{T}^{\omega_D}\left(\bar{f}(\mathbf{x}),\, \mathcal{M}(\phi(\mathbf{x}))\right) \frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}\right). \tag{4.5}$$

4.1.2. Results

We now present the results of our method. In Sec. 4.1.2, we evaluate our networks in terms of geometric detail representation by comparing with state-of-the-art methods on the single shape fitting task. We then evaluate various design components in an ablation study in Sec. 4.1.2. Finally, we validate the transferability of the displacement fields in a detail transfer task in Sec. 4.1.2.

Implementation details. By default, both the base and the displacement nets have 4 hidden layers with 256 channels each. The maximal displacement α, attenuation factors ν, and the switching training percentile is set to T_m are set to 0.05, 0.02 and 0.2 respectively; The loss weights $\lambda_{\{0,1,2,3\}}$ in Eq. (4.4) are set to 5, 400, 40 and 50. We train our models for 120 epochs using ADAM optimizer with initial learning rate of 0.0001 and decay to 0.00001 using cosine annealing [LH16] after finishing 80% of the training epochs. We presample

4 million surface points per mesh for supervision. Each training iteration uses 4096 subsampled surface points and 4096 off-surface points uniformly sampled from the $[-1, 1]^3$ bounding box. To improve the convergence rate, we initialize SIREN models by pre-training the base model $\mathcal{N}^{\omega_B}$ (and baseline SIREN, *c.f.* Sec. 4.1.2) to a sphere with radius 0.5. This initialization is optional for our training but is critical for baseline SIRENs.

Data. We test our method using 16 high-resolution shapes, including 14 from Sketchfab [Skeb] and 2 from Stanford 3DScanRepo [Sta]. Our transferable displacement model is tested using shapes provided by Berkiten et al. [Ber+17], Yang et al. [Yan+20], and Zhou and Jacobson [ZJ16].

Detail representation.

Chamfer distance points to point distance $\cdot 10^{-3}$ / normal cosine distance $\cdot 10^{-2}$

model	Progressive FFN	NGLOD (LOD4)	NGLOD (LOD6)	SIREN-3 $\omega = 60$	SIREN-7 $\omega = 30$	SIREN-7 $\omega = 60$	Direct Residual	D-SDF	Ours
angel	6.00/4.19	2.28/2.87	1.47/1.43	9.54/5.47	5.57/2.85	-/-	251/87.9	3.36/3.70	**1.30**/**0.89**
asian dragon	4.96/4.02	1.66/4.05	1.03/1.91	6.13/5.28	3.65/2.36	7.24/4.03	269/92.7	2.84/1.80	**0.93**/**1.36**
camera	4.62/1.51	1.56/1.15	1.32/0.62	6.50/2.38	4.11/1.10	-/-	281/38.5	1.99/2.01	**1.25**/**0.34**
compressor	5.55/1.35	1.64/0.88	1.52/0.44	8.83/2.82	4.63/0.82	-/-	330/56.9	2.66/3.71	**1.39**/**0.23**
dragon	5.10/4.00	1.80/3.77	1.39/2.37	7.04/4.75	3.80/2.49	-/-	263/76.4	2.68/**1.21**	**1.24**/1.50
dragon warrior	5.94/7.25	2.46/8.12	1.52/5.21	7.27/8.45	3.68/4.83	5.96/8.01	6.09/9.77	2.22/**4.46**	**1.47**/4.56
dragon wing	5.68/5.41	2.01/4.98	1.47/2.87	6.40/5.46	7.92/3.84	-/-	167/76.7	2.66/4.09	**1.31**/**1.49**
dragon china	6.20/2.31	2.32/1.75	**1.39**/1.01	9.15/4.35	6.20/2.29	-/-	272/69.7	3.31/7.06	1.40/**0.51**
dragon cup	4.39/3.13	1.83/3.57	1.24/1.17	7.67/4.69	5.50/2.15	-/-	173/86.9	3.21/4.93	**1.10**/**0.51**
helmet	4.79/1.02	1.70/0.871	1.40/0.410	8.40/2.72	5.19/0.83	-/-	263/94.6	2.59/1.99	**1.29**/**0.13**
hunter	4.17/4.66	2.03/5.08	1.18/2.08	8.96/6.66	3.40/2.58	-/-	3.39/3.64	2.61/4.89	**0.91**/**1.14**
luyu	7.22/4.29	2.19/3.30	1.53/1.81	8.98/6.83	6.16/3.16	-/-	206/94.6	5.10/9.76	**1.28**/**1.02**
pearl dragon	7.48/5.97	2.37/6.10	1.49/2.67	10.1/9.56	5.05/4.07	-/-	66.1 /49.8	3.26/6.24	**1.30**/**1.43**
ramesses	4.24/2.30	1.47/2.47	0.97/1.93	6.40/3.77	4.20/2.33	-/-	3.78/6.60	3.16/9.66	**0.92**/**1.58**
Thai Statue	5.23/7.01	7.16/16.7	1.30/4.77	6.27/7.48	3.81/4.17	-/-	117/45.2	1.76/**2.46**	**1.07**/2.92
Vase Lion	5.92/1.93	1.86/2.18	1.39/0.77	39.9/25.6	4.68/1.02	-/-	227/54.8	2.26/2.31	**1.31**/**0.43**
AVG	5.47/3.77	2.27/4.24	1.35/1.97	9.85/6.64	4.85/2.56	-/6.02	181/59.0	2.85/4.39	**1.22**/**1.25**

Table 4.1.: Quantitative comparison. Among the benchmarking methods, only NGLOD at LOD-6, using 256× number of parameters compared to our model, can yield results close to ours. SIREN models with larger ω have convergence issues: despite our best efforts, the models still diverged in most cases.

The methods we compare with are 1. Fourier feature network [Tan+20] with SOFTPLUS activation and 8 frequency bands trained from coarse-to-fine, as suggested by Park et al. [Par+20]; a total of 8 hidden layers each of size 256 are used to match our model size; additionally we apply a skip-connection in the middle layer as proposed in DeepSDF Park et al. [Par+19]; 2. NGLOD [Tak+21] trained using 4 and 6 levels of detail (LODs) corresponding to 64^3 and 256^3 spatial resolution respectively, with LOD4 comparable with our model in terms of the number of parameters, 3. baseline SIREN, for which we trained three different variations in hope of overcoming training divergence issues; 4. direct residual, where we compose the

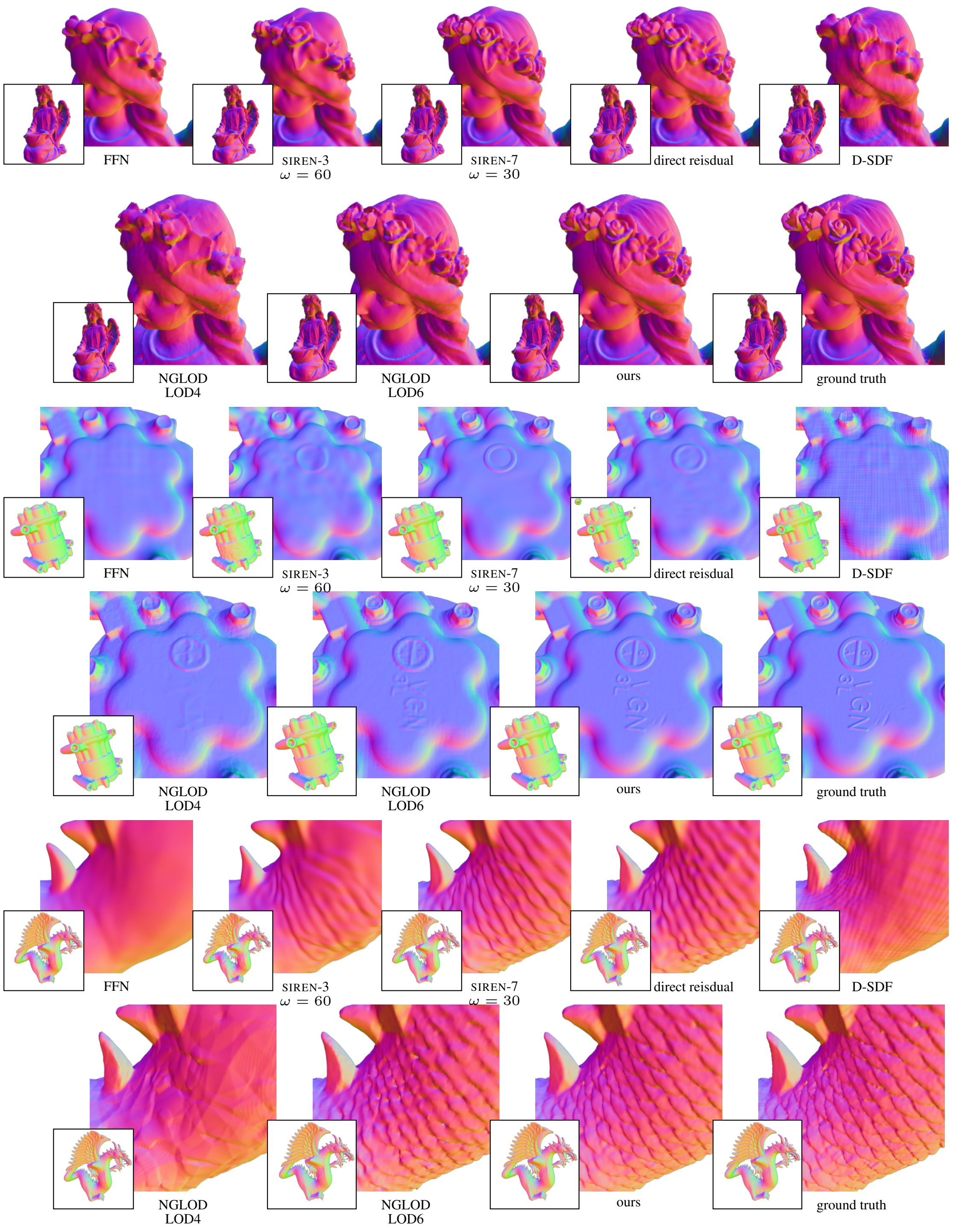


Figure 4.7.: Comparison of detail reconstruction (better viewed with zoom-in).

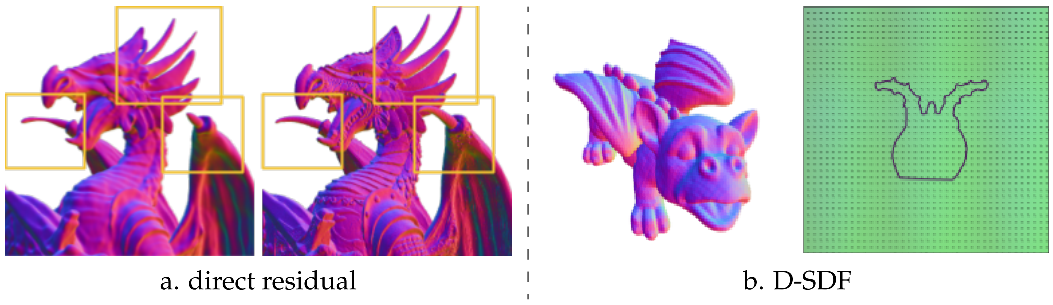


Figure 4.8.: Examples of the direct residual and D-SDF models. Direct residual doesn't enforce displacement directions and produces large structural artifacts; D-SDF fails to learn meaningful displacement due to the lack of constraints.

signed distance value simply as the sum of base and displacement nets, *i.e.* $\hat{f}(\mathbf{x}) = \mathcal{N}^{\omega_B}(\mathbf{x}) + \mathcal{N}^{\omega_D}(\mathbf{x})$; 5. D-SDF (inspired by [Pum+21; Par+20]), in which the displacement is a vector in arbitrary direction, *i.e.* $\hat{f} = f(\mathbf{x} + \Delta)$, where $\Delta \in \mathbb{R}^3$ is predicted in the second network. We follow network specs of D-Nerf [Pum+21], which contains 2 8-layer MLP networks with ReLU activation and positional encodings.

Among these, NGLOD, direct residual and D-SDF requires ground truth SDF for supervision, the rest are trained using our training loss. Two-way point-to-point distance and normal cosine distance are computed as the evaluation metrics on 5 million points randomly sampled from meshes extracted using marching cubes with 512^3 resolution. As shown in Table 4.1 and Fig. 4.7, our method outperforms the baseline methods with much higher reconstruction fidelity. NGLOD with 6 LODs is the only method onpar with ours in terms of detail representation, however it requires storing more than 300 times as many as parameters as our model. SIREN networks with larger ω have severe convergence issues even with sphere initialization (*c.f.* Implementation Details) and gradient clipping. Direct residual doesn't enforce displacement directions and produces large structural artifacts (Fig. 4.8a). D-SDF yields qualitatively poor results, as the displacement net is unable to learn meaningful information shown in Fig. 4.8b due to the lack of constraints.

Ablation study

We study the contributions of different design components described in Sec. 4.1.1, including the displacement scaling α tanh, the attenuation function χ and the progressive training Eq. (4.4), as well as the effect of the scaling factor α and attenuation speed ν.

First, as Table 4.2 shows, all the test modes converge within comparable range, even for the model with the least constraints. This shows that our

	α test (with $\nu = 0.02$)					ν test (with $\alpha = 0.05$)				
	0.01	0.02	0.05	0.1	0.2	0.01	0.02	0.05	0.1	0.2
point-to-point distance $(\cdot 10^{-3})$	1.178	1.171	1.147	1.146	1.149	1.146	1.147	1.147	1.149	1.152
normal cosine distance $(\cdot 10^{-2})$	1.525	1.490	1.252	1.251	1.260	1.254	1.253	1.251	1.250	1.274

Table 4.3.: Study of the hyperparameters α (left) and ν (left). The reconstruction accuracy remains stable and highly competetive throughout hyperparameters variation. The results are averaged over 8 test models.

model is robust against violations of theoretical assumptions specified in Sec. 4.1.1. At the same time, the performance rises with increasingly constrained architecture and progressive training, suggesting that the proposed mechanisms further boost training stability.

α tanh	χ	prog. training	average CD $\cdot 10^{-3}$
			1.44
✓			1.41
✓	✓		1.38
✓	✓	✓	1.24

Table 4.2.: Ablation study. Our model benefits from the proposed architectural and training designs, yet it is also robust against variations.

Table 4.3 shows that in a reasonable range of α and ν there is very little variance across different hyperparameter values, indicating again that the designed model is robust to violations of the assumptions. If α is too small (0.01, 0.02 in Table 4.3), the displacement distance is capped also in the small range, then the displacement may no longer be sufficient to correct the difference between the base and the ground truth surface. This explains the slight increase of chamfer distances in the table for $\alpha = \{0.01, 0.02\}$. Also, when ν is too large (0.2), i.e. the high frequency signal is not suppressed in the void region, the chamfer distances also increase due to off-surface high-frequency noise.

Noise tolerance

Chamfer distance points to point distance / normal cosine distance $\cdot 10^{-2}$			
training points	noise σ	ours	poisson reconstruction
40000	0.002	1.07/7.54	1.08/7.78
40000	0.005	1.05/7.57	1.08/7.82
400000	0.002	1.00/6.01	1.04/6.63
400000	0.005	1.00/5.99	1.04/6.60

Table 4.4.: Quantitative evaluation given sparse and noisy inputs.

While we used dense and clean sampled point clouds as inputs in the paper, as our focus is on detail representation, we examine the behavior of our method under noisy and sparse inputs. Specifically, we train our network with 400 thousand and 40 thousand sampled points (10% and 1% of the amount in our main experiment, respectively), and added $\sigma = 0.002$ and $\sigma = 0.005$ Gaussian noise on both the point normals and the point positions. From the

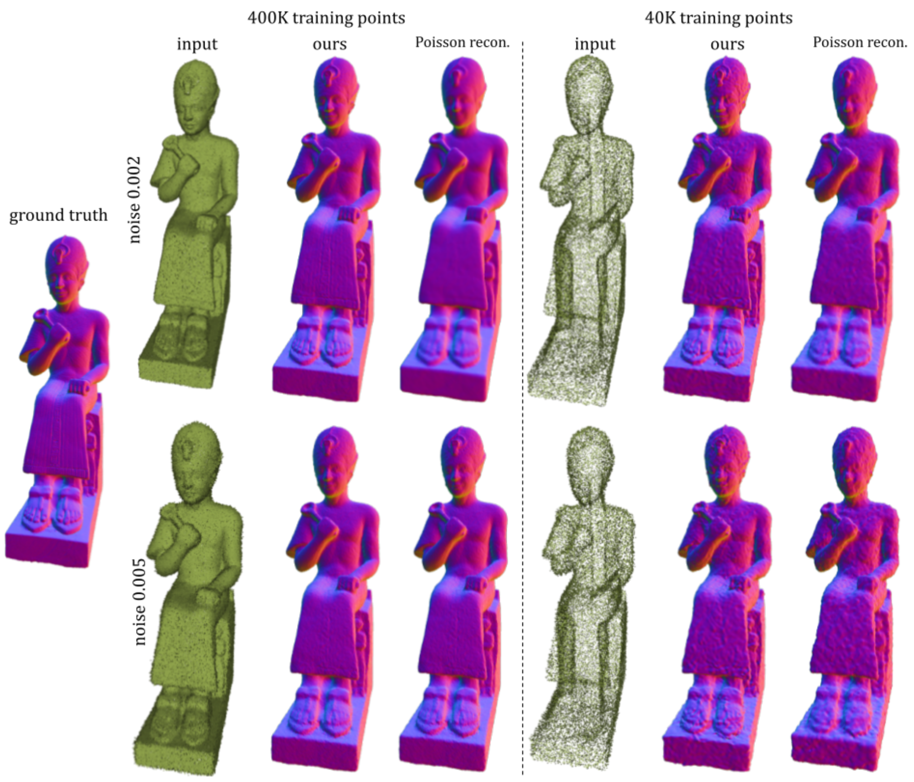


Figure 4.9.: Qualitative evaluation given sparse and noisy inputs.

qualitative and quantitative results shown in Fig. 4.9 and Table 4.4, we can see that our method recovers geometric details better than Poisson reconstruction given sufficient training data (c.f. the left half of the figure). When the training sample is sparse, our method tends to generate more high-frequency noise as a result of overfitting.

Transferability

We apply our method to detail transfer in order to validate the transferability of IDF. Specifically, we want to transfer the displacements learned for a source shape to an aligned but different target shape. In the first test scenario, the base shape is provided and lies closely to the ground truth detailed surface. In the second scenario, we are only provided with the detailed shapes and thus need to estimate the base and the displacements jointly. The pipeline consists of the following steps: 1) training $\mathcal{N}^{\omega_B}$ by fitting the source shape (or the source base shape if provided), 2) training $\mathcal{T}^{\omega_D}$, $\mathcal{M}$ and the query feature extractor ϕ jointly by fitting the source shape using Eq. (4.5) while keeping $\mathcal{N}^{\omega_B}$ fixed, 3) training $\mathcal{N}_{\text{new}}^{\omega_B}$ by fitting the target shape (or the target base shape if provided), 4) evaluating Eq. (4.5) by replacing $\mathcal{N}^{\omega_B}$ with $\mathcal{N}_{\text{new}}^{\omega_B}$. To

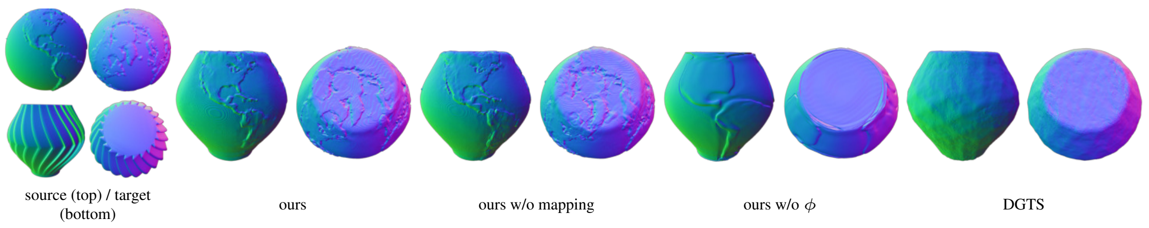
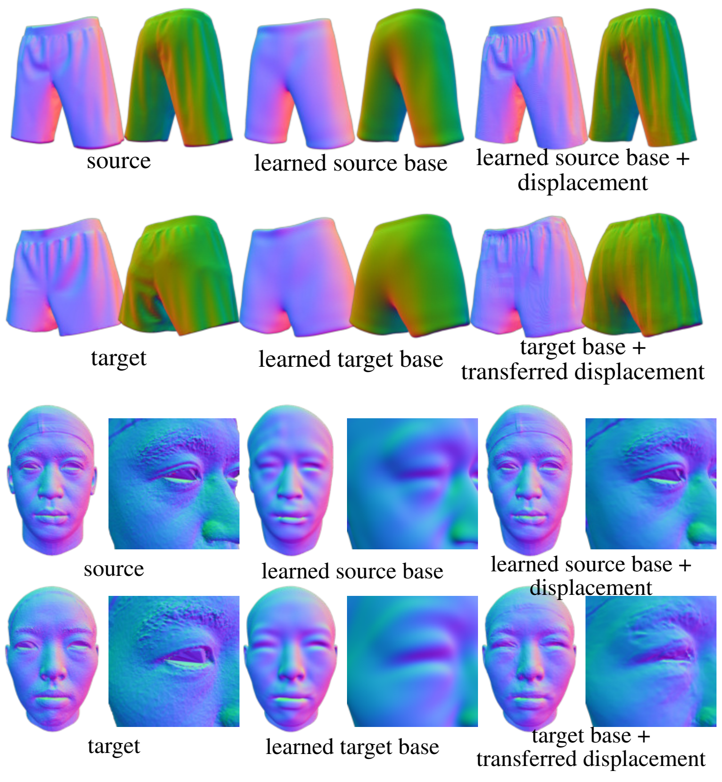


Figure 4.11.: Transferring spatially-variant geometric details using various methods. Small to severe distortions are introduced when removing different components of the proposed transferable IDF. Thanks to the combination of global/local query feature, our method transfers spatially-variant details while [Her+20a] only focuses on spatially-invariant isometric details.

prevent the base network from learning high-frequency details when the base is unknown, we use $\omega_B = 5$ and three 96-channel hidden layers for $\mathcal{N}^{\omega_B}$.



source
learned source base
learned source base + displacement
target
learned target base
target base + transferred displacement
source
learned source base
learned source base + displacement
target
learned target base
target base + transferred displacement

Figure 4.10.: Transferable IDF applied to detail transfer. Upper: the base shape is provided and lies closely to the ground truth detailed surface; Lower: only the detailed shapes are provided, thus the base and the displacements need to estimated jointly.

Example outputs for both scenarios are shown in Figure 4.10; the base shapes are provided for the *shorts* model. We use a 32^3 and a 128^2 grid (for the frontal view), for the *shorts* and *face* model respectively in ϕ to extract the query features. Our displacement fields, learned solely from the source shape, generate plausible details on the unseen target shape. The transferred details contain high-frequency signals (*e.g.* the eyebrows on the face), which is challenging for explicit representations. However, for the second scenario the performance degenerates slightly since the displacement field has to compensate errors stemming specifically from the base SDF.

In additional, we evaluate the design of the transferable IDF model by removing the mapping net and the convolutional context descriptor ϕ.

For the former case, we drop the FiLM conditioning and simply use concatenation of $\phi(\mathbf{x})$ and $\bar{f}(\mathbf{x})$ as the inputs to $\mathcal{N}^{\omega_D}$; for the latter we directly use the normal at the sampled position as the context descriptor, *i.e.* $\phi(\mathbf{x}) = \nabla f(\mathbf{x})$. As Figure 4.11 shows, the removal of mapping net and ϕ lead to different degrees of feature distortions. We also compare with the DGTS [Her+20a], which fails completely at this example since it only consumes local intrinsic features. Furthermore, the effect of scaling $\bar{f}$ is shown in Fig. 4.12, where using unscaled f as input to $\mathcal{T}^{\omega_D}$ leads to artifacts at the boundary.

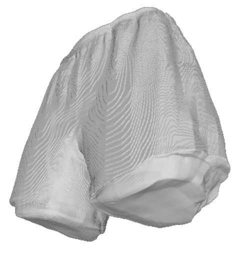


Figure 4.12.: Detail transfer without scaling $\bar{f}$.

Inference and training time

Training the models as described in the paper takes 2412 seconds (40 minutes), which amounts to 120 epochs, i.e., around 20 seconds per epoch, where each epoch comprises 4 million surface samples and 4 million off-surface samples. In comparison, the original implementation of NGLOD6 takes 110 minutes to train 250 epochs, where each epoch comprises 200000 surface samples and 300000 off-surface samples. As for inference, using the same evaluation setup, NGLOD6 takes 193.9s for 512^3 points, while our inference takes 250.06 seconds for 512^3 query points. All benchmarking is performed on a single Nvidia 2080 RTX GPU. These timings could be improved by optimizing the model for performance, which we did not. For example, instead of using autodiff for computing the base surface normals, one could exploit the fact that the differentiation of SIREN is also a SIREN and explicitly construct a computation graph for computing the base surface normals.

