FROM DISCRETE TO CONTINUOUS: LEARNING 3D GEOMETRY FROM UNSTRUCTURED POINTS BY RANDOM CONTINUOUS SPACE QUERIES

MENG JIA

A DISSERTATION SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

DECEMBER 2024

Abstract

In this dissertation, we focus on generalizing recent point convolution methods and building well-behaved point-cloud 3D shape features to achieve more robust, invariant, and versatile implicit neural representations (INR) of 3D shapes.

In recent efforts to explore point-cloud based learning methods to improve 3D shape analysis, there has been much attention paid to the use of INR-based frameworks. Existing methods, however, mostly formulate models with an encoder-decoder architecture that incorporates a global shape embedding space, which often fails to model fine-grained local details efficiently, limiting overall generalization performance. To overcome this problem, we propose a convolutional feature space sampling operation (Dual-Feature Sampling or DFS) and develop a novel INR learning framework (Stochastic Continuous Function Learning or SCFL).

This framework is first adapted and evaluated for its use in surface reconstruction of generic objects from sparsely sampled point clouds, which is a task that has been extensively used to bench-mark INR 3D shape learning methods. This study demonstrates impressive capabilities of our method, namely: 1) an ability to faithfully recover fine details and uncommon shape characteristics; 2) improved robustness to point-cloud rotation; 3) flexibility to handle different levels of sparsity in the input point clouds; 4) significantly better generalization in the presence of unseen shape categories. In addition, the proposed DFS operator proposed for this framework is well-formulated and general enough that it can be easily made compatible

for integration into existing systems designed to address more complex 3D shape tasks.

In this work, we harness this powerful ability to represent shape, within a newly proposed SCFL-based occupancy network, applied to shape based processing problems in medical image registration and segmentation.

Specifically, our network is adapted and applied to two different, traditionally challenging problems: 1) liver image-to-physical registration; and 2) tumour-bearing whole brain segmentation. In both of these tasks, significant deformation can severely degrade and hinder performance. We illustrate however, that accuracy in both tasks can be considerably improved over baseline methods using our proposed network.

Finally, through the course of the investigations conducted, an intensive effort has been made throughout the dissertation to review, analyze and offer speculative insights into the features of these proposed innovations, their role in the configurations presented, as well as possible utility in other scenarios and configurations that may warrant future investigation. It is our hope that the work in this dissertation may help to spark new ideas to advance the state of the art in learning-based representation of 3D shapes and encourage more interest in novel applications of INR to solve real-world problems.

Acknowledgements

Throughout the study and research of this dissertation, I have received a great deal of support and assistance.

First of all, I would like to express my highest appreciation to my supervisor, Professor Matthew Kyan, for his tremendous help and support to my research work. With almost ten years of study under Professor Kyan's supervision and direction, I developed a sufficient ability to conduct research independently in this field. Meanwhile, I have enough freedom to explore topics with my own interests and enthusiasm during this period. I will certainly benefit from these outcomes for a whole lifetime.

I'm also thankful to my supervisory committee members, Professor Hui Jiang and Professor Petros Faloutsos. They provided me with much inspiration through the committee meetings and helped me pass the qualifying examination and dissertation proposal.

I also appreciate York University and the Department of Electrical Engineering and Computer Science (EECS) for the wonderful environment they created for effective study and comfortable living. Their support for international students is impressive.

Lastly, the most important acknowledgment is my deepest gratitude and love for my family, including my parents, wife, and child. I would not achieve anything meaningful without their support from varied aspects of human life.

Table of Contents

List of Tables

List of Figures

Abbreviations

AR	Augmented Reality
VR	Virtual Reality
CAD	Computer Aided Design
CNN	Convolutional Neural Network
GNN	Graph Neural Network
RNN	Recurrent Neural Network
INR	Implicit Neural Representation
DFS	Dual Feature Sampling
SCFL	Stochastic Continuous Function Learning
FEM	Finite Element Method
SDF	Signed Distance Field
RBF	Radial Basis Function
MLS	Moving Least Square
SVF	Stationary Velocity Field
k-NN	k-Nearest Neighbor
MLP	Multi-Layer Perceptron
SVM	Support Vector Machine

VAE	Variational AutoEncoder
GAN	Generative Adversarial Network
RBM	Restricted Boltzmann Machine
MRF	Markov Random Field
GMM	Gaussian Mixture Model
SOM	Self-Organizing Map
ICP	Iterative Closest Point
APSS	Algebraic Point Set Surfaces
DPSR	Differentiable Poisson Surface Reconstruction
VASE	Volume-Aware Surface Evolution
PCNN	Point Convolution Neural Network
ONet	Occupancy Network
CBN	Conditional Batch-Normalization
ReLU	Rectified Linear Unit
MISE	Multiresolution IsoSurface Extraction
LE	Linear Elastic
IoU	Intersection-over-Union
TRE	Target Registration Error
CBCT	Cone-Beam Computed Tomography
MRI	Magnetic Resonance Imaging
DTI	Diffusion Tensor Image

Chapter 1

Introduction

The past decades have witnessed tremendous advances in hardware and algorithms relating to digital acquisition, decomposition and interpretation of the visual environment, that have revolutionized our ability to model, recognize, and analyze of the world around us. We can even readily construct virtual worlds (Augmented Reality (AR)/Virtual Reality (VR)) that seek to depict real-world environments with striking accuracy and realism, or to synthesize imagined or hyper-realistic environments, equally compelling in their ability to mimic the traits one would associate with real-world environments. Toward this end, the ability to map and extract models of real-world geometries to be used as a basis for digitizing of objects, structures and scenes, has long been key. It is made possible with the tremendous efforts to develop 3D acquisition, surface reconstruction, geometric segmentation and registration. At the same time, with 3D shape data becoming more available and ubiquitous, there emerges a demand for effectively analyzing 3D shapes with data-driven prior-knowledge-enhanced intelligent systems. Problems like 3D shape recognition, target detection, and semantic segmentation require modeling prior distributions and using features as distinctive as possible across different classes. The persistent work toward these goals has led to the emergence of many sub-fields of study. We can roughly cluster these topics into two categories with

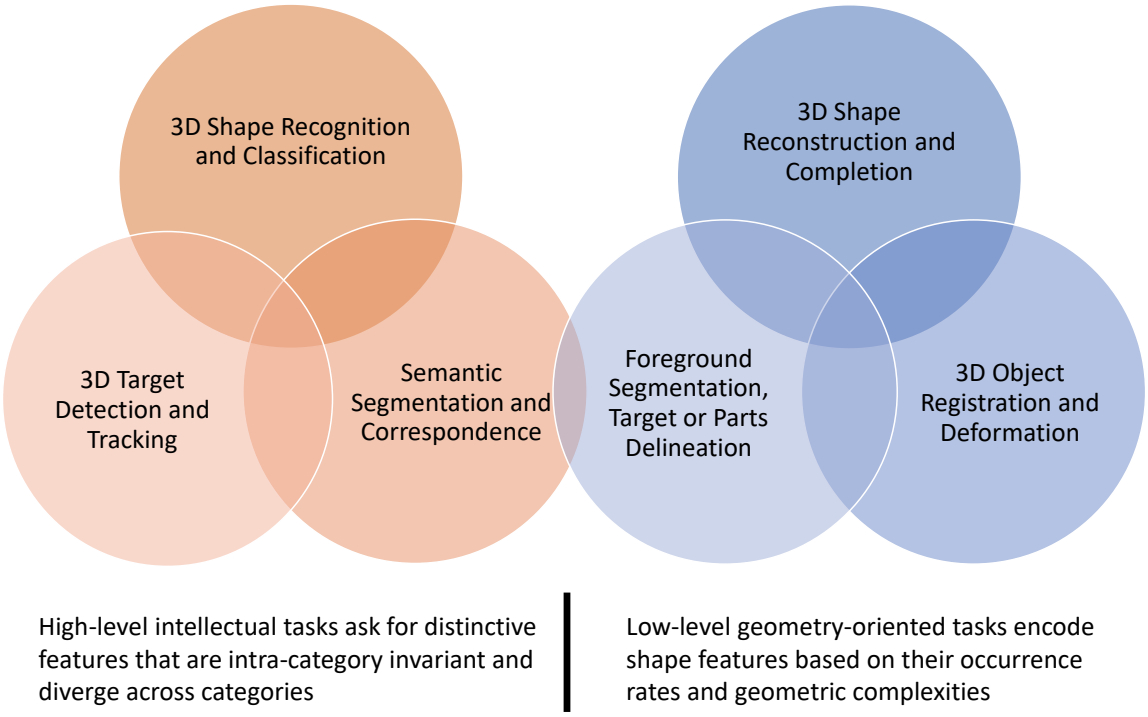


Figure 1.1: A diagram showing the two clusters of 3D shape tasks, which have many overlaps and are interconnected.

overlapping subcategories, as shown in Figure 1.1. The orange colored cycles indicate the ill-posed, semantic-level tasks such as inference, scene understanding, object recognition, and blue ones indicates the low-level geometry-oriented problems. Our study in this dissertation is scoped mainly to the second cluster. We progressively ventured into 3D object reconstruction, registration, and segmentation while introducing and evolving a new point cloud based deep learning innovation.

Surface reconstruction is the most characteristic problem in the second category. It refers to the task of capturing shapes from raw data retrieved from real-world objects and transforming that into a representative, concise, and usable form that fosters interpretation. Modern 3D acquisition technology has made it easy to scan and record the physical world as raw 3D data like sampled point clouds. However, the raw data see limited suitability

for numerical analyses, physical modeling, or computer graphics tasks. Instead, various applications require the scanned objects to be represented in the form of polygonal meshes or parametric surface representations. So, surface reconstruction is a crucial component of many real-world problems that require accurate environment or object geometry. The study itself has enjoyed a long and fruitful history as a sub-field of computer graphics. Moreover, the first cluster (orange) of topics in Figure 1.1, especially shape recognition, are originated mainly as pattern classification problems. The general probabilistic models and machine learning methods can easily be applied to the 3D domain. Some target detection and segmentation tasks (like those in Robotics and automated driving/control of vehicles) require semantic features and distinctive models commonly connected with both recognition and classification.

While the two clusters of topics have different origins, they are deeply interconnected in many aspects. For example, the semantic knowledge that goats are a four-limbed animal is clearly helpful for segmenting them from the grassland or finding their body correspondence with a horse, and a classifier will need other features to distinguish a four-footed chair from them. There are varied types of segmentation tasks. Some are closer to pattern recognition because they work with semantic or category-related features. For example, in indoor segmentation scenarios where desks may be present or not present in a room, a segment or just a labeled bounding box enclosing the region in which a desk exists in the scene immediately implies that a target of "desk" class is detected and recognized. In contrast, some segmentation tasks rely mainly on low-level features. Consider the segmentation (or delineation) of the hippocampus in healthy brain MR images as a example. The hippocampus or other anatomic structures always exist and are situated within a relatively fixed contextual background. Spatial location, characteristic shapes, and grayscale magnitude can effectively help to segment from the surrounding tissues.

Meanwhile, 3D registration and segmentation are sometimes highly entangled, particularly when considering information from different scanning/acquisition modalities, when dealing

with image scans from different frames of reference or captured via multiple sensors, or when turning raw geometric scans to mesh or volume shapes ready for inferring high-level information. For optical scanning that samples object surfaces from different viewpoints, registration operations are required to bring the range scans into one coordinate system. This procedure may further introduce misalignment artifacts. For medical scanning techniques, the raw data are volumetric images, or rather, slices of images collocated across a common volume. People usually have to segment the desired object first (delineation) before any medical analysis or data processing. Meanwhile, multi-modality registration (non-rigid) and fusion of data from across modalities could be conducted to overcome limitations in resolving particular structures encountered with individual modalities. In many applications, segmentation and registration should be considered jointly to achieve optimum results.

Through years of progress, studies in fields benefit mutually from each other. For example, the pattern recognition techniques and features developed for object classification may easily be extended to segmentation or target detection. The most noticeable and attractive advance is probably deep learning, which has been used in many active research topics: 3D shape classification [1, 2, 3, 4], indoor scene segmentation [5], 3D target detection [6], shape completion or reconstruction from partial views [7, 8], embedding of human pose [9]. In the last decade, Convolutional Neural Network (CNN) based deep learning approaches have provided state-of-the-art performance on many computer vision tasks, making deep learning approaches predominant over many pattern recognition problems. This success has also encouraged an enormous interest toward establishing deep learning architectures for 3D shape-related tasks.

However, when reviewing the broad literature, we identify that most learning models and 3D features initially proposed for high-level pattern recognition tasks are directly transferred into geometry-oriented problems like surface reconstruction and 2D/3D image registration. On one hand, these tasks typically require 3D shape features of local structural

representations, while high-level tasks need representations that are better suited to global semantic information and to support feature aggregation and fusion. On the other hand, traditional methods solve shape analysis problems with discriminative or generative models within feed-forward networks, which may often be unsuitable for geometry-oriented tasks that are usually formulated as optimization problems. A thriving topic, Implicit Neural Representation (INR), may often be proposed to fill this gap. INRs aim to learn implicit functions of 3D shapes with deep neural networks. Simply speaking, these methods train a network to model Signed Distance Fields (SDFs) [10, 11, 12, 13, 14] or occupancy functions [15, 16, 17, 18] of 3D shapes: parts of surface, objects, and scene environments. The works proposed in this dissertation are our attempts in this direction. Our initial work in INR is concurrent and independent of other early approaches. Its early development and evaluation are influenced by PCNN [3] and Occupancy-Net [16]. Our study is later accelerated with other inspirations from Flex-Convolution [19], DPC-ONet [20], Brewer et al. [21], Heiselman et al. [22], VoxelMorph [23], and SamSeg [24].

1.1 Motivations and Scope of Research

There are many reasons to prefer using point clouds as a universal form of 3D data, so we pay attention to developing networks that can directly take point clouds as input and extract point-based features. We notice some recent learning methods that define convolution operations for point sets to extract per-point features while still mathematically preserving a desired spatial paradigm. This observation helps us generalize such convolutional features to off-surface points (queries) randomly sampled from the space around input point clouds. We propose a method to infuse such networks with the ability to sample randomly within the input space to form queries.

As mentioned, implicit neural representations have received much attention and develop-

ment in recent studies. However, many INR-based 3D shape learning methods are developed initially for regular shape generative problems. If directly using the popular encoder-decoder network structure in surface reconstruction, they are usually incapable of capturing local geometry details [16] and not robust to data imperfections (e.g. due to noise, sample sparsity, or unexpected pose/orientation) and geometric variations. Instead, they attempt to reconstruct a generalized or eigen form from a set of learned exemplars within a category from a training set. In addition, they typically use a single MLP to model the implicit function. MLPs take a 3D coordinate (query point) as the input and return the scalar function value at this point, i.e., $MLP\,(q) : \mathbb{R}^3 \rightarrow \mathbb{R}$. As pointed out by [25, 26], this type of global function mapping is inefficient for modeling fine-grained local details.

To overcome this problem, we propose a random feature space sampling modification of the point-wise convolution-based PCNN [3] operator and Flex-Conv [19] architecture. This innovation shows great promise for learning shape implicit functions, enhancing the network's 3D shape reconstruction and representation ability. Our method is first validated with a well defined proof-of-concept task: 3D shape reconstruction from sparse point cloud samples. Moreover, this method maps unstructured 3D points to implicit continuous 3D functions in a versatile manner, making it applicable to many 3D shape-related tasks. The continuous and differentiable properties inherent in the INR representations are leveraged in two different applications, namely liver CT image-to-physical registration and tumor-bearing brain MR segmentation.

1.1.1 Machine Learning of Point Clouds

The study of 3D machine learning is an attractive and active yet challenging and complex topic, as 3D data involves different forms of representation: RGB-D images, voxel grids, point clouds, 2D surfaces represented as triangular or quadrilateral meshes, 3D volumetric

representations using tetrahedrons or hexahedrons, parametric surface, etc. Among these forms, directly analyzing or processing point clouds with deep learning approaches is appealing to researchers. There are at least three reasons for focusing on point clouds. First, because of their simplicity and flexibility, point clouds are among the most widely used data types for 3D geometry and shapes, representing objects from microscopic to space-science scales. Second, the raw data generated from depth sensors are commonly found in the form of point clouds or range images. Transforming point clouds to more structured volumetric grids or rendering them into a collection of multi-view images will often introduce many artifacts and obscure the natural invariance of 3D shapes under geometric transformations. Third, other 3D representations, multi-view RGB(D) images, voxel grids, polygonal meshes, and primitive-based models can easily be transformed into point clouds. In other words, they are more general than other 3D representations.

In the last five years, point cloud deep learning has received enormous attention [27]. Varied methods are proposed to solve or improve 3D-shape-involved problems like surface reconstruction, point cloud super-resolution, partial-view completion or mending, 3D shape classification, 3D object detection and tracking, and 3D segmentation [27]. Most notably, the success of PointNet [4] has inspired many variants and encouraged concerted efforts toward achieving a general point-wise learning framework that could be applied to point clouds [27], just like Convolutional Neural Networks (CNNs [28]) upon images. Researchers focus on benchmarking their methods with tasks and datasets like shape classification/detection, segmentation, and surface-normal estimation because those tasks are well-defined and easily fit into an end-to-end discriminative or regression model. However, these works are usually not directly helpful for solving real-world problems, which are generally difficult to formulate as a discriminative task or in circumstances that make collecting training data infeasible. In addition, point-wise MLP methods like PointNet [4] can not fully deliver the virtue of deep learning like hierarchical structure, local-global aggregation, and the ability of manipulating

inputs. The most noticeable difficulty of applying learning methods on point clouds is the unstructured (or order-less) nature of using a set of points to represent 3D shapes. Researchers in the field also proposed generalized convolution methods [29, 30, 3, 31, 32, 33, 34] and graph-based methods [20, 35, 36, 33, 32] to solve the problems.

1.1.2 Implicit Neural Representations

Implicit Neural Representations (INRs) are an emerging topic in machine learning. Essentially, these representations are a novel way of parameterizing continuous signals of varied kinds using neural networks. This paradigm offers several benefits over conventional methods with other 3D representations and brings many possibilities for solving tasks that are intractable with discriminative models and feed-forward-style inference mechanisms. Firstly, they are more memory efficient for modeling fine details than discrete representations (voxels or faces) restricted by the grid resolution. More importantly, they offer a more universal structure and a unified framework for low-level geometric tasks like 3D reconstruction, registration, segmentation, and delineation. The implicit representation is a continuous function in 3D space that is analytically computed, sometimes has closed-form continuous derivatives w.r.t. the spatial coordinates. This property makes such representations a suitable toolbox for solving optimization and inverse problems.

For a long time, implicit functions like signed distance fields and occupancy functions have shown to be useful 3D shape representations for numerous computer geometric and graphic tasks. Learning such implicit shape representation as a function in the Euclidean domain is the first topic in which research into INR's have focused. This early exploration has triggered research that applies INRs to signals on non-Euclidean domains like graphs [37]. In addition, there is a large branch of studies aimed at learning implicit representations of objects and scene appearance (also known as neural rendering [38, 39] or 4D light fields

[40]), or even to inclusion of dynamic temporal information [41]. Since our interests are in learning-based methods of processing shapes in point cloud form, we pay attention only to INRs in the Euclidean domain, especially ones that can from the point cloud domain and map to implicit function domains. The early research of point cloud INRs diverges into a wide variety of continuous functions for shape representations. Several different implicit representations of geometry are used in INR frameworks: signed distance functions [11, 14], unsigned distance functions [12, 13, 42], occupancy functions [16, 17, 25], on-surface priors [43, 44], radiance functions [39, 45], and vector fields modeling medial axis [46] and surface normals [8]. Each of them demonstrates unique advantages and drawbacks. Along with those representations, a lot of the recent work tries to harness implicit neural representations to build models for a variety of tasks: part segmentation [4, 47, 48], surface reconstruction [16, 11, 14], shape completion [7, 8], style transform [49], and 3D shape generation [50, 51].

Our INR framework shares many comment characteristics with other methods [11, 16, 44, 25]. So it's appropriate to describe them here, as an introduction to generalized Euclidean domain INR. A high-level illustration of INR learning is shown in Figure 1.2. We aim to learn the mapping from input point clouds to an occupancy function in a supervised manner, thus, the workflow can be split into training vs evaluation phases.

The network takes two signals (the input point cloud and a query point) as inputs and outputs a scalar as the predicted implicit function value. In other words, it maps the cross-product space of input shape domain and query in $\mathbb{R}^3$ to a scalar output that indicates the predicted binary occupancy state of this query point, i.e., $\mathcal{X} \times \mathbb{R}^3 \rightarrow [0, 1]$. Abstractly, it represents a mapping from input point cloud $X \in \mathcal{X}$ to a continuous 3D scalar function $\phi(X, \cdot) : \mathbb{R}^3 \rightarrow \mathbb{R}$. To help visual understanding of this mapping, we usually consider the input point cloud been clamped to one of the input slots of the network ϕ. Together with such an input, it parameterizes a function in $C(\mathbb{R}^3, \mathbb{R})$. The 3D function is already determined, even if we do not evaluate it with any query. In fact, infinitely many query points can be

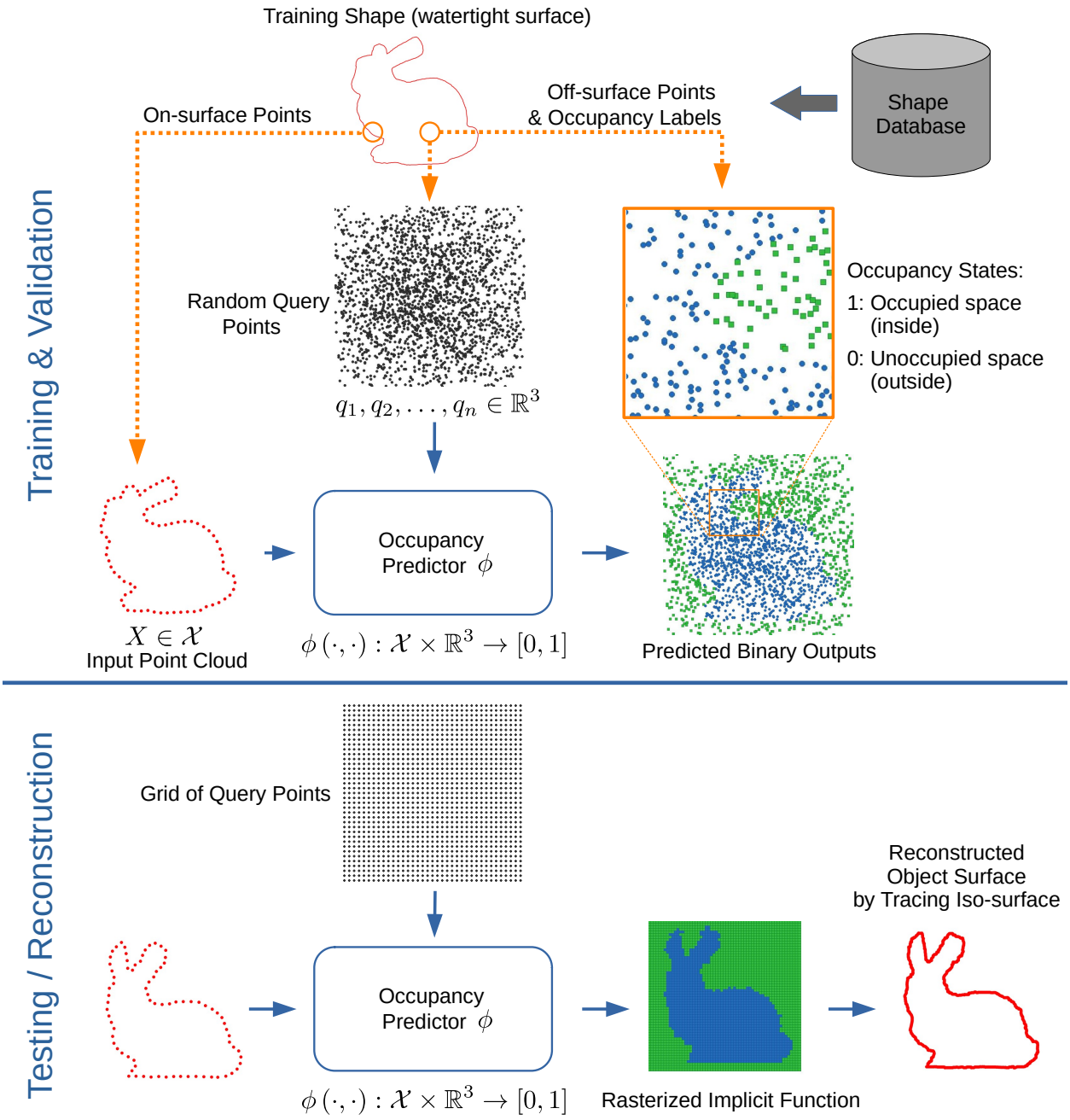


Figure 1.2: A high-level diagram of our INR frame for supervised learning occupancy functions.

sampled from continuous space. While, they are usually fed to the network in batch, the queries are answered as if they are individual actions.

Benefiting from this flexibility, we distribute query points differently in training/testing and reconstruction setups. For building training data, we take a collection of 3D shapes in water-tight meshes to sample points and occupancy labels. The point clouds are sampled randomly from the surface mesh, instance-by-instance of the shape dataset. The corresponding query points are sampled in space around the surface and their labels are determined with inside-outside test. The point cloud, the corresponding query points around it, and the labels associated with each set of queries are combined together to form one instance of training. The network is then trained as a binary classifier by minimizing errors of the predicted occupancy state. If some query points are miss-classified given an input (i.e., $\phi(X, q) \neq o(q)$), a miss-match is implied between the ground-truth mesh surface and the iso-surface of the predicted occupancy function $\phi(X, \cdot)$, at which point, weights of the network will be modified to eliminate this miss-match. Then, the next training point cloud is fed into the network to supervise it producing another occupancy function. In this way, $\phi(\cdot, \cdot)$ becomes capable for parameterizing a collection of varied shapes.

Often, training point clouds are sparse and noisy and traditional surface reconstruction methods may not recover the original shape accurately. In low-level terms, the network learns de-noising, interpolation, and sharpening in local space. A higher (more semantic) levels, the network can learn shape priors from the dataset which contains classes of objects. Shapes from the same category share similar global shape profiles. With proper training data, we expect the network can effectively learn feature maps that encode both local geometry-level and global semantic-level prior knowledge. In addition, the network is totally category-agnostic as it only works to map occupancy functions and minimize shape errors.

After training, the network weights are fixed. Given an unseen imperfect point cloud (from one of the categories learned, but not explicitly labeled), we can reconstruct its shape with

this trained network. For example, by feeding a grid of queries and collecting the outputs, a dense volumetric shape is constructed. The iso-surface can also be extracted, forming parametric surfaces or triangular meshes. Notice, the dense volume of query illustration is over-simplified. We implement two methods, coarse-to-fine octree structure and cross-surface triangulation, to efficiently sample and reconstruct the output surface.

Challenges of INR In the description above, we avoid talking about details about the network $\phi(\cdot,\cdot)$ that models implicit functions. Most 3D shape INR methods implement this function with an encoder and a decoder. They typically learn a low-dimensional shape embedding space in which vectors (acting as latent codes) are treated as global descriptions of the learned shape properties. The encoding component maps input shapes to the latent codes $\phi^E : X_i \mapsto Z_i, Z_i \in \mathbb{R}^{dimE}$. The decoder $\phi^D : \mathbb{R}^{dimE} \times \mathbb{R}^3 \mapsto \mathbb{R}$ is a feed-forward network that maps a query to an implicit function value, conditioned on the latent codes. There are mainly two ways to achieve this conditioning model: 1) conditioning via concatenation and 2) conditioning via hyper-networks.

The encoder-decoder architecture commonly used in recent INR works is an effective framework for shape generation and completion tasks that heavily rely on the learned object and semantic priors, like the symmetry of aircraft, the topology of a doughnut, and the number of limbs of goats and ducks. However, this global embedding design implies an assumption that objects and scenes of varied classes have shape and appearance properties that can be well characterized by the variation of a low-dimensional latent code. In challenging surface reconstruction tasks, this method may inevitably lose locality information, ignore uncommon characteristics, and be inefficient for encoding geometric details. This problem has long been realized by researchers in INRs [25].

Some solutions have been proposed to mitigate these shortcomings. Convolutional occupancy methods [17, 25] produce fine-grained implicit 3D reconstruction by combining

convolutional encoders with implicit occupancy decoders. Another similar solution uses hybrid voxel-grid and implicit representations to fit complex 3D structures. Basically, this approach extends the *single* latent code to *a set of* local codes on a coarse regular 3D grid. To answer an occupancy query, the local feature is then computed by interpolating its neighboring local codes. Some other studies suggest incorporating high-frequency functions into the network to fit the data containing high-frequency variations. For example, Mildenhall et al. [39] propose positional encoding that maps the input coordinates with sin/cos function in a range of frequency bands. Sitzmann et al. [26] build implicit neural representation systems with periodic activation functions that efficiently parameterize images, videos, audio, and 3D shapes.

Our Method and Its Applications The work presented in this dissertation could be viewed as an alternative solution to this problem. The main idea is to unify the input points' feature space and query points' feature space so that more fine-grained local details are preserved, and learning becomes more efficient. In so doing, we don't have to take the low dimensional global embedding assumption and pass a latent code from an encoder to a decoder. The global feature aggregation and high-level information representation is achieved by the point-based spatial pooling/un-pooling and the U-Net style network structure. Compared with conditioning-via-concatenation and conditioning-via-hypernetwork methods, the split of encoder and decoder is completely avoided. In our approach, the query points are incorporated in concert into the point-convolutional U-Net architecture that predicts the corresponding occupancy or other continuous values. We select the well-defined convolutional operation based point cloud networks, namely PCNN [3] and FlexConv [19], in which the discrete operation implies a hidden continuous feature field. We can then define a *sample operation* based on the hidden continuous field, in which the local feature vectors are sampled at the query points and mapped individually to the target occupancy value. Inherently, the

feature mapping and shape implicit prior preserves local information and irregular fine details, because the point convolutional feature mapping is a well-defined and well-behaved local operation. In our design, global shape is extracted with the U-Net-style architecture and aggregated hierarchically with k-Nearest Neighbor (k-NN) based spatial pooling.

We name this approach "conditioning via sampling" to emphasize its difference from "conditioning via concatenation" and "conditioning via hypernetwork". Details are explained following a brief literature review of INR in section 2.5.

We implement occupancy function learning based surface reconstruction with the proposed method. This is not only a fundamental validation of our method, but also acts as a proof-of-concept study for the remainder of this dissertation. The experiments conducted in this concept study show that our network can produce more reliable occupancy functions from sparse point cloud input, implying that the local geometric features and global distributions (shape prior) are all learned efficiently. We argue in this dissertation, that this is because of weight sharing, strong regularization, the local-receptive nature of the network configuration, and convenient feature aggregation of the points into convolutional operations.

With the success of the first surface reconstruction study, a Stochastic Continuous Function Learning (SCFL) paradigm is established. It is then applied to two medical applications that require processing shape of organs. First, we implement liver non-rigid image-to-physical registration system in which SCFL is used to produce an implicit representation from a sparse sampling of liver surface. Second, we use SCFL to learn tumor-induced tissue deformation in human brain, and improve tumor-bearing whole-brain segmentation with our learned model. In our studies, the proposed SCFL is easily adapted to be a key component within considerably different applications: object surface reconstruction, image-to-physical registration, and atlas-based brain segmentation. The conducted experiments show considerable performance improvements and other advantages. The significance of these applications is twofold. On one hand, the chosen applications validate and demonstrate the flexibility and versatility of

our SCFL framework. On the other hand, the way SCFL solves those different problems is allows for opportunities to integrate domain knowledge into the network when synthesizing training data, which suggests a practical approach for applying state-of-the-art deep learning methods to domain-specific real-world problems.

The key features of SCFL include 1) using point convolutional operators to learn 3D shape features directly from input point clouds; 2) equipping the point convolutional layers with Dual-Feature Sampling (DFS) operators to assist in the extraction of learnable features from across the 3D input space; 3) randomized query point selection strategy to control supervised training, with the training data being pairs of query points and corresponding function values, and 4) encoding a supervisory signal or task-related knowledge by properly synthesizing realistic data, overcoming the lack of real-world data available for training.

1.2 Contributions

An overview, summarizing the relationships between the research topics studied in this dissertation is shown in Figure 1.3. The proposed approach falls into the category of learning-based shape representation and reconstruction. Comparing with traditional (non-learning based) surface reconstruction methods, these methods may benefit from prior knowledge learned about the shape they are attempting to reconstruct, and can potentially rebuild shapes from very sparse samples or even partial views. More specifically, our work lies at the intersection of two sub-topics: 1) INR of 3D shapes and 2) deep learning of point clouds. We generalize recent point convolution learning methods (PCNN and FlexConvolution) by defining a DFS operation, which is used in INR to learn efficient localized feature maps that may be associated with query points. In addition to a benchmarking of this method in object shape reconstruction (Chapter 4), we also apply this method in two medical image tasks: liver registration (Chapter 5) and brain segmentation (Chapter 6).

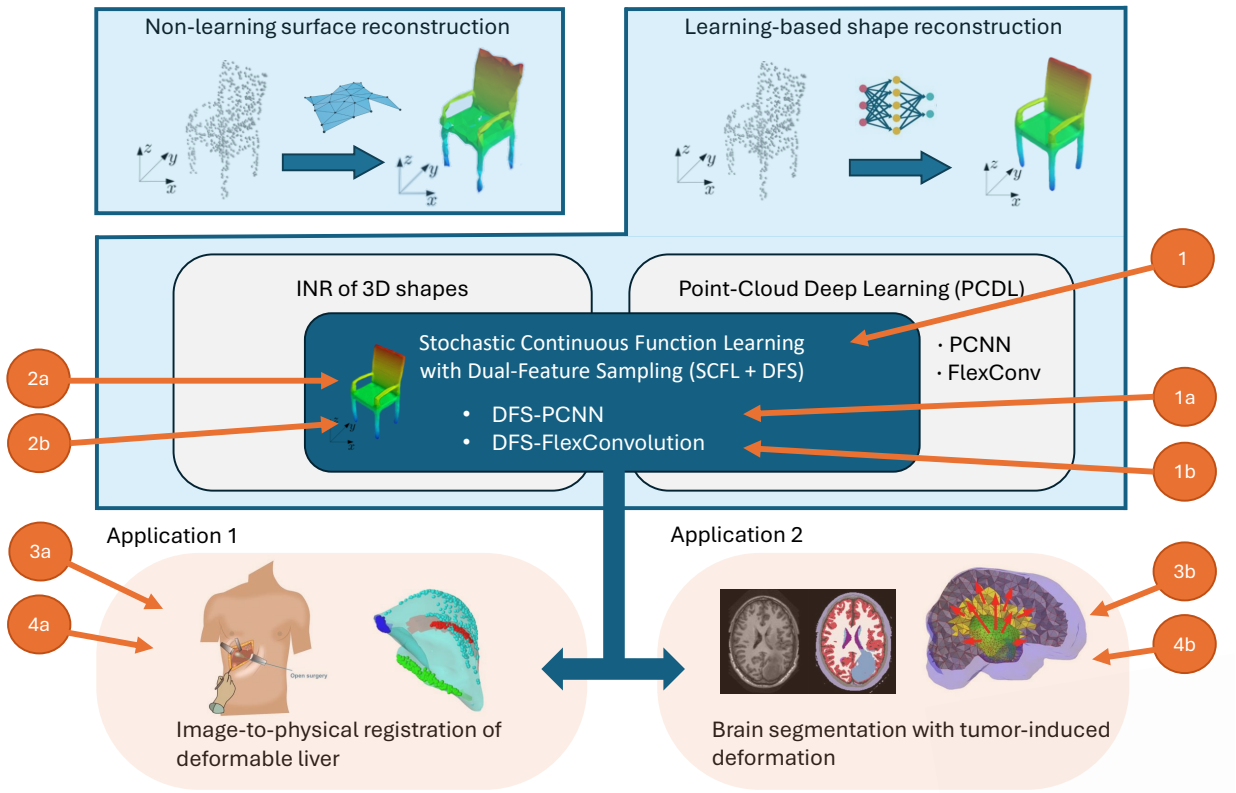


Figure 1.3: A diagram showing of the relation of the research topics in this study. The small orange circles enumerate the corresponding contributions made in the dissertation (as described in Section 1.2).

The main contributions of our work are enumerated below, grouped into three main categories (which are each indicated as corresponding circles illustrated in Figure 1.3):

1. Our work instantiates a new methodology of point cloud to implicit function mapping with deep neural networks. This novel INR learning method harnesses a local sampling operation (DFS) that unifies the feature space of input point cloud and associated query points. We implemented two different versions of DFS operation: 1a) PCNN and 1b) Flex-Convolution based. The proposed DFS operators in this framework are well-formulated and general enough that they may be compatible with many other point cloud convolution architectures.

2. An evaluation of the proposed INR with the sparse point surface reconstruction problem.

2a) Our method is evaluated for its performance in terms of reconstruction accuracy, robustness to deviations/asymmetries in observed versus learned shapes, sparseness of observations, and domain generalization. 2b) We propose an accelerated method for building triangle meshes from the output INR shapes.

3. We apply the SCFL paradigm to challenging medical problems involving deformable organ shape reconstruction, registration, and segmentation. 3a) In image-guided liver surgery, we train a network to reconstruct the liver from sparse samples and guide image-to-physical registration. 3b) In tumor-bearing brain anatomical segmentation, the network learns tumor-induced deformation to assist atlas-based brain segmentation. These works are published at top-level conferences gaining exposure to research communities in both machine learning and medical studies.

4. The applications in this dissertation all rely on the collation or generation of proper training data. In the two medical applications, we synthesize training data to pass the required information to the network to mitigate challenges when availability of training data is limited. Specifically, in 4a), we propose a linear elastic model for synthesizing training data (of randomly deformed liver shapes) for the INR model in liver registration task. Then, in 4b) we simulate brain tumor growth with a diffusion-reaction biochemical model to generate pairs of random tumors and the deformation fields that result, based on mass-effect, edema, and infiltration. In so, this work validates a paradigm that can help to solve complex problems in which human knowledge is hard to encode or where tasks are too expensive to collect real data with adequate ground truth.

1.3 Dissertation Outline

In this dissertation, we begin with a review of pertinent literature (Chapter 2) which provides some basic knowledge and background about 3D shape data representation and surface reconstruction. We also review point cloud deep learning, learning-based surface reconstruction, and implicit neural representation of 3D shapes. Then, Chapter 3 presents the foundational formulation and methodology of this dissertation, Dual-Feature Sampling (DFS) and Stochastic Continuous Function Learning (SCFL). Chapter 4 presents our study of using DFS and SCFL to build a novel occupancy network and surface reconstruction method. The outcome of this study leads to the medical applications explored in this dissertation. Chapter 5 presents the first of these applications, where SCFL is used to model sparsely sampled physical livers and serves as the guide for pre-surgery CT image registration. Chapter 6 presents a generalization of SCFL, which is used to learn deformation fields that can help guide and improve full brain segmentation in the presence of large tumor pathologies. Finally, in Chapter 7, we conclude our study, reviewing the contributions with a discussion for further research directions for each aspect of our work.

Chapter 2

Background and Literature Review

In this chapter, we review the history and related works, beginning with the concepts of shape representation and surface reconstruction. Some of the terminologies are overloaded and abused, so we try to clarify and use them consistently across this dissertation (Section 2.1). As a prerequisite for any 3D application, geometry acquisition techniques are briefly reviewed in Section 2.1.1. The remainder of this chapter deals exclusively with surface reconstruction and learning shape representations.

We acknowledge that there are a multitude of comprehensive and insightful surveys [52, 53, 54, 55, 56] of surface reconstruction, each have a different categorization of the existing methods. For example, Berger et al. [52] classify methods according to the shape priors on which they rely. Sulzer et al. [56] classify methods as surface-based versus volume-based reconstruction (sometimes also referred as interpolation approaches versus approximation approaches). In our work, we classify as non-learning deterministic versus learning-based methods.

We review traditional non-learning methods with attention paid more to their frameworks and mathematical formulations, serving as a baseline for a deeper look into learning based approaches. They are sub-classified according to either their computation of discrete points

and triangles (triangulation-based methods - section2.2.1) or on continuous domains (implicit functions and Moving Least Square (MLS) - section 2.2.2).

Section 2.3 can be viewed as a survey of deep learning methods generally targeted at point cloud data, in which generalized convolution approaches provide the foundation of our work. Lastly, a more focused review is made in learning based surface reconstruction (section 2.4) and implicit neural representation methods (section 2.5).