4.2. Optimizing neural implicit surfaces.

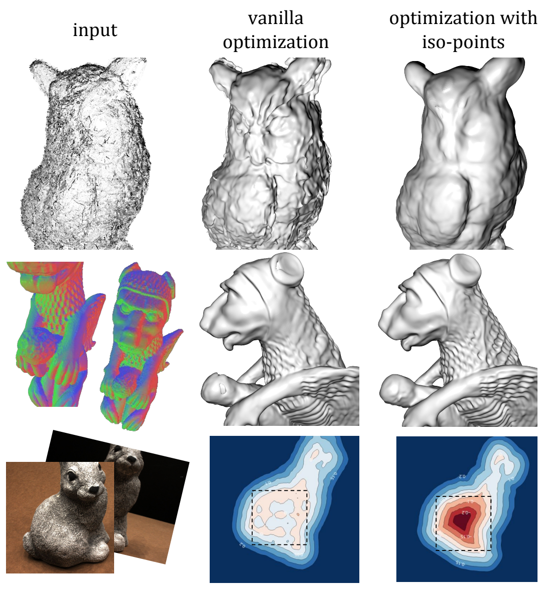


Figure 4.13.: The iso-points allow us to augment optimization pipelines in a variety of ways: geometry-aware regularizers are incorporated to reconstruct a surface from a noisy point cloud (first row); geometric details are preserved in multi-view reconstruction via feature-aware sampling (second row); iso-points can serve as a 3D prior to improve the topological accuracy of the reconstructed surface (third row).

Fitting neural implicit surfaces to various input observations has many practical applications, such as surface reconstruction from point clouds and multi-view reconstruction from images. For most cases, the observations are noisy and incomplete. This leads to fundamental geometric and topological problems in the final reconstructed surface, as the network overfits to the imperfect data. We observe that this problem remains, and can become more prominent with the recent powerful architectures, *e.g.* sine activations [SZW19] and Fourier features [PJH16].

We show examples of problems in fitting neural implicit functions in Fig. 4.13. When fitting a neural surface to a noisy point cloud, “droplets" and bumps emerge where there are outlier points and holes (first row); when fitting a surface to image observations, fine-grained geometric features are not captured due to under-sampling (second row); topological noise is introduced when inadequate views are available for reconstruction (third row).

In this work, we propose to alleviate these problems by introducing a *hybrid neural surface representation* using *iso-points*. The technique converts from an implicit neural surface to an explicit one via sampling iso-points, and goes back to the implicit representation via optimization. The two-way conversion is performed on-the-fly during training to introduce geometry-aware regularization and optimization. This approach unlocks a large set of fundamental tools from geometry processing to be incorporated for accurate and robust fitting of neural surfaces.

A key challenge is to extract the iso-points on-the-fly efficiently and flexibly during the training of a neural surface. Extending several techniques from point-based geometry processing, we propose a multi-stage strategy, consisting of projection, resampling, and upsampling. We first obtain a sparse point cloud on the implicit surface via projection, then resample the iso-points to fix severely under-sampled regions, and finally upsample to obtain a dense point cloud that covers the surface. As all operations are GPU friendly and the resampling and upsampling steps require only local point distributions, the entire procedure is fast and practical for training.

We illustrate the utility of the new representation with a variety of applications, such as multi-view reconstruction and surface reconstruction from noisy point clouds. Quantitative and qualitative evaluations show that our approach allows for fast convergence, robust optimization, and accurate reconstruction of details and topology.

4.2.1. Method

Given a neural implicit function $f_t(\mathbf{p};\theta_t)$ representing the surface $\mathcal{S}_t$, where θ_t are the network parameters at the t-th training iteration, our goal is to efficiently generate and utilize a dense and uniformly distributed point set on the zero level set, called *iso-points*, which faithfully represents the current implicit surface $\mathcal{S}_t$. Intuitively, we can deploy the iso-points back into the ongoing optimization to serve various purposes, *e.g.* improving the sampling of training data and providing regularization for the optimization, leading to a substantial improvement in the convergence rate and the final optimization quality.

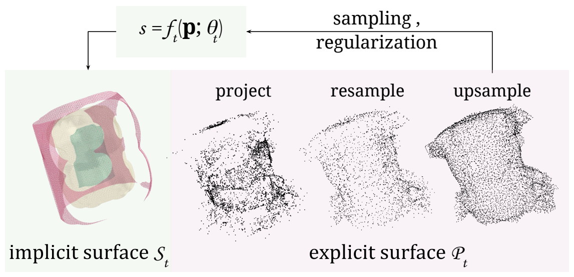

$s = f_t(\mathbf{p};\theta_t)$
sampling , regularization
project
resample
upsample
implicit surface $\mathcal{S}_t$
explicit surface $\mathcal{P}_t$

Figure 4.14.: Overview of our hybrid representation. We extract a dense, uniformly distributed set of *iso-points* as an explicit representation for a neural implicit surface. Since the extraction is fast, iso-points can be integrated back into the optimization as a 3D geometric prior, enhancing the optimization.

In this section, we first focus on how to extract the iso-points via projection and uniform resampling. We then explain how to utilize the iso-points for better optimization in practical scenarios.

Iso-surface sampling

As shown in Fig. 4.14, our iso-surface sampling consists of three stages. First, we project a set of initial points $\mathcal{Q}_t$ onto the zero level set to get a set of *base iso-points* $\tilde{\mathcal{Q}}_t$. We then resample $\tilde{\mathcal{Q}}_t$ to avoid clusters of points and fill large holes. Finally, we upsample the points to obtain dense and uniformly distributed iso-points $\mathcal{P}_t$.

Projection. Projecting a point onto the iso-surface can be seen as using Newton's method [BI66] to approximate the root of the function starting from a given point. Atzmon *et al.* [Atz+19] derive the projection formula for functions modeled by generic networks. For completeness, we recap the steps here, focusing on $f : \mathbb{R}^3 \to \mathbb{R}$.

Given an implicit function $f(\mathbf{p})$ representing a signed distance field and an initial point $\mathbf{q}_0 \in \mathbb{R}^3$, we can find an iso-point $\mathbf{p}$ on the zero level set of f using Newton iterations: $\mathbf{q}_{k+1} = \mathbf{q}_k - J_f(\mathbf{q}_k)^+ f(\mathbf{q}_k)$, where J_f^+ is the Moore-Penrose pseudoinverse of the Jacobian. In our case, J_f is a row 3-vector, so that $J_f^+(\mathbf{q}_k) = \frac{J_f^\top(\mathbf{q}_k)}{\|J_f(\mathbf{q}_k)\|^2}$, where the Jacobian $J_f(\mathbf{q}_k)$ can be conveniently evaluated in the network via backpropagation.

However, for some contemporary network designs, such as sine activation functions [Sit+20] and positional encoding [Mil+20], the signed distance field can be very noisy and the gradient highly non-smooth. Directly applying Newton's method then causes overshooting and oscillation. While one could attempt more sophisticated line search algorithms, we instead address this issue with a simple clipping operation to bound the length of the update, *i.e.*

$$\mathbf{q}_{k+1} = \mathbf{q}_k - \tau \left(\frac{J_f^\top(\mathbf{q}_k)}{\|J_f(\mathbf{q}_k)\|^2} f(\mathbf{q}_k) \right), \tag{4.6}$$

where $\tau(\mathbf{v}) = \frac{\mathbf{v}}{\|\mathbf{v}\|} \min(\|\mathbf{v}\|, \tau_0)$. We set $\tau_0 = \frac{D}{2|\mathcal{Q}_t|}$ with D denoting the diagonal length of the shape's bounding box.

In practice, we initialize $\mathcal{Q}_t$ with randomly sampled points at the beginning of the training and then with iso-points $\mathcal{P}_{t-1}$ from the previous training iteration. Similar to [Atz+19], at each training iteration, we perform a maximum of 10 Newton iterations and terminate as soon as all points have converged, *i.e.* $|f(\mathbf{q}_k)| < \epsilon, \forall \mathbf{q} \in \mathcal{Q}_t$. The termination threshold ϵ is set to 10^{-4} and gradually reduced to 10^{-5} during training.

Uniform resampling. The projected base iso-points $\tilde{\mathcal{Q}}_t$ can be sparse and hole-ridden due to the irregularity present in the neural distance field, as shown in Fig. 4.14. Such irregular sample distribution prohibits us from many downstream applications described later.

The resampling step aims at avoiding over- and undersampling by iteratively moving the base iso-points away from high-density regions, *i.e.*

$$\tilde{\mathbf{q}} \leftarrow \tilde{\mathbf{q}} - \alpha \mathbf{r}, \tag{4.7}$$

where $\alpha = \sqrt{D/|\tilde{\mathcal{Q}}_t|}$ is the step size. The update direction $\mathbf{r}$ is a weighted average of the normalized translations between $\tilde{\mathbf{q}}$ and its K-nearest points (we set $K = 8$):

$$\mathbf{r} = \sum_{\tilde{\mathbf{q}}_i \in \mathcal{N}(\tilde{\mathbf{q}})} w(\tilde{\mathbf{q}}_i, \tilde{\mathbf{q}}) \frac{\tilde{\mathbf{q}}_i - \tilde{\mathbf{q}}}{\|\tilde{\mathbf{q}}_i - \tilde{\mathbf{q}}\|}. \tag{4.8}$$

The weighting function is designed to gradually reduce the influence of faraway neighbors, specifically

$$w(\tilde{\mathbf{q}}_i, \tilde{\mathbf{q}}) = e^{-\frac{\|\tilde{\mathbf{q}}_i - \tilde{\mathbf{q}}\|^2}{\sigma_p}}, \tag{4.9}$$

where the density bandwidth σ_p is set to be $16D/|\tilde{\mathcal{Q}}_t|$.

Upsampling. Next, we upsample the point set to the desired density while further improving the point distribution. Our upsampling step is based on edge-aware resampling (EAR) proposed by Huang *et al.* [Hua+13b]. We explain the key steps and our main differences to EAR as follow.

First, we compute the normals as the normalized Jacobians and apply bilateral normal filtering, just as in EAR. Then, the points are pushed away from the edges to create a clear separation. We modify the original optimization formulation with a simpler update consisting of an attraction and a repulsion term. The former pulls points away from the edge and the latter prevents the points from clustering.

$$\Delta\mathbf{p}_{\text{attraction}} = \frac{\sum_{\mathbf{p}_i \in \mathcal{N}(\mathbf{p})} \phi(\mathbf{n}_i, \mathbf{p}_i - \mathbf{p})(\mathbf{p} - \mathbf{p}_i)}{\sum_{\mathbf{p}_i \in \mathcal{N}(\mathbf{p})} \phi(\mathbf{n}_i, \mathbf{p}_i - \mathbf{p})}, \tag{4.10}$$

$$\Delta\mathbf{p}_{\text{repulsion}} = 0.5 \frac{\sum_{\mathbf{p}_i \in \mathcal{N}(\mathbf{p})} w(\mathbf{p}_i, \mathbf{q})(\mathbf{p}_i - \mathbf{p})}{\sum_{\mathbf{p}_i \in \mathcal{N}(\mathbf{p})} w(\mathbf{p}_i, \mathbf{q})}, \tag{4.11}$$

$$\mathbf{p} \leftarrow \mathbf{p} - \tau(\Delta\mathbf{p}_{\text{repulsion}}) - \tau(\Delta\mathbf{p}_{\text{attraction}}), \tag{4.12}$$

where $\phi(\mathbf{n}_i, \mathbf{p} - \mathbf{p}_i) = e^{-\frac{(\mathbf{n}_i^\top(\mathbf{p} - \mathbf{p}_i))^2}{\sigma_p}}$ is the anisotropic projection weight, $\mathbf{n}_i$ is the point normal of neighbor $\mathbf{p}_i$ and w is the spatial weight defined in (4.9). We use the same directional clipping function τ as before to bound the two update terms individually, which improves the stability of the algorithm for sparse point clouds.

By design, new points are inserted in areas with low density or high curvature. The trade-off is controlled by a unifying priority score $P(\mathbf{p}) = \max_{\mathbf{p}_i \in \mathcal{N}(\mathbf{p})} B(\mathbf{p}, \mathbf{p}_i)$, where B is a distance measure (see [Hua+13b] for the exact definition). Denoting the point with the highest priority as $\mathbf{p}^*$, a new point is inserted at the midpoint between $\mathbf{p}^*$ and neighbor $\mathbf{p}^*_{i^*}$, where $\mathbf{p}^*_{i^*} = \arg\max_{\mathbf{p}_i \in \mathcal{N}(\mathbf{p}^*)} B(\mathbf{p}^*, \mathbf{p}_i)$. In the original EAR method, the insertion is done iteratively, requiring an update of the neighborhood information and recalculation of B after every insertion. Instead, to allow parallel computation at GPU, we approximate this process by simultaneously inserting a maximum of $|\mathcal{P}|/10$ points each step. In this way, however, inserting the midpoint of point pairs would lead to duplicated new points. Thus, we insert asymmetrically at $\frac{1}{3}(\mathbf{p}_{i^*}^* + 2\mathbf{p}^*)$ instead.

We then project the upsampled iso-points to the iso-surface. As shown in Fig. 4.14, the final iso-points successfully reflect the 3D geometry of the current implicit surface.

Compared with using marching cubes to extract the iso-surface, our adaptive sampling is efficient. Since both the resampling and upsampling steps only require information of local neighborhood, we implement it on the GPU. Furthermore, since we use the iso-points from the previous iteration for initialization, the overall point distribution improves as the training stabilizes, requiring fewer (or even zero) resampling and upsampling steps at later stages.

Utilizing iso-points in optimization

We introduce two scenarios of using iso-points to guide neural implicit surface optimization: (i) importance sampling for multi-view reconstruction and (ii) regularization when reconstructing neural implicit surfaces from raw input point clouds.

Iso-points for importance sampling. Optimizing a neural implicit function to represent a high-resolution 3D shape requires abundant training samples – specifically, many supervision points sampled close to the iso-surface to capture the fine-grained geometry details accurately. However, for applications where the explicit 3D geometry is not available during training, the question of how to generate training samples remains mostly unexplored.

We exploit the geometry information and the prediction uncertainty carried by the iso-points during training. The main idea is to compute a saliency metric on iso-points, then add more samples in those regions with high saliency. To this end, we experiment with two types of metrics: curvature-based and loss-based. The former aims at emphasizing geometric features, typically reflected by high curvature. The latter is a form of hard example mining, as we sample more densely where the higher loss occurs, as shown in Fig. 4.15.

Since the iso-points are uniformly distributed, the curvature can be approximated by the norm of the Laplacian, *i.e.* $\mathcal{R}_{\text{curvature}}(\mathbf{p}) = \|\mathbf{p} - \sum_{\mathbf{p}_i \in \mathcal{N}(\mathbf{p})} \mathbf{p}_i\|$.

For the loss-based metric, we project the iso-points on all training views and compute the average loss at each point, *i.e.* $\mathcal{R}_{\text{loss}}(\mathbf{p}) = \frac{1}{N}\sum_i^N \text{loss}(\mathbf{p})$, where N is the number of occurrences of $\mathbf{p}$ in all views.

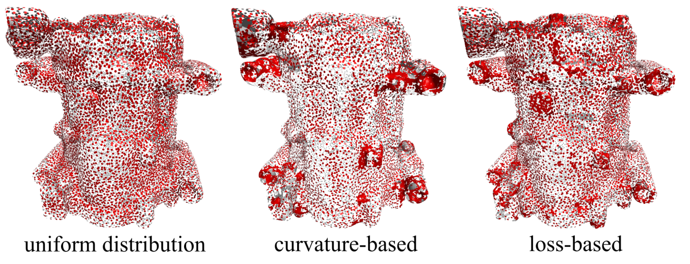


Figure 4.15.: Examples of importance sampling based on different saliency metrics. Uniform iso-points treat different regions of the iso-surface equally. We compute different metrics on the uniform iso-points, based on which more samples can be gathered in certain regions. The curvature-based metric emphasizes geometric details, while the loss-based metric allows for hard example mining.

As both metrics evolve smoothly, we need not update them in each training iteration. Denoting the iso-points at which we computed the metric as $\mathcal{T}$ and the subset of template points with high metric values by $\mathcal{T}^* = \{\mathbf{t}^*\}$, the metric-based insertion for each point $\mathbf{p}$ in the current iso-point set $\mathcal{P}_t$ can be written as

$$\mathbf{p}_{\text{new},i} = \tfrac{2}{3}\mathbf{p} + \tfrac{1}{3}\mathbf{p}_i, \forall \mathbf{p}_i \in \mathcal{N}(\mathbf{p}), \text{ if } \min_{\mathbf{t}^*} \|\mathbf{p} - \mathbf{t}^*\| \leq \sigma.$$

The neighborhood radius σ is the same one used in (4.8).

Iso-points for regularization. The access to an explicit representation of the implicit surface also enables us to incorporate geometry-motivated priors into the optimization objective, exerting finer control of the reconstruction result.

Let us consider fitting a neural implicit surface to a point cloud. Depending on the acquisition process, the point cloud may be sparse, distorted by noise, or distributed highly unevenly. Similar to previous works [UVL18; Wil+19], we observe that the neural network tends to reconstruct a smooth surface at the early stage of learning, but then starts to pick up higher frequency signals from the corrupted observations and overfits to noise and other caveats in the data, as shown in Fig. 4.16. This characteristic is consistent across network architectures, including those designed to accommodate high-frequency information, such as SIREN.

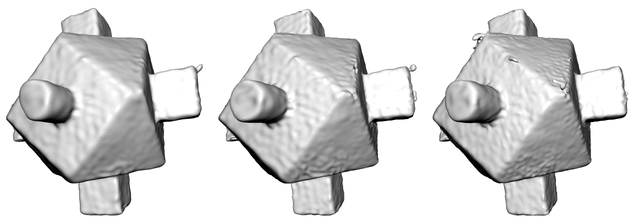


Figure 4.16.: Progression of overfitting. When optimizing a neural implicit surface on a noisy point cloud, the network initially outputs a smooth surface, but increasingly overfits to the noise in the data. Shown here are the reconstructed surfaces after 1000, 2000 and 5000 iterations. The input point cloud is acquired in-house using an Artec Eva scanner.

Existing methods that address overfitting include early stopping, weight decay, drop-out *etc*. [Goo+16]. However, whereas these methods are generic tools designed to improve network generalizability for training on large datasets, we propose a novel regularization approach that is specifically tailored to training neural implicit surfaces and serves as a platform to integrate a plethora of 3D priors from classic geometry processing.

Our main idea is to use the iso-points as a consistent, smooth, yet dynamic approximation of the reference geometry. Consistency and smoothness ensure that the optimization does not fluctuate and overfit to noise; the dynamic nature lets the network pick up consistent high-frequency signals governed by underlying geometric details.

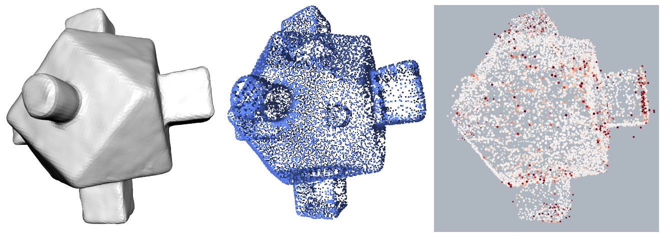


Figure 4.17.: Iso-points for regularization. At the early stage of the training, the implicit surface is smooth (left), and we extract iso-points (middle) as a reference shape, which can facilitate various regularization terms. In the example on the right, we use the iso-points to reduce the influence of outliers (shown in red).

To this end, we extract iso-points after a short warmup training (*e.g*. 300 iterations). Because of the aforementioned smooth characteristic of the network, the noise level in the initial iso-points is minimal. Then, during subsequent training, we update the iso-points periodically (*e.g*. every 1000 iterations) to allow them to gradually evolve as the network learns more high-frequency information.

The utility of the iso-points includes, but is not limited to 1) serving as additional training data to circumvent data scarcity, 2) enforcing additional geometric constraints, 3) filtering outliers in the training data.

Specifically, for sparse or hole-ridden input point clouds, we take advantage of the uniform distribution of iso-points and augment supervision in under-sampled areas by enforcing the signed distance value on all iso-points to be zero:

$$\mathcal{L}_{\text{isoSDF}} = \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p} \in \mathcal{P}} |f(\mathbf{p})|. \tag{4.13}$$

Given the iso-points, we compute their normals from their local neighborhood using principal component analysis (PCA) [Hop+92a]. We then increase surface smoothness by enforcing consistency between the normals estimated by PCA and those computed from the gradient of the network, *i.e.*

$$\mathcal{L}_{\text{isoNormal}} = \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p} \in \mathcal{P}} (1 - |\cos(J_f^\top(\mathbf{p}), \mathbf{n}_{\text{PCA}})|). \tag{4.14}$$

The larger the PCA neighborhood is, the smoother the reconstruction becomes. Optionally, additional normal filtering can be applied after PCA to reduce over-smoothing and enhance geometric features. Finally, we can use the iso-points to filter outliers in the training data. Specifically, given a batch of training points $\mathcal{Q} = \{\mathbf{q}\}$, we compute a per-point loss weight based on their alignment with the iso-points. Here, we choose to use bilateral weights to take both the Euclidean and the directional distance into consideration. Denoting the normalized gradient of an iso-point $\mathbf{p}$ and a training point $\mathbf{q}$ as $\mathbf{n_p}$ and $\mathbf{n_q}$, respectively, the bilateral weight can written as

$$v(\mathbf{q}) = \min_{\mathbf{p}\in\mathcal{P}} \phi(\mathbf{n_p}, \mathbf{p}-\mathbf{q})\psi(\mathbf{n_p}, \mathbf{n_q}), \quad \text{with} \tag{4.15}$$

$$\psi(\mathbf{n_p}, \mathbf{n_q}) = e^{-\left(1-\frac{1-\mathbf{n_p}^\top\mathbf{n_q}}{1-\cos(\sigma_n)}\right)^2}, \tag{4.16}$$

where σ_n regulates the sensitivity to normal difference; we set $\sigma_n = 60^\circ$ in our experiments. This loss weight can be incorporated into the existing loss functions to reduce the impact of outliers. A visualization of the outliers detected by this weight is shown in Fig. 4.17.

4.2.2. Results

Iso-points can be incorporated into the optimization pipelines of existing state-of-the-art methods for implicit surface reconstruction. In this section, we demonstrate the benefits of the specific techniques introduced in Sec. 4.2.1.

We choose state-of-the-art methods as the baselines, then augment the optimization with iso-points. Results show that the augmented optimization outperforms the baseline quantitatively and qualitatively.

Sampling with iso-points

We evaluate the benefit of utilizing iso-points to generate training samples for multi-view reconstruction. As the baseline, we employ the ray-tracing algorithm from a state-of-the-art neural implicit renderer IDR [Yar+20], which generates training samples by ray-marching from the camera center through uniformly sampled pixels in the image. As shown in Fig. 4.18, three types of samples are used for different types of supervision: on-surface samples, which are ray-surface intersections inside the object's 2D silhouette, in-surface samples, which are points with the lowest signed distance on the non-intersecting rays inside 2D silhouette, and out-surface samples, which are on the rays outside the 2D silhouette either at the surface intersection or at the position with the lowest signed distance.

	baseline (ray-tracing)	uniform	curvature-based	loss-based
CD $\cdot 10^{-4}$(position)	17.24	1.80	1.83	**1.71**
CD $\cdot 10^{-1}$(normal)	1.51	1.10	0.99	**0.95**

Table 4.5.: Quantitative effect of importance sampling with iso-points. Compared to the baseline, which generates training points via ray-marching, we use iso-points to draw more attention on the implicit surface. The result is averaged over 10 models selected from the Sketchfab dataset [Yif+19b].

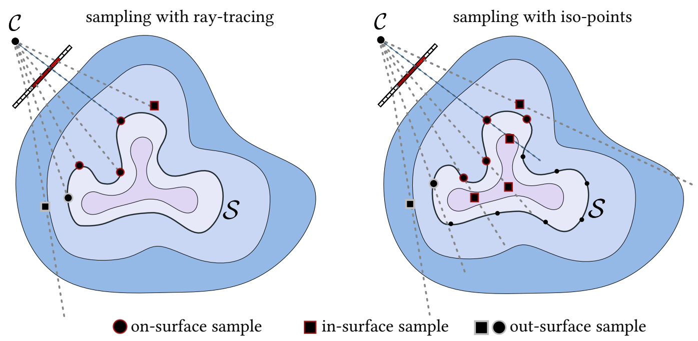


Figure 4.18.: A 2D illustration of two sampling strategies for multi-view reconstruction. Ray-tracing (left) generates training samples by shooting rays from camera $\mathcal{C}$ through randomly sampled pixels; depending on whether an intersection is found and whether the pixel lies inside the object silhouette, three types of samples are generated: on-surface, in-surface and out-surface points. We generate on-surface samples directly from iso-points, obtaining evenly distributed samples on the implicit surface, and also use the iso-points to generate more reliable in-surface samples.

On this basis, we incorporate the iso-points directly as on-surface samples. We can direct the learning attention by varying the distribution of iso-points using the saliency metrics described in Sec. 4.2.1. The iso-points also provide us prior knowledge to generate more reliable in-surface samples. More specifically, as shown in Fig. 4.18 (right), we generate the three types of samples as follows: a) on-surface samples: we remove occluded iso-points by visibility testing using a point rasterizer, and select those iso-points whose projections are inside the object silhouette; b) in-surface samples: on the camera rays that pass through the on-surface samples, we determine the point with the lowest signed distance on the segment between the on-surface sample and the farther intersection with the object's bounding sphere. c) out-surface samples: we shoot camera rays through pixels outside the object silhouette, and choose the point with the lowest signed distance.