2.1 Basics: Geometric Acquisition and Representations of 3D Shapes

2.1.1 Geometry Acquisition and Processing

Real-world applications typically start from geometry acquisition: scanning objects, scenes and tissues with specialty sensors and devices. Techniques for 3D scanning range from mechanical (touch probes), range or remote sensing, optical scanning or acquisition of light through traditional lens based optics, through to non-invasive medical imaging techniques like Ultrasound, CT, and MRI. Mechanical touch probes are the most accurate, however their use involves tedious and prohibitive manual human effort, so its application is restricted. Optical methods, including active-lighting-based scanning [1], passive multi-view stereo, and even the collection of multiple static images via traditional cameras, is relatively cheap and thus, more available. Nowadays, consumer level devices like Microsoft Kinect and LiDAR sensors are being used actively by hobbyists and professionals alike, generating large amounts of 3D data every day. The raw scanning data of these types of devices includes range images (depth maps), that usually contain extensive noise and outliers. Moreover, scanning a complex non-convex object will often yield artifacts such as malformed geometries, or holes due to

[1] An incomplete list of active-lighting methods: Time of flight laser, Triangulation laser, Structured light

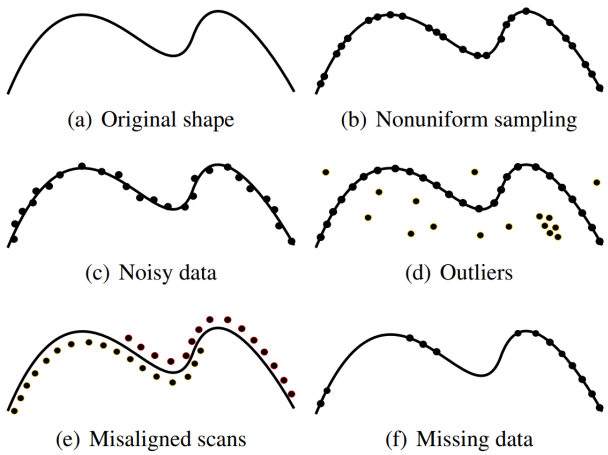


Figure 2.1: Different forms of point cloud defects, illustrated in the case of a curve in 2D. Reprinted from Berger et al. [52].

missing data, occlusion and non-uniform sampling. Figure 2.1 sketches many kinds of these defects that are generally emphasized in reconstruction of surface geometries from 3D point clouds. Volumetric acquisition such as those found in many 3D medical imaging modalities provide abilities to sample the inside of objects in a non-invasive manner. The resolution of these techniques is relative low (compared with what is required for applications like computer-assisted surgery). Often one modality of scanning reveals a particular aspect of the material or tissue under investigation. Ultrasound Sonography is sensitive to reflection of ultrasound waves at the boundary between tissues, while CT is sensitive to a tissue's ability to absorb X-rays, and MRI is sensitive to water content/water density within different tissues (detected via a tissue's respondance within a changing magnetic field).

In addition to being noisy and incomplete, the raw data produced from these 3D sensors are typically nonparametric, discrete, and unstructured. There is a high demand to efficiently store, process, and analyze acquired 3D shapes in varied applications. Surface reconstruction methods, as a result, are often framed with respect to how they address problematic aspects inherent in the raw data. Generally, these methods work to recover 3D shape from imperfect

raw data to produce noise-free and complete output in forms that are suitable for particular applications.

2.1.2 Representations of Curves and Surfaces

A mathematic representation of curves and surfaces is a fundamental component in Computer Aided Design (CAD), computer graphics, and 3D computer vision. Formally, there are two main types of analytic representation of surfaces: explicit and implicit.

Explicit Representation This type can be sub-divided as axis-independent versus axis-dependent explicit representation.

The coordinates (x, y, z) of on-surface points are expressed as functions of two parameters u and v

$$x = X(u, v), y = Y(u, v), z = Z(u, v). \tag{2.1}$$

This type of representation is the most convenient in computer graphics practice as it is an axis-independent form that can be easily transformed, traced and manipulated [57]. For such reason, surface parameterization is well studied [58] and is dedicated to task that build a one-to-one mapping from a parameter domain (such as a sphere) to the observed/scanned object surface (mostly stored as unstructured meshes). Recently, deep learning approaches are used for fitting B-spline curves and surfaces [59, 60] and surface parameterization [61].

An axis-dependent explicit representation is given by

$$z = g(x, y), \tag{2.2}$$

such that the x and y axis *directly* act as the parameters u, v in Equation 2.2. The main disadvantage of this form is the infinite slope problem: where the locus of the surface perpendicular to $x - y$ plane becomes impossible to represent. The surface in this form is

also hard to transform and manipulate. However, it is powerful if used locally in the surface tangent space. Moving Least Square (MLS) is an example of such a local surface-normal-based approach for point cloud reconstruction. In Projection MLS [62, 27] for instance, the polynomial height functions are defined in the estimated surface normal and fitted to a local surface. Most noticeably, as such functions are 2D, they become very convenient for applying convolutional methods in the local tangent space. In recent point cloud deep learning, Roveri et al. [63] render points to multiple view-based depth images (like $g(\cdot)$ in Equation 2.2) and feed them to ResNet [64], a convolutional neural network used to accumulate global shape features.

Implicit Representation Given a 3D scalar function $f(\cdot) : \mathbb{R}^3 \rightarrow \mathbb{R}$, an implicit surface is defined as the level set of SDFs:

$$f(x, y, z) = C,$$

or the boundary of an occupancy/indicator functions:

$$f'(x, y, z) \neq 0.$$

From this definition, the efficiency of tracing contours from an implicit function is immediately obvious. However, it still remains extremely challenging to achieve high-resolution reconstructions. Generally, piece-wise linear continuation methods [65] can be used to solve this problem. Relatively simple methods have been used thus far in learning-based surface reconstruction research. Most of the works (reviewed in section 2.4) transform learned implicit surface to meshes by sampling an indicator function over the voxelized space and extracting an iso-surface with the classical marching cubes algorithm [66].

While there are many parametric methods for analytically modeling 3D shape with explicit

and implicit representations, sometimes it is more convenient to use discrete, nonparametric and piece-wise approximation for 3D shapes (due to their ease of enumeration and modification, ability to capture arbitrary topology, and regular data format). Polygon meshes and voxel cartesian 3D grids are the discrete approximation of explicit and implicit 3D shapes, respectively.

The main focus of this work is on using implicit representation of shapes, and equipping them with a deep neural network's universal approximation capability. 3D shape representations using point clouds and implicit functions have the advantage that they do not impose a predetermined resolution, thus avoid the problem of cubic increases in complexity with 3D voxel grid representations. Our work moves toward an improved INR framework for surface reconstruction. In the rest of this chapter, we review methods that recover surface from sampled points. A rich and diverse array of surface reconstruction methods have been proposed in the past several decades. Based on what type of surface is produced as output, early approaches of surface reconstruction can be categorized as 1) Discrete (e.g. triangulation-based), or 2) Continuous (e.g. Moving-Least-Square MLS, and implicit surface reconstruction).

2.2 Traditional Non-learning Methods for Surface Reconstruction

2.2.1 Discrete Surface Reconstruction

Triangulation (or interpolation) based surface reconstruction methods are sometimes concisely referred to as "connect the points". The early work by Greg and Marc [67] is a famous and illustrative example of this category, in which they designed a 3D acquisition procedure that combines a collection of triangulation laser-scanned range images into a single polygonal

mesh. This connect-the-dots approach comes down to establishing connectivity between sampled points. Geometrically partitioning structures like Voronoi diagram or Delaunay triangulation can elegantly serve as the discrete representation of point-wise relations, and thus help to construct local point connectivity [68].

Surface based triangulation Tangent-plane triangulation methods [69, 70, 71] simply project the sampled points onto the locally estimated 2D tangent space and compute a Delaunay triangulation over the projected points. Restricted Delaunay based methods (like the Crust algorithm [72, 73]) come from the observation that tangent plane Delaunay methods only take a subset of triangulation $D(P)$, and aim to compute the Delaunay triangulation directly *restricted* to the surface. The problems that need to be addressed in these methods are: finding appropriate neighbourhood size and good centres for the estimating the tangent plane, making orientations across tangent planes consistent, and stitching the patches into a global mesh that maintains a topological manifold.

Volume based triangulation Inside/Outside labeling methods (like Boissonnat [74], Power-Crust [75], and Eigen-crust [76]) try to find an interface between the tetrahedrons inside (occupied) the 3D body and outside. They generate an approximate inside/outside labeling to sculpt the outside Delaunay tetrahedrons, and take the out-faces of remaining tetrahedrons to be the reconstructed triangle mesh. A famous method called Empty-Ball or α-Shape [77] represents a special type of inside/outside labeling, constructing surface triangles with the so called *empty-ball property*, i.e. a Voronoi polyhedron is formed from a set of points whose circumscribing ball is empty of additional sample points. This approach has been studied in-depth, and there are many variants that have been developed [78, 79, 80].

Triangulation methods take a large amount of sample points as input (more than needed for acting as vertices to reconstruct a fair triangle mesh), and take a subset (by eliminating

the noise and outlier points) to define faces and reconstruct a surface. However, if the input point clouds are non-uniform, sparse, incomplete, noisy, or misaligned, failures may happen. Without implementing extra interpolation and de-noising modules, such "connect the points" approaches are restricted to draw output vertices *only* from the input point set.

2.2.2 Continuous Surface Reconstruction

Instead of directly interpolating the surface with a special triangulation structure established upon the sampled points, it is usually advantageous to approximate the underlying surface with some parametric representations. Explicit approximations directly fit the observed points with a parameterized surface $(u, v) \mapsto p, p \in S$, while implicit representations aim to find a scale volumetric function $C(\mathbb{R}^3, \mathbb{R})$ that best explains the observed points with a level-set or iso-surface.

Explicit Surface-based Approximation

The goal in these approaches is to find a projection that maps a point in local reference domain (e.g. a tangent plane or simply $\mathbb{R}^3$) onto the target surface. For example, we can use a 2D polynomial function to approximate a patch of regular surface. Projection Moving Least squares (MLS) methods [62, 81] directly fit polynomials to the local surface itself. Given a point in the neighbourhood of sampled points $\mathbf{x} \in \Omega$, this method aims to find a function that *projects* $\mathbf{x}$ onto the surface S, i.e. $S = range(f(\mathbf{x})), \mathbf{x} \in \Omega$. Two steps are involved in a tangent plane based MLS method: 1) estimate the tangent plane H_x (parameterized as $n, q \in \mathbb{R}^3, \|n\| = 1$) of S near x, and 2) find local polynomial approximation to map from H_x to S, by minimizing an error term defined locally in that tangent plane reference space.

Theoretically, projection MLS is equivalent to a discrete-to-continuous interpolation or curve fitting from scattered points. In that sense, Alexa et al. [62] introduce a projection-

based method to advocate the use of up-sampled point sets to represent 3D shapes and apply them to a splatting rendering scheme. The original MLS methods include only positional constraints, which enforces exact approximation at the sample points. Guennebaud and Gross [82] (Algebraic Point Set Surfaces, APSS) incorporate derivative constraints, matching the gradients of implicit function and normals of corresponding sample points. Moreover, APSS fit local spheres instead of tangent planes, achieve better reconstruction quality and robustness.

Implicit Volume-based Approximation

Implicit function methods define the reconstructed surface as the level-set of a three-dimensional scalar function built from the input points. According to the implicit surface theorem, if c is a regular value of $f(p)$ (i.e. the differential of f is surjective at every preimage of c), then the c-level set is a two-dimensional manifold embedded in three-dimensional space. Visually, the contour lines on topographical maps are 1D curves embedded in 2D topographic height maps, excluding all extreme-value and saddle (irregular) points. Given a point set $P = \{p_i \in \mathbb{R}^3\}, i \in \{1, 2, \ldots, n\}$ that is sampled from an unknown surface S, the goal of implicit surface methods is to find a 3D scalar function $f(p)$ such that its level set combats imperfections in the point cloud P and best approximates surface S. This framework is corresponding to the implicit represent mentioned in Section 2.1.

One of the earliest works in this category is Hoppe et al. [69], in which the signed distance function $f(x)$ is approximated by the distance to the tangent plane associated with the sample point in P that is closet to x:

$$\begin{aligned} f(x) &= \operatorname{dist}(x, \operatorname{tangent}(p)) = (p - x), \\ p &: \min(\|p - x\|), p \in P. \end{aligned} \tag{2.3}$$

While this method reconstructs the surface through approximating signed distance function, the reconstructed surface is non-smooth at the interface of two Voronoi cells, making it not robust to imperfect scans and point cloud defects because every sample point (including outliers) is taken into account.

Radial Basis Function (RBF) Interpolation RBFs are frequently used for scattered data interpolation. Given a set of points with prescribed SDF values, this method reproduces the function with a linear combination of RBFs which retains a high-degree of smoothness. In [83, 84], SDFs are approximated as:

$$f(x) = t(x) + \sum_{i=0}^{n-1} w_i \varphi(\|x - c_i\|), \tag{2.4}$$

in which φ represents RBFs and c_i indicates the constraint points. $t(x)$ is a low-degree polynomial in 3D that accounts for thresholding of the iso-surface, and sample points can act as simple constraints $f(p_i) = 0, p_i \in P$. The type of RBF and constraint parameter must be carefully selected to avoid artifacts and achieves an appropriate trade-off between smoothness and detail preservation.

Poisson Surface Reconstruction Poisson surface reconstruction is arguably the most popular test-of-time approach. Kazhdan [85] considers an indicator function as the implicit representation of shapes, and the oriented point cloud could be viewed as a discrete gradient field of an indicator function χ. The surface reconstruction problem is thus turned into an optimization problem: to find a *softened* indicator function χ such that

$$\text{argmin}_{\chi} \int \|\nabla \chi(x) - \mathcal{N}(x)\|_2 \, dx, \tag{2.5}$$

in which $\mathcal{N}(x)$ indicates the gradient field revealed by the oriented point cloud $\{P, N\}$. Since the gradient is enforced to be zero in the area away from the surface, χ is smooth and tends to well-behaved in these areas.

There are a lot of variants of Poisson reconstruction. The method of [85] reforms the reverse derivative to estimate the Fourier transform of χ (represented by its Fourier coefficients). Kazhdan et al. [86] later introduced another method that directly solves this Poisson problem in a coarse-to-fine multi-grid hierarchical approach. Manson et al. [87] further improved this idea and made reconstructions on a subset of points at a time. The main problem with Poisson methods [85, 86, 87] is over-smoothing effect in directly attempting to fit $\nabla\chi$ with data points. Guennebaud and Gross [82] incorporate derivative constraints, and Kazhdan and Hoppe [88] explicitly use point clouds as positional constraints into the optimization (the Screened Poisson problem), so that this optimization encourages a tighter fit to the sample points. Similarly, Calakli and Taubin [89] prevent over-fitting by introducing a Hessian term of χ in Smooth Signed Distance (SSD) surface reconstruction.

2.3 Point Cloud Feature Learning with Networks

After witnessing the breakthrough and success made by deep learning in image processing and computer vision, point cloud deep-learning has received much recent attention. Most notably, the success of the pioneering work PointNet [4] and PointCNN [30] encouraged concerted efforts toward achieving a general point-based learning framework that could be applied to point cloud 3D shape like "pixel-based" Convolutional Neural Networks (CNNs) upon images. Most of these works try to learn an effective feature map or a shape embedding space for various point cloud understanding tasks: 3D shape classification [1, 4], object detection and tracking [6], point cloud segmentation [5]. From a machine learning perspective, these tasks are relatively easy to define and fit into an end-to-end supervised learning framework

with deep discriminative models, and the results and performance of methods are easy to benchmark with available labeled datasets [27]. In addition to these typical high-level tasks that require capture of the semantic properties of point clouds, machine learning methods have been applied to 1) low-level reconstruction problems like point cloud denoising, consolidation and up-sampling [90, 91, 92, 93], 2) 3D registration [94, 95, 96], 3) data-driven prior based shape recognition [97], and 4) generative models of point clouds [10, 98, 99, 100].

While low-level 3D task studies obviously have a more direct relation with our intended surface reconstruction purpose, the outcome from works in point cloud understanding could be useful for surface reconstructions and other low-level tasks. For example, knowing what an object is and what properties are likely presented obviously help reconstructing an object and in its manipulation. More specifically for the interests of this dissertation, learning a good latent representation and local shape feature map is definitely an important factor for learning-based reconstruction and generation. Due to our surface reconstruction purpose, we focus our attention to methods that have the potential to be used generally as shape features for point clouds.

2.3.1 PointNet and Pointwise-MLP Learning Methods

PointNet [4] and related methods use a symmetric function (invariant to permutation) to aggregate point-level features into a global shape descriptor. They first feed every point to a learned differentiable function:

$$f\left(p_n\right) : \mathbb{R}^3 \rightarrow \mathbb{R}^m$$

in which m indicates the dimension of features. In PointNet, this function is implemented as a cascade of multi-layer perceptrons (MLPs). Through this function, the point set $\mathbb{R}^{n\times 3}$ is mapped to $\mathbb{R}^{n\times m}$ (m is 1024 in [4]). For classification tasks the architecture then employs a max-pooling (guaranteed to be invariant to the order of points) over point features,

$\mathbb{R}^{n\times m} \rightarrow \mathbb{R}^m$, aggregating per-point features to form an overall global feature vector. Max-pooling acts like bagging of features into a global description (or signature) of the input shape. This signature is ultimately associated with an appropriate class label through one final MLP mapping. After training on labeled point sets, unknown point sets are fed through this trained network yielding a set of scores for each class category. For point segmentation tasks, per-point features are formed by concatenating a point's MLP mapping together with its global features, such that information describing local point neighbourhoods can integrated together with global semantics.

The idea behind this method seems to be related to the universal approximation property of MLPs and implicit function theory. The point-level MLP projects a low-dimensional (3D) point to a high-dimensional ($m = 1024$ in PointNet) feature space that has large capacity for representing many shapes. Considering each dimension (one of 1024) of the output feature as a scalar function (field) in 3D, its level sets (or contour surfaces) represent 2D manifolds in the input 3D space. A winner point p_0 in max-pooling of a particular dimension will confirm a piece of information about the shape: all of the points in this shape are bounded by a specific contour surface $f(p) = f(p_0)$. m such activated contours may form a good shape descriptor of the input point cloud, and a general deep learning classifier could be built on this feature to learn recognition based on the shape. Figure 2.2 shows a 2D illustration of this shape encoding. The individual dimension of the feature mappings are plotted as the first column, along with its contour lines and the 2D points. The second column shows the winner points and corresponding contour lines of each dimension. The enclosed spaces of each active contour lines are combined (into a $m \times 1$ vector) to be a representation of the shape.

Despite its simplicity, PointNet is quite effective on point cloud classification and segmentation. In addition, it is very efficient for computation, because the point sets can be processed in their unstructured form (i.e. without requiring extensive structuring within regular volumetric grids or other reference frames typically required for understanding spatial

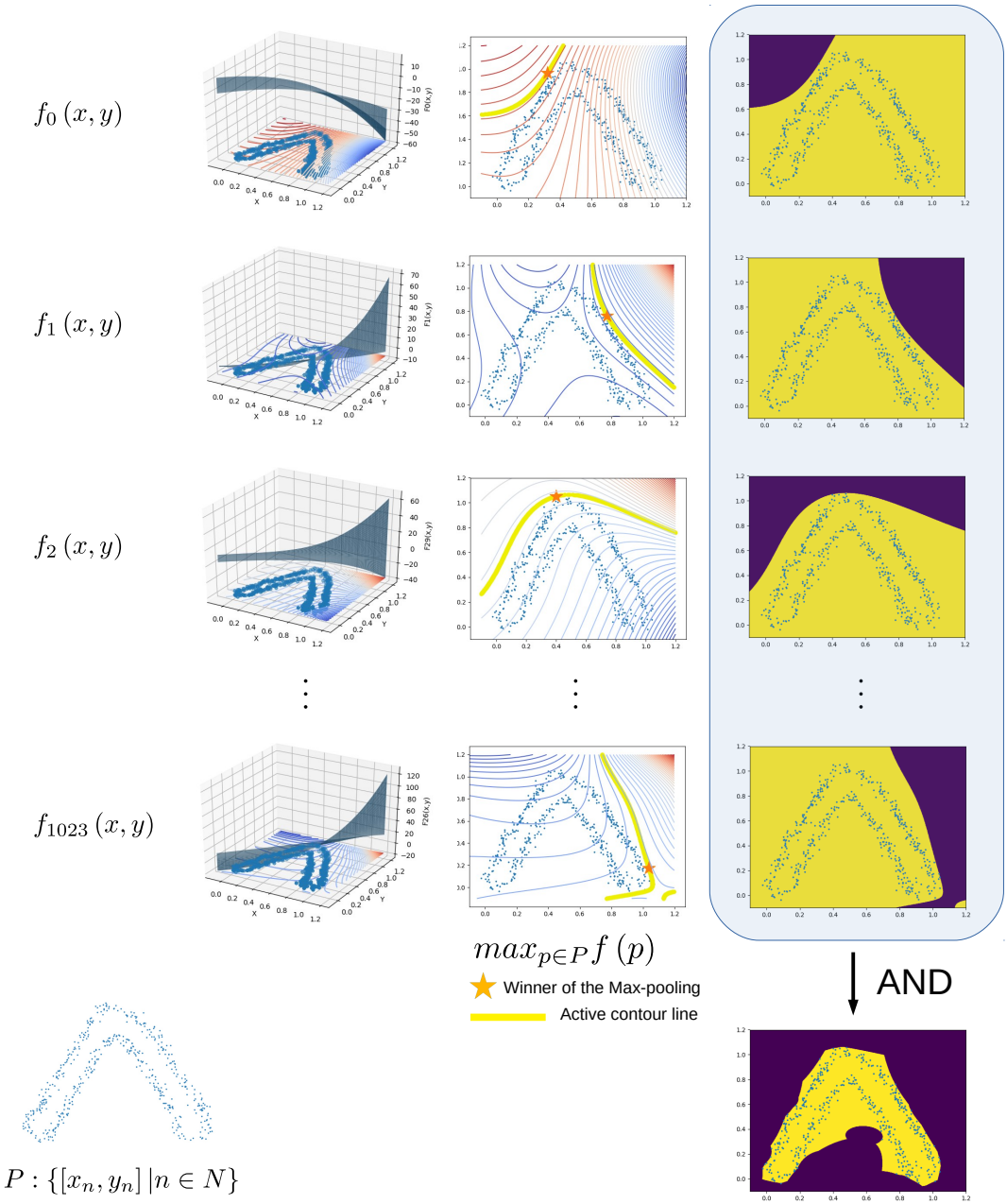


Figure 2.2: An illustration of the way shape is encoded with MLPs and max-pooling.

proximity). PointNet++ [101] enhanced the representation ability by proposing a hierarchical architecture for PointNet. Other variations like Frustum PointNets [6] and Engelmann et al. [47] extend the PointNet architecture to incorporate larger-scale spatial context. PointNet and its variants are extensively used in point cloud generative methods as the shape encoder, i.e. the output of the PointNet-like model represents the learned shape embedding space.