Below, we demonstrate two benefits of the proposed sampling scheme.

Surface details from importance sampling. First, we examine the effect of drawing on-surface samples using iso-points by comparing the optimization results under fixed optimization time and the same total sample count.

As inputs, we render 512 × 512 images per object under known lighting and material from varied camera positions. When training with the iso-points,

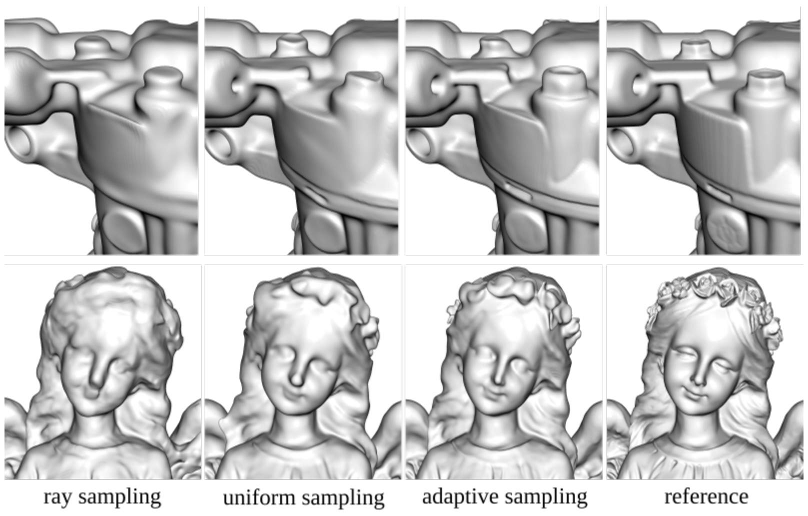


Figure 4.19.: Qualitative comparison between sampling strategies for multi-view reconstruction. Using the same optimization time and similar total sample count, sampling the surface points with uniformly distributed iso-points considerably improves the reconstruction accuracy. A loss based importance sampling further improves the recovery of small-scale structures. The models shown here are the COMPRESSOR and ANGEL2 from the Sketchfab dataset [Yif+19b].

we extract 4,000 iso-points after 500 iterations, then gradually increase the density until reaching 20,000 points. To match the sample count, in the ray-tracing variation, we randomly draw 2,048 pixels per image and then increase the sample count until reaching 10,000 pixels. We use a 3-layer SIREN model with the frequency multipliers set to 30, and optimize with a batch size of 4.

We evaluate our method quantitatively using 10 watertight models from the Sketchfab dataset [Yif+19b]. As shown in Table 4.5, we compute 2-way chamfer point-to-point distance ($\|\mathbf{p}_i - \mathbf{p}_j\|^2$) and normal distance $(1 - \cos(\mathbf{n}_i, \mathbf{n}_j))$ on 50K points, uniformly sampled from the reconstructed meshes.

Results show that using uniform iso-points as on-surface samples compares favorably against the baseline, especially in the normal metric. It suggests that we achieve higher fidelity on the finer geometric features, as our surface samples overcome under-sampling issues occurring at small scale details. We also see that importance sampling with iso-points and loss based upsampling exhibits a substantial advantage over other variations, demonstrating the effectiveness of smart allocation of the training samples according to the current learning state. In comparison, curvature-based sampling performs similarly to the baseline, but notability worse than with the uniform iso-points. We observe that the iso-points, in this case, are highly concentrated on a few spots on the surface and ignore regions where the current reconstruction is problematic (Fig. 4.15).

The improvement is more pronounced qualitatively, as shown in Fig. 4.19. Sampling on-surface with uniform iso-points clearly enhances reconstruction accuracy compared to the baseline with ray-tracing. The finer geometric details further improve with loss-based importance sampling.

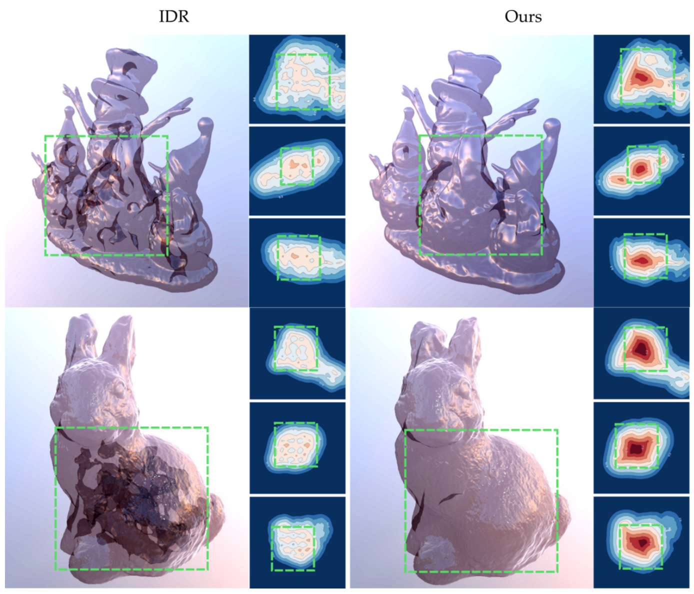


Figure 4.20.: Topological correctness of the reconstructed surface in multi-view reconstruction. The erroneous inner structures of IDR are indicated by the artifacts in the images rendered by Blender [Hes10] (using glossy and transparent material) and by the contours of the signed distance field on cross-sections.

IDR can reconstruct impressive geometric details on the DTU dataset [Jen+14], but a closer inspection shows that the reconstructed surface contains a considerable amount of topological errors inside the visible surface. We use iso-points to improve the topological accuracy of the reconstruction.

Topological correctness from 3D prior. We use the same network architecture and training protocol as IDR, which samples 2048 pixels from a randomly chosen view in each optimization iteration. We use uniform iso-points in this experiment. To keep a comparable sample count, we subsample the visible iso-points to obtain a maximum of 1500 on-surface samples per iteration. Since our strategy automatically creates more in-surface samples (as shown in Fig. 4.18), we halve the loss weight on the in-surface samples compared to the original implementation.

We visualize the topology of the reconstructed surface in Fig. 4.20. To show the inner structures of the surface, we render it in transparent and glossy material with a physically-based renderer [Hes10] and show the back faces of the mesh. Dark patches in the rendered images indicate potentially erroneous light transmission caused by inner structures. Similarly, we also show the contour lines of the iso-surface on a cross-section to indicate the irregularity of the reconstructed implicit function. In both visualizations, the incorrect topology in IDR reconstruction is apparent. In contrast, our sampling enables more accurate reconstruction of the signed distance field inside and outside the surface with more faithful topological structure.

Regularization with iso-points

We evaluate the benefit of using iso-points for regularization. As an example application, we consider surface reconstruction from a noisy point cloud.

As our baseline method, we use the publicly available SIREN codebase and adopt their default optimization protocol. The noisy input point clouds are either acquired with a 3D scanner (Artec Eva) or reconstructed [FP09] from the multi-view images in the DTU dataset.

In each optimization iteration, the baseline method randomly samples an equal number of oriented surface points $\mathcal{Q}_s = \{\mathbf{q}_s, \mathbf{n}_s\}$ from the input point cloud and unoriented off-surface points $\mathcal{Q}_o = \{\mathbf{q}_o\}$ from bounding cube's interior. The optimization objective is comprised of four parts:

$$\mathcal{L} = \gamma_{\text{onSDF}}\mathcal{L}_{\text{onSDF}} + \gamma_{\text{normal}}\mathcal{L}_{\text{normal}} + \gamma_{\text{offSDF}}\mathcal{L}_{\text{offSDF}} + \gamma_{\text{eikonal}}\mathcal{L}_{\text{eikonal}},$$

where

$$\mathcal{L}_{\text{onSDF}} = \frac{\sum_{\mathbf{q}_s \in \mathcal{Q}_s} |f(\mathbf{q}_s)|}{|\mathcal{Q}_s|}, \mathcal{L}_{\text{normal}} = \frac{\sum_{\mathbf{q}_s \in \mathcal{Q}_s} |1 - \cos(J_f^{\top}(\mathbf{q}_s), \mathbf{n}_s)|}{|\mathcal{Q}_s|},$$

$$\mathcal{L}_{\text{offSDF}} = \frac{\sum_{\mathbf{q}_o \in \mathcal{Q}_o} e^{-\alpha|f(\mathbf{q}_o)|}}{|\mathcal{Q}_o|}, \mathcal{L}_{\text{eikonal}} = \frac{\sum_{\mathbf{q} \in \mathcal{Q}_o \cup \mathcal{Q}_s} |1 - \|J_f^{\top}(\mathbf{q})\||}{|\mathcal{Q}_s| + |\mathcal{P}_o|}$$

and $\gamma_{\text{onSDF}} = 1000, \gamma_{\text{normal}} = 100, \gamma_{\text{offSDF}} = 50, \gamma_{\text{eikonal}} = 100$. We alter this objective with the outlier-aware loss weight defined in (4.16), and then add the regularizations on iso-points $\mathcal{L}_{\text{isoSDF}}$ (4.13) and $\mathcal{L}_{\text{isoNormal}}$ (4.14). The final objective becomes

$$\mathcal{L} = \gamma_{\text{onSDF}}(\mathcal{L}_{\text{onSDF}} + \mathcal{L}_{\text{isoSDF}}) + \gamma_{\text{normal}}(\mathcal{L}_{\text{normal}} + \mathcal{L}_{\text{isoNormal}}) + \\ \gamma_{\text{offSDF}}\mathcal{L}_{\text{offSDF}} + \gamma_{\text{eikonal}}\mathcal{L}_{\text{eikonal}},$$

where the loss terms with on-surface points are weighted as follows:

$$\mathcal{L}_{\text{onSDF}} = \frac{1}{|\mathcal{Q}_s|} \sum_{\mathbf{q}_s \in \mathcal{Q}_s} v(\mathbf{q}_s)|f(\mathbf{q}_s)|,$$

$$\mathcal{L}_{\text{normal}} = \frac{1}{|\mathcal{Q}_s|} \sum_{\mathbf{q}_s \in \mathcal{Q}_s} v(\mathbf{q}_s)|1 - \cos(J_f^{\top}(\mathbf{q}_s), \mathbf{n}_s)|.$$

The iso-points are initialized by subsampling the input point cloud by 1/8 and updated every 2000 iterations.

The comparison with the baseline, i.e., vanilla optimization without regularization, is shown in Fig. 4.21. For the DTU-MVS data, we also conduct quantitative evaluation following the standard DTU protocol as shown in Table 4.6, i.e., L1-Chamfer distance between the reconstructed and reference point cloud within a predefined volumetric mask [Aan+16]. For clarity, we also show the results of screened Poisson reconstruction [KH13a] and

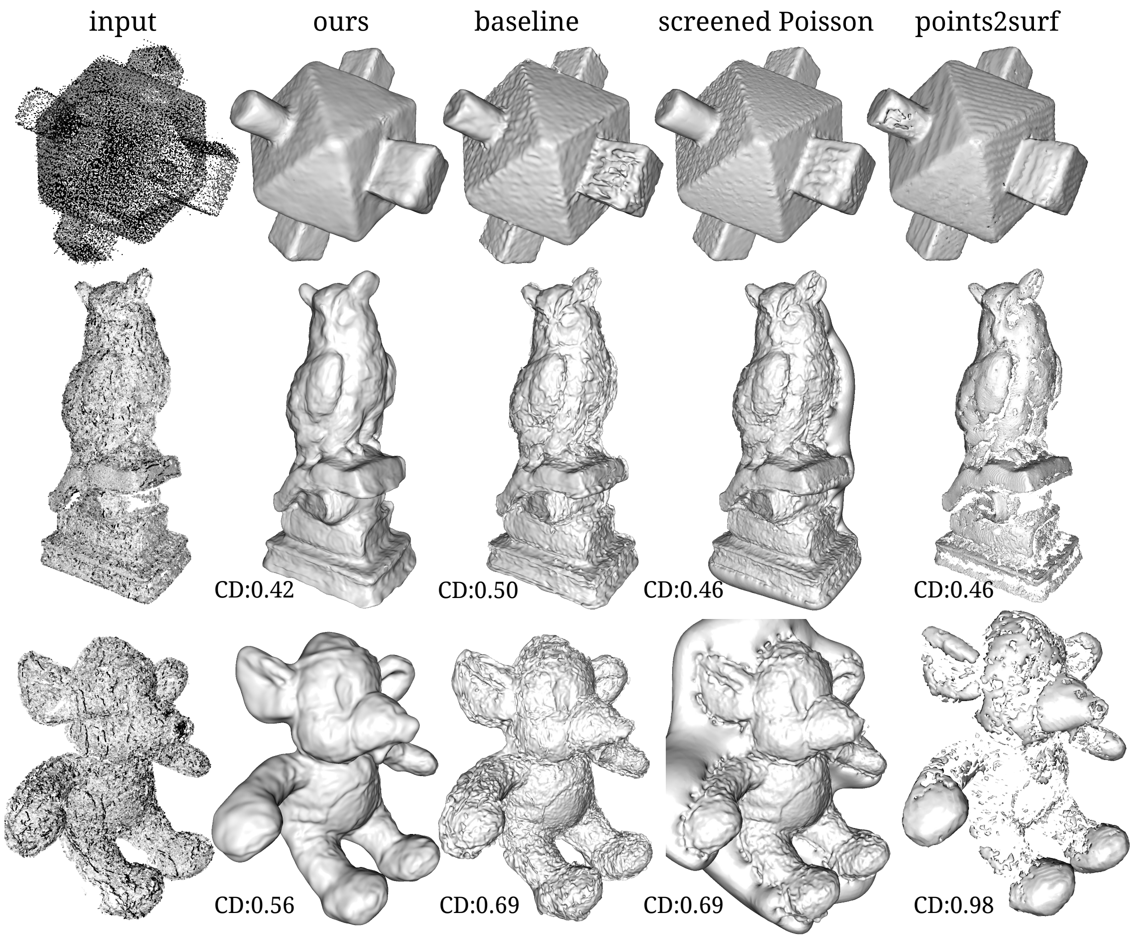


Figure 4.21.: Implicit surface reconstruction from noisy and sparse point clouds. From left to right: input, reconstruction with our proposed regularizations, baseline reconstruction without our regularizations, screened Poisson reconstruction and Points2Surf. CD denotes L1-Chamfer distance. The sparse point cloud in the first row is acquired with an Artec Eva scanner, whereas the inputs in the second row and third row are reconstructed from DTU dataset (model 105 and 122) using [FP09].

Points2Surf [Erl+20]. The former reconstructs a watertight surface from an oriented point set by solving local Poisson equations; the latter fits an implicit neural function to an unoriented point set in a global-to-local manner. Compared with the baseline and screened Poisson, our proposed regularizations significantly suppresses noise. Points2Surf can handle noisy input well, but the sign propagation appears to be sensitive to the point distribution, leading to holes in the reconstructed mesh. Moreover, since their model does not use the points' normal information, the reconstruction lacks detail.

ID	ours	baseline	point2surf	screened Poisson
55	**0.37**	0.41	0.56	0.42
69	**0.59**	0.65	0.61	0.63
105	**0.56**	0.69	0.98	0.69
110	0.54	**0.51**	0.61	0.55
114	**0.38**	0.45	0.45	0.37
118	**0.45**	0.49	0.59	0.55
122	**0.42**	0.50	0.46	0.46
Average	0.53	0.61	0.52	**0.47**

Table 4.6.: Quantitative evaluation for surface reconstruction from a noisy sparse point cloud. We evaluate the two-way L1-chamfer distance on a subset of the DTU-MVS dataset.

Performance analysis

The main overhead in our approach is the projection step. One newton iteration requires a forward and a backward pass. On average, the projection terminates within 4 iterations. This procedure is optimized by only considering points that are not yet converged at each iteration. Empirically, the computation time of extracting the iso-points once is typically equivalent to running 3 training iterations. In practice, as we extract iso-points only periodically, the total optimization time only increases marginally.

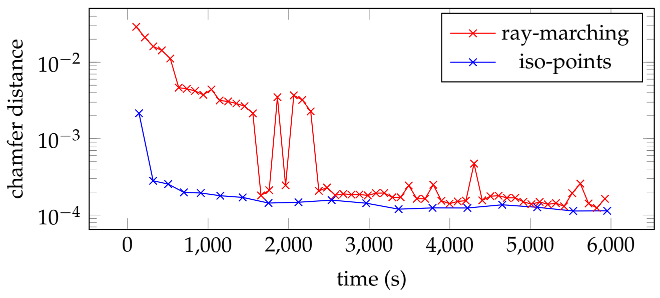


Figure 4.22.: Validation error in relation to optimization time. The first time stamp is at the 100-th iteration.

In the case of multi-view reconstruction, since the ray-marching itself is an expensive operation, involving multiple forward passes per ray, the overhead of our approach is much less notable. As discussed, we filter the occluded iso-points before the projection, which also saves optimization time. The trade-off between optimization speed and quality is depicted in a concrete example in Fig. 4.22, where we plot the evolution of the chamfer distance during the optimization of the COMPRESSOR model (first row of Fig. 4.19). Compared to the baseline optimization with ray-tracing, it is evident that the iso-points augmented optimization consistently achieves better results at every timestamp. In other words, with iso-points we can reach a given quality threshold faster.

4.3. Concluding remarks

Contributions. In this chapter, we presented two algorithms for surface reconstruction via neural implicit surface representations.

First, we proposed a new parameterization of neural implicit functions for detailed geometry representation, named implicit displaced field (IDF). Extending displacement mapping, a classic shape modeling technique, our formulation represents a given shape by a smooth base surface and a high-frequency displacement field that offsets the base surface along its normal directions. This resulting frequency partition enables the network to concentrate on regions with rich geometric details, significantly boosting its representational power. Thanks to the theoretically grounded network design, the high-frequency signal is well constrained. As a result our model shows improved convergence qualities compared to other models leveraging high-frequency signals, such as SIREN and positional encoding. Furthermore, emulating the deformation-invariant quality of classic displacement mapping, we extend our method to enable transferability of the implicit displacements, thus making it possible to use implicit representations for new geometric modeling tasks.

In the second algorithm, we introduced a complementary representation, iso-points, for learning implicit surfaces. Implicit surfaces can represent 3D structures in arbitrary resolution and topology but lack an explicit form to adapt the optimization process to input data. Iso-points, as a point cloud adaptively distributed on the underlying surface, are fairly straightforward and efficient to manipulate and analyze the underlying 3D geometry.

We presented effective algorithms to extract and utilize iso-points. Extensive experiments show the power of our hybrid representation. We demonstrate that iso-points can be readily employed by state-of-the-art neural 3D reconstruction methods to significantly improve optimization efficiency and reconstruction quality.

Limitations. The main limitation of IDF centers around its transferability and the application of detail transfer. Since we used normals as a descriptor, which is not orientation invariant, the demonstrated detail transfer application assumes pre-align shape inputs, thus limiting the applicability to arbitrary deformable and unaligned shapes. To address this limitation, one option is to consider exploring sparse correspondences as part of the input, which is a common practice in computer graphics applications, to facilitate subsequent automatic shape alignment. Another limitation of the transferable IDF is associated with the use of 3D convolutions as the context descriptor. 3D convolutions have a large memory footprint and also induce slower training and inference. More importantly, while convolutions are effective in propagating translation-invariant features to off-surface areas, they also tend to oversmooth features. One promising direction to address this issue is to consider the features of an off-surface point as a learnable aggregation of obtained on-surface features. To this end, attention-based mechanisms such as Transformers [Vas+17] may serve as a potential apparatus.

In the iso-points project, one limitation lies in the proposed sampling strategy in that it is mainly determined by the geometry of the underlying surface and does not explicitly model the appearance. In the future, we would like to extend our hybrid representation to model the joint space of geometry and appearance, which can in turn allow us to apply path-tracing for global illumination, bridging the gap between existing neural rendering approaches and classic physically based rendering. Secondly, in the current implementation, we experimented with object-level reconstruction. Further optimization is required to scale up to multi-object scene-level reconstruction.

CHAPTER 5

Shape Deformation

Deformation of 3D shapes is a ubiquitous task, arising in many vision and graphics applications. For instance, *deformation transfer* [SP04] aims to infer a deformation from a given pair of shapes and apply the same deformation to a novel target shape. As another example, a small dataset of shapes from a given category (*e.g.*, chairs) can be augmented by *synthesizing variations*, where each variation deforms a randomly chosen shape to the proportions and morphology of another while preserving local detail [Xu+10; Wan+19].

In this chapter, we present a learning-based shape deformation algorithm, which deforms a shape in a given shape category, *e.g.* chairs, to match an arbitrary shape in the same category without correspondence information. In particular, unlike previous learned deformation methods, the proposed approach is detail-preserving by construction, which is achieved by leveraging a classic technique in interactive shape modeling: *cage-based deformation*.

5.1. Neural cage deformation

Deformation techniques usually need to simultaneously optimize at least two competing objectives. The first is alignment with the target, *e.g.*, matching limb positions while deforming a human shape to another human in a different pose. The second objective is adhering to quality metrics, such as distortion minimization and preservation of local geometric features, such as the human's face. These two objectives are contradictory, since a perfect alignment of a deformed source shape to the target precludes preserving the original details of the source.

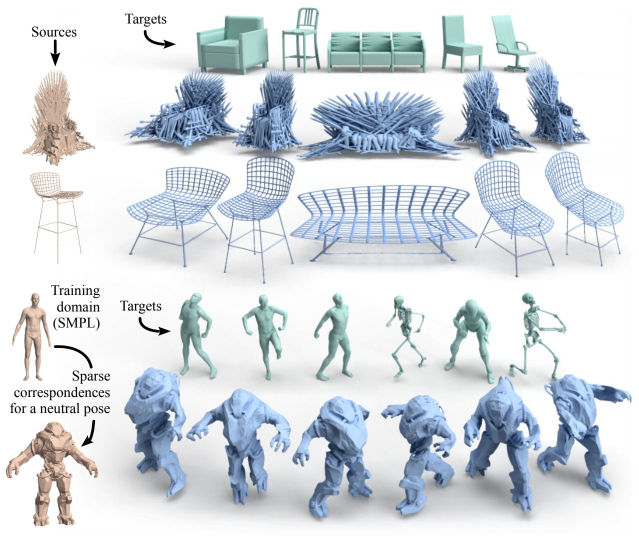


Figure 5.1.: Applications of our neural cage-based deformation method. *Top:* Complex source chairs (brown) deformed (blue) to match target chairs (green), while accurately preserving detail and style with non-homogeneous changes that adapt different regions differently. No correspondences are used at any stage. *Bottom:* A cage-based deformation network trained on many posed humans (SMPL) can transfer various poses of novel targets (SCAPE, skeleton, X-Bot, in green) to a very dissimilar robot of which only a single neutral pose is available. A few matching landmarks between the robot and a neutral SMPL human are required. Dense correspondences between SMPL humans are used only during training.

Due to these conflicting objectives, optimization techniques [LSP08] require parameter tuning to balance the two competing terms, and are heavily reliant on an inferred or manually supplied correspondence between the source and the target. These parameters vary based on the shape category, representation, and the level of dissimilarity between the source and the target.

To address these limitations, recent techniques train a *neural network* to predict shape deformations. This is achieved by predicting new positions for all vertices of a template shape [Tan+18] or by implicitly representing the deformation as a mapping of all points in 3D, which is then used to map each vertex of a source shape [Wan+19; Gro+19]. Examples of the results of some of these methods can be seen in Fig 5.4, which demonstrates the limitations of such approaches: the predicted deformations corrupt features and exhibit distortion, especially in areas with thin structures, fine details or gross discrepancies between source and target. These artifacts are due to the inherent limitations of neural networks to capture, preserve, and generate high frequencies.

In this work, we circumvent the above issues via a classic geometry processing technique called *cage-based deformation* [JSW05; LLCO08; Jos+07], abbreviated to *CBD*. In CBD, the source shape is enclosed in a very coarse scaffold mesh called the *cage* (Fig 5.2). The deformation of the cage is transferred to the enclosed shape by interpolating the translations of the cage vertices. Fittingly, the interpolation schemes in these classic works are carefully designed to preserve details and minimize distortion.

Our main technical contribution is a novel neural architecture in which, given a source mesh, learnable parameters are optimized to predict both the positioning of the cage around the source shape, as well as the deformation of that cage, which drives the deformation of the enclosed shape in order to match a target shape. The source shape is deformed by deterministically interpolating the new positions of its surface points from those of the cage vertices, via a novel, differentiable, cage-based deformation layer. The pipeline is trained end-to-end on a collection of randomly chosen pairs of shapes from a training set.

The *first key advantage* of our method is that cages provide a much more natural space for predicting deformations: CBD is feature-preserving by construction, the degrees of freedom in deformation only depends on the number of vertices on the coarse cage. In short, our network makes a prediction in a low-dimensional space of highly regular deformations.

The *second key advantage* is that our method is not tied to a single source shape, nor to a single mesh topology. As the many examples in this paper demonstrate, the trained network can predict and deform cages for similar shapes not observed during training. The target shape can be crude and noisy, *e.g.*, a shape acquired with cheap scanning hardware or reconstructed from an image. Furthermore, dense correspondences between the source and target shapes are not required in general, though they can help when the training set has very varied articulations. Thus the method can be trained on large datasets that are not co-registered and do not have consistently labeled landmarks.

We show the utility of our method in two main applications. We generate shape variations by deforming a 3D model using other shapes as well as images as targets. We also use our method to pose a human according to a target humanoid character, and, given a few sparse correspondences, perform deformation transfer and pose an arbitrary novel humanoid. See Figures 5.1, 5.7, 5.9 and 5.4 for examples.

5.1.1. Method

We now detail our approach for learning cage-based deformations (CBD). We start with a brief overview of the principles of CBD, and then explain how we train a network to control these deformations from data.

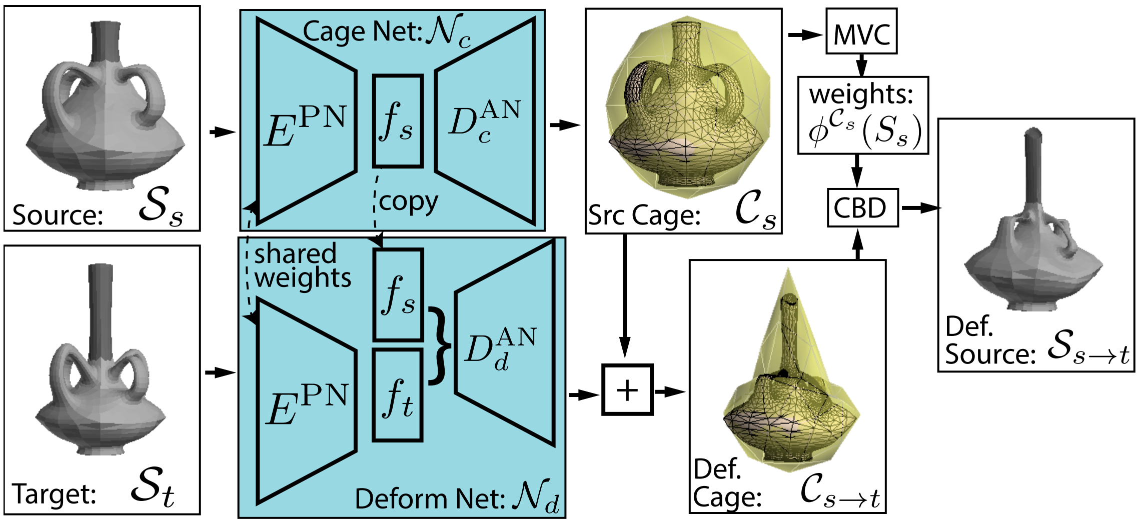

Source: $\mathcal{S}_s$
Target: $\mathcal{S}_t$
Cage Net: $\mathcal{N}_c$
E^{PN}
f_s
D_c^{AN}
copy
shared weights
f_t
D_d^{AN}
Deform Net: $\mathcal{N}_d$
Src Cage: $\mathcal{C}_s$
MVC
weights: $\phi^{\mathcal{C}_s}(S_s)$
CBD
+
Def. Cage: $\mathcal{C}_{s\to t}$
Def. Source: $\mathcal{S}_{s\to t}$

Figure 5.2.: Overview. A source $\mathcal{S}_s$ and a target $\mathcal{S}_t$ are encoded by the same PointNet encoder E^{PN} into latent codes f_s and f_t, resp. An AtlasNet-style decoder D_c^{AN} decodes f_s to a source cage $\mathcal{C}_s$ in the cage module $\mathcal{N}_c$. Another decoder D_d^{AN} creates the offset for $\mathcal{C}_s$ in the deformation module $\mathcal{N}_d$ from the concatenation of f_s and f_t, yielding a deformed cage $\mathcal{C}_{s\to t}$. Given a source cage and shape, our novel MVC layer computes the mean value coordinates $\phi^{\mathcal{C}_s}(\mathcal{S}_s)$, which are used to produce a deformed source shape $\mathcal{S}_{s\to t}$ from the cage deformation $\mathcal{C}_{s\to t}$.

Cage-based deformations

CBD are a type of freeform space deformations. Instead of defining a deformation solely on the surface $\mathcal{S}$, space deformations warp the entire ambient space in which the shape $\mathcal{S}$ is embedded. In particular, a CBD controls this warping via a coarse triangle mesh, called a *cage* $\mathcal{C}$, which typically encloses $\mathcal{S}$. Given the cage, any point in ambient space $\mathbf{p} \in \mathbb{R}^3$ is encoded via generalized barycentric coordinates, as a weighted average of the cage vertices $\mathbf{v}_j$: $\mathbf{p} = \sum \phi_j^{\mathcal{C}}(\mathbf{p})\,\mathbf{v}_j$, where the weight functions $\left\{\phi_j^{\mathcal{C}}\right\}$ depend on the relative position of $\mathbf{p}$ w.r.t. to the cage vertices $\{\mathbf{v}_j\}$. The deformation of any point in ambient space is obtained by simply offsetting the cage vertices and interpolating their new positions $\mathbf{v}'_j$ with the pre-computed weights, *i.e.*

$$\mathbf{p}' = \sum_{0\le j<|V_{\mathcal{C}}|} \phi_j^{\mathcal{C}}(\mathbf{p})\,\mathbf{v}'_j. \tag{5.1}$$

Previous works on CBD constructed various formulae to attain weight functions $\left\{\phi_j^{\mathcal{C}}\right\}$ with specific properties, such as interpolation, linear precision, smoothness and distortion minimization. We choose mean value coordinates (MVC) [JSW05] for their feature preservation and interpolation properties, as well as simplicity and differentiability w.r.t. the source and deformed cages' coordinates.

Learning cage-based deformation

As our goal is an end-to-end pipeline for deforming shapes, we train the network to predict both the source cage and the target cage, in order to optimize the quality of the resulting deformation. Given a source shape $\mathcal{S}_s$ and a target shape $\mathcal{S}_t$, we design a deep neural network that predicts a cage deformation that warps $\mathcal{S}_s$ to $\mathcal{S}_t$ while preserving the details of $\mathcal{S}_s$. Our network is composed of two branches, as illustrated in Fig 5.2: a cage-prediction model $\mathcal{N}_c$, which predicts the initial cage $\mathcal{C}_s$ around $\mathcal{S}_s$, and a deformation-prediction model $\mathcal{N}_d$, which predicts an offset from $\mathcal{C}_s$, yielding the deformed cage $\mathcal{C}_{s\to t}$, *i.e.*

$$\mathcal{C}_s = \mathcal{N}_c\left(\mathcal{S}_s\right) + \mathcal{C}_0, \quad \mathcal{C}_{s\to t} = \mathcal{N}_d\left(\mathcal{S}_t, \mathcal{S}_s\right) + \mathcal{C}_s \tag{5.2}$$

Since both branches are differentiable, they can be both learned jointly in an end-to-end manner.

The branches $\mathcal{N}_c$ and $\mathcal{N}_d$ only predict the cage and do not directly rely on the detailed geometric features of the input shapes. Hence, our network does not require high-resolution input nor involved tuning for the network architectures. In fact, both $\mathcal{N}_c$ and $\mathcal{N}_d$ follow a very streamlined design: their encoders and decoders are simplified versions of the ones used in AtlasNet [Gro+18a]. We remove the batch normalization and reduce the channel sizes, and instead of feeding 2D surface patches to the decoders, we feed a template cage $\mathcal{C}_0$ and the predicted initial cage $\mathcal{C}_s$ to the the cage predictor and deformer respectively, and let them predict the offsets. By default, $\mathcal{C}_0$ is a 42-vertex sphere.

Loss terms

Our loss incorporates three main terms. The first term optimizes the source cage to encourage positive mean value coordinates. The two latter terms optimize the deformation, the first by measuring alignment to target and the second by measuring shape preservation. Together, these terms comprise our basic loss function:

$$\mathcal{L} = \alpha_{\text{MVC}}\mathcal{L}_{\text{MVC}} + \mathcal{L}_{\text{align}} + \alpha_{\text{shape}}\mathcal{L}_{\text{shape}}. \tag{5.3}$$

We use $\alpha_{\text{MVC}} = 1$, $\alpha_{\text{shape}} = 0.1$ in all experiments.

To optimize the mean value coordinates of the source cage, we penalize negative weight values, which emerge when the source cage is highly concave, self-overlapping, or when some of the shape's points lie outside the cage:

$$\mathcal{L}_{\text{MVC}} = \frac{1}{|\mathcal{C}_s||\mathcal{S}_s|} \sum_{i=1}^{|\mathcal{S}_s|} \sum_{j=1}^{|\mathcal{C}_s|} \left|\min\left(\phi_{ji}, 0\right)\right|^2, \tag{5.4}$$

where α_{MVC} is the loss weight, and ϕ_{ji} denotes the coordinates of $\mathbf{p}_i \in \mathcal{S}_s$ w.r.t. $\mathbf{v}_j \in \mathcal{C}_s$.

$\mathcal{L}_{\text{align}}$ is measured either via chamfer distance in the unsupervised case sans correspondences, or as the L2 distance when supervised with correspondences.

The above two losses drive the deformation towards alignment with the target, but this may come at the price of preferring alignment over feature preservation. Therefore, we add terms that encourage shape preservation. Namely, we draw inspiration from Laplacian regularizers [Gro+18b; Wan+19; Liu+19b], but propose to use a point-to-surface distance as an orientation-invariant, second-order geometric feature. Specifically, for each point $\mathbf{p}$ on the source shape, we fit a PCA plane to a local neighborhood $\mathcal{B}$ (we use the one-ring of the mesh), and then compute the point-to-plane distance as $d = \|\mathbf{n}^T\left(\mathbf{p} - \mathbf{p}_{\mathcal{B}}\right)\|$, where $\mathbf{n}$ denotes the normal of the PCA plane and $\mathbf{p}_{\mathcal{B}} = \frac{1}{|\mathcal{B}|}\sum_{\mathbf{q}\in\mathcal{B}(\mathbf{p})} \mathbf{q}$ is the centroid of the local neighborhood around $\mathbf{p}$. We then penalize change in the distance d_i for each vertex on the surface:

$$\mathcal{L}_{\text{p2f}} = \frac{1}{|\mathcal{S}_s|} \sum_{i=1}^{|\mathcal{S}_s|} \|d_i - d_i'\|^2 \tag{5.5}$$

where d_i' is the distance post deformation. In contrast to the uniform Laplacian, which considers the distance to the centroid and hence yields a non-zero value whenever the local neighborhood is not evenly distributed, the proposed point-to-surface distance better describes the local geometric features.

For man-made shapes, we use two additional losses that leverage priors of this shape class. First, normal consistency is important for, *e.g.*, preserving the planarity of elements like tabletops. To encourage this, we penalize the angular difference of PCA normals before and after deformation:

$$\mathcal{L}_{\text{normal}} = \frac{1}{|\mathcal{S}_s|} \sum_{i}^{|\mathcal{S}_s|} (1 - \mathbf{n}_i^T \mathbf{n}_i'), \tag{5.6}$$

where $\mathbf{n}'$ denotes the PCA-normal after the deformation. As demonstrated later, this normal penalty considerably improves the perceptual quality of the deformation. Second, similarly to Wang *et al.* [Wan+19], we also use the symmetry loss $\mathcal{L}_{\text{symm}}$, measured as the chamfer distance between the shape and its reflection around the $x = 0$ plane. We apply this loss to the deformed shape $\mathcal{S}_{s\to t}$ as well as the cage $\mathcal{C}_s$. Thus, our final shape preservation loss is:

$\mathcal{L}_{\text{shape}} = \mathcal{L}_{\text{p2f}} + \mathcal{L}_{\text{normal}} + \mathcal{L}_{\text{symm}}$ for man-made shapes and $\mathcal{L}_{\text{shape}} = \mathcal{L}_{\text{p2f}}$ for characters.

5.1.2. Applications

We now showcase two applications of the trained cage-based deformation network.

Stock amplification via deformation

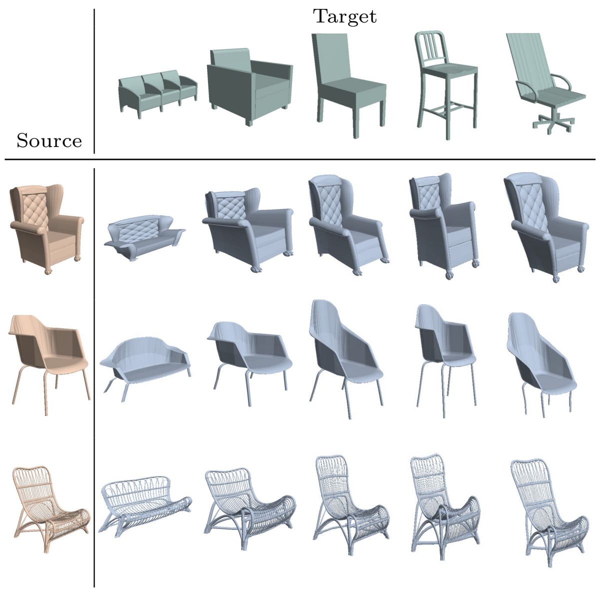


Figure 5.3.: Synthesizing variations of source shapes (brown), by deforming them to match targets (green).

Creating high-quality 3D assets requires significant time, technical expertise, and artistic talent. Once the asset is created, the artist commonly deforms the model to create several variations of it. Inspired by prior techniques on automatic stock amplification [Wan+19], we use our method to learn a meaningful deformation space over a collection of shapes within the same category, and then use random pairs of source and target shapes to synthesize plausible variations of artist-generated assets.

Training details. We train our model on the *chair, car* and *table* categories from ShapeNet [Cha+15] using the same splitting into training and testing sets as in Groueix et al. [Gro+19]. We then randomly sample 100 pairs from the test set. Each shape is normalized to fit in a unit bounding box and is represented by 1024 points.

Variation synthesis examples. Fig 5.3 shows variations generated from various source-target pairs, exhibiting the regularizing power of the cages: even though our training omits all semantic supervision such as part labels, these variations are plausible and do not exhibit feature distortions; fine details, such as chair slats, are preserved.

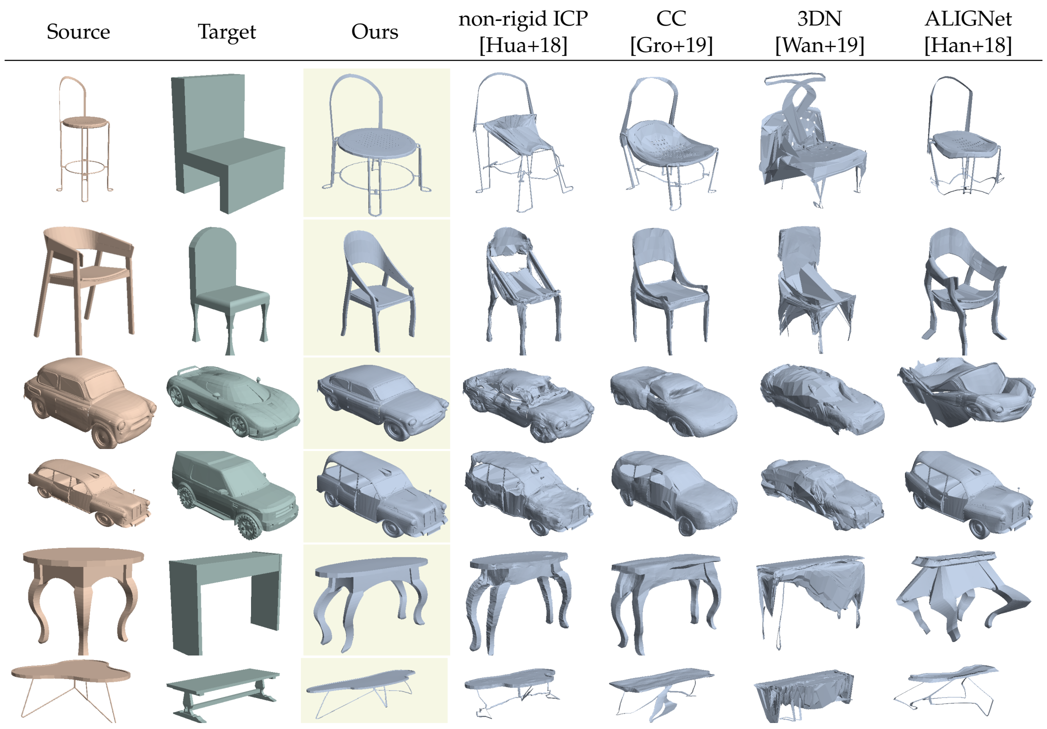


Figure 5.4.: Comparison of our method with other non-homogeneous deformation methods. Our method achieves superior detail preservation of the source shape in comparison to optimization-based [Hua+18] and learning-based [Gro+19; Wan+19; Han+18] techniques, while still aligning the output to the target.

Comparisons. We compared our target-driven deformation method to other methods that strive to achieve the same goal. Results are shown in Fig 5.4. While in many cases alternative techniques do align the deformed shape the target, in all cases they introduce significant artifacts in the deformed meshes.

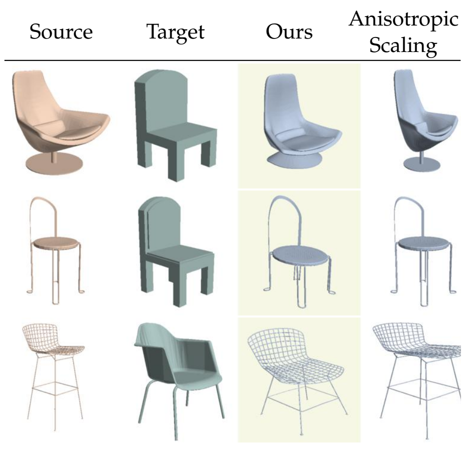


Figure 5.5.: Comparison of our method with anisotropic scaling. Our method better matches corresponding semantic parts.

We first compare to a non-learning-based approach: non-rigid ICP [Hua+18], a classic registration technique that alternates between correspondence estimation and optimization of a non-rigid deformation to best align corresponding points. We show results with the optimal registration parameters we found to achieve detail preservation. Clearly, ICP is sensitive to wrong correspondences that cause convergence to artifact-ridden local minima. We also compare to learning-based methods that directly predict per-point transformations and leverage cycle-consistency (CC) [Gro+19] or

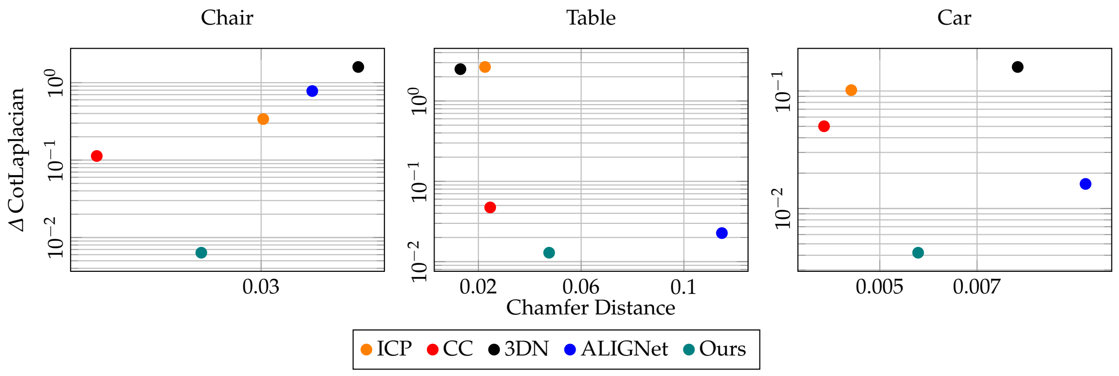


Figure 5.6.: Quantitative evaluation of our method vs alternative methods. Each point represents a method, embedded according to its average alignment error (Chamfer Distance) and distortion (ΔCotLaplacian). Points near the bottom-left corners are better.

feature-preserving regularization (3DN) [Wan+19] to learn low-distortion shape deformations. Both methods blur and omit features, while also creating artifacts by stretching small parts. We also compare to ALIGNet [Han+18], a method that predicts a freeform deformation over a voxel grid, yielding a volumetric deformation of the ambient space similarly to our technique. Contrary to our method, the coarse voxel grid cannot capture the fine deformation of the surface needed to avoid large artifacts. Our training setup is identical to CC, and we retrained 3DN and ALIGNet with the same setup using parameters suggested by the authors.

In Fig 5.5 we compare our results to the simplest of deformation methods – anisotropic scaling, achieved by simply rescaling the source bounding box to match that of the target. While local structure is well preserved, this method cannot account for the different proportion changes required for different regions, highlighting the necessary intricacy of the optimal deformation in this case.

Quantitative comparisons. in Fig 5.6, we quantitatively evaluate the various methods using two metrics: distance to the target shape, and detail preservation, measured via chamfer distance (computed over a dense set of 5000 uniformly sampled points) and difference in cotangent Laplacians, respectively. Note that these metrics do not favor any method, since all optimize for a variant of chamfer distance, and none of the methods optimize for the difference in the cotangent Laplacian. Each 2D point in the figure represents one method, with the point's coordinates prescribed with respect to the two metrics, the origin being ideal. This figure confirms our qualitative observations: our method is more effective at shape preservation than most alternatives while still capturing the gross structure of the target.

Using images as targets. Often, a 3D target is not readily available. Images

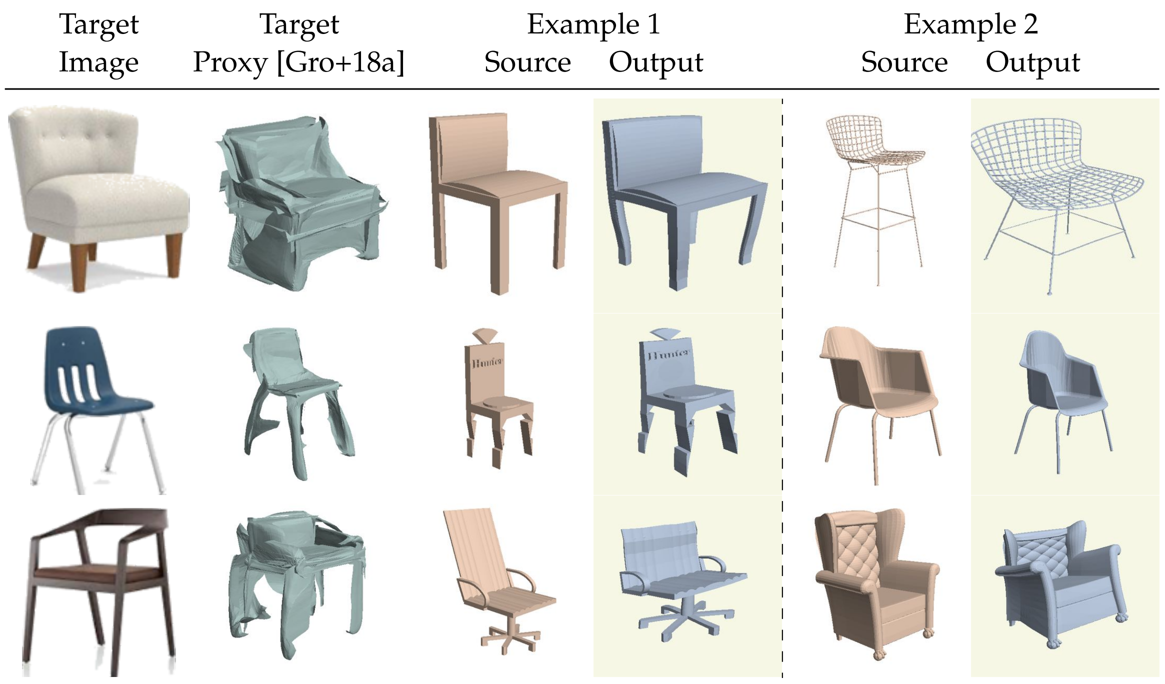


Figure 5.7.: We use our method to deform a 3D shape to match a real 2D image. We first use AtlasNet [Gro+18a] to reconstruct a 3D proxy target. Despite the poor quality of the proxy, it still serves as a valid target for our network to generate a matching output preserving the fine details of the source.

are more abundant and much easier to acquire, and thus pose an appealing alternative. We use a learning-based single-view reconstruction technique to create a proxy target to use with our method to find appropriate deformation parameters. We use publicly available product images of real objects and execute AtlasNet's SVR reconstruction [Gro+18a] to generate a coarse 3D proxy as a target. Fig 5.7 shows that even though the proxy has coarse geometry and many artifacts, these issues do not affect the deformation, and the result is still a valid variation of the source.