Variants and Applications of PointNet-Like Networks Ravanbakhsh et al. [102] formally introduce the properties of permutation invariant and equivariant operations of sets, and they propose a Deep Set architecture. This is an approach similar to PointNet (developed and published concurrently) but proposes a layer-wise symmetric layer to form a network. They apply this method to predict the sum of digits, point cloud classification, and set anomaly detection. Deep-RBFNet [103] is inspired by PointNet [4], RBF-based surface representation [84] and Radial basis function networks [104]. They propose to, within a PointNet-like architecture, implement the feature mapping of each input 3D point with RBF kernels scattered in the space, where an RBF kernel's centre and width are trainable parameters of the network. This method has number of parameters and increased computational complexity over PointNet but achieves *better performance* than PointNet++ [101], implying that RBFs are good at capturing and representing 3D geometric properties. Like RBFs, PointSIFT [48] is another local feature (inspired by SIFT [105]) that can be integrated in PointNet-based architecture. A network built with PointSIFT has recently achieved state-of-the-art in tasks such as point cloud segmentation.

While these methods have promising performance on point cloud classification tasks, there is a noticeable disadvantage: these methods fail to form a data structure to describe the unordered point sets efficiently - the max-pooling aggregates everything into an unified descriptor. Not only do they give up the benefit of leveraging the underlying structural information, but the non-structured architecture of these methods makes them less useful for

generalizing to other point cloud problems like surface reconstruction.

2.3.2 Point Graph and Tree Based Learning Frameworks

Another class of methods tries to endow graph or tree structure to unstructured points to construct efficient learning architectures. These methods leverage the point-wise local or global structure, extracting nearest neighbours and building (hierarchical) local connectives. To be specific, they treat each point in the set as a vertex (graph approach) or leaf-node (tree approach).

The main problem is looking for optimal structures of points for general or particular tasks. For example, density of connections and distance metrics play an important role in both spatial and spectral domain graph features of shapes. When modeling point clouds with hierarchies like Kd-tree [106] or Self-Organizing Map (SOM) [32], the structures (winner node and root node) is not stable with tiny changes of sampled points. To achieve robust learning with unstable representation is a challenging problem. A lot of different structures are explored: the point relationships could be extracted locally from a bottom-up direction [20, 35], globally from a top-down direction [36], or even as an optimized structure with SOM. In a graph-based method, the feature learning can be performed in spatial or spectral domains. Wang et al. [20] construct k-NN graph that is built in the point-wise feature space and makes local feature learning with EdgeConv, an operation defined with point-wise connectivity. RGCNN [33] is a spectral method that maps the k-NN point graph to a per-point feature space with spectral filtering layers. Klokov and Lempitsky [36] propose a Kd-Net that builds a kd-tree for the point cloud and equips hierarchical feature extractions from the leaves to root at each branch. Li et al. [32] use SOM to represent point cloud geometric structure. They first construct a SOM with $m \times m$ nodes (having a 2D grid structure) randomly located in the space around the point cloud, and the SOM is updated with an unsupervised competitive

learning rule to represent the points.

These structure based approaches have attracted increasingly more attention in the last few years, because graph and tree representations are inherently effective to handle irregular data.

Some other methods [107, 108, 109] directly impose regular and compact structures onto the shape with a network. They use meshes composed from finite discrete elements and deform the mesh to fit target shapes. 2D regular grids, spherical triangle meshes, and volumetric tetrahedral meshes are used in FoldingNet [107], Pixel2Mesh [108], and DefTet [109], respectively. Advantages of this approach, including topology-preservation and easy-manipulation of shapes, are desired properties for generative applications.

2.3.3 Generalized Convolution Operation for Point Sets

Generalizing convolution to points is arguably the most popular approach of point cloud deep learning. Because Convolutional Neural Networks (CNNs) are so successful for challenging image-understanding tasks, it is appealing to apply CNN-like operation to point cloud learning problems. However, regular CNN kernels are defined with discrete pixel grids that can be written as 2D arrays, while point clouds are sets with order-less and structure-less points, preventing necessary convolution operations for extracting spatial context. Many solutions [29, 30, 3, 31, 32, 33] have been explored for this problem.

Li et al. [30] propose to use a so-called χ-transformation to map the features of each point into a canonical order before making regular convolution operations. Hua et al. [110] implement convolution operations on point clouds by considering the clustered points as a sampled density signal, and accumulating them with $3 \times 3 \times 3$ grid of bins (as a kernel operation). Wu et al. [34] propose PointConv using a different way to achieve a similar density signal and define 3D continuous convolution with Monte Carlo approximation. Shen

et al. [31] propose to use small point sets as convolution kernels. Image convolution could be interpreted as computing the score to measure the similarity between image patches and the convolution kernel (the kernel could be viewed as another patch of image). So, they believe generalizing a convolution to points should be done simply as a measure of affinity between points' neighbouring point sets (patches) and a point set kernel. The affinity is measured based on pair-wise distance between the points of the patch and points of the kernel. Some other methods [3] use tricks to project discrete points or kernels to continuous function spaces, and compute continuous convolution using the projected signal and kernel.

From the methods above, we can see there are several different assumptions made (or implied) for generalize regular discrete convolution to the point cloud domain:

- Canonical order could be endowed onto order-less point clouds (at least for small clusters of points) [30];
- Point clouds are considered as a density sampling of a hidden signal. So we can access the hidden shape by accumulating the points in space [110, 34];
- Continuous convolution could be conducted if discrete points are projected back to a signal (3D Gaussian) in *continuous* function space [3];
- Convolution is nothing but measuring the similarity between a small patch of data (local point clusters) and a kernel (also a cluster of points) [31].

2.3.4 Limitations and Issues with Point Cloud Learning

Unlike in the image domain where similar CNN architectures are ubiquitously used, there is no single architecture that dominates the point cloud domain. Various methods are proposed for different point cloud tasks. We group the recent point cloud deep learning method into three categories: point-wise MLP, point graph/tree based, and point convolution. However,

they are not strictly exclusive to one another. For example, point-wise MLP method's characteristic global-pooling is used in PointGCN [111], a Gaussian k-NN graph method, to capture global and local features of the point cloud. MLP is used in convolution-based methods PointConv [34] and MCCNN [112] for estimation of the point densities. So how to efficiently learn 3D shapes with deep networks, or equivalently, what is the best neural operator for capturing shape features remains an open problem.

The second unsolved problem is how to achieve effective hierarchy in learning models. Most convolution-based and point-wise MLP methods have to carefully form a hierarchical structure to aggregate multi-scale shape features of the point cloud. Many adaptive sampling schemes (Gumbel subset sampling [113], Furthest point sampling (FPS) [114], and Poisson disc sampling [112]) are developed for more efficient representation or challenging situations like non-uniform point clouds. Meanwhile, there is not enough work for exploring learning methods to handle flexible, or large numbers of points.

Another noticeable problem with state-of-the-art of point cloud learning is limited evaluation. There are already a large amount of point cloud learning methods, however, their properties, advantages, and draw-backs are still not clear. There is still a lack of good metrics and benchmarks for evaluation. Most noticeably, their robustness when processing imperfect input point clouds with a pretrained network have not been well-studied and compared. Because of this, the use of point cloud deep learning in autonomous driving, robotics, remote sensing, and medical treatment, has had neither a handy and versatile method (like CNN for visual understanding) nor a clear guide for which to pick.

Lastly, the research in point cloud deep learning has unbalanced interests toward 1) supervised over unsupervised learning and 2) discriminative over generative models, leaving some important problems not as well-studied. The topics reviewed in section 2.4 and 2.5 can be viewed as efforts to mitigate this problem.

2.4 Learning Methods for Surface Reconstruction and Generative Models of 3D Shapes

2.4.1 Challenging Surface Reconstruction Tasks and Structural Priors

As reviewed in Section 2.2.2, approximation based surface reconstruction aims to find a curved plane or a scalar function whose zero set best approximates a surface, i.e. as close to the observed on-surface points as possible. In principle, this is an ill-posed problem because an infinite number of functions can make its zero set pass through a given set of points. Assumptions and priors have to be imposed to make such a task practical.

The design of traditional non-learning methods typically implies a surface smoothness prior. Significant progress has been made recently, on reconstruction through more advanced versions of these methods. Various kinds of information and priors have been typically employed to achieve better performance and robustness:

- **Surface visibility during acquisition** - Cone Carving [115], Multi-Scale Scan Merge [116]
- **Volume smoothness prior** - Volume-Aware Surface Evolution (VASE) [117]
- **Geometric primitives** - Reconstructing Indoor Scene with Volume Primitives [118]
- **Global regularity** - Regular Arrangements of Planes (RAPter) [5]

In this list, the first two are usually harnessed to overcome missing data defects, and the last two are mainly used to handle the task of reconstructing large-scale cluttered indoor scans. In addition, for some specific tasks and scenarios, interactive methods have been developed, and these methods can achieve reconstruction for more challenging tasks that

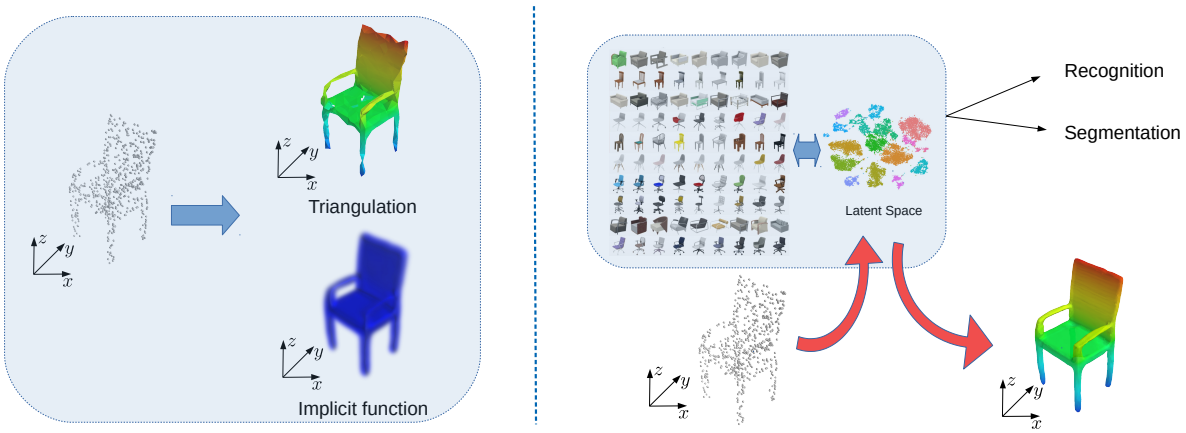


Figure 2.3: Diagrammatic illustration of the distinction between traditional and learning-based reconstruction methods. The left subfigure represents non-learning, while right one represents learning-based methods.

neither human nor machine can comfortably handle independently. Recently, an emerging topic, solving structure reconstruction with data-driven and machine learning techniques, have begun to receive attention.

The main distinction between a traditional non-learning and a learning-based reconstruction is illustrated in Figure 2.3. Non-learning methods recover surfaces by applying knowledge and assumptions directly to formulate either a structure of points (e.g. triangulation) or optimization of an objective function (e.g. implicit function). Learning-based methods, by contrast, involve a learned mapping from data space to some kind of latent space, which encode class/semantic and geometrical/topological properties of shapes (the box in Figure 2.3 right). This latent space mapping is crucial for discriminative or semantic tasks like shape recognition and segmentation. Such models have the potential to capture shape priors, semantic and categorization knowledge learned from data. This approach becomes helpful for reconstruction in scenarios involving extreme noise/sparsity/non-uniform and even missing data, because the model may be better able to "hallucinate" shape with its direct knowledge.

2.4.2 Early Learning Approaches and Low-Level Shape Learning

Before the age of deep learning, neural network structures had been applied to shape/surface reconstruction. As recovering shape from point clouds has primarily been formulated as a optimization problem, we can imagine that competitive learning principles, leveraged by: genetic algorithm, simulated annealing, and particle swarm optimization, could be useful tools to solve this problem. For example, Self Organizing Map (SOM) is a non-linear vector quantization with topological preservation. It's used in Yu [119] to build a multiple resolution mesh from an input point cloud. The fixed structure of node in the original SOM-based methods sometimes does not allow it to fit multi-resolution and different scale of details. And the global update scheme is not efficient if the network is large. To overcome these problems, Ivrissimtzis et al. [120] proposed a dynamic version of SOM approaches, by growing cells. Simulated annealing is proposed as a probabilistic method of finding global minimum, hence it is suitable to guide a network to approximate point clouds. Wu et al. [121] use simulated annealing to adjust the parameters of an RBF reconstruction method. As population-based stochastic optimization methods, particle swarm and differential evolution have also been implemented as the optimizer of spline fitting [122, 123].

Based on the distinction specified in section 2.4.1, the above methods are not learning-based methods that are at the same level as other later approaches. They use machine learning techniques as an optimization and regulation term, mapping points from 3D spaces to discrete surface space (instead of shape instance space to feature space). As they do not deliberately learn shape instance level knowledge or prior from data, it's hard to generalize the learned parameters to other instances of shapes.

Point Cloud Consolidation and Upsampling Point cloud consolidation or up-sampling used to be an intermediate step toward shape reconstruction. The terminology *consolidation* places emphasis on producing a set of denoised, outlier-free, and evenly distributing points

over the surface [124], whereas *upsampling* refers more to increasing density of the point clouds. This problem sometimes is considered as equivalent to projective parametric surface reconstruction like projection MLS [62, 81].

Although a well-studied topic [124, 125, 126], it has been recently explored from a machine learning perspective [93, 92, 127, 128]. Roveri et al. [128] proposed to transform a patch of point cloud into a height-maps structure and 2D grids to accommodate regular a 2D CNN, which is trained to "consolidate" 2D patches. Other methods like Yu et al. [92], EC-Net [127], and Yifan et al. rely on a point cloud network (like PointNet++ [101]) to extract multi-scale per-point features and upsample the feature space to solve the upsampling problem. He et al. [129] upsample the surface with a point cloud auto-encoder (as a feature extractor) with a geodesic matching component to learn geodesics from sampled surface points.

Point cloud consolidation/upsampling, by it's definition, is a local operation. So these methods [93, 92, 127, 128] only consider local shape structures and work with patches. This is the main difference between [93, 92, 127, 128] and those methods [11, 16, 15, 10, 130, 131] that are more related to our work.

Learning to Recover Surface Normals Boulch and Marlet [132] in 2016 proposed a CNN method to estimate robust surface normal. This method first implements a Hough transform component to make the points vote for planes in spherical 2D accumulator grids. The CNN acts as a filter to extract the *real* peak in the Hough space as the tangent plane at a sample point. In addition, many point cloud understanding works reviewed in Section 2.3 also benchmark the proposed network by predicting a point-wise surface normal. However, like the common problem of deep learning models, the end-to-end learning models' properties are not clear. The lack of comprehensive benchmarking and evaluation (current metrics only consider the point-wise surface normal individually and calculate MSE) results in a lack of clarity around the generalization and robustness of these methods.

2.4.3 Triangulation and Interpolation Methods

Simply speaking, triangulation approaches mentioned in Section 2.2.1 aim to build mesh across a subset of input points to approximate the underlying surface. Learning-based versions of these methods can potentially help this procedure by imposing loss functions corresponding to the shape priors learned from training data. We find some works [133, 134] that use a candidate triangle proposal & filtering framework. Sharp and Ovsjanikov [133] train MLPs to estimate the probability of a candidate triangle as belonging to part of the surface. Liu et al. [134] propose to learn geodesic distance between points and define the so-called intrinsic-extrinsic ratio (IER) as the quotient of geodesic and Euclidean distance between points of a triangle. This ratio is used as a criterion for selecting on-surface triangles. The main drawback with this framework is that the reconstructed triangle surface mesh may not be water-tight or even non-manifold. Filtering the triangles can also be done from a volume perspective, i.e. labeling tetrahedrons as inside or outside of the 3D body and selecting the interface triangles. Traditional methods typically define a inside-outside potential for each volume element and segment the space (graph-cut) to achieve a global energy minimization. Sulzer et al. [135] suggest to use learning-based potentials to replace handcrafted ones, and label inside-outside with a graph neural network (GNN). While those triangulation and interpolation methods can directly produce a triangular mesh, they suffer from the same problem as the traditional triangulation based methods mentioned in Section 2.2.1.

2.4.4 Learning to Recover Parametric Surface

Machine learning methods can also be used to reconstruct parametric surfaces. For example, DeepSpline [60] uses a hierarchical Recurrent Neural Network (RNN) to solve the optimization problem that minimizes the output-target mismatch. This method is especially robust at initialization for approximating complex curves. Laube et al. [59] proposed to include

B-spline curve parameters directly into a neural network architecture to solve the curve fitting problem. AtlasNet [136] proposes to represent 3D shapes with a collection of parametric surface elements (a set of 2D squares fitting locally to the surface), which are used to train a shape auto-encoder to reconstruct models of an atlas. Deep Geometric Prior [97] is another parametric surface reconstruction method. This method is inspired by deep image priors [137], and is able to do image denoising, inpainting, and super-resolution *without* training with any data. They use a similar deep point cloud prior by overfitting a neural network representation to a local point cloud. Then a manifold atlas is computed from many such networks on overlapping point cloud patches.

2.5 Implicit Neural Representation for 3D Generation and Surface Reconstruction

Recently, the implicit surface representation has been adopted in learning-based shape reconstruction methods [16, 11, 15, 10]. These methods aim to learn a network that acts as implicit representation, mostly signed distance function or occupancy function, of the underlying 3D shape. The network is a scalar function whose domain is the continuous 3D space around the input point cloud: $\Omega : \mathbb{R}^3$, and range:

$$\forall p \in \Omega, f\left(p\right) \in \left(-\infty, +\infty\right),$$

for learning SDFs [11, 10],

$$\forall p \in \Omega, f\left(p\right) \in \left\{0, 1\right\},$$

for learning indicator functions [16, 15]. In other words, given any 3D point, the network is trained to predict how far away the point is from the surface (SDF), or whether this point is

located inside or outside of the shape (occupancy).

2.5.1 Early Methods

Park et al. [11] propose a framework to generate 3D shape through learning SDFs. The SDF of a query point is produced by a fully-connected neural network. Instead of SDFs, IM-NET [15] uses a fully connected network to predict the implicit function. The shape embedding (latent code) in IM-NET is modeled with an autoencoder. Mescheder et al. [16] introduce another indicator function learning approach, Occupancy Network or ONet. This differs from IM-NET [15] and DeepSDF [11] who use fully connected networks to approximate the target implicit function, ONet is based on Conditional Batch-Normalization (CBN) [138], where a latent code is mapped via means and variances of batch normalization to learn an implicit function network. Atzmon et al. [10] propose a different framework that directly controls the decision boundary of a network (so called *neural level sets*). This is achieved by locally parameterizeing the level set around subsets of points and defining a geometric SVM loss function to encourage large margins between query points and the decision boundary.

2.5.2 Varied Applications and Interests of INR

The impressive results and interesting properties shown in those pioneering works has triggered novel applications and studies that focus on specific aspects of INRs.