Deformation transfer

Given a novel 3D model, it is much more time-efficient to automatically deform it to mimic an existing example deformation, than having an artist deform the novel model directly. This automatic task is called *deformation transfer*. The example deformation is given via a model in a rest pose $\mathcal{S}_s$, and a model in the deformed pose $\mathcal{S}_t$. The novel 3D model is given in a corresponding rest post $\mathcal{S}_{s'}$. The goal is to deform the novel model to a position $\mathcal{S}_{t'}$ so that the deformation $\mathcal{S}_{s'} \rightarrow \mathcal{S}_{t'}$ is analogous to $\mathcal{S}_s \rightarrow \mathcal{S}_t$. This task can be quite challenging, as the example deformation $\mathcal{S}_t$ may have very different geometry, or even come from an ad-hoc scan, and thus dense

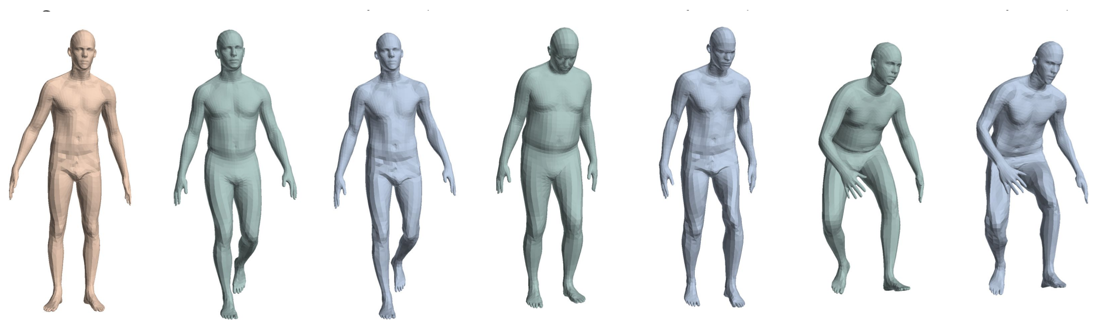


Figure 5.8.: The deformation model, trained to deform a fixed source (left) to various articulations.

correspondences between $\mathcal{S}_s$ and $\mathcal{S}_t$ are unavailable, preventing the use of traditional mesh optimization techniques such as [SP04]. Furthermore, as the novel character $\mathcal{S}_{s'}$ may be significantly different from all models observed during training, it is impossible to a-priori learn a deformation subspace for $\mathcal{S}_{s'}$ unless sufficient pose variations of $\mathcal{S}_{s'}$ is available, as in Gao *et al*. [Gao+18].

We demonstrate that our learning-based approach can be used to perform deformation transfer on arbitrary humanoid models. The network infers the deformation from the source $\mathcal{S}_s$ to the target $\mathcal{S}_t$, without any given correspondences, and then an optimization-based method transfers this deformation to a novel shape $\mathcal{S}_{s'}$ to obtain the desired deformation $\mathcal{S}_{t'}$. Hence, given *any* arbitrarily-complex novel character, all our method requires are sparse correspondences supplying the necessary alignment between the two rest poses, $\mathcal{S}_s$ and $\mathcal{S}_{s'}$. We now overview the details of our learned cage-based human deformation model and the optimization technique used to transfer the deformations.

Learning cage-based human deformation. To train our human-specific deformation model, we use the dataset [Gro+18b] generated using the SMPL model [Bog+14] of 230K models of various humans in various poses. Since our application assumes that the exemplar deformation is produced from a single canonical character, we picked one human in the dataset to serve as $\mathcal{S}_s$. Subsequently, since we only have one static source shape $\mathcal{S}_s$, we use a *static* cage $\mathcal{C}_s$ manually created with 77 vertices, and hence do not need the cage prediction network $\mathcal{N}_c$ and only use the deformation network $\mathcal{N}_d$. We train $\mathcal{N}_d$ to deform the static $\mathcal{S}_s$ using the static $\mathcal{C}_s$ into exemplars $\mathcal{S}_t$ from the dataset (with targets not necessarily stemming from the same humanoid model as $\mathcal{S}_s$). We then train with the loss in Eq. (5.3), but with one modification: in similar fashion to prior work, during training we use ground truth correspondences and hence replace the chamfer distance with the L2 distance w.r.t the known correspondences. Note that these correspondences are *not* used at inference time.

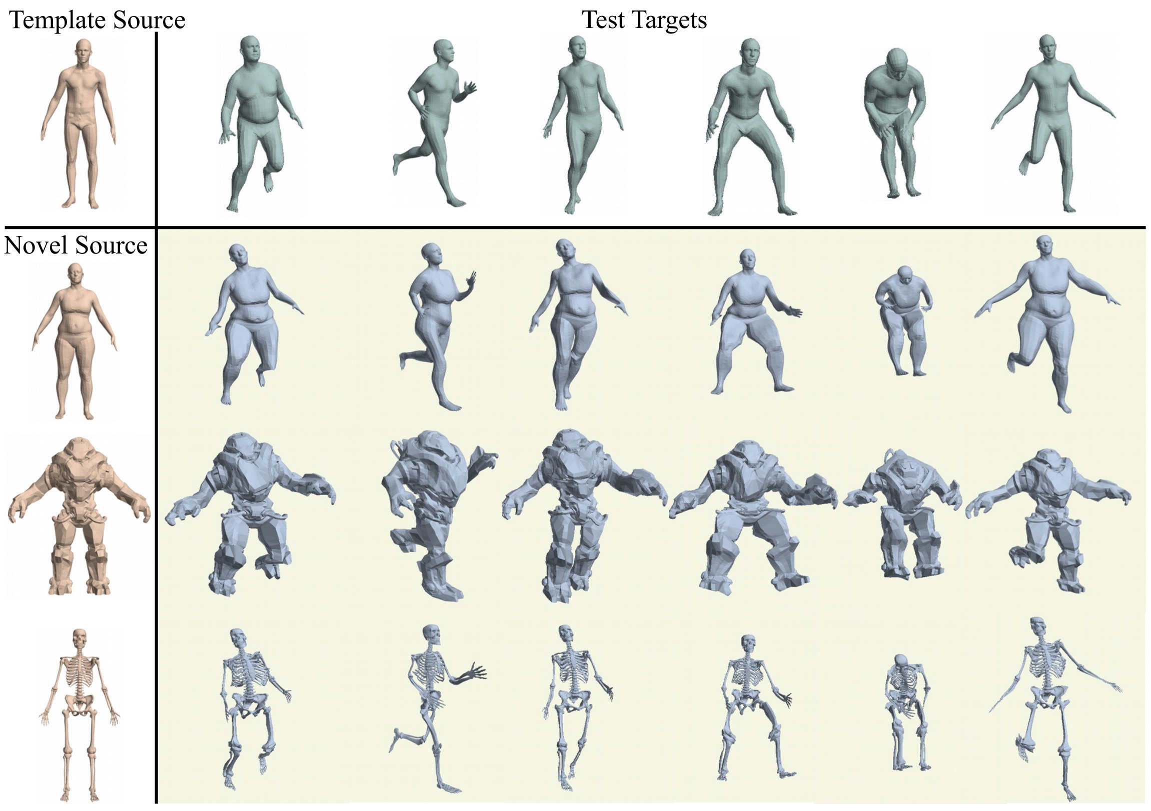


Figure 5.9.: Deformation transfer. We first learn the cage deformation space for a template source shape (top left) with known pose and body shape variations. Then, we annotate predefined landmarks on new characters in neutral poses (left column, rows 2-4). At test time, given novel target poses (top row, green) without known correspondences to the template, we transfer their poses to the other characters (blue).

Lastly, during training we also optimize the static source cage $\mathcal{C}_c$ by treating its vertices as degrees of freedom and directly optimizing them to reduce the loss so as to attain a more optimal, but still static cage after training.

Fig 5.8 shows examples of human-specific cage deformations predicted for test targets (not observed while training). Note how our model successfully matches poses even without knowing correspondences at inference time, while preserving fine geometric details such as faces and fingers.

Transferring cage deformations. After training, we have at our disposal the deformation network $\mathcal{N}_d$ and the static $\mathcal{C}_s, \mathcal{S}_s$. We assume to be given a novel character $\mathcal{S}_{s'}$ with 83 landmark correspondences aligning it to $\mathcal{S}_s$, and an example target pose $\mathcal{S}_t$. Our goal is to deform $\mathcal{S}_{s'}$ into a new pose $\mathcal{S}_{t'}$ that is analogous to the deformation of $\mathcal{S}_s$ into $\mathcal{S}_t$.

We first generate a new cage $\mathcal{C}_{s'}$ for the character $\mathcal{S}_{s'}$. Instead of a network-based prediction, we simply optimize the static cage $\mathcal{C}_s$, trying to match mean value coordinates between corresponding points of $\mathcal{S}_s, \mathcal{S}_{s'}$:

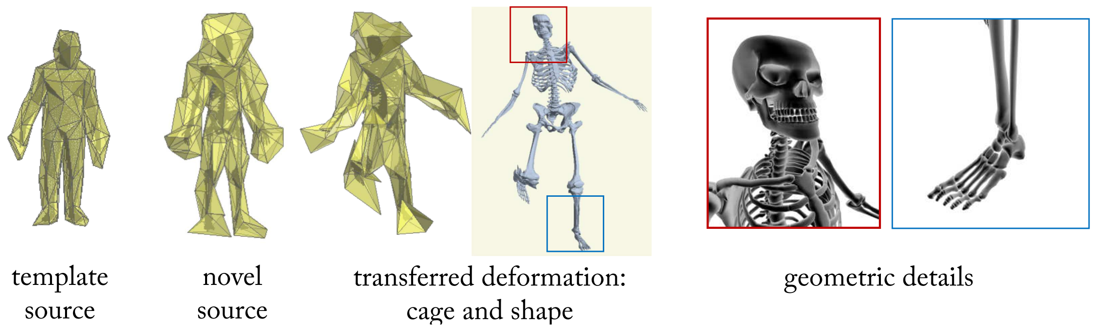


Figure 5.10.: In deformation transfer, the manually created cage for a template shape (leftmost) is fitted to a novel source shape (second left) by optimizing MVC of a sparse set of aligned landmarks. The learnt deformation can be directly applied to the fitted source cage (columns 3-4), preserving rich geometric features (right).

$$\mathcal{L}_{\text{consistency}} = \sum_j \sum_{(p,q)} \|\phi_j^{\mathcal{C}_s}(p) - \phi_j^{\mathcal{C}_s'}(q)\|^2 \tag{5.7}$$

where (p,q) are corresponding landmarks. We also regularize with respect to the cotangent Laplacian of the cage:

$$\mathcal{L}_{\mathcal{C}\text{lap}} = \sum_{0 \le j < |\mathcal{C}_s|} \left(\|L_{\cot}\mathbf{v}_j\| - \|L_{\cot}\mathbf{v}_j'\|\right)^2. \tag{5.8}$$

Then, we compute $\mathcal{C}_{s'}$ by minimizing $\mathcal{L} = \mathcal{L}_{\text{consistency}} + 0.05\mathcal{L}_{\mathcal{C}\text{lap}}$, with $\mathcal{C}_s$ used as initialization, solved via the Adam optimizer with step size $5 \cdot 10^{-4}$ and up to 10^4 iterations (or until $\mathcal{L}_{\text{consistency}} < 10^{-5}$).

Finally, given the cage $\mathcal{C}_{s'}$ for the novel character, we compute the deformed cage $\mathcal{C}_{s' \to t'}$, using our trained deformation network, by applying the predicted offset to the optimized cage: $\mathcal{C}_{s' \to t'} = \mathcal{N}_d\left(\mathcal{S}_t, \mathcal{S}_{s'}\right) + \mathcal{C}_{s'}$. The final deformed shape $\mathcal{S}_{t'}$ is computed by deforming $\mathcal{S}_{s'}$ using the cage $\mathcal{C}_{s' \to t'}$ via Eq. (5.1). This procedure is illustrated in Fig 5.10, while more examples can be found in the supplemental material. Due to the agnostic nature of cage-deformations to the underlying shape, we are able to seamlessly combine machine learning and traditional geometry processing to generalize to never-observed characters. To demonstrate the expressiveness of our method, we show examples on extremely dissimilar target characters in Figures 5.1 and 5.9.

5.1.3. Evaluation

In this section, we study the effects and necessity of the most relevant components of our methods. To measure the matching error we use chamfer distance computed on 5000 uniformly resampled points, and to measure the feature distortion we use the distance between cotangent Laplacians. All models are normalized to a unit bounding box.

Benefit of learning CBD from data. Instead of learning the CBD from a

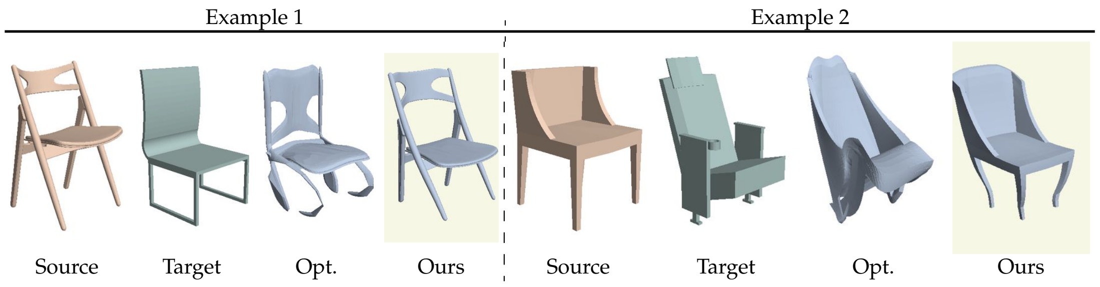
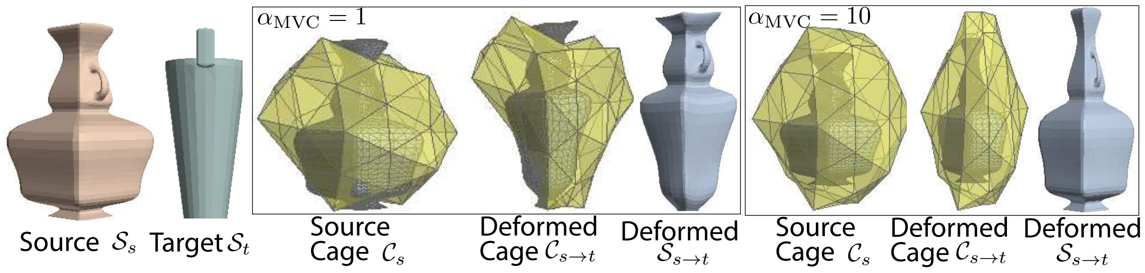


Figure 5.11.: Our approach produces more plausible inter-shape correspondences and deformations than per-pair optimization.



$\alpha_{\text{MVC}} = 1$
$\alpha_{\text{MVC}} = 10$
Source $\mathcal{S}_s$
Target $\mathcal{S}_t$
Source Cage $\mathcal{C}_s$
Deformed Cage $\mathcal{C}_{s\to t}$
Deformed $\mathcal{S}_{s\to t}$
Source Cage $\mathcal{C}_s$
Deformed Cage $\mathcal{C}_{s\to t}$
Deformed $\mathcal{S}_{s\to t}$

Figure 5.12.: Effect of $\mathcal{L}_{\text{MVC}}$. Higher regularization yields more conservative deformations.

collection of data, one could minimize Eq. (5.3) for a single pair of shapes, which is essentially a non-rigid Iterative-Closest-Point (ICP) parameterised by cage vertices. As shown in Fig 5.11, when correct correspondence estimation becomes challenging, the optimization alternative produces non-plausible outputs. In contrast, the learnt approach utilizes domain knowledge embedded in the network's parameters [ZF14; SDM19], amounting to better reasoning about the plausibility of inter-shape correspondences and deformations. The learned domain knowledge can generalize to new data. As demonstrated in Sec 5.1.2, even though our network is trained with ground-truth correspondences, it is able to automatically associate the source shape to a new target *without correspondences* during inference, while optimization methods require accurate correspondence estimation for every new target.

Effect of the negative MVC penalty, $\mathcal{L}_{\text{MVC}}$. in Fig 5.12 we show the effect of penalizing negative mean value coordinates. We train our architecture on 300 vase shapes from COSEG [Wu+14], while varying the weight $\alpha_{\text{MVC}} \in \{0, 1, 10\}$. Increasing this term brings the cages closer to the shapes' convex hulls, leading to more conservative deformations. Quantitative results in Table 5.1a also suggest that increasing the weight α_{MVC} favors shape preservation over alignment accuracy. Completely eliminating this term hurts convergence, and increases the alignment error further.

Effect of the shape preservation losses, $\mathcal{L}_{\text{shape}}$. In Fig 5.13 we compare deformations produced with the full loss ($\mathcal{L}_{\text{shape}} = \mathcal{L}_{\text{p2f}} + \mathcal{L}_{\text{normal}} + \mathcal{L}_{\text{symm}}$)

Ablation	CD	ΔCotLaplacian
$\alpha_{\text{MVC}} = 0$	1.64	9.04
$\alpha_{\text{MVC}} = 1$	1.44	8.74
$\alpha_{\text{MVC}} = 10$	2.65	8.27
(a) Effect of the MVC loss, $\mathcal{L}_{\text{MVC}}$		
$\mathcal{L}_{\text{shape}} = \mathcal{L}_{\text{lap}} + \mathcal{L}_{\text{symm}}$	5.16	4.75
$\mathcal{L}_{\text{shape}} = \mathcal{L}_{\text{p2f}} + \mathcal{L}_{\text{symm}}$	4.86	4.70
$\mathcal{L}_{\text{shape}} = \mathcal{L}_{\text{normal}} + \mathcal{L}_{\text{symm}}$	5.45	4.33
(b) Effect of the shape preservation losses, $\mathcal{L}_{\text{shape}}$		
$\mathcal{N}_c$=Identity	3.27	5.65
$\mathcal{N}_c$=Source-invariant	3.11	12.05
$\mathcal{N}_c$=Ours	3.06	10.45
(c) Design choices for cage prediction network, $\mathcal{N}_c$		

Table 5.1.: We evaluate effect of different losses ($\mathcal{L}_{\text{MVC}}, \mathcal{L}_{\text{shape}}$) and components ($\mathcal{N}_c$) of our pipeline with respect to chamfer distance (CD, scaled by 10^2) and cotangent Laplacian (scaled by 10^3).

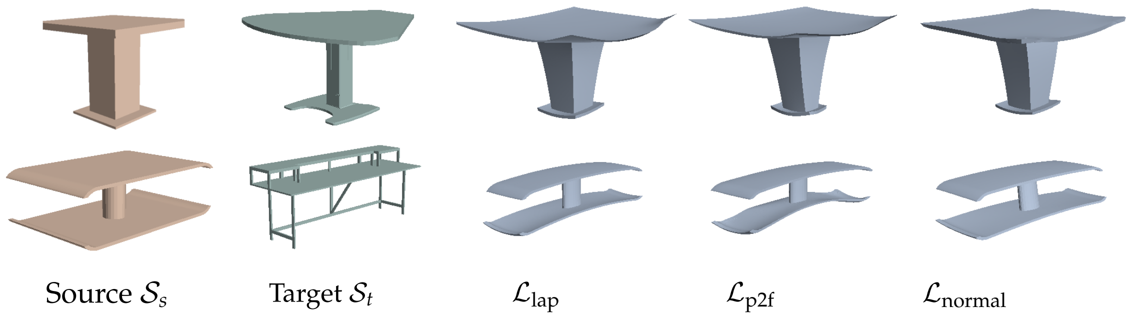
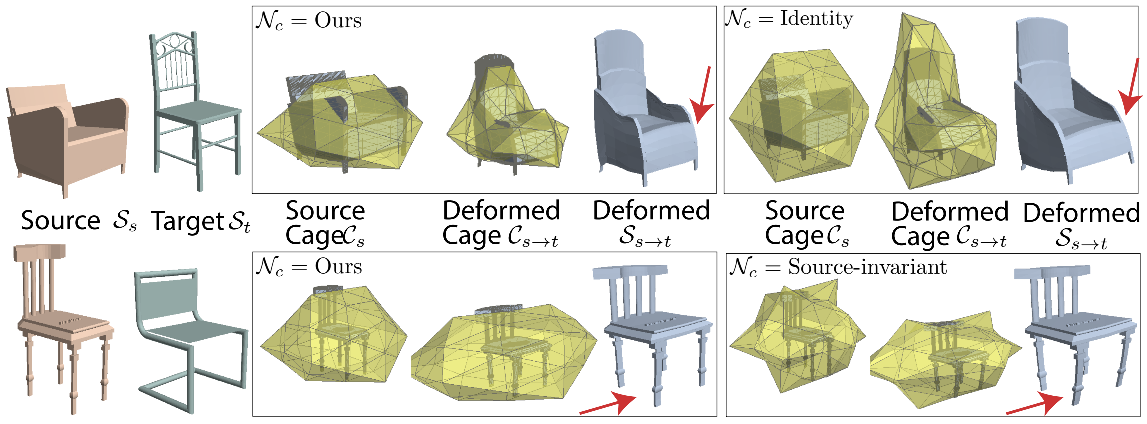

Source $\mathcal{S}_s$
Target $\mathcal{S}_t$
$\mathcal{L}_{\text{lap}}$
$\mathcal{L}_{\text{p2f}}$
$\mathcal{L}_{\text{normal}}$

Figure 5.13.: The effect of different shape preservation losses, note that all results include $\mathcal{L}_{\text{symm}}$.

to ones produced with only one of the first two loss terms. While we did not use the Laplacian regularizer $\mathcal{L}_{\text{lap}}$ as in [Wan+19], it seems to have an effect equivalent to $\mathcal{L}_{\text{p2f}}$. As expected, $\mathcal{L}_{\text{normal}}$ prevents bending of rigid shapes. We quantitatively evaluate these regularizers in Table 5.1b, which suggests that $\mathcal{L}_{\text{p2f}}$ is slightly better as the deformed shape is more aligned with the target than $\mathcal{L}_{\text{lap}}$, even though shape preservation has not been sacrificed. $\mathcal{L}_{\text{normal}}$ reduces distortion even further.

Design choices for the cage prediction network, $\mathcal{N}_c$. The cage prediction network $\mathcal{N}_c$ morphs the template cage mesh (a 42-vertex sphere) into the initial cage enveloping the source shape. In Fig 5.14 and Table 5.1c we compare to two alternative design choices for this module: an *Identity* module retains the template cage, and a *source-invariant* module in which we optimize the template cage's vertex coordinates with respect to all targets in the dataset,



$\mathcal{N}_c$ = Ours
$\mathcal{N}_c$ = Identity
Source $\mathcal{S}_s$
Target $\mathcal{S}_t$
Source Cage $\mathcal{C}_s$
Deformed Cage $\mathcal{C}_{s\to t}$
Deformed $\mathcal{S}_{s\to t}$
Source Cage $\mathcal{C}_s$
Deformed Cage $\mathcal{C}_{s\to t}$
Deformed $\mathcal{S}_{s\to t}$
$\mathcal{N}_c$ = Ours
$\mathcal{N}_c$ = Source-invariant

Figure 5.14.: The effect of source-cage prediction. We compare our per-instance prediction of $\mathcal{N}_c$ with (1) a static spherical cage (top right) and (2) a single optimized cage prediction over the entire training set (bottom right). Our approach achieves better alignment with the target shape.

but then use the same fixed cage for testing. Learning source-specific cages produces deformations closest to the target with minimum detail sacrifice. As expected, fixing the template cage produces more rigid deformations, yielding lower distortion at the price of less-aligned results.

5.2. Concluding remarks.

Contributions. We show that classical cage-based deformation provides a low-dimensional, detail-preserving deformation space directly usable in a deep-learning setting. We implement cage weight computation and cage-based deformation as differentiable network layers, which could be used in other architectures. Our method succeeds in generating feature-preserving deformations for synthesizing shape variations and deformation transfer, and better preserves salient geometric features than competing methods.

Limitations. One limitation of our approach is that our losses are not quite sufficient to always ensure rectilinear/planar/parallel structures in man-made shapes are perfectly preserved (Fig 5.13).

Also, we would like to incorporate alternative cage weight computation layers, such as Green Coordinates [LLCO08]. Unlike MVC, this technique is not affine-invariant, and thus would introduce less affine distortion for large articulations (see the second row fourth column in Fig 5.9).

CHAPTER 6

Conclusion

The momentous trend of reality digitization has made geometry processing a ubiquitous and crucial technological subject. A general geometry processing pipeline encapsulates a line of techniques to create and modify 3D content, which are principally divided into raw data processing, surface reconstruction, and shape modeling. This thesis investigated all the above sub-domains and presented a set of novel solutions to various relevant geometry processing tasks, which not only pushed forward the state-of-the-art performance of the task in question, but also offered brand-new use-cases. In particular, the contributions made in the thesis belong to the pioneering effort to revolutionize geometry processing by incorporating emerging and evolving toolkits from the machine learning community. In all five of the algorithms we presented in the thesis, we have developed paradigms that either directly improve the way deep learning was deployed, or created new possibilities to utilize deep learning for the task at hand. The incorporation of deep learning is motivated by its ability to discover domain knowledge, which is typically impossible to rigorously define otherwise. The result, as we have demonstrated in this thesis, is consistent improvements on the output quality and especially notable advantages when the input is under-constrained, *e.g.* noise-ridden, undersampled or unaligned. The use of neural networks also relieves the system from dense user inputs, thus paving the way for autonomous and large-scale deployment, which is essential for the ongoing transition to digitized reality.

6.1. Recapitulation of core contributions

Raw data processing and enhancement. In Chapter 3, we introduced two algorithms to process and enhance point clouds. Point cloud processing and enhancing has particular significance in many in-demand industries such as robotics and autonomous driving, since point clouds are the major output form for most industrial 3D acquisition devices. The first algorithm, introduced in Sec. 3.1, tackles a common problem in acquired point clouds - undersampling, which leads to lost or distorted geometric features in the downstream tasks. Previous filtering-based approaches is limited to fitting local point distributions, hence do not handle very sparse inputs well. We presented a deep neural network to upsample point clouds. It circumvents the difficulty of handling very sparse inputs using a patch-based progressive multi-scale neural network, which dedicates a chain of specialized subnet to upsample the point cloud in each progressive step. It brought considerable improvement compared to previous state-of-the-art methods, as the multi-stage network is able to learn scale-variant features, and – through the interconnections between the subnets and end-to-end training – simultaneous optimization across different scales.