1. Using INR for human poses [9] and animatable avatars [139, 140]
2. Using INR for shape completion and shape mending [141, 7, 8]
3. Learning INR for complex structure and clustered scenes [17, 142]
4. Unsupervised or self-supervised learning of INR [42, 12]
5. Leaning INR from 2D supervision only [38, 143]
6. Rotation equivariant INR [144]

7. Formulating INR as Meta-learning [14, 145]
8. Formulating implicit representations other than occupancy functions or SDFs [44, 43, 46, 8, 42, 18]

2.5.3 Problem and Limitations of Point Cloud INR and Proposed Solution

Architectures of Point Cloud INR Early INR methods mostly predict the surface by first building a shape embedding (the encoder) at the whole object level and using a second conditional network (the decoder) to produce the implicit function $\phi : \mathbb{R}^3 \rightarrow \mathbb{R}$. The decoder is used to parameterize the target implicit function. When implementing INR for shape generalization and reconstruction tasks, feedforward neural networks, mostly Multi-Layer Perceptrons (MLPs), directly map a 3D coordinate (the "query") onto the functional value at this point, i.e., $MLP\left(\mathbf{Z}_i \cdot\right) : \mathbb{R}^3 \rightarrow \mathbb{R}$. The MLP is conditioned with a low-dimensional latent code $\mathbf{Z}_i$. This low-dimension embedding of the global shape could be either regressed by an encoder, jointly optimized in the auto-decoder framework, or produced by generative models like Variational AutoEncoders (VAEs) and Generative Adversarial Networks (GANs). These models learn data distribution and form an embedding for the respective high-dimensional implicit function domain. This part of the framework can be viewed as the encoder as shown in Figure 2.4 top. For decoding, there exist two ways for conditioning an MLP: 1) conditioning via concatenation [10, 11, 12] and 2) conditioning via hypernetwork [146, 16, 26, 38].

These two types of decoding are illustrated in Figure 2.4 separately. In conditioning via concatenation methods, the coordinates of query points q are simply concatenated with a latent embedding $\mathbf{Z}_i$ and are fed as an input vector to an MLP whose parameters φ are fixed after training and shared across the entire learned shape domain. Because of the latent code conditioning, this shared network $\phi_\varphi\left(Z_i, \cdot\right)$ may represent and decode multiple classes and

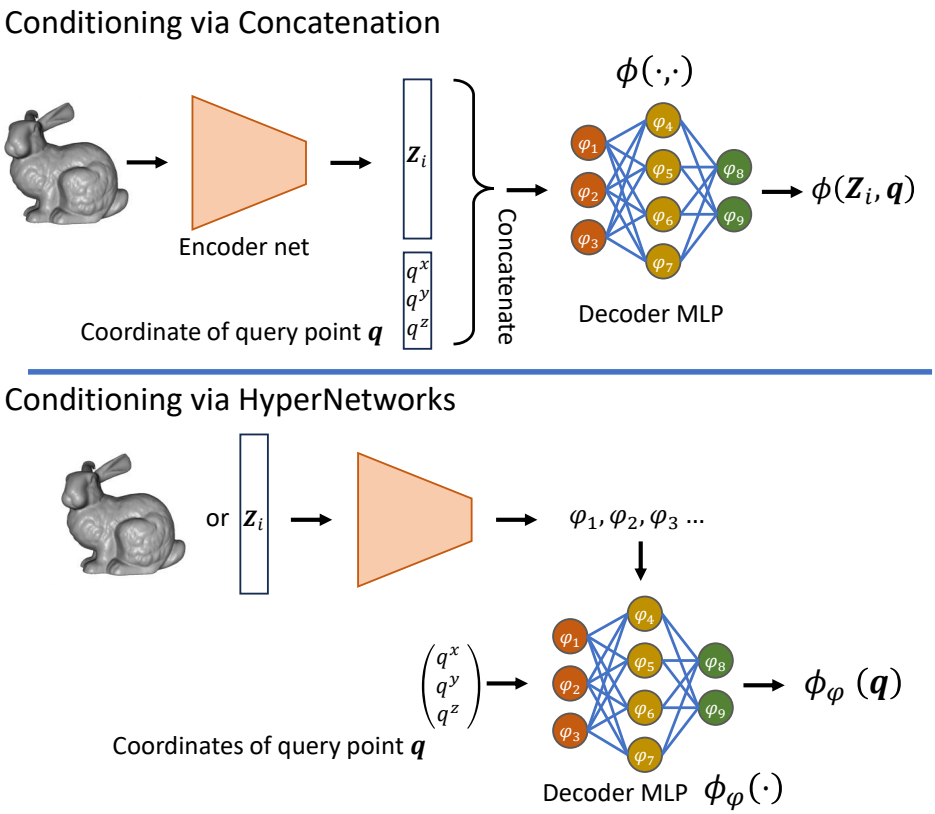


Figure 2.4: Diagrams illustrating the network architectures of existing INR works. They parameterize continuous functions in two slightly different ways: conditioning-via-concatenation (top) and conditioning-via-hypernetwork (bottom).

innumerable shape instances. In hypernetwork methods, a (hyper) neural network $\mathcal{H}$ is used to regress the parameters of another neural network. The input to $\mathcal{H}$ can be instances of either raw shape domain (reconstruction and generalization tasks, $\mathcal{H} : X \mapsto \varphi$) or the embedding domain (generative learning, $\mathcal{H} : Z \mapsto \varphi$). The decoder network then parameterizes the target implicit function as $\phi_{\varphi(X \text{or} Z)}(\cdot)$.

Limitations of INR This design is limited by its inability to efficiently incorporate local point cloud information, thus may cause issues like not being able to generate fine-grained details (tendency to over-smooth the surface) and generalize to unseen object categories (tendency to hallucinate *eigen-shapes*). One approach to solve this problem [17, 25] is to

extract shape features from a local neighborhood and then aggregate the feature in 2D or 3D grids like in image CNNs. Some methods [147, 142] also incorporate local shape-part embedding code to the grid cells to enhance representation ability. However, as Boulch and Marlet state, this makes the inferences "lose the direct connection with the input points sampled on the surface of objects". Moreover, those discretization methods with a fixed resolution may lose the continuous and irregular properties of INR learning, thus limit their representation ability and applications.

Some other studies suggest incorporating high-frequency functions into the network to fit the data containing high-frequency variations. For example, Mildenhall et al. [39] propose positional encoding that maps the input coordinates with sine/cosine functions in a range of frequency bands. Sitzmann et al. [26] build implicit neural representation systems with periodic activation functions (instead of ReLU activation) that efficiently parameterize images, videos, audio, and 3D shapes.

Lastly, there have been attempts to improve the learned local shape prior and learning efficiency of INR. These efforts may also mitigate the aforementioned global embedding problem. They craft mathematical/geometrical mechanisms and use some sophisticated network structures or loss functions. For example, Peng et al. [148] propose Differentiable Poisson Surface Reconstruction (DPSR) that uses a neural network to estimate offset and surface normals for input points and reconstructs volumetric indicator functions with a linear rasterization Poisson solver. This framework is feasible because the entire DPSR pipeline is differentiable. In POCO [149], the shape embedding code is learned at each input point, and the occupancy for a query point is predicted based on a composition of its neighboring latent codes. Atzmon and Lipman [12] introduce a sign agnostic loss to learn SDFs directly from raw, unsigned point clouds. Neural-Pull [43] is a method that trains a network to pull query points to their closest points on the zero-set (target surface) of the underlying signed distance field. This pulling is conducted according to the gradient (projecting direction) and

the values (projecting distance) of SDF at query points. Based on this query projection idea, Ma et al. [44] later propose another implicit neural representation (on-surface prior), that can be learned without using binary occupancy labels or ground truth signed distances. This method shows impressive reconstruction accuracy and great robustness for sparse and noisy input.

In this dissertation, we propose an alternative solution that samples point-wise feature space at the query location to learn a more localized implicit function, so that we can preserve the benefit of local-observations (more faithful representation of observed details) together with the general advantage of INR (the ability to resolve/hallucinate from memory). For such a purpose, we develop a point convolution based local operator, Dual Feature Sampling (DFS), and achieve a more efficient and robust INR learning for 3D object shapes. This novel approach is outlined in the next chapter.

Chapter 3

Foundations: Dual Feature Sampling (DFS) and Stochastic Continuous Function Learning (SCFL)

The core methodology of this dissertation is a Dual Feature Sampling (DFS) based Stochastic Continuous Function Learning (SCFL) framework. In this chapter, we first introduce the necessary concepts and notions of SCFL in Section 3.1. The motivation of our "conditioning-via-sampling" method and a system-level overview of the DFS operation based network are introduced in Section 3.2. In Section 3.3, we show how to adapt PCNN [3] to be our building-block and build PCNN-based DFS. In Section 3.4, we use a FlexConvolution [19] operator to take PCNN's place and make FlexConvolution-based DFS. These two versions of networks using different computational elements demonstrate the flexibility of the proposed SCFL framework.

3.1 Notions and Formulations

The Stochastic Continuous Function Learning (SCFL) paradigm plays a foundational role in our study. Roughly speaking, by *randomly* and *sparsely* sampling the target output space, pairs of random query points and target values are produced as training data. With such forms of data, we can train useful models by stochastically supervising the network to approach a target function value at these random query points. Before diving into the details of our work, we have to review some general machine learning concepts and notions to see the characteristics of point cloud deep learning and our SCFL method.

In supervised learning, training data is a set of observations $\mathcal{D} = \{(X_i, Y_i) \text{ for } i = 1, ..., N_D\}$ drawn from the product space of data and label domains $\mathcal{X} \times \mathcal{Y}$. The task of machine learning is to find a function $F : \mathcal{X} \to \mathcal{Y}$ from a parameterized hypothesis set $\mathcal{F} = \{F_\theta \mid \theta \in \Theta\}$ to *best explain* (according to some specific loss $L(\cdot, \cdot) : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$) the training data as $Y_i = F_\theta(X_i)$. Typically, the data domain is assumed as a Euclidean space $\mathcal{X} = \mathbb{R}^d$. For example, feature extraction techniques produce a feature vector intended to be informative and non-redundant for a specific data type, and classifiers or regressors take the feature space as their input domain. In contrast, deep learning methods map the original data directly to the output space with a network. To overcome the-curse-of-dimensionality [150], additional structures within the input data have to be leveraged. For example, an RGB image is a signal $\Omega \to \mathcal{C}$, where $\Omega = \mathbb{Z}_n \times \mathbb{Z}_m$ is a two-dimensional $n \times m$ grid, and $\mathcal{C}$ is the RGB color space, i.e. being $\mathbb{Z}_n \times \mathbb{Z}_m \to \mathbb{R}^3$ other than $\mathbb{R}^{n \times m \times 3}$). Based on the neighborhood defined in $\mathbb{Z}_n \times \mathbb{Z}_m$, convolutional operations in CNNs are shift-equivariant, and pooling makes the feature maps scale- and deformation- invariant. Point clouds, on the other hand, contain even less structure than $\mathbb{R}^{n \times d}$. The order of these n points in a point cloud is irrelevant to the shape of P, which implies that point clouds are just *sets* with no explicit structures.

Within such a context, implicit function based shape learning of point clouds is formulated

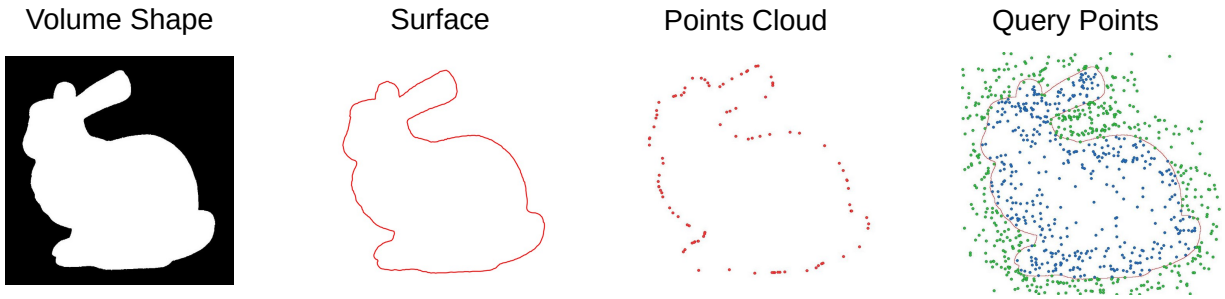


Figure 3.1: The structure of the training data used in our method illustrated in 2D.

as obtaining a mapping:

$$X \mapsto F[X], F[X](\cdot) : \mathbb{R}^3 \to \mathbb{R}, \tag{3.1}$$

in which the point cloud $X = \{x_i\}_{i=1}^{N_X}$ sampled from the surface of an object is mapped to a 3D continuous scalar function in $F[X] \in C(\mathbb{R}^3, \mathbb{R})$. If $q \in \mathbb{R}^3$ is a 3D coordinate, $F[X](q) \in (-\infty, +\infty)$ for regression models of SDFs and $F[X](q) \in [0, 1]$ for classifying occupancies[2]. If we follow the common supervised learning scheme of feed-forward models, the loss can be immediately constructed as an integral over the 3D physical space:

$$\mathcal{L}(f[X], Y) := \int_R L(F_\theta[X](q), Y(q)) \, \mathrm{d}q,$$

where $L(\cdot, \cdot) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a measurement of differences and $R \subset \mathbb{R}^3$ indicates the region-of-interest, e.g. bounding-box of X. Generally, this loss is intractable, except for discretizing it with voxelization or finite element methods. Instead, we proposed to spawn random query

[2]Typically, the occupancy networks' last component is a soft-max classifier, its output is interpreted as the probability of q being internal/external of the surface. In a network containing C^0 non-linear activation functions like Rectified Linear Unit (ReLU), the output function is also C^0 continuous

points $q_1, q_2, \ldots$ in R, and supervise the network as:

$$F_\theta = \arg\min_{\theta \in \Theta} \sum_{(P,Y) \in D} \tilde{L}\left(F_\theta[X], Y\right), \tag{3.2}$$

$$\text{where } \tilde{L}\left(F_\theta[X], Y\right) := \sum_{i=1}^{N_Q} L(F_\theta[X](q_i), Y(q_i)).$$

This could be viewed as a Monte-Carlo (MC) estimation of the loss. The target Y is sparsely endowed to train the network so that $F_\theta[X]$ stochastically approaches the real SDFs or occupancy functions. In this dissertation, we are particularly interested in training a feed-forward network mapping point clouds to the objects' occupancy functions. For each query point q_i, we associate it with a target $Y(q_i) \in \{0, 1\}$ indicating whether it's inside or outside the surface. In Figure 3.1, the target function, object surface, point cloud and query points are illustrated in $\mathbb{R}^2$. Like other works in this field [16, 10], our method relies on properly generated training data from a collection of explicit shapes, e.g. water-tight triangle meshes. It could be viewed as a stochastic approximation of the target continuous function. So we call this method Stochastic Continuous Function Learning (SCFL).

3.2 Motivation and Overview

In light of the limitations and solutions discussed in Section 2.5.3, we argue that answering a query for reconstruction doesn't have to go that deep and complex, up to a global shape embedding (encoding) then down to 3D space for occupancy prediction (decoding). Intuitively, finding occupancy state of a query point is equivalent to answering the question: is this query location enveloped by the input point cloud. The input 3D points can be roughly assumed to be distributed in a *closed* hidden 2D manifold which envelopes a piece of 3D space. To answer such a query, the model should look at its peripheral space and reason its relative

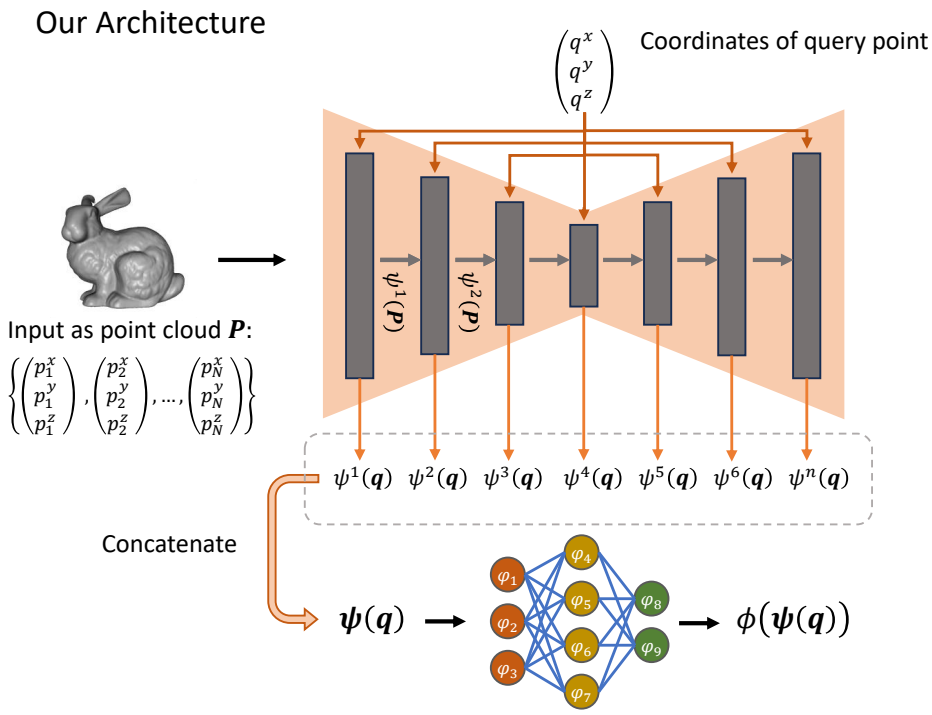


Figure 3.2: Illustration of our conditioning-via-sampling architecture. The stacked DFS layers in this network are high-lighted to emphasize the difference between our method and other architectures.

position with respect to the surrounding input points. In other words, we need the network to learn a local association and expect an effective and faithful representation of fine details.

From this observation, we propose DFS operation that samples point-wise feature space at the query location. DFS is a point convolution based computation component that unifies the input points feature space and query points feature space. We then stack multiple DFS layers to form a deep network for INR learning. It relies on point-wise pooling and the U-Net style hierarchy to achieve feature aggregation and capture global shape representation. We name this paradigm "Conditioning via Sampling" to emphasize the main idea of sampling query point features for learning localized occupancy function.

This network architecture is illustrated in Figure 3.2. Our DFS operation makes use of recent works that generalize convolution operations to the point cloud domain (related works

are reviewed in Section 2.3). In these methods, feature maps are formulated as functions on discrete points. In 2D image CNNs, each pixel (with pooling and multi-resolution) is associated with a feature vector. Similarly, each input point p is associated with a feature vector. So, the output of the n-th point convolutional component is a set of mappings:

$$\psi^n(\mathbf{P}) : \{p_1 \mapsto \psi^n(p_1), p_2 \mapsto \psi^n(p_2), \ldots, p_N \mapsto \psi^n(p_N)\},$$

where $\psi^n(p) \in \mathbb{R}^J$ can be viewed as a J-dimensional feature function *evaluated* at the spatial point p. While there are many different ways of formulating the operation ψ across existing methods, we notice one can easily be evaluated at *any* spatial point q without introducing too much complexity or losing feature representation. This is a favorable characteristic of this formulation, particularly for our intention which is to unify the feature space of input points and off-surface query points. It is made possible because ψ is spatially well-defined (i.e. no singularity, efficient computation everywhere) and well-behaved (i.e. continuous, smooth derivative).

One single query point is distributed into and flows through every layer of the network to compute a query point feature vector (indicated with the vertical arrows in Figure 3.2). In our DFS operator, there is a hidden feature space parameterized implicitly with a continuous function $\psi^n(\cdot) : \mathbb{R}^3 \mapsto \mathbb{R}^J$. As a matter of fact, the point-wise feature vector $\psi^n(p) \in \mathbb{R}^J$ is just the value of this function evaluated at this particular point. With $\psi^n(\cdot)$ well-defined, the query point feature is simply defined as the value of the same function $\psi^n(q)$ evaluated at the query [3].

The shape features in this hierarchy represent different level abstractions of the shape. So the feature vectors from each layer are concatenated to aggregate information about this query location. The concatenated vector is then fed to an MLP to regress the final implicit function

[3]While they come from the same hidden feature function, their computations are different. The computation of $\psi^n(q)$ is subject to how $\psi^n(P)$ is defined

of the shape. Notice, the MLP in our architecture is simply a mapping from the query point feature vector to the target function value. Unlike MLPs in conditioning-via-concatenation and conditioning-via-hypernetwork approaches, it has no conditional input.

Given the formulation above, it's not hard to see that each DFS layer can have dual purposes. On one hand, the feed-forward connections aim to extract shape characteristics from the input point cloud P. The learned features should encode point-based geometry of the underlying shapes, as well as shape priors and semantic knowledge about objects in each category. The union of all hidden feature vectors $\left\{\psi^n\left(p\right)|p \in \mathbf{P}, n = 0, 1, \ldots N^{depth}\right\}$ forms a comprehensive representation of the input shape. On the other hand, features are sampled via random off-surface points from the same hidden feature space $\psi^n\left(p\right)$ to infer implicit function values of such a query. It means the feature is trained to provide information that's most relevant to one point's occupancy state precisely. Because of this, we call this computation component the Dual-Feature-Sampling (DFS) operator.

In this dissertation, we adapt recent point convolutional methods, namely PCNN [3] and Flex-Convolution [19], to propose two instances of DFS operators and build two versions of occupancy function networks. The technical details of our approach are explained in the following sections.