In Sec. 3.2, we presented the first differentiable point renderer, which provides a way to change the attributes of point clouds, *e.g.* point positions, normals and colors, through image-level updates. This seemingly indirectly point processing approach provides means to utilize a vast pool of image processing techniques including, importantly, a plethora of tremendously successful neural techniques. In this thesis, we demonstrated in a point denoising task how a generative image-to-image translation network can be used hand in hand with the proposed differentiable renderer to produce more geometric details in the denoised point clouds. Given the relative maturity of neural image processing and practically unlimited data resource, it is certain that many challenging tasks in the 3D domain will benefit from differentiable rendering methods like the one proposed in this thesis.

Surface reconstruction. In Chapter 4 we delved into implicit surface reconstruction, which is the most widely adopted approach to convert 3D acquisition, *e.g.* volumetric or point clouds, to polygon meshes for further processing and modeling tasks. Recently, neural networks have emerged as a powerful tool to approximate implicit surface functions, which are traditionally modeled as a series of polynomial functions. We offered two algorithms that improve the state of this highly active research field. In Sec. 4.1, we introduced a new formulation of implicit surface functions to improve the representation of geometric details. Named implicit displacement field, this new formulation draws inspiration from displacement mapping, a classic method from surface deformation, to represent the surface details as an offset from a base smooth shape. Guided by the mathematical formulation, we proposed an elegantly simple neural network design that inherently supports the base/detail separation. The result is a novel neural implicit representation that demonstrates superior representational power for geometric details while being extremely memory efficient and resilient to common optimization difficulties. Building upon the base/detail separation, we demonstrated how the proposed approach can be extended for detail transfer application. Compared to previous methods, it can transfer spatially-variant high-resolution geometric details without parameterization or correspondences, with a trade-off that the input shapes should be aligned.

While implicit displacement field addresses the surface representation itself, in Sec. 4.2 we focused on the optimization of existing popular neural implicit functions when the training data is imperfect. We demonstrated that the quality and the speed for learning neural implicit functions can benefit from an explicit representation of the shape under optimization. The challenge, however, is to extract such an explicit representation efficiently and accurately on-the-fly, since popular existing surface extraction methods, such as Marching Cubes, are slow and hence unsuited for our online application. Thereby we proposed a method to extract dense uniformly distributed point clouds, called iso-points, on the iso-surface of the implicit function. We demonstrated that iso-points can boost the performance and quality of the optimization in various surface reconstruction applications using different input modalities, such as multiview images and noisy point clouds, through shape-aware sampling and regularizers.

Shape manipulation. In Chapter 5, we visited one of the most important subjects in shape modeling, shape deformation. In particular, we presented a novel algorithm for exemplar-based deformation. Given two arbitrary shapes, it deforms one to match the other without correspondence information while, importantly, preserving the geometric details of the original shape. Since human inputs are spared, exemplar-based deformation methods find applications in many large-scale modeling tasks such as character posing and shape generation via variation. The core of the method proposed in this thesis is a neural network that operates on a low dimension deformation space based on cage-based deformation from classic shape modeling. It addresses two critical shortcomings of previous approaches. One one hand, by learning to align random pairs of a specific shape category, the neural network essentially learns to estimate semantic correspondences. This is extremely challenging with conventional optimization-based learning approaches, since the two shapes may be semantically compatible yet drastically dissimilar geometrically and topologically. Furthermore, by having a novel network design that performs cage-based deformation, it effectively reduces feature distortions, and preserves surface details and complex topologies with constant computation. The efficacy of the cage-based neural deformation method was demonstrated in shape generation and deformation transfer applications.

6.2. Future work

Neural geometry processing is an expansive venue for many future research directions.

Dynamic point clouds. The modeling of dynamic point clouds is a highly valuable yet relatively less explored subject. Development of this field will have a pivotal impact in extended fields such as robotics. The analysis and processing of dynamic point clouds may benefit from additional input modalities, such as videos. Recently, researches in artificial general intelligence have shown promising advances in multi-model learning, where multiple input modalities are jointly analyzed and modelled in a common latent space. Adopting similar ideas for dynamic point cloud processing is an extremely stimulating direction.

Neural implicit surface editing Although implicit surface representations have many compelling advantages over the polygon meshes as stated in the earlier chapters, they can not yet be sufficiently utilized in shape modeling, where polygon meshes are still indisputably the most adopted representation. Therefore, developing a paradigm to directly model and edit the implicit surfaces is a relevant topic for the community. Transferable IDF and some other recent works such as DualSDF [Hao+20] can be regarded as an initial step in this direction. Yet much remains to be accomplished to make implicit representations really compatible for shape modeling. In this regard, it is crucial to improve the interpretability and controllability of existing neural representations so as to provide tangible "handles" for more intuitive and targeted editing. This may be addressed by learning a structured latent space in a generative model analog to controllable image translation, or alternatively via a tightly coupled implicit-explicit hybrid representation, where the explicit representation can actively and directly alter its implicit counterpart (whereas in the iso-points project, iso-points assert influence over the implicit functions indiretly via sampling and regularization of the optimization process). At the same time, when considering potential integration to interactive shape modeling, real-time response is one important factor, which may be addressed with more modular factorizations of the implicit representation.

APPENDIX A

Appendix

A.1. Proof for Sec. 4.1.1

Theorem 1. *If function $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable, Lipschitz-continuous with constant L and Lipschitz-smooth with constant M, then $\|\nabla f(\mathbf{x} + \delta \nabla f(\mathbf{x})) - \nabla f(\mathbf{x})\| \le |\delta| LM$.*

Proof. If a differentiable function f is Lipschitz-continuous with constant L and Lipschitz-smooth with constant M, then

$$\|\nabla f(\mathbf{x})\| \le L \tag{A.1}$$

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \le M\|\mathbf{x} - \mathbf{y}\|. \tag{A.2}$$

$$\begin{aligned} \|\nabla f(\mathbf{x} + \delta \nabla f(\mathbf{x})) - \nabla f(\mathbf{x})\| &\le M\|\delta \nabla f(\mathbf{x})\| && \text{by (A.2)} \\ &\le |\delta| LM && \text{by (A.1)} \end{aligned}$$

□

Corollary 1. *If a signed distance function f satisfying the eikonal equation up to error $\epsilon > 0$, $\big|\|\nabla f\| - 1\big| < \epsilon$, is Lipschitz-smooth with constant M, then $\|\nabla f(\mathbf{x} + \delta \nabla f(\mathbf{x})) - \nabla f(\mathbf{x})\| < (1 + \epsilon)|\delta| M$.*

Proof. $\big|\|\nabla f\| - 1\big| < \epsilon \;\Rightarrow\; \|\nabla f\| < \epsilon + 1$. This means f is Lipschitz-continuous with constant $\epsilon + 1$. Then by Theorem 1, $\|\nabla f(\mathbf{x} + \delta \nabla f(\mathbf{x})) - \nabla f(\mathbf{x})\| < |\delta|(1 + \epsilon)M$. □

Finally we show the upper bound for the normalized gradient, *i.e.*,

$$\|\hat{\mathbf{n}} - \mathbf{n}\| \leq \frac{1+\epsilon}{1-\epsilon} |\delta| \, M,$$

where $\mathbf{n} = \frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}$, $\hat{\mathbf{n}} = \frac{\nabla f(\hat{\mathbf{x}})}{\|\nabla f(\hat{\mathbf{x}})\|}$ and $\hat{\mathbf{x}} = \mathbf{x} + d(\mathbf{x})\,\mathbf{n}$ with $d(\mathbf{x})$ denoting the small displacement.

Proof.

$$\|\hat{\mathbf{n}} - \mathbf{n}\| = \left\| \frac{\nabla f(\hat{\mathbf{x}})}{\|\nabla f(\hat{\mathbf{x}})\|} - \frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|} \right\|. \tag{A.3}$$

For brevity, we denote $\nabla f(\hat{\mathbf{x}})$ and $\nabla f(\mathbf{x})$ as $\mathbf{u}$ and $\mathbf{v}$. Without loss of generality, we assume $\|\mathbf{u}\| \leq \|\mathbf{v}\|$. Then

$$\|\hat{\mathbf{n}} - \mathbf{n}\| = \left\| \frac{\mathbf{u}}{\|\mathbf{u}\|} - \frac{\mathbf{v}}{\|\mathbf{v}\|} \right\| \tag{A.4}$$

$$\overset{(*)}{\leq} \frac{1}{\|\mathbf{u}\|} \|\mathbf{u} - \mathbf{v}\| \tag{4.6}$$

$$\leq \frac{1}{1-\epsilon} \|\mathbf{u} - \mathbf{v}\| \qquad \text{by Eikonal constraint} \tag{A.5}$$

$$= \frac{1}{1-\epsilon} \|\nabla f(\hat{\mathbf{x}}) - \nabla f(\mathbf{x})\| \tag{A.6}$$

$$= \frac{1}{1-\epsilon} \left\| \nabla f\left(\mathbf{x} + \frac{d(\mathbf{x})\,\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}\right) - \nabla f(\mathbf{x}) \right\|. \tag{A.7}$$

Since $|d(\mathbf{x})|$ is a small and $\|\nabla f(\mathbf{x})\|$ is close to 1, we can set $\delta = \dfrac{d(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}$. Thereby using Corrolary 1, we conclude

$$\frac{1}{1-\epsilon} \left\| \nabla f\left(\mathbf{x} + \frac{d(\mathbf{x})\,\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}\right) - \nabla f(\mathbf{x}) \right\| \leq \frac{1+\epsilon}{1-\epsilon} |\delta| M, \tag{A.8}$$

thus

$$\|\hat{\mathbf{n}} - \mathbf{n}\| \leq \frac{1+\epsilon}{1-\epsilon} |\delta| M. \tag{A.9}$$

Eq. (4.6) can be proved as follows

$$\left\| \overbrace{\frac{\mathbf{u}}{\|\mathbf{u}\|} - \frac{\mathbf{v}}{\|\mathbf{v}\|}}^{\mathbf{d}} \right\| = \left\| \overbrace{\frac{\mathbf{u}}{\|\mathbf{u}\|} - \frac{\mathbf{v}}{\|\mathbf{u}\|}}^{\mathbf{d}'} + \overbrace{\left(\frac{\mathbf{v}}{\|\mathbf{u}\|} - \frac{\mathbf{v}}{\|\mathbf{v}\|}\right)}^{\mathbf{e}} \right\|, \tag{A.10}$$

which depicts the distance of the unit sphere projections of $\mathbf{u}$ and $\mathbf{v}$. Obviously, as shown in Figure A.1, $\|\mathbf{d}\| \leq \|\mathbf{d}'\|$ if $\sphericalangle(\mathbf{d}, \mathbf{e}) \geq 90°$.

Since $\mathbf{e} = (\frac{1}{\|\mathbf{u}\|} - \frac{1}{\|\mathbf{v}\|})\mathbf{v}$ and $(\frac{1}{\|\mathbf{u}\|} - \frac{1}{\|\mathbf{v}\|}) \geq 0)$, to show that $\sphericalangle\langle \mathbf{d}, \mathbf{e} \rangle \geq 90°$ is the same as to show that $\sphericalangle\langle \mathbf{d}, \mathbf{v} \rangle \geq 90°$. Indeed:

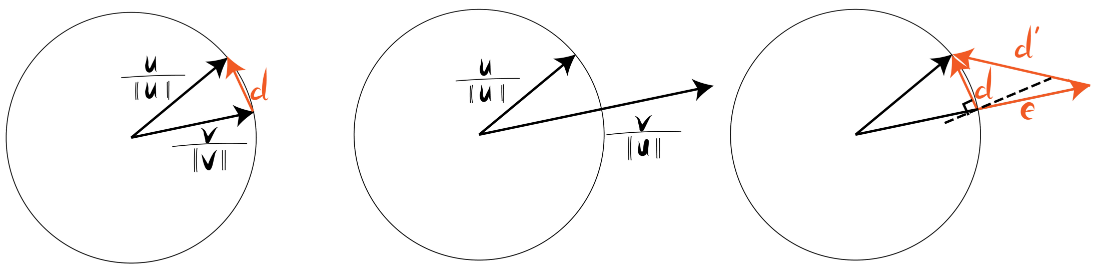


Figure A.1.: Sketch for proof.

$$\langle \mathbf{d}, \mathbf{v} \rangle = \left\langle \frac{\mathbf{u}}{\|\mathbf{u}\|} - \frac{\mathbf{v}}{\|\mathbf{v}\|}, \mathbf{v} \right\rangle \tag{A.11}$$

$$= \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{u}\|} - \|\mathbf{v}\| \tag{A.12}$$

$$\leq \frac{\|\mathbf{u}\|\|\mathbf{v}\|}{\|\mathbf{u}\|} - \|\mathbf{v}\| \qquad \text{by Cauchy-Schwarz inequality} \tag{A.13}$$

$$= 0 \tag{A.14}$$

□

Bibliography

[Aan+16] Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. "Large-Scale Data for Multiple-View Stereopsis". In: *International Journal of Computer Vision* (2016), pp. 1–16 (cit. on p. 93).

[Ach+12] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. "SLIC superpixels compared to state-of-the-art superpixel methods". In: *IEEE transactions on pattern analysis and machine intelligence* 34.11 (2012), pp. 2274–2282 (cit. on p. 59).

[Ach+18] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. "Learning Representations and Generative Models for 3D Point Clouds". In: *Proc. Int. Conf. on Machine Learning* (2018) (cit. on pp. 16, 18).

[AG21] animatico AG. *animatico*. 2021. URL: `https://animati.co/` (visited on 07/01/2021) (cit. on p. 1).

[AL20a] Matan Atzmon and Yaron Lipman. "Sal: Sign agnostic learning of shapes from raw data". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2020, pp. 2565–2574 (cit. on pp. 21, 22).

[AL20b] Matan Atzmon and Yaron Lipman. *SALD: Sign Agnostic Learning with Derivatives*. 2020. arXiv: `2006.05400 [cs.CV]` (cit. on p. 22).

[Ale+03] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T Silva. "Computing and rendering point set surfaces". In: *IEEE Trans. Visualization & Computer Graphics* 9.1 (2003), pp. 3–15 (cit. on p. 18).

[AML18a] Matan Atzmon, Haggai Maron, and Yaron Lipman. "Point convolutional neural networks by extension operators". In: *arXiv preprint arXiv:1803.10091* (2018) (cit. on p. 16).

[AML18b] Matan Atzmon, Haggai Maron, and Yaron Lipman. "Point Convolutional Neural Networks by Extension Operators". In: *ACM Trans. on Graphics (Proc. of SIGGRAPH)* (2018) (cit. on p. 28).

[Atz+19] Matan Atzmon, Niv Haim, Lior Yariv, Ofer Israelov, Haggai Maron, and Yaron Lipman. "Controlling neural level sets". In: *Proc. IEEE Int. Conf. on Neural Information Processing Systems (NeurIPS)*. 2019, pp. 2034–2043 (cit. on pp. 23, 84).

[Azi+19] Dejan Azinovic, Tzu-Mao Li, Anton Kaplanyan, and Matthias Nießner. "Inverse path tracing for joint material and lighting estimation". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2019, pp. 2447–2456 (cit. on p. 20).

[Bas+20] Ronen Basri, Meirav Galun, Amnon Geifman, David Jacobs, Yoni Kasten, and Shira Kritchman. "Frequency bias in neural networks for input of non-uniform density". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 685–694 (cit. on p. 21).

[BBH08] Derek Bradley, Tamy Boubekeur, and Wolfgang Heidrich. "Accurate multi-view reconstruction using robust binocular stereo and surface meshing". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2008, pp. 1–8 (cit. on p. 6).

[Ber+13] Matthew Berger, Joshua A Levine, Luis Gustavo Nonato, Gabriel Taubin, and Claudio T Silva. "A benchmark for surface reconstruction". In: *ACM Trans. on Graphics* 32.2 (2013), p. 20 (cit. on p. 35).

[Ber+17] Sema Berkiten, Maciej Halber, Justin Solomon, Chongyang Ma, Hao Li, and Szymon Rusinkiewicz. "Learning detail transfer based on geometric features". In: *Computer Graphics Forum* 36.2 (2017), pp. 361–373 (cit. on pp. 22, 75).

[BI66] Adi Ben-Israel. "A Newton-Raphson method for the solution of systems of equations". In: *Journal of Mathematical analysis and applications* 15.2 (1966), pp. 243–252 (cit. on p. 84).

[Bie+02] Henning Biermann, Ioana Martin, Fausto Bernardini, and Denis Zorin. “Cut-and-paste editing of multiresolution surfaces”. In: *ACM Transactions on Graphics (TOG)* 21.3 (2002), pp. 312–321 (cit. on p. 22).

[Bog+14] Federica Bogo, Javier Romero, Matthew Loper, and Michael J Black. “FAUST: Dataset and evaluation for 3D mesh registration”. In: *CVPR*. 2014, pp. 3794–3801 (cit. on p. 109).

[Boi+02] Jean-Daniel Boissonnat, Olivier Devillers, Sylvain Pion, Monique Teillaud, and Mariette Yvinec. “Triangulations in CGAL”. In: *Computational Geometry* 22 (2002), pp. 5–19 (cit. on p. 35).

[Bot+10] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon mesh processing*. CRC press, 2010 (cit. on p. 22).

[Bro+17] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. “Geometric deep learning: going beyond euclidean data”. In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42 (cit. on p. 16).

[Bro+21] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. “Geometric deep learning: Grids, groups, graphs, geodesics, and gauges”. In: *arXiv preprint arXiv:2104.13478* (2021) (cit. on p. 16).

[BW10] Z.M. Bi and Lihui Wang. “Advances in 3D data acquisition and processing for industrial applications”. In: *Robotics and Computer-Integrated Manufacturing* 26.5 (2010), pp. 403–413. ISSN: 0736-5845. DOI: `https://doi.org/10.1016/j.rcim.2010.03.003`. URL: `https://www.sciencedirect.com/science/article/pii/S073658451000013X` (cit. on p. 5).

[Car+01] Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. “Reconstruction and representation of 3D objects with radial basis functions”. In: *Proc. of SIGGRAPH*. 2001, pp. 67–76 (cit. on pp. 6, 21).

[CB17] Stéphane Calderon and Tamy Boubekeur. “Bounding proxies for shape approximation”. In: *ACM Trans. Graph.* 36.4 (2017), p. 57 (cit. on p. 24).

[CCC87] Robert L Cook, Loren Carpenter, and Edwin Catmull. “The Reyes image rendering architecture”. In: *ACM SIGGRAPH Computer Graphics* 21.4 (1987), pp. 95–102 (cit. on p. 68).

[CCS12] Massimiliano Corsini, Paolo Cignoni, and Roberto Scopigno. "Efficient and flexible sampling with blue noise properties of triangular meshes". In: *IEEE Trans. Visualization & Computer Graphics* 18.6 (2012), pp. 914–924 (cit. on p. 36).

[Cha+15] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. *ShapeNet: An Information-Rich 3D Model Repository*. Tech. rep. arXiv:1512.03012 [cs.GR]. 2015 (cit. on p. 105).

[Cha+20a] Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. "Deep local shapes: Learning local sdf priors for detailed 3d reconstruction". In: *European Conference on Computer Vision*. Springer. Springer International Publishing, 2020, pp. 608–625 (cit. on pp. 17, 21, 68).

[Cha+20b] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. "pi-GAN: Periodic Implicit Generative Adversarial Networks for 3D-Aware Image Synthesis". In: *arXiv preprint arXiv:2012.00926* (2020) (cit. on pp. 17, 74).

[Che+12] Xiaobai Chen, Tom Funkhouser, Dan B Goldman, and Eli Shechtman. "Non-parametric texture transfer using meshmatch". In: *Technical Report Technical Report 2012-2* (2012) (cit. on p. 22).

[Che+19] Chao Chen, Guanbin Li, Ruijia Xu, Tianshui Chen, Meng Wang, and Liang Lin. "Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4994–5002 (cit. on p. 16).

[Che+21] Zhiqin Chen, Vladimir Kim, Matthew Fisher, Noam Aigerman, Hao Zhang, and Siddhartha Chaudhuri. "DecorGAN: 3D Shape Detailization by Conditional Refinement". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021 (cit. on pp. 15, 22).

[Cig+08] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. "MeshLab: an Open-Source Mesh Processing Tool". In: *Eurographics Italian Chapter Conference*. 2008 (cit. on p. 36).

[CL96] Brian Curless and Marc Levoy. “A volumetric method for building complex models from range images”. In: *ACM Trans. on Graphics (Proc. of SIGGRAPH)*. 1996, pp. 303–312 (cit. on p. 21).

[Coo84] Robert L Cook. “Shade trees”. In: *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*. 1984, pp. 223–231 (cit. on p. 68).

[Cor+19] Victor Cornillere, Abdelaziz Djelouah, Wang Yifan, Olga Sorkine-Hornung, and Christopher Schroers. “Blind image super-resolution with spatially variant degradations”. In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), pp. 1–13.

[CTZ20] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. “Bsp-net: Generating compact meshes via binary space partitioning”. In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2020, pp. 45–54 (cit. on pp. 17, 21).

[Cyb89] George Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314 (cit. on pp. 3, 6).

[CZ19] Zhiqin Chen and Hao Zhang. “Learning implicit fields for generative shape modeling”. In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2019, pp. 5939–5948 (cit. on pp. 6, 7, 17, 21).

[Dai+18] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. “Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4578–4587 (cit. on p. 15).

[DBI18] Haowen Deng, Tolga Birdal, and Slobodan Ilic. “PPF-FoldNet: Unsupervised Learning of Rotation Invariant 3D Local Descriptors”. In: *arXiv preprint arXiv:1808.10322* (2018) (cit. on p. 19).

[DBV16] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. “Convolutional neural networks on graphs with fast localized spectral filtering”. In: *Advances in neural information processing systems* 29 (2016), pp. 3844–3852 (cit. on p. 16).

[Den+20] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. “Cvxnet: Learnable convex decomposition”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 31–44 (cit. on pp. 17, 21, 67).

[Don+16] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. “Image super-resolution using deep convolutional networks”. In: *IEEE Trans. Pattern Analysis & Machine Intelligence* 38.2 (2016), pp. 295–307 (cit. on p. 27).

[DRQN17] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. “Shape completion using 3d-encoder-predictor cnns and shape synthesis”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5868–5877 (cit. on p. 15).

[Dua+20] Yueqi Duan, Haidong Zhu, He Wang, Li Yi, Ram Nevatia, and Leonidas J Guibas. “Curriculum deepsdf”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 51–67 (cit. on p. 17).

[Dum+18] Vincent Dumoulin, Ethan Perez, Nathan Schucher, Florian Strub, Harm de Vries, Aaron Courville, and Yoshua Bengio. “Feature-wise transformations”. In: *Distill* 3.7 (2018), e11 (cit. on p. 74).

[Erl+20] Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Niloy J Mitra, and Michael Wimmer. “Points2Surf Learning Implicit Surfaces from Point Clouds”. In: *Proc. Euro. Conf. on Computer Vision*. Springer. 2020, pp. 108–124 (cit. on pp. 17, 21, 23, 94).

[Est+14] Joan Bruna Estrach, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. “Spectral networks and deep locally connected networks on graphs”. In: *2nd International Conference on Learning Representations, ICLR*. Vol. 2014. 2014 (cit. on p. 16).

[Est+21] Andre Esteva, Katherine Chou, Serena Yeung, Nikhil Naik, Ali Madani, Ali Mottaghi, Yun Liu, Eric Topol, Jeff Dean, and Richard Socher. “Deep learning-enabled medical computer vision”. In: *NPJ digital medicine* 4.1 (2021), pp. 1–9 (cit. on p. 1).

[Fan+17] Yuchen Fan, Honghui Shi, Jiahui Yu, Ding Liu, Wei Han, Haichao Yu, Zhangyang Wang, Xinchao Wang, and Thomas S Huang. “Balanced two-stage residual networks for image super-resolution”. In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition Workshops*. IEEE. 2017, pp. 1157–1164 (cit. on pp. 28, 30).

[Fig+92] Luiz Henrique de Figueiredo, Jonas de Miranda Gomes, Demetri Terzopoulos, and Luiz Velho. “Physically-based methods for polygonization of implicit surfaces”. In: *Graphics Interface*. Vol. 92. 1992, pp. 250–257 (cit. on p. 23).

[FLBMB90] Olivier D. Faugeras, Elisabeth Le Bras-Mehlman, and Jean-Daniel Boissonnat. "Representing stereo data with the delaunay triangulation". In: *Artificial Intelligence* 44.1-2 (1990), pp. 41–87 (cit. on p. 6).

[FP09] Yasutaka Furukawa and Jean Ponce. "Accurate, dense, and robust multiview stereopsis". In: *IEEE Trans. Pattern Analysis & Machine Intelligence* 32.8 (2009), pp. 1362–1376 (cit. on pp. 93, 94).

[Fri+00] Sarah F Frisken, Ronald N Perry, Alyn P Rockwood, and Thouis R Jones. "Adaptively sampled distance fields: A general representation of shape for computer graphics". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000, pp. 249–254 (cit. on p. 21).

[FSG17] Haoqiang Fan, Hao Su, and Leonidas J Guibas. "A Point Set Generation Network for 3D Object Reconstruction from a Single Image." In: vol. 2. 4. 2017, p. 6 (cit. on p. 16).

[Gao+18] Lin Gao, Jie Yang, Yi-Ling Qiao, Yu-Kun Lai, Paul L Rosin, Weiwei Xu, and Shihong Xia. "Automatic unpaired shape deformation transfer". In: *SIGGRAPH Asia*. 2018 (cit. on pp. 25, 109).

[Gao+19] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao(Richard) Zhang. "SDM-NET: Deep Generative Network for Structured Deformable Mesh". In: *ACM Trans. Graph.* 38.6 (2019), 243:1–243:15 (cit. on p. 25).

[GCS06] M. Goesele, B. Curless, and S.M. Seitz. "Multi-View Stereo Revisited". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. 2006, pp. 2402–2409. DOI: `10.1109/CVPR.2006.199` (cit. on p. 6).

[GEM18] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. "3D Semantic Segmentation With Submanifold Sparse Convolutional Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on p. 8).

[Gen+19] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. "Learning shape templates with structured implicit functions". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7154–7164 (cit. on pp. 17, 21).

[Gen+20] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. "Local deep implicit functions for 3d shape". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 4857–4866 (cit. on pp. 17, 21).

[Gla+16] Oliver Glauser, Wan-Chun Ma, Daniele Panozzo, Alec Jacobson, Otmar Hilliges, and Olga Sorkine-Hornung. "Rig Animation with a Tangible and Modular Input Device". In: *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)* 35.4 (2016) (cit. on p. 2).

[Goo+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets". In: *In Advances in Neural Information Processing Systems (NIPS)*. 2014 (cit. on p. 59).

[Goo+16] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. MIT press Cambridge, 2016 (cit. on p. 88).

[Gro+18a] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. "AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2018 (cit. on pp. 16, 23, 24, 33, 103, 108).

[Gro+18b] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. "3D-CODED: 3D correspondences by deep deformation". In: *ECCV*. 2018 (cit. on pp. 8, 16, 24, 104, 109).

[Gro+19] Thibault Groueix, Matthew Fisher, Vladimir Kim, Bryan Russell, and Mathieu Aubry. "Unsupervised cycle-consistent deformation for shape matching". In: *SGP*. 2019 (cit. on pp. 16, 24, 25, 100, 105, 106).

[Gro+20] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. *Implicit geometric regularization for learning shapes*. arXiv preprint arXiv:2002.10099. 2020 (cit. on p. 22).

[Gue+18] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J Mitra. "PCPNet Learning Local Shape Properties from Raw Point Clouds". In: *Computer Graphics Forum* 37.2 (2018), pp. 75–85 (cit. on p. 54).

[GWM18] Matheus Gadelha, Rui Wang, and Subhransu Maji. "Multiresolution Tree Networks for 3D Point Cloud Processing". In: *arXiv preprint arXiv:1807.03520* (2018) (cit. on p. 19).

[Hai+19] Niv Haim, Nimrod Segol, Heli Ben-Hamu, Haggai Maron, and Yaron Lipman. "Surface networks via general covers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 632–641 (cit. on p. 16).

[Han+17] Xiaoguang Han, Zhen Li, Haibin Huang, Evangelos Kalogerakis, and Yizhou Yu. "High-resolution shape completion using deep neural networks for global structure and local geometry inference". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 85–93 (cit. on p. 15).

[Han+18] Rana Hanocka, Noa Fish, Zhenhua Wang, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. "ALIGNet: partial-shape agnostic alignment via unsupervised learning". In: *ACM Trans. Graph.* 38.1 (2018), p. 1 (cit. on pp. 24, 25, 106, 107).

[Han+19] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. "Meshcnn: a network with an edge". In: *ACM Transactions on Graphics (TOG)* 38.4 (2019), pp. 1–12 (cit. on p. 17).

[Han+20] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. "Point2Mesh: A Self-Prior for Deformable Meshes". In: *ACM Trans. Graph.* 39.4 (2020). ISSN: 0730-0301. DOI: `10.1145/3386569.3392415`. URL: `https://doi.org/10.1145/3386569.3392415` (cit. on p. 17).

[Hao+20] Zekun Hao, Hadar Averbuch-Elor, Noah Snavely, and Serge Belongie. "Dualsdf: Semantic shape manipulation using a two-level representation". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2020, pp. 7631–7641 (cit. on pp. 21, 118).

[Har96] John C Hart. "Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces". In: *The Visual Computer* 12.10 (1996), pp. 527–545 (cit. on p. 23).

[Has+95] Mohamad H Hassoun et al. *Fundamentals of artificial neural networks*. MIT press, 1995 (cit. on pp. 3, 6).

[HBL15] Mikael Henaff, Joan Bruna, and Yann LeCun. "Deep convolutional networks on graph-structured data". In: *arXiv preprint arXiv:1506.05163* (2015) (cit. on p. 16).

[He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2016, pp. 770–778 (cit. on pp. 19, 31).

[Hec89] Paul S Heckbert. “Fundamentals of texture mapping and image warping”. In: (1989) (cit. on p. 45).

[Her+18] P. Hermosilla, T. Ritschel, P-P Vazquez, A. Vinacua, and T. Ropinski. “Monte Carlo Convolution for Learning on Non-Uniformly Sampled Point Clouds”. In: *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia)* 37.6 (2018). DOI: 10.1145/3272127.3275110 (cit. on p. 28).

[Her+20a] Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. “Deep Geometric Texture Synthesis”. In: *ACM Trans. Graph.* 39.4 (July 2020). ISSN: 0730-0301. DOI: 10.1145/3386569.3392471. URL: https://doi.org/10.1145/3386569.3392471 (cit. on pp. 17, 22, 80, 81).

[Her+20b] Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. “Pointgmm: A neural gmm network for point clouds”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 12054–12063 (cit. on p. 16).

[Her+21] Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. “Progressive Encoding for Neural Optimization”. In: *arXiv preprint arXiv:2104.09125* (2021) (cit. on pp. 22, 72).

[Hes10] Roland Hess. *Blender Foundations: The Essential Guide to Learning Blender 2.6*. Focal Press, 2010. ISBN: 0240814304, 9780240814308 (cit. on p. 92).

[HL94] Simon Haykin and Richard Lippmann. “Neural networks, a comprehensive foundation”. In: *International journal of neural systems* 5.4 (1994), pp. 363–364 (cit. on pp. 3, 6).

[Hop+92a] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. “Surface reconstruction from unorganized points”. In: *ACM Trans. on Graphics (Proc. of SIGGRAPH)* (1992), pp. 71–78 (cit. on pp. 6, 88).

[Hop+92b] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. “Surface reconstruction from unorganized points”. In: *Proc. of SIGGRAPH* (1992), pp. 71–78 (cit. on p. 36).

[Hor91] Kurt Hornik. “Approximation capabilities of multilayer feed-forward networks”. In: *Neural networks* 4.2 (1991), pp. 251–257 (cit. on pp. 3, 6).

[HRR19] Pedro Hermosilla, Tobias Ritschel, and Timo Ropinski. “Total Denoising: Unsupervised Learning of 3D Point Cloud Cleaning”. In: *arXiv preprint arXiv:1904.07615* (2019) (cit. on p. 16).

[HTM17] Christian Häne, Shubham Tulsiani, and Jitendra Malik. "Hierarchical Surface Prediction for 3D Object Reconstruction". In: *2017 International Conference on 3D Vision (3DV)*. 2017, pp. 412–420. DOI: `10.1109/3DV.2017.00054` (cit. on p. 15).

[HTY18] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. "Pointwise convolutional neural networks". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2018, pp. 984–993 (cit. on p. 28).

[Hu+19] Tao Hu, Zhizhong Han, Abhinav Shrivastava, and Matthias Zwicker. "Render4Completion: Synthesizing Multi-view Depth Maps for 3D Shape Completion". In: *arXiv preprint arXiv:1904.08366* (2019) (cit. on p. 20).

[Hua+08] Qi-Xing Huang, Bart Adams, Martin Wicke, and Leonidas J. Guibas. "Non-rigid Registration Under Isometric Deformations". In: *SGP*. 2008 (cit. on p. 25).

[Hua+09] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or. "Consolidation of Unorganized Point Clouds for Surface Reconstruction". In: *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia)* 28.5 (2009), 176:1–176:7 (cit. on pp. 18, 35, 41, 52, 54, 60).

[Hua+13a] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. Zhang. "Edge-Aware Point Set Resampling". In: *ACM Trans. on Graphics* 32.1 (2013), 9:1–9:12 (cit. on pp. 5, 18, 36, 60).

[Hua+13b] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao Zhang. "Edge-aware point set resampling". In: *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia)* 32.1 (2013), pp. 1–12 (cit. on pp. 23, 85).

[Hua+16] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. "Deep networks with stochastic depth". In: *Proc. Euro. Conf. on Computer Vision*. Springer. 2016, pp. 646–661 (cit. on p. 31).

[Hua+17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. "Densely Connected Convolutional Networks." In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2017 (cit. on pp. 19, 31).

[Hua+18] Haibin Huang, Evangelos Kalogerakis, Siddhartha Chaudhuri, Duygu Ceylan, Vladimir G Kim, and Ersin Yumer. "Learning local shape descriptors from part correspondences with multi-view convolutional networks". In: *ACM Trans. Graph.* 37.1 (2018) (cit. on p. 106).

[HWK15] Qixing Huang, Hai Wang, and Vladlen Koltun. "Single-view Reconstruction via Joint Analysis of Image and Shape Collections". In: *ACM Trans. Graph.* 34.4 (2015), 87:1–87:10 (cit. on p. 25).

[ID18] Eldar Insafutdinov and Alexey Dosovitskiy. "Unsupervised learning of shape and pose with differentiable point clouds". In: *Proc. IEEE Int. Conf. on Neural Information Processing Systems (NeurIPS)*. 2018, pp. 2802–2812 (cit. on pp. 20, 43, 58).

[Inc19] Apple Inc. *How to use Animoji on your iPhone and iPad Pro*. 2019. URL: `https://support.apple.com/en-us/HT208190` (visited on 07/01/2021) (cit. on p. 1).

[Iso+17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. "Image-To-Image Translation With Conditional Adversarial Networks". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. July 2017 (cit. on p. 59).

[Jac+14] Alec Jacobson, Daniele Panozzo, Oliver Glauser, Cédric Pradalier, Otmar Hilliges, and Olga Sorkine-Hornung. "Tangible and Modular Input Device for Character Articulation". In: *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)* 33.4 (2014), 82:1–82:12 (cit. on p. 2).

[Jac+18] Dominic Jack, Jhony K Pontes, Sridha Sridharan, Clinton Fookes, Sareh Shirazi, Frederic Maire, and Anders Eriksson. "Learning free-form deformations for 3D object reconstruction". In: *ACCV*. 2018 (cit. on p. 24).

[Jen+14] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. "Large scale multi-view stereopsis evaluation". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. IEEE. 2014, pp. 406–413 (cit. on p. 92).

[Jia+19] Chiyu Max Jiang, Jingwei Huang, Karthik Kashinath, Prabhat, Philip Marcus, and Matthias Nießner. "Spherical CNNs on Unstructured Grids". In: *International Conference on Learning Representations*. 2019. URL: `https://openreview.net/forum?id=Bkl-43C9FQ` (cit. on p. 16).

[Jia+20] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. "Local implicit grid representations for 3d scenes". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 6001–6010 (cit. on p. 68).

[Jos+07] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. "Harmonic coordinates for character articulation". In: *ACM Trans. Graph.* 26.3 (2007) (cit. on pp. 24, 100).

[JSW05] Tao Ju, Scott Schaefer, and Joe Warren. "Mean value coordinates for closed triangular meshes". In: *ACM Trans. Graph.* 24.3 (2005), pp. 561–566 (cit. on pp. 24, 100, 102).

[JWL18] Mingyang Jiang, Yiran Wu, and Cewu Lu. "PointSIFT: A SIFT-like Network Module for 3D Point Cloud Semantic Segmentation". In: *arXiv preprint arXiv:1807.00652* (2018) (cit. on p. 19).

[Kar+18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. "Progressive Growing of GANs for Improved Quality, Stability, and Variation". In: *Proc. Int. Conf. on Learning Representations*. 2018 (cit. on pp. 19, 31).

[Kat+20] Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. *Differentiable Rendering: A Survey*. arXiv preprint arXiv:2006.12057. 2020 (cit. on p. 23).

[Kaz+20] Misha Kazhdan, Ming Chuang, Szymon Rusinkiewicz, and Hugues Hoppe. "Poisson surface reconstruction with envelope constraints". In: *Computer Graphics Forum*. Vol. 39. 5. Wiley Online Library. 2020, pp. 173–182 (cit. on p. 6).

[KBH06] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. "Poisson surface reconstruction". In: *Proc. Eurographics Symp. on Geometry Processing*. Vol. 7. 2006 (cit. on pp. 6, 21).

[KH13a] Michael Kazhdan and Hugues Hoppe. "Screened poisson surface reconstruction". In: *ACM Trans. on Graphics* 32.3 (2013), pp. 1–13 (cit. on pp. 6, 93).

[KH13b] Michael Kazhdan and Hugues Hoppe. "Screened Poisson surface reconstruction". In: *ACM Trans. on Graphics* 32.1 (2013), 29:1–29:13 (cit. on p. 36).

[KKLML16] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. "Accurate image super-resolution using very deep convolutional networks". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2016, pp. 1646–1654 (cit. on p. 28).

[KL17] Roman Klokov and Victor Lempitsky. "Escape from cells: Deep kd-networks for the recognition of 3D point cloud models". In: *Proc. Int. Conf. on Computer Vision*. IEEE. 2017, pp. 863–872 (cit. on p. 19).

[Kos+21] Adam R. Kosiorek, Heiko Strathmann, Daniel Zoran, Pol Moreno, Rosalia Schneider, Soňa Mokrá, and Danilo J. Rezende. *NeRF-VAE: A Geometry Aware 3D Scene Generative Model*. 2021. arXiv: `2104.00587 [stat.ML]` (cit. on p. 17).

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *In Advances in Neural Information Processing Systems (NIPS)*. 2012, pp. 1097–1105 (cit. on p. 19).

[KUH18] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. "Neural 3D mesh renderer". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2018, pp. 3907–3916 (cit. on pp. 19, 43, 49, 50, 56, 57).

[Lai+17] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. "Deep laplacian pyramid networks for fast and accurate superresolution". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2017 (cit. on pp. 19, 28, 30).

[LB14] Matthew M Loper and Michael J Black. "OpenDR: An approximate differentiable renderer". In: *Proc. Euro. Conf. on Computer Vision*. Springer. 2014, pp. 154–169 (cit. on pp. 19, 43, 50, 56, 57).

[LB20] Ryan Lege and Euan Bonner. "Virtual reality in education: The promise, progress, and challenge." In: *JALT CALL Journal* 16.3 (2020) (cit. on p. 1).

[LC10] Yann LeCun and Corinna Cortes. *MNIST handwritten digit database*. http://yann.lecun.com/exdb/mnist/. 2010. URL: `http://yann.lecun.com/exdb/mnist/` (cit. on p. 35).

[LC87] William E Lorensen and Harvey E Cline. "Marching cubes: A high resolution 3D surface construction algorithm". In: *ACM Trans. on Graphics (Proc. of SIGGRAPH)* 21.4 (1987), pp. 163–169 (cit. on p. 6).

[LCL18] Jiaxin Li, Ben M Chen, and Gim Hee Lee. "SO-Net: Self-Organizing Network for Point Cloud Analysis". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2018, pp. 9397–9406 (cit. on p. 32).

[Led+17] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network." In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2017 (cit. on p. 28).

[Lev98] David Levin. "The approximation power of moving least-squares". In: *Mathematics of computation* 67.224 (1998), pp. 1517–1531 (cit. on p. 21).

[LGS06] Florian Levet, Xavier Granier, and Christophe Schlick. "Fast sampling of implicit surfaces by particle systems". In: *IEEE International Conference on Shape Modeling and Applications*. IEEE. 2006, pp. 39–39 (cit. on p. 23).

[LH16] Ilya Loshchilov and Frank Hutter. "Sgdr: Stochastic gradient descent with warm restarts". In: *arXiv preprint arXiv:1608.03983* (2016) (cit. on p. 74).

[Li+12] Hao Li, Linjie Luo, Daniel Vlasic, Pieter Peers, Jovan Popović, Mark Pauly, and Szymon Rusinkiewicz. "Temporally Coherent Completion of Dynamic Shapes". In: *ACM Trans. Graph.* 31.1 (2012) (cit. on p. 25).

[Li+18a] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. "Differentiable monte carlo ray tracing through edge sampling". In: *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia)*. ACM. 2018, p. 222 (cit. on p. 20).

[Li+18b] Yangyan Li, Rui Bu, Mingchao Sun, and Baoquan Chen. "PointCNN". In: *arXiv preprint arXiv:1801.07791* (2018) (cit. on pp. 16, 28).

[Li+19] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. "Pu-gan: a point cloud upsampling adversarial network". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7203–7212 (cit. on pp. 16, 65).

[Li+21a] Peizhuo Li, Kfir Aberman, Rana Hanocka, Libin Liu, Olga Sorkine-Hornung, and Baoquan Chen. "Learning Skeletal Articulations with Neural Blend Shapes". In: *ACM Transactions on Graphics (TOG)* 40.4 (2021), p. 1 (cit. on p. 17).

[Li+21b] Ruihui Li, Xianzhi Li, Pheng-Ann Heng, and Chi-Wing Fu. "Point Cloud Upsampling via Disentangled Refinement". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 344–353 (cit. on p. 65).

[Lin+18] Di Lin, Yuanfeng Ji, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. "Multi-Scale Context Intertwining for Semantic Segmentation". In: *Proc. Euro. Conf. on Computer Vision*. 2018, pp. 603–619 (cit. on p. 32).

[Lip+07] Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. "Parameterization-free projection for geometry reconstruction". In: *ACM Trans. on Graphics (Proc. of SIGGRAPH)* 26.3 (2007), 22:1–22:6 (cit. on p. 18).

[Liu+17] Guilin Liu, Duygu Ceylan, Ersin Yumer, Jimei Yang, and Jyh-Ming Lien. "Material editing using a physically based rendering network". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2017, pp. 2261–2269 (cit. on p. 43).

[Liu+18] Xinhai Liu, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. "Point2Sequence: Learning the Shape Representation of 3D Point Clouds with an Attention-based Sequence to Sequence Network". In: *arXiv preprint arXiv:1811.02565* (2018) (cit. on p. 19).

[Liu+19a] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. "Soft rasterizer: A differentiable renderer for image-based 3D reasoning". In: *Proc. Int. Conf. on Computer Vision*. 2019, pp. 7708–7717 (cit. on pp. 19, 50).

[Liu+19b] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. "Soft Rasterizer: A Differentiable Renderer for Image-Based 3D Reasoning". In: *ICCV*. 2019 (cit. on p. 104).

[Liu+19c] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. "Learning to infer implicit surfaces without supervision". In: *Proc. IEEE Int. Conf. on Neural Information Processing Systems (NeurIPS)*. 2019, pp. 8295–8306 (cit. on p. 23).

[Liu+20a] Hsueh-Ti Derek Liu, Vladimir G. Kim, Siddhartha Chaudhuri, Noam Aigerman, and Alec Jacobson. "Neural Subdivision". In: *ACM Trans. Graph.* 39.4 (July 2020). ISSN: 0730-0301. DOI: `10.1145/3386569.3392418`. URL: `https://doi.org/10.1145/3386569.3392418` (cit. on p. 17).

[Liu+20b] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. "Neural Sparse Voxel Fields". In: *Proc. IEEE Int. Conf. on Neural Information Processing Systems (NeurIPS)*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 15651–15663. URL: `https://proceedings.neurips.cc/paper/2020/file/b4b758962f17808746e9bb832a6fa4b8-Paper.pdf` (cit. on pp. 21, 68).

[Liu+20c] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. "DIST: Rendering deep implicit signed distance function with differentiable sphere tracing". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2020, pp. 2019–2028 (cit. on p. 23).

[LKL18] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. "Learning efficient point cloud generation for dense 3D object reconstruction". In: *AAAI Conference on Artificial Intelligence*. 2018 (cit. on pp. 20, 43).

[LLCO08] Yaron Lipman, David Levin, and Daniel Cohen-Or. "Green coordinates". In: *ACM Trans. Graph.* 27.3 (2008) (cit. on pp. 24, 100, 114).

[Lom+19] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. *Neural volumes: Learning dynamic renderable volumes from images*. arXiv preprint arXiv:1906.07751. 2019 (cit. on p. 15).

[Lon] Imperial College London. *Virtual Labs and Simulation Tools for Remote Education | Staff | Imperial College London*. `https://www.imperial.ac.uk/immersive-technology-initiative/virtual-labs-and-simulation-tools-for-remote-education/`. (Accessed on 07/08/2021) (cit. on p. 1).

[Loo87] Charles Loop. "Smooth subdivision surfaces based on triangles". In: *Master's thesis, University of Utah, Department of Mathematics* (1987) (cit. on p. 35).

[LSP08] Hao Li, Robert W. Sumner, and Mark Pauly. "Global Correspondence Optimization for Non-Rigid Registration of Depth Scans". In: *SGP*. 2008 (cit. on pp. 25, 100).

[LTJ18] Hsueh-Ti Derek Liu, Michael Tao, and Alec Jacobson. "Paparazzi: surface editing by way of multi-view image processing." In: *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia)* 37.6 (2018), pp. 221–1 (cit. on pp. 19, 43, 50, 56, 57, 59).

[Lu+18] Xuequan Lu, Shihao Wu, Honghua Chen, Sai-Kit Yeung, Wenzhi Chen, and Matthias Zwicker. "GPF: GMM-inspired feature-preserving point set filtering". In: *IEEE Trans. Visualization & Computer Graphics* 24.8 (2018), pp. 2315–2326 (cit. on p. 60).

[LWL20] Chen-Hsuan Lin, Chaoyang Wang, and Simon Lucey. *SDF-SRN: Learning Signed Distance 3D Object Reconstruction from Static Images*. arXiv preprint arXiv:2010.10505. 2020 (cit. on p. 21).

[LZ21a] Christoph Lassner and Michael Zollhofer. “Pulsar: Efficient Sphere-based Neural Rendering”. In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2021, pp. 1440–1449 (cit. on pp. 19, 65).

[LZ21b] Manyi Li and Hao Zhang. “D^{2}IM-Net: Learning Detail Disentangled Implicit Fields from Single Images”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021 (cit. on p. 22).

[Mar+17] Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G Kim, and Yaron Lipman. “Convolutional neural networks on surfaces via seamless toric covers.” In: *ACM Trans. Graph.* 36.4 (2017), pp. 71–1 (cit. on p. 16).

[Mar+21] Julien N. P. Martel, David B. Lindell, Connor Z. Lin, Eric R. Chan, Marco Monteiro, and Gordon Wetzstein. “Acorn: Adaptive Coordinate Networks for Neural Scene Representation”. In: *ACM Trans. Graph.* 40.4 (July 2021). ISSN: 0730-0301. DOI: `10.1145/3450626.3459785`. URL: `https://doi.org/10.1145/3450626.3459785` (cit. on pp. 17, 21).

[Mas+15] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. “Geodesic convolutional neural networks on riemannian manifolds”. In: *Proceedings of the IEEE international conference on computer vision workshops*. 2015, pp. 37–45 (cit. on p. 16).

[MB+21] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. “NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 7210–7219 (cit. on p. 17).

[Mes+19] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. “Occupancy networks: Learning 3D reconstruction in function space”. In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2019, pp. 4460–4470 (cit. on pp. 6, 7, 17, 21, 24).

[Mil+20] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. “Nerf: Representing scenes as neural radiance fields for view synthesis”. In: *European*

Conference on Computer Vision. Springer. Springer International Publishing, 2020, pp. 405–421 (cit. on pp. 17, 21, 67, 68, 84).

[MO14] Mehdi Mirza and Simon Osindero. “Conditional generative adversarial nets”. In: *arXiv preprint arXiv:1411.1784* (2014) (cit. on p. 32).

[Mo+19] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. “PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019 (cit. on p. 8).

[Mon+17] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. “Geometric deep learning on graphs and manifolds using mixture model cnns”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5115–5124 (cit. on p. 16).

[NG21] Michael Niemeyer and Andreas Geiger. “GIRAFFE: Representing Scenes As Compositional Generative Neural Feature Fields”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 11453–11464 (cit. on p. 17).

[Nie+19] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. “Occupancy flow: 4d reconstruction by learning particle dynamics”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 5379–5389 (cit. on p. 17).

[Nie+20] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. “Differentiable volumetric rendering: Learning implicit 3D representations without 3D supervision”. In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2020, pp. 3504–3515 (cit. on pp. 17, 21, 67).

[NP+18] Thu H Nguyen-Phuoc, Chuan Li, Stephen Balaban, and Yongliang Yang. “Rendernet: A deep convolutional network for differentiable rendering from 3D shapes”. In: *Proc. IEEE Int. Conf. on Neural Information Processing Systems (NeurIPS)*. 2018, pp. 7891–7901 (cit. on p. 43).