3.3 PCNN-based continuous DFS layer

At the core of our method is the modified Point Convolution Neural Network (PCNN) operator that samples the continuous feature space with random point convolutions. PCNN is proposed by Atzmon et al. [3] as a general point cloud deep learning framework that is utilized in 3D shape classification, part segmentation and surface normal prediction. We will briefly review PCNN and derive how to make it the building-block of learning continuous functions.

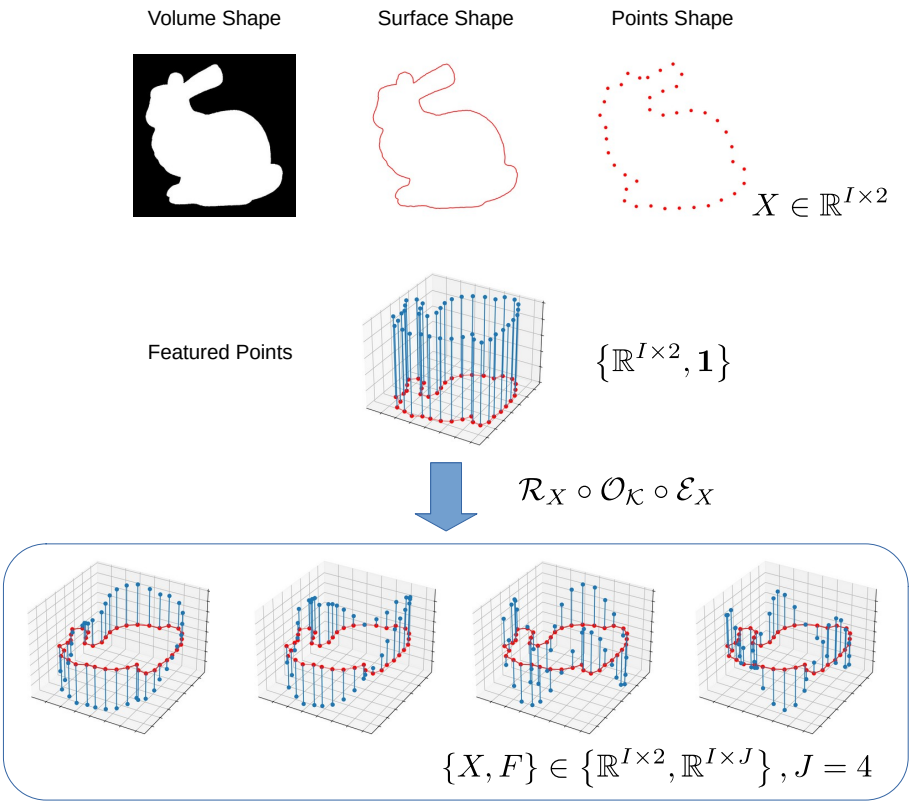


Figure 3.3: An illustration of *points with per-point features*.

3.3.1 PCNN Operators

Notations Let's denote the PCNN operator as $\mathcal{R}_X \circ \mathcal{O}_\mathcal{K} \circ \mathcal{E}_X$, a cascade of *extension* operator $\mathcal{E}_X$, continuous convolution $\mathcal{O}_\mathcal{K}$, and *restriction* operator $\mathcal{R}_X$. Note both its input and output are *points with per-point features*. Because of that, multiple PCNN operators (layers) can be stacked to form a deep hierarchical architecture like CNNs.

Given a point cloud, i.e. a set of 3D points $X : \{x_i\}_{i=1}^{I}, x \in \mathbb{R}^3$, each point could be associated with a feature vector $f_i \in \mathbb{R}^{\mathrm{J}}$. For the input points, f could be color, surface normal or other properties acquired together with the 3D points. For "hidden points", i.e. outputs of the middle layers, f represents shape features learned by PCNN. So *points with*

per-point features are denoted by tuples $\{X, F\} \in \left\{\mathbb{R}^{I\times 3}, \mathbb{R}^{I\times J}\right\}$ where I indicates the number of points and J indicates the length of the feature vector. Figure 3.3 shows a 2D example where the PCNN operator extracts 4-channel features from the input point cloud. In this case, the input points are unoriented and have no color information. So there is a constant **1** feature that indicates every point has a same weight. From a signal processing point-of-view, the point cloud is a digital sample of the (continuous) volumetric or surface shape, i.e. a sequence of unit impulses. Note the order of I impulses has no effect on the shape represented. So PCNN operator $\mathcal{R}_X \circ \mathcal{O}_\mathcal{K} \circ \mathcal{E}_X$ is designed to be invariant to permutation.

PCNN - Extension Operator In [3], the extension operator is defined as

$$\mathcal{E}_X^j[f](x) = \sum_i f_{ij} l_i(x) \tag{3.3}$$

i.e. a sum of continuous basis functions $l \in C\left(\mathbb{R}^3, \mathbb{R}\right)$ that is shifted by the points and scaled by the features. In the original PCNN, the basis functions are weighted Gaussians,

$$l_i = \omega_i \Phi\left(|x - x_i|\right),$$

where the weight factors $\omega_i = 1/\sum_{i'} \Phi\left(|x_{i'} - x_i|\right), i' = 1, 2, \ldots, I$ normalize the global magnitude. It is helpful to view this as the impulses $f_{ij}\delta(-x_i)$ convolved with RBF basis functions. See the top row of Figure 3.4 for a 2D illustration of extension operator using Gaussian as the basis. We may have J channels of features in the input, thus the output of the extension operator is a J-dimensional vector field on $\mathbb{R}^3$, i.e. $\mathcal{E}_X\left[f\right](\cdot) \in C\left(\mathbb{R}^3, \mathbb{R}^J\right)$.

The output field is a continuous and flattened signal approximating the scattered (at the sampled on-surface points $\{X\}$) feature vectors $\{F\}$. This signal tends to have extrema when crossing the surface and vanishes when moving off-surface (Gaussian basis $max\Phi\left(x\right) = \Phi\left(0\right)$ and $\Phi\left(x\right) \to 0$ when $x \to \pm\infty$). The width of basis function is a hyper-parameter that plays

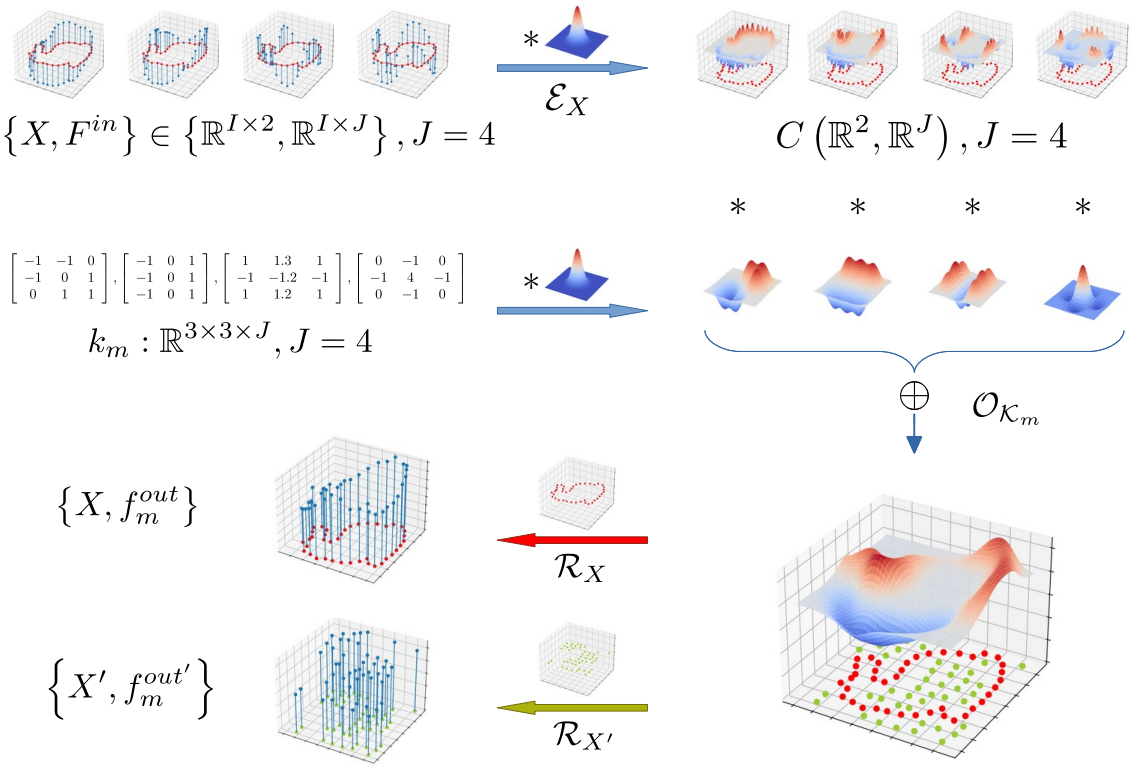


Figure 3.4: An illustration of PCNN operator.

a significant role for the approximation. A basis that's too narrow fails to approximate over the surface, while one too wide tends to over-smooth the feature field. So it's better to be determined according to the density of the sampled on-surface points. In the original PCNN [3] and our work [151], variance of the Gaussian basis is inversely propositional to the square root of the number of points.

From the definitions above, we can see that the extension operator is invariant to the order of input points. Moreover, the extension operator is able to approximate continuous feature functions on the manifold where the points are sampled from.

PCNN - Kernels and Convolution Continuous convolution could be applied to the result of extension $\psi = \mathcal{E}_X[f](x)$ to extract more complex point features. The convolution kernel is based on a $3 \times 3 \times 3 \times J \times M$ lookup table the elements of which, are learnable parameters. Taking out one slice $k_m \in \mathbb{R}^{3\times3\times3\times J}$ from the last dimension, it could be mapped to a continuous function (denoted as $\mathcal{K}_m \in C\left(\mathbb{R}^3, \mathbb{R}^J\right)$) as a grid of overlapped Gaussian bases. In Figure 3.4, the second row visualizes such mapping in a 2D situation. The convolution operation $\mathcal{O}_\mathcal{K}$ is then defined as

$$\mathcal{O}_\mathcal{K}^m[\psi](x) = \psi * \mathcal{K}_m = \int_{R^3} \sum_j \psi_j(y) * \mathcal{K}_{mj}(x-y)\, dy \tag{3.4}$$

where $m = 1, 2, \ldots, M$,and M represents the number of output channels of the convolution operator. At the right-hand-side of Figure 3.4, we plot the process of Equation 3.4. The input features are correlated with a number of learned kernels (which themselves may act as edge and ridge detectors or Laplacian operators) in the feature space and the output is a combination of those correlations. Note this is just one output channel of $\mathcal{O}_\mathcal{K}$, and the same process replicates for M times with different kernels. Eventually, the operation maps continuous functions $C\left(\mathbb{R}^3, \mathbb{R}^J\right) \to C\left(\mathbb{R}^3, \mathbb{R}^M\right)$.

We can see this convolution operation is similar to 2D image convolution in many respects. First, they are all translation invariant. Second, J and M are correlated to the number of channels of the image CNN's input/output feature maps. Third, the kernel size is also configurable ($3 \times 3 \times 3 \to 5 \times 5 \times 5$ or adjusting the Gaussian grids' scale) just like image CNN.

PCNN - Restriction Operator Restriction is simply sampling a continuous function at some given points,

$$\mathcal{R}_X^m[\psi] = \psi_m(X), \tag{3.5}$$

where ψ represents the result of previous convolution, and X is the sample points. Figure 3.4 (at the bottom) shows two examples that restriction operation with the original point cloud (X, red) and random off-surface points (X', green). Because ψ is continuous, this operation can be performed with ANY sample points. This makes it straight forward to define pooling and up-sampling: the sampling points being either a subset or a superset of the input point cloud.

Computation of PCNN The operations defined above have an important property that they can be computed efficiently with a closed-form solution. As we know, the convolution of two Gaussian functions is still Gaussian [152]:

$$\Phi_\alpha\left(|x-a|\right) * \Phi_\beta\left(|x-b|\right) = \Phi_{\sqrt{\alpha^2+\beta^2}}\left(|x-a-b|\right).$$

Beacuse of that, there is no need to compute anything numerically in the continuous domain ψ (the right-hand-side of Figure 3.4). The operators in Eqn. (3.3), (3.4) and (3.5) together could be equivalently computed with discrete points and features as:

$$O_{X,X'}\left[F\right] = \sum_{ijl} f_{ij} k_{ljm} w_i q_{ii'l}, \tag{3.6}$$

where q is a tensor in which each element corresponds to the convolution of Gaussians:

$$q_{i'il} = \Phi_\gamma\left(|x_{i'} - x_i - y_l|\right),$$

where $y_l \in \mathbb{R}^3$ indicates the spatial offsets of the Gaussian bases in the convolution kernels, and $l = 1, 2, \ldots, 27$ to be the offsets of the kernel centres in the $3 \times 3 \times 3$ grid. The combined deviation γ is one of the most important hyperparameters in the network. In Eqn. (3.6) the indices i, j, l are summed-up and $O_{X,X'}\left[F\right] \in \mathbb{R}^{I' \times M}$ forms the output features vectors of the

restriction points $X' \in \mathbb{R}^{I' \times 3}$.

3.3.2 Generalizing PCNN for DFS Operation

From the definition above, we can see the output of PCNN or Flex-Convolution operator has the same form as its input, i.e. points and the per-point features associated with each point. So the PCNN layers could be stacked to form a deep architecture. Atzmon et al. [3] develop their method as a generalization of more traditional image CNNs. The point up-sampling, pooling and deconvolution layers are defined to form a hierarchical point network. Moreover, image deep learning techniques such as Relu Non-linear Activation, Dropout, and Batch Normalization, can be seamlessly integrated into PCNNs. With those methods, a standard CNN architecture can be built for point cloud shape classification, and a U-net architecture can be built for part segmentation.

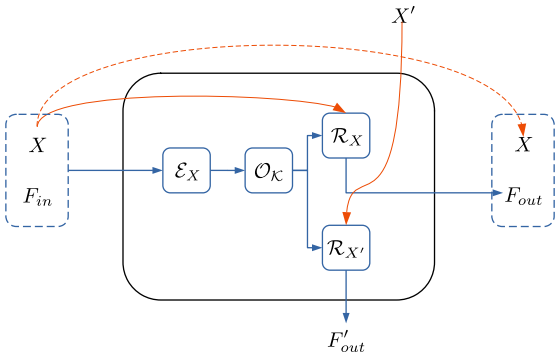


Figure 3.5: The computational chart of the dual-restriction convolution units.

After evaluating this PCNN method, we realize its potential of learning continuous spatial functions. In the original scenarios in [3], the restriction operation (3.5) is only applied with on-surface points, either the input point cloud X or its subset/superset. To achieve dual-feature sampling, we proposed to use random off-surface points in a second restriction operator to form a side stream, or simply speaking, dual-sampling via dual-restriction. The

dual-restriction convolution units are defined as the operations:

$$D_{X,Q}\left[F_{in}\right] = \begin{bmatrix} O_{X,X}\left[F_{in}\right] \\ O_{X,Q}\left[F_{in}\right] \end{bmatrix} = \begin{bmatrix} \mathcal{R}_X \circ \mathcal{O}_\mathcal{K} \circ \mathcal{E}_X\left[F_{in}\right] \\ \mathcal{R}_Q \circ \mathcal{O}_\mathcal{K} \circ \mathcal{E}_X\left[F_{in}\right] \end{bmatrix} \tag{3.7}$$

or, equivalently shown as the diagram in Figure 3.5. Q is a query point set distributed *randomly* in the region-of-interest (usually the space around X). A second restriction operator conducted after $\mathcal{O}_\mathcal{K} \circ \mathcal{E}_X$ samples the continuous feature space produced by the kernel operation $\mathcal{O}_\mathcal{K}$ and produces F'_{out}. Figure 3.4 is a graphical illustration where the green arrow is a restriction operation with random query points. Sampling the feature space at random points enables us to continuously supervise the target function ψ anywhere in the 3D space.

3.4 FlexConvolution-based continuous DFS layer

FlexConvolution is an efficient convolutional point-cloud learning method. Groh et al. [19] proposed this method to process a million-point scale real-world dataset and demonstrate competitive performance with lower memory consumption. This method is utilized for object part segmentation (ShapeNet) and real-world indoor semantic point cloud segmentation. We will briefly review Flex-Convolution and derive how to utilize it as a building-block for learning continuous functions.

3.4.1 FlexConvolution

The idea of FlexConvolution comes from the observation that a discrete 2D convolution applied to images is a linear operation that has a kernel that can be represented as a $3 \times 3 \times J^{in} \times J^{out}$ weight matrix. Watching only on the j'-th output channel, the operation is performed as

$$conv\left(K, F\right)\left[x\right]_{j'} = \sum_{j \in [1, J^{in}]} \sum_{\tau \in \{-1,0,1\}^2} k_{j'}\left(\tau, j\right) f\left(x - \tau, j\right), \tag{3.8}$$

where $j' \in [1, J^{out}]$ indicates one dimension of the output channel, and $f(x - \tau, j)$ indicates input color images or feature maps with J^{in} channels. In most deep networks, the kernel has spatial size of 3 (i.e. 3×3 pixels). It could be viewed as a look-up table or a function in a discrete domain:

$$K_{j'} : J^{in} \times \{-1, 0, 1\}^2 \to \mathbb{R}.$$

If we want to extend a convolution to points, a kernel function's domain has to be continuous and unbounded:

$$\tilde{K}_{j'} : J^{in} \times \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R},$$

where $d = 3$ for points in 3D space. Groh et al. [19] simply implement this function as:

$$\tilde{k}(c, x, x' | \theta_c, \theta_b) = \langle \theta_c, x - x' \rangle + \theta_b, \tag{3.9}$$

i.e. an inner product followed by an offset, in which $\theta_c \in \mathbb{R}^d$ and $\theta_b \in \mathbb{R}$ are learnable parameters. This kernel is computed w.r.t. neighboring points (x and x') that are spatially close to each other, just like 2D image convolution kernel does on a 3×3 receptive field. With the kernel defined above, the Flex-Convolution shall be:

$$conv\left(\tilde{K}, F\right)[x]_{c'} = \sum_{c \in C} \sum_{p' \in \mathcal{N}_k(p)} \tilde{k}(c, p, p') \cdot f(c, p'), \tag{3.10}$$

where $\mathcal{N}_k(p)$ is the k-nearest neighbour points of p and $f(c, p')$ represents the feature associated with point p'. Comparing Flex-Convolution with PCNN in Equation 3.6, we can see that Flex-Convolution is less computational expensive, which makes it possible to stack many layers to form a *deep* architecture to process large-scale point clouds containing millions of points.

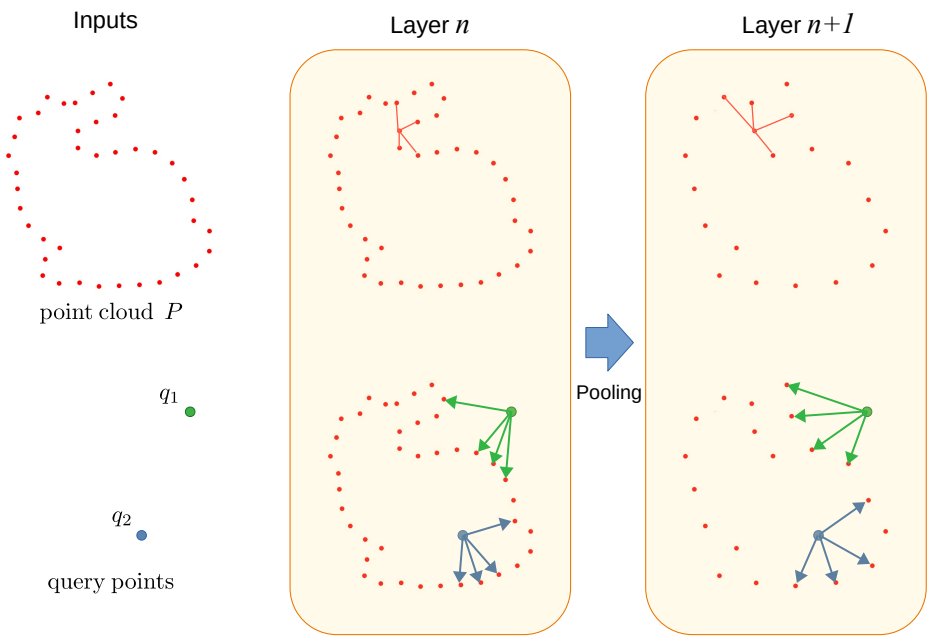


Figure 3.6: The k-NN graphs are built respectively for the on-surface points and query points.