[Oec+19] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. “Texture fields: Learning texture representations in function space”. In: *Proceedings of the*

IEEE/CVF International Conference on Computer Vision. 2019, pp. 4531–4540 (cit. on p. 17).

[Oec+20] Michael Oechsle, Michael Niemeyer, Christian Reiser, Lars Mescheder, Thilo Strauss, and Andreas Geiger. "Learning Implicit Surface Light Fields". In: *2020 International Conference on 3D Vision (3DV)*. 2020, pp. 452–462. DOI: 10.1109/3DV50981.2020.00055 (cit. on p. 17).

[ÖGG09] A Cengiz Öztireli, Gael Guennebaud, and Markus Gross. "Feature preserving point set surfaces based on non-linear kernel regression". In: *Computer Graphics Forum (Proc. of Eurographics)*. Vol. 28. 2. 2009, pp. 493–501 (cit. on pp. 18, 52, 54, 60).

[Oht+03] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. "Multi-Level Partition of Unity Implicits". In: *ACM Trans. Graph.* 22.3 (2003), 463–470. ISSN: 0730-0301. DOI: 10.1145/882262.882293. URL: https://doi.org/10.1145/882262.882293 (cit. on p. 21).

[Par+19] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. "DeepSDF: Learning continuous signed distance functions for shape representation". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2019, pp. 165–174 (cit. on pp. 6, 7, 17, 21, 75).

[Par+20] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. "Deformable Neural Radiance Fields". In: *arXiv preprint arXiv:2011.12948* (2020) (cit. on pp. 22, 71, 72, 75, 77).

[Pas+17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. "Automatic differentiation in PyTorch". In: *NIPS-W*. 2017 (cit. on p. 56).

[Pas+18] Despoina Paschalidou, Osman Ulusoy, Carolin Schmitt, Luc Van Gool, and Andreas Geiger. "Raynet: Learning volumetric 3d reconstruction with ray potentials". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3897–3906 (cit. on p. 15).

[Pen+20] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. "Convolutional Occupancy Networks". In: *European Conference on Computer Vision (ECCV)*. Cham: Springer International Publishing, Aug. 2020 (cit. on p. 73).

[Per+18] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. “Film: Visual reasoning with a general conditioning layer”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2018 (cit. on p. 74).

[Pet+19] Felix Petersen, Amit H Bermano, Oliver Deussen, and Daniel Cohen-Or. “Pix2Vex: Image-to-Geometry Reconstruction using a Smooth Differentiable Renderer”. In: *arXiv preprint arXiv:1903.11149* (2019) (cit. on pp. 19, 50).

[Pfi+00] Hanspeter Pfister, Matthias Zwicker, Jeroen Van Baar, and Markus Gross. “Surfels: Surface elements as rendering primitives”. In: *Proc. Conf. on Computer Graphics and Interactive techniques*. 2000, pp. 335–342 (cit. on p. 20).

[PJH16] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016 (cit. on p. 82).

[PJS07] João Proença, Joaquim A Jorge, and Mario Costa Sousa. “Sampling Point-Set Implicits.” In: *Eurographics Symposium on Point-Based Graphics*. 2007, pp. 11–18 (cit. on p. 23).

[Pon+17] Jhony K Pontes, Chen Kong, Sridha Sridharan, Simon Lucey, Anders Eriksson, and Clinton Fookes. “Image2Mesh: A Learning Framework for Single Image 3D Reconstruction”. In: *arXiv preprint arXiv:1711.10669* (2017) (cit. on p. 20).

[Pou+20] Omid Poursaeed, Matthew Fisher, Noam Aigerman, and Vladimir G Kim. *Coupling explicit and implicit surface representations for generative 3D modeling*. arXiv preprint arXiv:2007.10294. 2020 (cit. on p. 23).

[Pum+21] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. “D-NeRF: Neural Radiance Fields for Dynamic Scenes”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 10318–10327 (cit. on pp. 17, 77).

[Qi+17a] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. “Pointnet: Deep learning on point sets for 3d classification and segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660 (cit. on pp. 16, 28).

[Qi+17b] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. "PointNet++: Deep hierarchical feature learning on point sets in a metric space". In: *In Advances in Neural Information Processing Systems (NIPS)*. 2017, pp. 5099–5108 (cit. on pp. 16, 28, 32).

[Rah+19] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. "On the spectral bias of neural networks". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 5301–5310 (cit. on pp. 10, 21).

[Raj+18] Sai Rajeswar, Fahim Mannan, Florian Golemo, David Vazquez, Derek Nowrouzezahrai, and Aaron Courville. "Pix2Scene: Learning Implicit 3D Representations from Images". In: (2018) (cit. on p. 43).

[Rak+19] Marie-Julie Rakotosaona, Vittorio La Barbera, Paul Guerrero, Niloy J Mitra, and Maks Ovsjanikov. "POINTCLEANNET: Learning to Denoise and Remove Outliers from Dense Point Clouds". In: *arXiv preprint arXiv:1901.01060* (2019) (cit. on pp. 16, 60).

[Rav+20] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. "Accelerating 3D Deep Learning with PyTorch3D". In: *arXiv:2007.08501* (2020) (cit. on p. 65).

[Ret+18] Dario Rethage, Johanna Wald, Jürgen Sturm, Nassir Navab, and Federico Tombari. "Fully-Convolutional Point Networks for Large-Scale Point Clouds". In: *arXiv preprint arXiv:1808.06840* (2018) (cit. on p. 19).

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical Image Computing and Computer-assisted Intervention*. Springer. 2015, pp. 234–241 (cit. on p. 19).

[Ric+17] Elad Richardson, Matan Sela, Roy Or-El, and Ron Kimmel. "Learning detailed face reconstruction from a single image". In: *IEEE Trans. Pattern Analysis & Machine Intelligence*. 2017, pp. 1259–1268 (cit. on p. 20).

[ROUG17] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. "Octnet: Learning deep 3d representations at high resolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 3577–3586 (cit. on p. 16).

[Rov+18a] Riccardo Roveri, A Cengiz Öztireli, Ioana Pandele, and Markus Gross. "PointProNets: Consolidation of Point Clouds with Convolutional Neural Networks". In: *Computer Graphics Forum* 37.2 (2018), pp. 87–99 (cit. on pp. 43, 58).

[Rov+18b] Riccardo Roveri, Lukas Rahmann, Cengiz Oztireli, and Markus Gross. "A network architecture for point cloud classification via automatic depth images generation". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2018, pp. 4176–4184 (cit. on pp. 20, 58).

[SA07] Olga Sorkine and Marc Alexa. "As-Rigid-As-Possible Surface Modeling". In: *SGP*. 2007 (cit. on p. 24).

[Sai+19] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. "Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2304–2314 (cit. on pp. 17, 21).

[Sai+20] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. "Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 84–93 (cit. on pp. 17, 21).

[SB09] Olga Sorkine and Mario Botsch. "Tutorial: Interactive Shape Modeling and Deformation". In: *EUROGRAPHICS*. 2009 (cit. on pp. 22, 24).

[Sch+20] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. "GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis". In: *Proc. IEEE Int. Conf. on Neural Information Processing Systems (NeurIPS)*. 2020 (cit. on p. 17).

[SD99] Steven M Seitz and Charles R Dyer. "Photorealistic scene reconstruction by voxel coloring". In: *International Journal of Computer Vision* 35.2 (1999), pp. 151–173 (cit. on p. 6).

[SDM19] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. "Singan: Learning a generative model from a single natural image". In: *ICCV*. 2019 (cit. on p. 112).

[She+18] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. "Mining point cloud local structures by kernel correlation and graph pooling". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2018 (cit. on p. 32).

[Shi+16] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2016, pp. 1874–1883 (cit. on p. 28).

[Sit+18] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. "DeepVoxels: Learning Persistent 3D Feature Embeddings". In: *arXiv preprint arXiv:1812.01024* (2018) (cit. on pp. 15, 20).

[Sit+20] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. "Implicit Neural Representations with Periodic Activation Functions". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 7462–7473. URL: `https://proceedings.neurips.cc/paper/2020/file/53c04118df112c13a8c34b38343b9c10-Paper.pdf` (cit. on pp. 21, 68, 84).

[Skea] Sketchfab. *Sketchfab*. `https://sketchfab.com` (cit. on p. 36).

[Skeb] *Sketchfab*. `https://sketchfab.com`. 2021 (cit. on p. 75).

[Sor+04] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and H-P Seidel. "Laplacian surface editing". In: *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. 2004, pp. 175–184 (cit. on p. 22).

[SOS04] Chen Shen, James F. O'Brien, and Jonathan R. Shewchuk. "Interpolating and Approximating Implicit Surfaces from Polygon Soup". In: *Proceedings of ACM SIGGRAPH 2004*. Los Angeles, California: ACM Press, Aug. 2004, pp. 896–904. URL: `http://graphics.cs.berkeley.edu/papers/Shen-IAI-2004-08/` (cit. on p. 6).

[SP04] Robert W. Sumner and Jovan Popović. "Deformation Transfer for Triangle Meshes". In: *ACM Trans. Graph.* 23.3 (2004), pp. 399–405 (cit. on pp. 25, 99, 109).

[SR20] Weijing Shi and Raj Rajkumar. "Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020 (cit. on p. 16).

[Sta] *Stanford 3D scan repository*. `http://graphics.stanford.edu/data/3Dscanrep/`. 2021 (cit. on p. 75).

[Sun+20] Yongbin Sun, Yue Wang, Ziwei Liu, Joshua Siegel, and Sanjay Sarma. "Pointgrow: Autoregressively learned point cloud generation with self-attention". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020, pp. 61–70 (cit. on p. 16).

[Sut+13] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. "On the importance of initialization and momentum in deep learning". In: *Proc. Int. Conf. on Machine Learning*. 2013, pp. 1139–1147 (cit. on p. 56).

[SVJ15] Leonardo Sacht, Etienne Vouga, and Alec Jacobson. "Nested Cages". In: *ACM Trans. Graph.* 34.6 (2015), 170:1–170:14 (cit. on p. 24).

[SWL19] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. "PointR-CNN: 3D Object Proposal Generation and Detection From Point Cloud". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019 (cit. on p. 16).

[SZW19] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. "Scene representation networks: Continuous 3D-structure-aware neural scene representations". In: *Proc. IEEE Int. Conf. on Neural Information Processing Systems (NeurIPS)*. 2019, pp. 1121–1132 (cit. on pp. 17, 21, 82).

[Tak+11] Kenshi Takayama, Ryan Schmidt, Karan Singh, Takeo Igarashi, Tamy Boubekeur, and Olga Sorkine. "Geobrush: Interactive mesh geometry cloning". In: *Computer Graphics Forum* 30.2 (2011), pp. 613–622 (cit. on p. 22).

[Tak+21] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. "Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021 (cit. on pp. 21, 68, 75).

[Tan+18] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. "Variational Autoencoders for Deforming 3D Mesh Models". In: *CVPR*. 2018 (cit. on pp. 24, 100).

[Tan+20] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains". In: *Advances in Neural Information Processing*

Systems. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 7537–7547. URL: `https://proceedings.neurips.cc/paper/2020/file/55053683268957697aa39fba6f231c68-Paper.pdf` (cit. on pp. 22, 75).

[Tew+20] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. *State of the Art on Neural Rendering*. arXiv preprint arXiv:2004.03805. 2020 (cit. on p. 23).

[TL94] Greg Turk and Marc Levoy. "Zippered polygon meshes from range images". In: *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. 1994, pp. 311–318 (cit. on p. 6).

[Tre+20] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Carsten Stoll, and Christian Theobalt. "PatchNets: Patch-Based Generalizable Deep Implicit 3D Shape Representations". In: *European Conference on Computer Vision*. Springer. Springer International Publishing, 2020, pp. 293–309 (cit. on pp. 17, 21).

[TTB12] Jean-Marc Thiery, Julien Tierny, and Tamy Boubekeur. "CageR: Cage-based Reverse Engineering of Animated 3D Shapes". In: *Computer Graphics Forum* 31.8 (2012), pp. 2303–2316 (cit. on p. 24).

[Tul+17] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. "Multi-view supervision for single-view reconstruction via differentiable ray consistency". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2017, pp. 2626–2634 (cit. on pp. 15, 43).

[Uni21] European Union. *The European Commission launches a unique study on 3D digitisation of tangible cultural heritage*. 2021. URL: `https://digital-strategy.ec.europa.eu/en/news/european-commission-launches-unique-study-3d-digitisation-tangible-cultural-heritage` (visited on 07/01/2021) (cit. on p. 1).

[UVL18] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Deep image prior". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2018, pp. 9446–9454 (cit. on p. 87).

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008 (cit. on p. 97).

[Vog+07] George Vogiatzis, Carlos Hernández Esteban, Philip HS Torr, and Roberto Cipolla. "Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency". In: *IEEE transactions on pattern analysis and machine intelligence* 29.12 (2007), pp. 2241–2246 (cit. on p. 6).

[Vog+18] Thijs Vogels, Fabrice Rousselle, Brian McWilliams, Gerhard Röthlin, Alex Harvill, David Adler, Mark Meyer, and Jan Novák. "Denoising with kernel prediction and asymmetric loss functions". In: *ACM Trans. on Graphics* 37.4 (2018), p. 124 (cit. on p. 54).

[VSH19] Floor Verhoeven and Olga Sorkine-Hornung. "RodMesh: Two-handed 3D Surface Modeling in Virtual Reality". In: *Proceedings of the Symposium on Vision, Modeling and Visualization (VMV)*. Eurographics Association, 2019 (cit. on p. 2).

[WAJL20] Xiaogang Wang, Marcelo H Ang Jr, and Gim Hee Lee. "Cascaded refinement network for point cloud completion". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 790–799 (cit. on pp. 16, 65).

[Wan+17a] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. "O-cnn: Octree-based convolutional neural networks for 3d shape analysis". In: *ACM Transactions On Graphics (TOG)* 36.4 (2017), pp. 1–11 (cit. on p. 16).

[Wan+17b] Weiyue Wang, Qiangui Huang, Suya You, Chao Yang, and Ulrich Neumann. "Shape inpainting using 3d generative adversarial network and recurrent convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2298–2306 (cit. on p. 15).

[Wan+18a] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. "Pixel2mesh: Generating 3D mesh models from single RGB images". In: *Proc. Euro. Conf. on Computer Vision*. 2018, pp. 52–67 (cit. on p. 24).

[Wan+18b] Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. "Adaptive O-CNN: A Patch-based Deep Representation of 3D Shapes". In: *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia)* (2018) (cit. on p. 19).

[Wan+18c] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. “High-resolution image synthesis and semantic manipulation with conditional GANs”. In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2018 (cit. on p. 19).

[Wan+18d] Yifan Wang, Federico Perazzi, Brian McWilliams, Alexander Sorkine-Hornung, Olga Sorkine-Hornung, and Christopher Schroers. “A fully progressive approach to single-image super-resolution”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018, pp. 864–873 (cit. on pp. 19, 31, 34).

[Wan+18e] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. “Dynamic graph CNN for learning on point clouds”. In: *arXiv preprint arXiv:1801.07829* (2018) (cit. on pp. 16, 32).

[Wan+19] Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. “3DN: 3D deformation network”. In: *CVPR*. 2019 (cit. on pp. 16, 24, 25, 99, 100, 104–107, 113).

[Wen+20] Xin Wen, Tianyang Li, Zhizhong Han, and Yu-Shen Liu. “Point cloud completion by skip-attention network with hierarchical folding”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 1939–1948 (cit. on p. 16).

[WH94] Andrew P Witkin and Paul S Heckbert. “Using particles to sample and control implicit surfaces”. In: *ACM Trans. on Graphics (Proc. of SIGGRAPH)*. 1994, pp. 269–277 (cit. on p. 23).

[Wil+19] Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. “Deep geometric prior for surface reconstruction”. In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2019, pp. 10130–10139 (cit. on p. 87).

[Wil+20] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. “SynSin: End-to-End View Synthesis From a Single Image”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020 (cit. on p. 19).

[WQF19] Wenxuan Wu, Zhongang Qi, and Li Fuxin. “PointConv: Deep Convolutional Networks on 3D Point Clouds”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019 (cit. on p. 16).

[Wu+14] Zizhao Wu, Ruyang Shou, Yunhai Wang, and Xinguo Liu. “Interactive shape co-segmentation via label propagation”. In: *Computers & Graphics* 38 (2014), pp. 248–254 (cit. on p. 112).

[Wu+15a] Shihao Wu, Hui Huang, Minglun Gong, Matthias Zwicker, and Daniel Cohen-Or. “Deep points consolidation”. In: *ACM Trans. on Graphics* 34.6 (2015), p. 176 (cit. on p. 18).

[Wu+15b] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. “3d shapenets: A deep representation for volumetric shapes”. In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2015, pp. 1912–1920 (cit. on pp. 15, 35).

[Wu+19] Zhijie Wu, Xiang Wang, Di Lin, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. “SAGNet: Structure-Aware Generative Network for 3D-Shape Modeling”. In: 38.4 (July 2019). ISSN: 0730-0301. DOI: `10.1145/3306346.3322956`. URL: `https://doi.org/10.1145/3306346.3322956` (cit. on p. 8).

[XDZ17] Haotian Xu, Ming Dong, and Zichun Zhong. “Directionally convolutional networks for 3d shape segmentation”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2698–2707 (cit. on p. 16).

[XLG12] Chuhua Xian, Hongwei Lin, and Shuming Gao. “Automatic cage generation by improved OBBs for mesh deformation”. In: *The Visual Computer* 28.1 (2012), pp. 21–33 (cit. on p. 24).

[Xu+10] Kai Xu, Honghua Li, Hao Zhang, Daniel Cohen-Or, Yueshan Xiong, and Zhi-Quan Cheng. “Style-content Separation by Anisotropic Part Scales”. In: *ACM Trans. Graph.* 29.6 (2010), 184:1–184:10 (cit. on p. 99).

[Xu+11] Li Xu, Cewu Lu, Yi Xu, and Jiaya Jia. “Image smoothing via L 0 gradient minimization”. In: *ACM Transactions on Graphics (TOG)*. Vol. 30. 6. ACM. 2011, p. 174 (cit. on p. 59).

[Xu+18] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. “SpiderCNN: Deep Learning on Point Sets with Parameterized Convolutional Filters”. In: *Proc. Euro. Conf. on Computer Vision* (2018) (cit. on pp. 16, 28).

[Xu+19] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. “DISN: Deep Implicit Surface Network for High-quality Single-view 3D Reconstruction”. In: *Proc. IEEE Int. Conf. on Neural Information Processing Systems (NeurIPS)*.

Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/39059724f73a9969845dfe4146c5660e-Paper.pdf (cit. on p. 17).

[Xu+20] Yifan Xu, Tianqi Fan, Yi Yuan, and Gurprit Singh. *Ladybird: Quasi-Monte Carlo Sampling for Deep Implicit Field Based 3D Reconstruction with Symmetry*. arXiv preprint arXiv:2007.13393. 2020 (cit. on pp. 17, 21).

[Yan+16] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. "Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision". In: *In Advances in Neural Information Processing Systems (NIPS)*. 2016, pp. 1696–1704 (cit. on p. 20).

[Yan+18] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. "Foldingnet: Point cloud auto-encoder via deep grid deformation". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. Vol. 3. 2018 (cit. on pp. 16, 33).

[Yan+20] Haotian Yang, Hao Zhu, Yanru Wang, Mingkai Huang, Qiu Shen, Ruigang Yang, and Xun Cao. "FaceScape: a Large-scale High Quality 3D Face Dataset and Detailed Riggable 3D Face Prediction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 (cit. on p. 75).

[Yar+20] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. "Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance". In: *Advances in Neural Information Processing Systems* 33 (2020) (cit. on pp. 17, 21, 67, 89).

[Yif+19a] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. "Differentiable Surface Splatting for Point-based Geometry Processing". In: *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA)* 38.6 (2019).

[Yif+19b] Wang Yifan, Shihao Wu, Hui Huang, Daniel Cohen-Or, and Olga Sorkine-Hornung. "Patch-based Progressive 3D Point Set Upsampling". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5958–5967 (cit. on pp. 16, 90, 91).

[Yif+20a] Wang Yifan, Noam Aigerman, Vladimir G. Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. "Neural Cages for Detail-Preserving 3D Deformations". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[Yif+20b] Wang Yifan, Noam Aigerman, Vladimir G Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. "Neural cages for detail-preserving 3d deformations". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 75–83 (cit. on p. 22).

[Yif+21] Wang Yifan, Shihao Wu, Cengiz Oztireli, and Olga Sorkine-Hornung. "Iso-Points: Optimizing Neural Implicit Surfaces With Hybrid Representations". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 374–383.

[Yin+01] Lexing Ying, Aaron Hertzmann, Henning Biermann, and Denis Zorin. "Texture and shape synthesis on surfaces". In: *Eurographics Workshop on Rendering Techniques*. Springer. 2001, pp. 301–312 (cit. on p. 22).

[Yin+19] Kangxue Yin, Zhiqin Chen, Hui Huang, Daniel Cohen-Or, and Hao Zhang. "LOGAN: Unpaired Shape Transform in Latent Overcomplete Space". In: *ACM Trans. Graph.* 38.6 (2019), 198:1–198:13 (cit. on p. 25).

[YM16] M. E. Yumer and N. J. Mitra. "Learning Semantic Deformation Flows with 3D Convolutional Networks". In: *ECCV*. 2016 (cit. on p. 24).

[YRSH21] Wang Yifan, Lukas Rahmann, and Olga Sorkine-Hornung. *Geometry-Consistent Neural Shape Representation with Implicit Displacement Fields*. 2021. arXiv: `2106.05187 [cs.CV]`.

[Yu+18a] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. "EC-Net: an Edge-aware Point set Consolidation Network". In: *Proc. Euro. Conf. on Computer Vision* (2018) (cit. on pp. 16, 18, 36).

[Yu+18b] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. "PU-Net: Point Cloud Upsampling Network". In: *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2018, pp. 2790–2799 (cit. on pp. 16, 18, 28, 32, 36).

[Yu+21] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. "pixelNeRF: Neural Radiance Fields From One or Few Images". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 4578–4587 (cit. on p. 17).

[Yua+18] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. "PCN: Point Completion Network". In: *Proc. Int. Conf. on 3D Vision*. IEEE. 2018, pp. 728–737 (cit. on p. 16).

[ZF14] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *ECCV*. 2014 (cit. on p. 112).

[Zha+18a] Wentai Zhang, Haoliang Jiang, Zhangsihao Yang, Soji Yamakawa, Kenji Shimada, and Levent Burak Kara. "Data-driven Upsampling of Point Clouds". In: *arXiv preprint arXiv:1807.02740* (2018) (cit. on p. 18).

[Zha+18b] Yang Zhao, Guoqing Li, Wenjun Xie, Wei Jia, Hai Min, and Xiaoping Liu. "GUN: Gradual upsampling network for single image super-resolution". In: *IEEE Access* 6 (2018), pp. 39363–39374 (cit. on pp. 28, 30).

[Zha+19] Yongheng Zhao, Tolga Birdal, Haowen Deng, and Federico Tombari. "3D point capsule networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1009–1018 (cit. on p. 16).

[Zho+06] Kun Zhou, Xin Huang, Xi Wang, Yiying Tong, Mathieu Desbrun, Baining Guo, and Heung-Yeung Shum. "Mesh quilting for geometric texture synthesis". In: *ACM SIGGRAPH 2006 Papers*. 2006, pp. 690–697 (cit. on p. 22).

[Zho+07] Howard Zhou, Jie Sun, Greg Turk, and James M Rehg. "Terrain synthesis from digital elevation models". In: *IEEE transactions on visualization and computer graphics* 13.4 (2007), pp. 834–848 (cit. on p. 22).

[Zhu+17] Rui Zhu, Hamed Kiani Galoogahi, Chaoyang Wang, and Simon Lucey. "Rethinking reprojection: Closing the loop for pose-aware shape reconstruction from a single image". In: *Proc. Int. Conf. on Computer Vision*. 2017, pp. 57–65 (cit. on p. 20).

[Zhu+18] Chenyang Zhu, Kai Xu, Siddhartha Chaudhuri, Renjiao Yi, and Hao Zhang. "SCORES: Shape Composition with Recursive Substructure Priors". In: *ACM Trans. Graph.* 37.6 (2018) (cit. on p. 24).

[ZJ16] Qingnan Zhou and Alec Jacobson. “Thingi10K: A Dataset of 10,000 3D-Printing Models”. In: *arXiv preprint arXiv:1605.04797* (2016) (cit. on p. 75).

[Zwi+01] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. “Surface splatting”. In: *Proc. Conf. on Computer Graphics and Interactive techniques*. ACM. 2001, pp. 371–378 (cit. on pp. 20, 44–46).

[Zwi+02] Matthias Zwicker, Mark Pauly, Oliver Knoll, and Markus Gross. “Pointshop 3D: An interactive system for point-based surface editing”. In: *ACM Trans. on Graphics*. Vol. 21. 3. ACM. 2002, pp. 322–329 (cit. on p. 20).

[Zwi+04] Matthias Zwicker, Jussi Räsänen, Mario Botsch, Carsten Dachsbacher, and Mark Pauly. “Perspective accurate splatting”. In: *Proc. of Graphics interface*. Canadian Human-Computer Communications Society. 2004, pp. 247–254 (cit. on p. 20).