3.4.2 Generalizing Flex-Convolution for DFS Operation

To make use of Flex-Convolution inour SCFL method, we also formulate a DFS operator with it. Because the original Flex-Convolution operation is defined based on a (k-Nearest Neighbor) k-NN graph of the individual points, it is natural to consider using nearest point based neighborhood for the query points. For any query point, we can sample the feature space at this location in a manner similar to that of Flex-Convolution with on-surface points (Equation 3.10). The only difference would be the neighborhood set $\mathcal{N}_k(p)$ would be replaced by the k closest points of the query point q, denoted as $\mathcal{N}_k|_P(q)$:

$$\begin{aligned} &\mathcal{N}_k|_P(q) \in P, \\ &s.t.\ \forall p \in P \ominus \mathcal{N}_k|_P(q), \operatorname{dist}(q,p) \geq \max_{p' \in \mathcal{N}_k|_P(q)} \operatorname{dist}(q,p') \\ &\text{and } |\mathcal{N}_k|_P(q)| = k \end{aligned} \tag{3.11}$$

This neighborhood relation is determined only with the point cloud P and each individual query point, independent from the structure or size of the query point set. Also note the neighborhood of on-surface points is a subset of P that contains the centre point p itself, which is slightly different from the neighborhood of query points. In our networks, the input point cloud is progressively down-scaled through multiple pooling operations. So all of the neighbourhood relations are computed with respect to the point cloud structure of each layer. This nearest point method is illustrated in Figure 3.6.

Overall, the random point sampling operation is

$$conv\left(\tilde{K}, F\right)[q]_{c'} = \sum_{c \in C} \sum_{p' \in \mathcal{N}_k|_P} \tilde{k}\left(c, q, p'\right) \cdot f\left(c, p'\right). \tag{3.12}$$

The feature map sampled at point q is a spatial restricted (by $\mathcal{N}_k\ |_P\ (q)$) convolution of kernel $\tilde{k}\left(c, q, p'\right)$ (Equation 3.9) and the point-wise feature vectors $f\left(c, p'\right)$.

3.5 SCFL using Continuous DFS operations

3.5.1 Architecture of SCFL Networks

Our occupancy prediction network is inspired by U-Net [153] and adapted from the segmentation network in [3]. The architecture is illustrated in Figure 3.2. The network is constructed as a cascade of DFS units (3.7). The first input to this network is a point cloud P representing a geometrical object, and the second is a set of query points randomly scattered in the ROI (region of interest).

For each query point q, the network produces a concatenated feature vector $\psi\left(q|P\right)$ and a predicted local function value $\phi\left(q_1|P\right)$. In Figure 3.2, the horizontal arrows represent the main flow that processes and passes the implicit shape information of the input point cloud. Referring to the chart of dual-restriction in Figure 3.5, we can see that the main

stream is only related to the input geometry and feature of P. While the vertical arrows represent the side stream that the second restriction operator uses to sample the main streams signal ψ at the query point q, i.e. $\mathcal{R}_q[\psi]$. The feature vectors $\psi(q|P)$ are a concatenation of $\psi^1(q|P), \psi^2 i(q|P), \ldots$, i.e. sampling (at q) the main stream signal of the 1st, 2nd, ... dual-restriction blocks. Note these feature vectors are independent from one query point q_1 to another q_2, but they depend globally on points in P.

Our network architecture is constructed following U-Net principles: 1) The down-sampling stage has several blocks that increase the feature depth and using pooling to aggregate features into a set of sparser points that each represent an enlarged receptive field. 2) The up-sampling stage works in opposite way, by instead inserting points previously pooled in the down-sampling stage. 3) The mirrored layers in down- and up-sampling part of network have the same resolution (number of points). They are connected with lateral connections. The skip feed-forward connections feed the local fine features from the down-sampling stage forward to the up-sampling stage, maximizing the interaction between the information in multi-scale. Then these multi-scale global-local features are bundled as a query point feature. At the last stage of the side stream, a Multi-Layer Perception (MLP) with optionally non-linear activations or Softmax maps the feature vector to per-point output:

$$\phi(q|P; \{K_f, K_g\}) = h(\mathrm{MLP}(F(q|P))), \tag{3.13}$$

where K_f and K_g represent the learnable parameters of the dual-restriction U-net and the MLP respectively. The output g could be either a scalar or a vector.

Because the query points could potentially be any point in the 3D space, this network $\phi(\cdot|P; \{K_f, K_g\})$ actually maps an unstructured point cloud to a continuous function $P \rightarrow G, G \in C\left(\mathbb{R}^3, \mathbb{R}^M\right)$.

3.5.2 Training Algorithm of SCFL

SCFL is trained in a supervised manner, just like any binary classification networks are trained with a dataset of instances of input examples and their corresponding labels. This requires a dataset where each instance is a pair of a full point cloud and a target continuous function $\{P^{(i)}, G^{(i)}\}$. For each instance, we sample G_i with random query point set $q_j^{(i)} \in Q^{(i)}$ and record the sparse function values $o_j^{(i)}$. The SCFL network is trained by minimizing the batch loss function below:

$$\mathcal{L}_B(K) = \frac{1}{B}\sum_{i=1}^{B}\sum_{j=1}^{J}\mathcal{L}\left(\phi\left(q_j^{(i)} \mid P^{(i)}; K\right), o_j^{(i)}\right), \tag{3.14}$$

note the point-level and instance-level summation are up to J, the number of points in $Q^{(i)}$), and B, the batch size, respectively. The loss function $\mathcal{L}(\cdot)$ is task-related. For example, we use cross entropy loss in our occupancy learning methods (shape reconstruction task outlined in Chapter 4 and implicit liver registration in Chapter 5), and mean absolute difference loss in other applications such as learning of 3D deformation fields (chapter 6)

3D shape datasets typically store them as triangle surface meshes. To achieve training data in form (3.14), synthesis methods are crafted to sample the original into point clouds and capture target functions with query-value pairs. Domain knowledge (like shape priors and noise distribution) is thus implicitly encoded in the generated training data.

Learning an occupancy function or SDF is the simplest to work with. We generate training data in a similar way as existing research [16, 17, 25] in that sample points and the target function value are extracted from an objects' triangle mesh models (where source meshes are made water-tight first). In the liver registration task (chapter 5), we synthesize a spectrum of deformed 3D liver models following a linear elastic model (details in chapter 5), so that the sampled occupancy function encodes physical laws relating to elastic deformation. In the MRI brain segmentation task (chapter 6), we train the network to predict tumor induced

volume deformation from the tumor's shape and position, in which a 3D deformation field used for training is generated with a brain tumor simulation software.

3.5.3 Network Usage and Evaluation

Network prediction and usage After training, the network can be store with its trained weights for use in evaluation or real applications.

For our surface reconstruction task in Chapter 4, the goal is to build a mesh from the input point cloud. Unlike in training, the query points in reconstruction / evaluation are not random. Given an input point cloud, the query points should be a uniform dense sample of the space of interest (see the illustration shown in Figure 1.2). In principle, they form a regular 3D grid over the bounding box around the input shape. In reality, there are varied ways to reduce the number of query points to make mesh reconstruction faster, like a hierarchical or progressive coarse-to-fine split of voxels, or by interactively querying the network to insert new query points into the inferred neighborhood of the target surface. The network cannot take in all query points at once, so the query points are parsed to fixed-size batches, and fed into the network batch-by-batch. Similarly, the network's outputs have to be collected batch-by-batch and reorganized to form a volumetric sampling of the space or an explicit triangle mesh.

In the liver registration task in Chapter 5, there is no need to build the liver mesh from the reconstructed INR liver. As the goal is fitting a pre-surgery liver scan onto the sampled physical liver, we should move the surface vertices of the pre-surgery liver mesh to the (implicit) reconstruction surface of the INR liver shape. So these surface vertices are fed into the network as query points, and we optimize / register the translation, rotation, and deformation to make the output of the network approach 0.5 (the iso-surface / level-set of the sampled physical liver).

In the MRI brain segmentation task (Chapter 5), the INR network learns to produce a deformation field from the input tumor point cloud. The deformation is used to modify the geometry of the atlas (normal brain) to involve the change cause by the tumor. We simply sample a dense volume ($256 \times 256 \times 256$) in the skull space and interpolate the displacement or SVF of any point we need. That's because 1) we don't want to compromise any accuracy in this task, and 2) this INR prediction only runs once for each tumorous image segmentation and takes only a small fraction of time over the whole pipeline.

Evaluation Metrics Because the networks are trained for different purposes in our applications, we have to use different methods and metrics to make evaluations.

The surface reconstruction task is relatively well-studied and easy to benchmark. So we use widely adopted mesh based metrics to compare the reconstructed shape with ground truth, based on the testing split approach used on the ShapeNet dataset. Metrics used in the experiment are Intersection over Union (IoU), L1 Chamfer distance, and surface normal consistency score. Their definition and meaning are detailed in Section 4.3.1. Meanwhile, we also conduct experiments for different number of input points, domain generalization (unseen object shape categories), and the affect of input rotation for our network. Such evaluations provide some insight about the network's generalization and robustness.

In the liver registration task, we have access to data that contains the ground-truth displacement of discrete metal target beads embedded in a liver phantom. So we can use the simplest metric, the mean target error, to study the registration performance. We mainly compare our method with baseline ICP to see the boost in performance resulting from incorporating our INR liver reconstruction network.

In the MRI brain segmentation task, the evaluation is relatively difficult because of the lack of ground-truth segmentation of tumor-bearing images. To overcome this problem, we use three compromised ways: synthesizing tumorous brain images with ground truth, tracking

the change of brain inter-hemisphere symmetric score, and subjective visual comparison with base-line methods. For the former two ways, we use a segment over-lapping based metric, Dice score, to compare the results quantitatively.

A more comprehensive evaluation of the proposed SCFL network may involve ablations studies with respect to these tasks. However, this dissertation doesn't include this aspect of experiments because of workload issues in our research. It is our hope that further evaluation can be conducted in the future or by other researchers.

3.5.4 Computational Performance and Complexity Analysis

Mesh reconstruction speed problem The intensive computation is one of the most noticeable limitations of the proposed DFS based INR learning framework (as currently implemented).

Our method is considerably slower than INR reconstruction methods implemented in encoder-decoder architecture. Our PCNN-based network usually takes about 10-20 seconds to build the output mesh depending on the input shape and number of input points. (we implement it with Multiresolution IsoSurface Extraction (MISE) method proposed by [16] on a system of a Xeon CPU working at 4Ghz and a GTX1080ti GPU deploying the network). As a reference, Conv-ONet [17] takes less than a second for each instance of reconstruction. Notice, this speed is still relatively faster than most of auto-decoder based INR methods (taking between 30-40 seconds to several minutes, according to the running time comparison reported in [149]), which requires an optimization of the latent code during the reference time.

This relative slower speed can be easily explained by looking at the encoding of an input point cloud in the encoder-decoder methods. As shown in Figure 2.4 top, the encoder transform the point cloud into a low-dimensional code $\mathbf{Z}_i$ as an abstraction of the input

shape. This encoder only needs to run once for each new input shape to be reconstructed. This dim$E$ (dimension of the embedding space) vector is concatenated with a 3-dimensional query point coordinate (so becomes batches of dim$E$+3 vectors) to form the input of the MLP decoder, in which the hidden layers simply conduct matrix multiplications that can be hugely accelerated with GPU parallelization. In contrast, our DFS operation is point convolution based (referring to Figure 3.2), and query point feature $\psi_P(q)$ is sampled locally from the continuous feature $\psi(P)$. While some of the intermediate results (like $\psi(P)$) can be saved for inference of new query points, sampling the feature space requires a computation of the distance from query to all the input point (DFS-PCNN) and building a k-NN structure (DFS-FlexConvolution). There are some independent computations that need to be done in each DFS layer $\psi^1, \psi^2, ...$, and their time complexity is linear w.r.t. the number of input points (DFS-PCNN) or the size of the neighborhood (DFS-FlexConvolution). It's also linear w.r.t. the feature map depth (the dimension of $\psi^n(\cdot)$) of each DFS layer.

In Section 4.5, we alleviate this computational expensive problem by proposing a fast iso-surface reconstructing algorithm that reduces the number of query points needed in mesh reconstruction. Further ablation studies of the network may bring a lot of insight about the computational performance, left for future work.

Computational complexity Because our conditioning-via-sampling method is different from the MLP used in encoder-decoder or auto-decoder methods, it is hard to theoretically compare their computational complexity. But we attempt to make a preliminary comparison between DFS-PCNN and DFS-FlexConvolution proposed in this chapter. And we argue that the fundamental bottle-neck for improving computational performance lies mainly the trade-off between time (for both training and inference) and space (mainly GPU memory consumption) complexity.

Comparing Flex-Convolution in Equation 3.10 with PCNN in Equation 3.6, we can see

that Flex-Convolution is less computational expensive. First, in terms of model memory space consumption, each Flex-Convolution layer requires a set of learnable parameters with cardinality equal to $C \times C' \times (d+1)$, where C and C' indicate the input and output feature dimension and $d = 3$, while PCNN requires $C \times C' \times (d^3+1)$. Second, in terms of computation, the feed-forward layer for each output point (either on-surface points or query points) generating an output feature is theoretically equivalent to conducting a $|\mathcal{N}_k| \times (d+1)$-dimensional element-wise summation ($|\mathcal{N}_k|$ indicates the size of the neighborhood) and a $C \times |\mathcal{N}_k| \times d$-dimensional inner product. While each PCNN layer requires a $|\mathcal{N}_k| \times d^3$-dimensional summation followed by an element-wise exponential and equivalently a $C \times |\mathcal{N}_k| \times d^3$-dimensional inner product. The difference between DFS-PCNN and DFS-FlexConvolution seems not very significant, but the neighborhood $\mathcal{N}_k$ in PCNN is the entail point cloud (150, 300, 600, 1000 points in our experiments) and Flex-Convolution usually has $|\mathcal{N}_k|$ being 20-40. So DFS-FlexConvolution is significantly light-weight w.r.t DFS-PCNN. The network built with DFS-FlexConvolution may have a much deeper structure (number of DFS layers) and deeper feature maps (C and C' in the network).

Because of the properties stated above, the strategy of implementing PCNN based network is using large neighborhood ($\mathcal{N}_k$) with relative shallow network structure (C and C'). There is a straight-forward way to reduce computation: reduced the size of $\mathcal{N}_k$ from hundreds to tens of points. In theory, this should not sacrifice much its performance, because the PCNN operation is spatial restricted (Gaussian curve approaching 0 when moving away from the center) and there is no need to incorporate the points far away from the query point. However, the DFS lay will have to maintain as many inner continuous features as the number of query points in the batch (each generated form its own neighborhood points, $\psi_{\mathcal{N}_k^1}(\cdot), \psi_{\mathcal{N}_k^2}(\cdot), etc.$), instead of a single $\psi_P(\cdot)$ for all query points. In our test, this change can result in the need for significantly higher GPU memory usage.

The strategy in our FlexConvolution based network is opposite: using small neighborhood

($\mathcal{N}_k$) with deep network structure (C and C'). This is also our only choice, because 1) The FlexConvolution kernel is unbounded (see Equation 3.9, without restricting the size of receptive field with k-NN, the points far from the centre region will dominate the convolutional output); and 2) FlexConvolution is not as representative as PCNN, which makes it necessary to have larger and deeper structures (so that there are more complex combinations and non-linearity).

Chapter 4

Learning Implicit Function for Surface Reconstruction

4.1 Overview

As reviewed in Chapter 2, implicit representation of surface or 3D shape is relatively advantageous and widely used for reconstructing underlying surfaces from sampled points. We pay attention to this problem at the beginning of our study and propose a novel method for learning occupancy functions from sparse point clouds. Our method achieves vastly improved performance on challenging surface reconstruction tasks.

Unlike the previous methods which predict point occupancy with fully-connected multi-layer networks, we adapt the point cloud deep learning architecture, Point Convolution Neural Network (PCNN), to build our learning model, natively reflecting the geometric nature of point cloud data. Our occupancy function learning can be easily fit into procedures of point cloud up-sampling and surface reconstruction. Furthermore, we propose a Delaunay triangulation based fast algorithm to achieve efficient surface reconstruction, without sacrificing much accuracy. Our experiments show state-of-the-art performance for reconstructing with ShapeNet dataset,

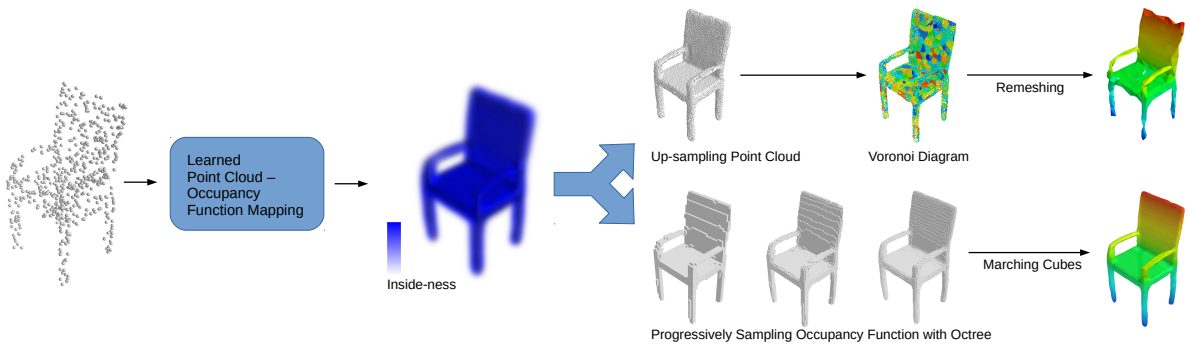


Figure 4.1: The reconstruction pipeline of the proposed method. The pretrained SCFL network directly maps the input sparse point cloud to the space occupancy functions of the underlying shapes. The continuous occupancy function can then be discretely sampled or used in other ways to build output meshes.

and we demonstrate the versatility of this method by implementing three simple applications: point cloud spectral segmentation, boolean and morphological operations on 3D shapes.

Our method is sketched in Figure 4.1. The proposed PCNN-based occupancy network is treated as a front-end module, and we use the Marching Cubes algorithm [154] to extract an iso-surface as the reconstructed structure. The network takes a point cloud as input and maps the shape represented by this point cloud to its indicator function. The predicted indicator function is then voxelized into three grids (sampled coarse-to-fine hierarchically). Lastly, we run Marching Cubes to produce the final triangle mesh.

Experiments conducted on the ShapeNet dataset [1] indicate that our method outperforms traditional deterministic methods [79, 86] and recent learning-based methods [130, 131, 16]. Primarily, traditional methods can not work well with very sparse or noisy point cloud. In applications where we don't have access to a dense scan, our study suggests there are obvious advantages to learning-based over non-learning methods. Meanwhile, previous learning methods implement an encoder-decoder framework and thus can only generate stereotypical shapes of each category of objects. In contrast, by integrating shape features in the geometry domain and relying more on local shape information carried by point clouds, our method

may capture and recover unique shape details. That also gives it better generalization ability and achieves a certain degree of rotation invariance like traditional surface reconstruction methods.

4.2 Occupancy Function Learning and Data Preparation

Modeling occupancy function with a binary classifier An occupancy function is a two-valued function in 3D space, $o : \mathbb{R}^3 \to \{0, 1\}$, indicating whether this point is an occupant of a solid M (interior of the object) or not (exterior), i.e., $\mathbf{1}(q \in M)$, and is therefore called an indicator function in a computer graphics context. ∂M indicates the surface of solid M, and it is what we want to recover from an observed point cloud $P : \{p_1, p_2, \ldots p_n\}$, where $p_n \in \mathbb{R}^3$. Following [16], we formulate occupancy learning as a 3D point classification problem, learning a network $\phi(q \mid P; \theta)$ to approximate $o(q; P)$, i.e., for a query point $q \in Q$, its likelihood of interior or exterior is predicted as the network's output.

Since this is a binary classification problem, we could easily form it as the model shown in Figure 3.2. Those query points $q \in \mathbb{R}^3$ are first fed to a conditional feature function $\psi : \mathbb{R}^3 \to \mathbb{R}^N$, and then classified. We use a fully connected Relu network with a softmax layer as the classifier:

$$\phi(q|P; \{\theta_\psi, \theta_\phi\}) = \text{softmax}\left[\text{CF}_{\theta_\phi}\left(\psi_{\theta_\psi}(q|P;)\right)\right]. \tag{4.1}$$

Note that outputs of those individual query points are conditioned only on the input point cloud P which represents the shape we reconstruct.

Training of this entire model is done by minimizing the loss function:

$$\mathcal{L}_{\mathcal{B}}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \sum_{j=1}^{\mathcal{K}} \mathcal{L}\left(\phi\left(q_j^{(i)} \mid P_i; \theta\right), o\left(q_j^{(i)}; P_i\right)\right), \tag{4.2}$$

where $q_j^{(i)}$ denotes the jth random query point corresponding to the ith point cloud in a batch $\mathcal{B}$. We use cross-entropy as the loss function $\mathcal{L}(\cdot)$.

Training data Query points q could potentially be any point in the 3D space. Training data of this network is a number of point clouds representing various shape corpora. Each point cloud (on-surface point set) is associated with a set of arbitrary points (queries during training) sampled and labeled inside and outside of the surface defined by the point cloud. This form of data is not like any sensor's raw data and thus does not naturally exist. However, there are many 3D shape datasets that contain thousands of high-quality 3D mesh models. Given a water-tight 3D mesh (one in which there is no ambiguity of interior and exterior), we can randomly take an arbitrary amount of points (queries) around it and make Ray-tests for each query point to determine whether a point is inside or outside (ground-truth label) the mesh. The input point clouds could be sampled from the mesh randomly, much like drawing from a uniform distribution, or deliberately, like simulating partially viewed point clouds. In our study and experiments, we take a subset from ShapeNet [1], a multi-category 3D mesh dataset, and use the method proposed in [155] to make the meshes water-tight. Point clouds P and query points Q are sampled from the mesh surface and space (in/out) around it.

Theoretically, an infinite number of random query points could be sampled around a water-tight mesh. However, the way of sampling those points matters for learning unbiased occupancy functions. The classification model (3.13) itself, is agnostic to geometry/topology or any point ordering, so query points fed to it are treated as individual entries. Suppose the sample points distribute different density inside and outside of M. In that case, we could not expect the network to learn an unbiased surface around ∂M. An illustration of this issue is shown in Figure 4.2. A possible solution to this problem is to sample the bounding box of

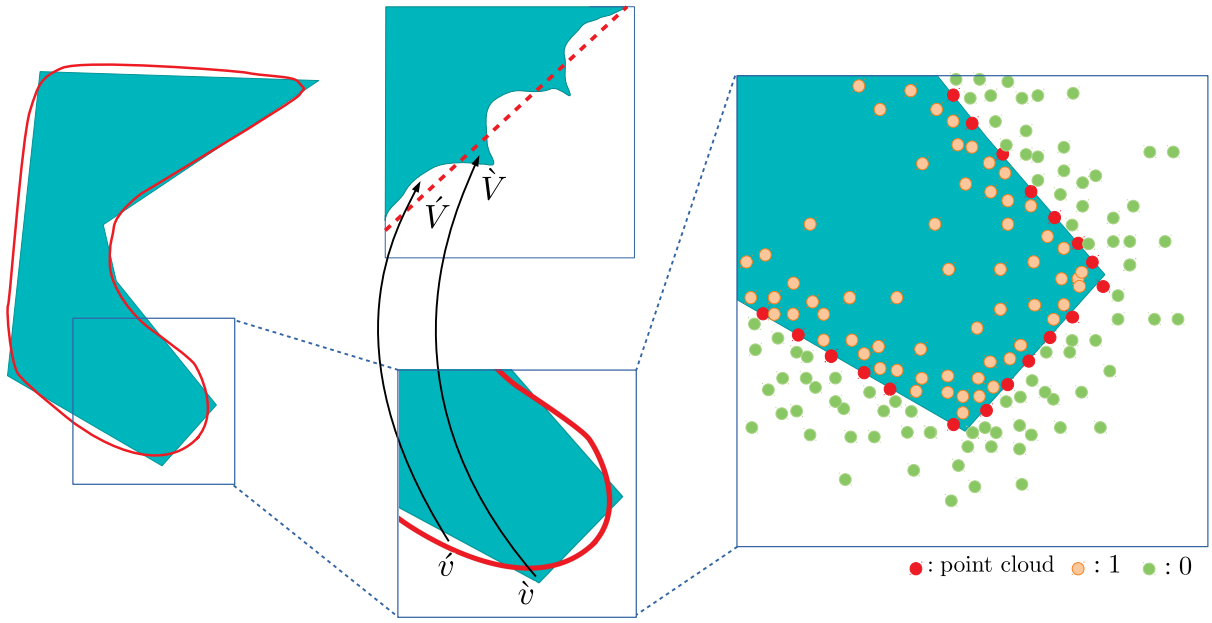


Figure 4.2: 2D illustration of the classifier's decision boundary (in 2D and feature space) and the way of generating training data. On the left side, a target shape (cyan area) is shown with an error-containing decision boundary (red curve) representing the learned occupancy function. Each point is mapped to feature space and classified. The red dotted line indicates the decision boundary of the softmax classifier. The right figure shows how the individual training query points are generated from a particular shape.

the objects uniformly. The drawback of this solution is obvious: points located far from the surface (inside and outside) are easy to classify, and error happens more around the surface ∂M. Uniformly sampling the bounding box will make network training less efficient (due to the proportion of training allocated to empty space within/outside the surface defining the volume). As shown in Figure 4.2, if training converges, areas of positive error $\acute{v}$ and negative error $\grave{v}$ contribute most to the loss function (4.2). In feature space, the corresponding area $\acute{V}$ and $\grave{V}$ push the decision boundary towards different directions. Imprecisely, if the negative and positive samples are symmetrically distributed around the local mesh surface, the decision boundary will overlap with it without bias. So, we first sample points from the object surface, add a small offset to move them off-surface and label them correspondingly. The offsets are drawn from zero-mean normal distributions. See Figure 4.2 (right) for a demonstration (in

2D) of this process's result. As most natural and human-made objects consist of more convex curvature surfaces than concave, the approach will tend to over-sample outside (some small object parts could be narrower than the width of the offset's distribution).

4.3 Generic Object Reconstruction Results

4.3.1 3D Mesh Reconstruction of ShapeNet

In this experiment, we evaluate our occupancy learning approach with the task of reconstructing shapes from unoriented point clouds. We take a subset of ShapeNet [1] (as the same of 3D-R2N2 [130] and Occupancy-Network [16]) and preprocess each mesh to make it water-tight with the code provided by [155]. We refer to it as ShapeNet-13 hereafter, as it consists of 13 classes of shapes. Our training/validation/testing split is the same as [16]. All of the meshes are rescaled and shifted into a unit cube centred at the origin of the coordinate system. 2048 points are randomly sampled from each mesh surface for data augmentation. Each input point cloud P contains 300 points randomly taken from these 2048 points, and Gaussian noise is applied to each point. For generating query points, we add random offsets to on-surface points to make them off-surface. In practice, we draw 6144 points with $\delta = 0.02$ normal distribution to capture details of shape and draw 2048 points with $\delta = 0.1$ that reflecting the overall shape contour.

The architecture of our network is inspired by the segmentation model in [3]. The input point cloud is fed through a cascade of shrinking and enlarging streams. At each convolution block of the shrinking stream, we decrease the size of point cloud (with point max-pooling [3]) and double the depth of features. The opposite is done in the enlarging stream, with point deconvolution [3]. The skipping feed-forward features from the shrinking stream are concatenated to the outputs of corresponding deconvolution layers. These size-related

hyper-parameters are summarized as below:

$$N \to (64, 256) \to (128, 128) \to \overrightarrow{(256, 64) \Rightarrow (256, 128)} \to (128, 256) \to (64, N),$$

where the long arrows on the head represent the lateral connections in U-Net based structures. The skipping (lateral) feed-forward connections from the shrinking (encoding) stream concatenated to outputs of the corresponding deconvolutional layers in the expansion (decoding) stream. Each pair of brackets is interpreted as (#feature dimension, #output points) and $N = 300, 600, \text{or} 1000$ represents the size of the input point cloud. It follows a "factor of two" principle except for the first and last layers. These dimensions are determined by mimicing a similar structure of the point cloud segmentation network in [3]. The justification of this setup emerges from CNN-type of applications where there is a decimation process that happens on the convolutional receptive fields and resolution of the feature map.

After training, our network is used as a frontend, and we use Octree hierarchical sampling and Marching Cubes to produce triangle meshes from each point cloud. We compare our approach against other reconstruction or shape generation methods, including non-learn methods (Ball-Pivoting [79] and Poisson reconstruction [86]) and learning based method (3D-R2N2 [130], DMC [131], and ONet [16]). For the learning based methods, they are all trained with the same dataset ShapeNet-13. We report the reconstruction performance in Table 4.1.

	IoU	Chamfer-L1	Normal Consistency
Ball-Pivoting	-	0.415	0.720
Poisson	0.349	0.534	0.717
3D-R2N2	0.565	0.169	0.719
DMC	0.674	0.117	0.848
ONet	0.778	0.079	0.895
Our	**0.817**	**0.060**	**0.905**

Table 4.1: Shape Reconstruction Results on ShapeNet-13 Dataset, 300 Points, Noise SD=0.05

The reconstruction quality is evaluated in Intersection-over-Union (IoU), average L1 Chamfer distance (Chamfer-L1), and Normal Consistency score calculated against the ground truth, as defined below:

1. Intersection-over-Union: Quotient of output mesh and ground truth mesh's intersection's volume and their union's, i.e., $\text{vol}\left(A \cap B\right) / \text{vol}\left(A \cup B\right)$. IoU compares the overlapped volume with to the union of two shapes. A value of 1 indicates a perfect alignment of the two surfaces, because the intersection approaches the union of the two volumes.

2. L1 Chamfer distance: Sum of an *accuracy measurement*, mean distance from points on the output mesh to ground-truth, and a *completeness measurement*, the mean distance from ground-truth mesh to output. Formally, $\sum_{x \in A} \min_{\text{y} \in \text{B}} |x - y| / N_x + \sum_{y \in B} \min_{\text{x} \in \text{A}} |x - y| / N_y$. It computes average error of the distribution of points that sit in one surface versus another. So Chamfer distance is more directly related to the difference of the surfaces itself. It approaches 0 when the surfaces align perfectly.

3. Normal consistency score: Normal consistency score consists of two terms, *normal accuracy* and *normal completeness*. They are the mean dot product of the surface normals on output mesh and the surface normals at the corresponding nearest points on the ground-truth mesh, and the opposite. Formally, $\sum_{x \in A, y = \text{argmin}_{y \in B} \|x - y\|} \mathcal{N}(x) \cdot \mathcal{N}(y)/2 + \sum_{y \in B, x = \text{argmin}_{x \in A} \|x - y\|} \mathcal{N}(x) \cdot \mathcal{N}(y)/2$. Similar as Chamfer distance, this is a measurement of the divergence of two surfaces, looking at the misalignment between two sets of normals on the surfaces. Considered from a signal point of view, normal consistency score captures difference in high-frequency components.

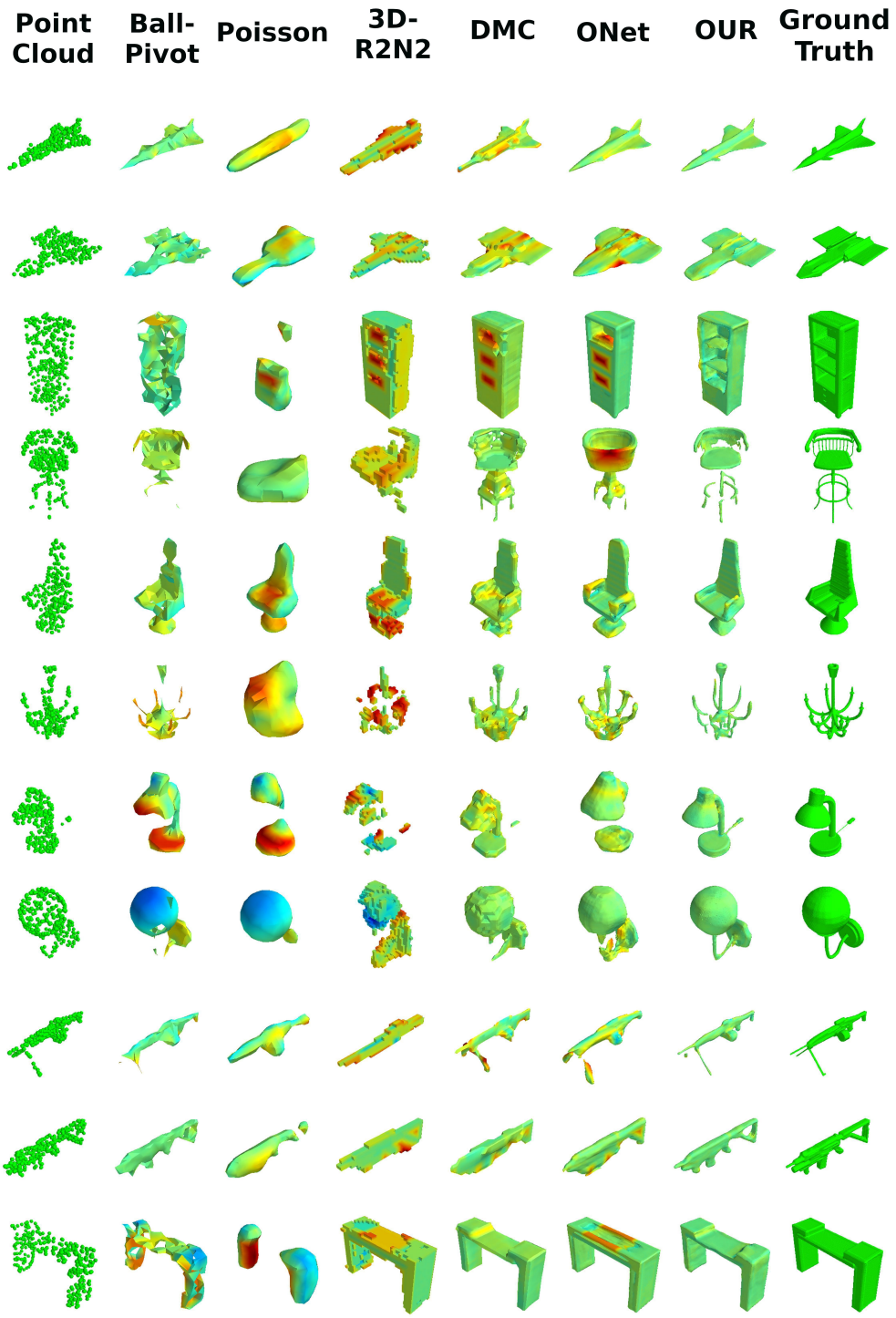


Figure 4.3: Part one of comparisons of our method versus Ball-Pivot, Poisson reconstruction, 3D-R2N2, DMC, and ONet.

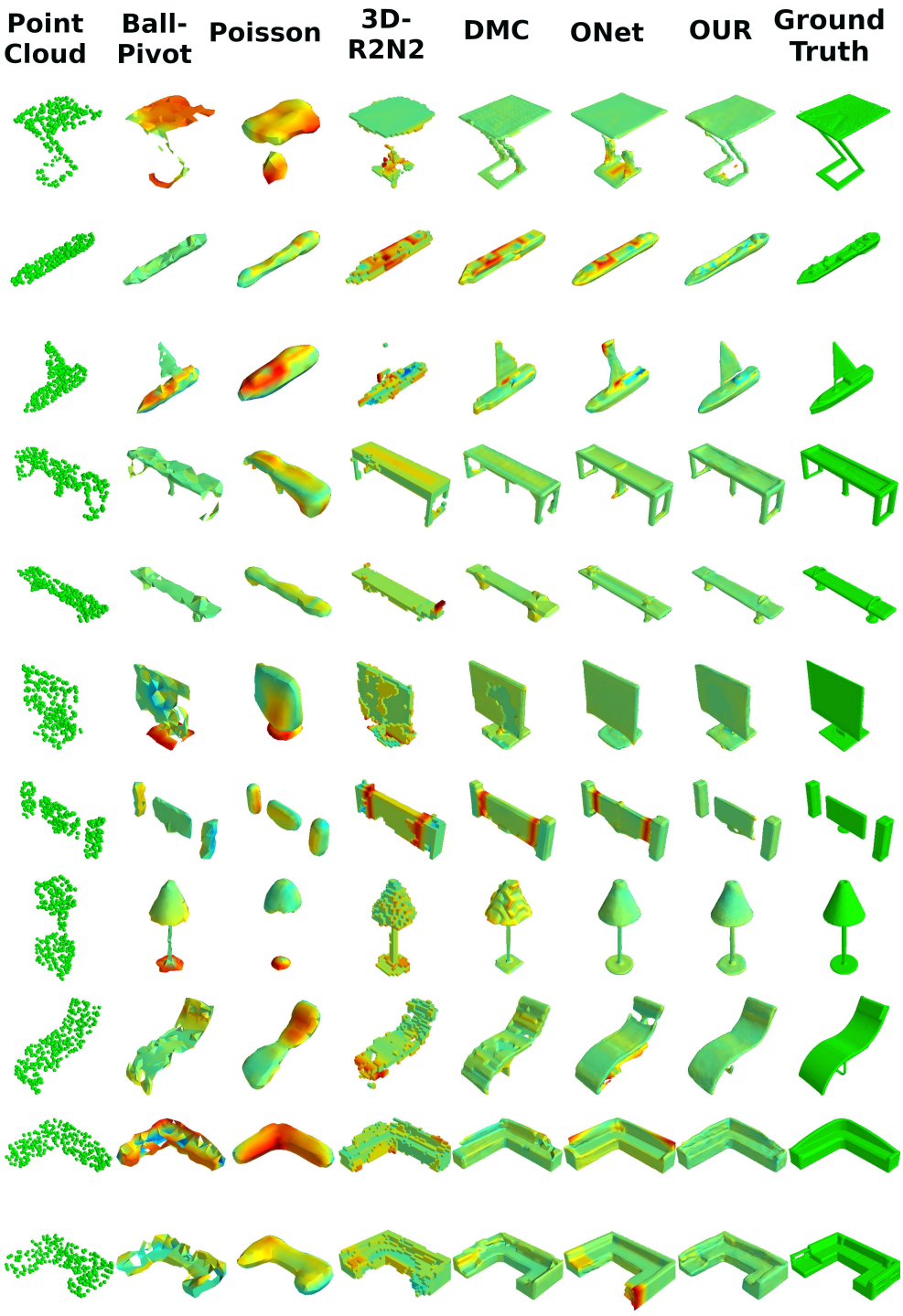


Figure 4.4: Part two of comparisons of our method versus Ball-Pivot, Poisson reconstruction, 3D-R2N2, DMC, and ONet.

	category	airplane	bench	cabinet	car	chair	display	lamp	speaker	rifle	sofa	table	phone	vessel	**mean**
size class	train	2832	1101	4746	1624	1661	5958	1359	1272	5248	767	1134	2222	737	-
	val	404	157	677	231	237	850	193	181	749	109	161	317	105	-
	test	809	314	1355	463	474	1701	387	363	1499	219	323	634	210	-
IoU	Ball-Pivoting	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Poisson	0.388	0.241	0.450	0.543	0.211	0.328	0.268	0.415	0.485	0.426	0.163	0.458	0.536	0.378
	3D-R2N2	0.459	0.425	0.717	0.673	0.505	0.585	0.328	0.701	0.413	0.698	0.468	0.665	0.542	0.552
	DMC	0.546	0.523	0.798	0.736	0.636	0.740	0.480	0.791	0.563	0.798	0.611	0.840	0.608	0.667
	ONet	0.757	**0.715**	**0.861**	**0.831**	0.731	0.813	0.564	**0.823**	0.690	0.867	0.755	0.910	0.745	0.774
	Our	**0.800**	0.694	0.855	0.816	**0.765**	**0.852**	**0.687**	0.814	**0.782**	**0.885**	**0.774**	**0.931**	**0.800**	**0.804**
Chamfer-L1 (less is better)	Ball-Pivoting	0.245	0.241	0.544	0.251	0.575	0.446	0.467	0.711	0.075	0.401	0.633	0.209	0.164	0.382
	Poisson	0.423	0.400	0.579	0.287	0.817	0.571	0.590	0.744	0.193	0.454	0.756	0.318	0.240	0.490
	3D-R2N2	0.153	0.157	0.171	0.201	0.202	0.180	0.283	0.203	0.144	0.161	0.187	0.145	0.181	0.182
	DMC	0.111	0.102	0.114	0.165	0.121	0.099	0.167	0.150	0.083	0.102	0.103	0.064	0.147	0.118
	ONet	0.056	0.059	**0.073**	0.098	0.089	0.076	0.138	0.116	0.060	0.069	0.071	0.041	0.085	0.079
	Our	**0.045**	**0.057**	0.076	**0.076**	**0.076**	**0.062**	**0.078**	**0.107**	**0.039**	**0.058**	**0.064**	**0.032**	**0.058**	**0.064**
Consistency Normal	Ball-Pivoting	0.753	0.704	0.719	0.703	0.669	0.739	0.717	0.726	0.811	0.702	0.716	0.799	0.785	0.734
	Poisson	0.702	0.661	0.768	0.763	0.656	0.741	0.690	0.766	0.786	0.743	0.677	0.802	0.763	0.732
	3D-R2N2	0.651	0.683	0.784	0.714	0.667	0.749	0.583	0.746	0.676	0.751	0.728	0.841	0.626	0.708
	DMC	0.804	0.802	0.886	0.825	0.837	0.900	0.769	0.880	0.771	0.885	0.870	0.947	0.797	0.844
	ONet	0.897	**0.878**	**0.915**	**0.874**	**0.889**	0.925	0.812	**0.897**	0.863	**0.927**	**0.916**	0.970	0.859	0.894
	Our	**0.910**	0.858	0.904	0.848	0.885	**0.928**	**0.859**	0.881	**0.902**	0.925	0.909	**0.974**	**0.880**	**0.897**

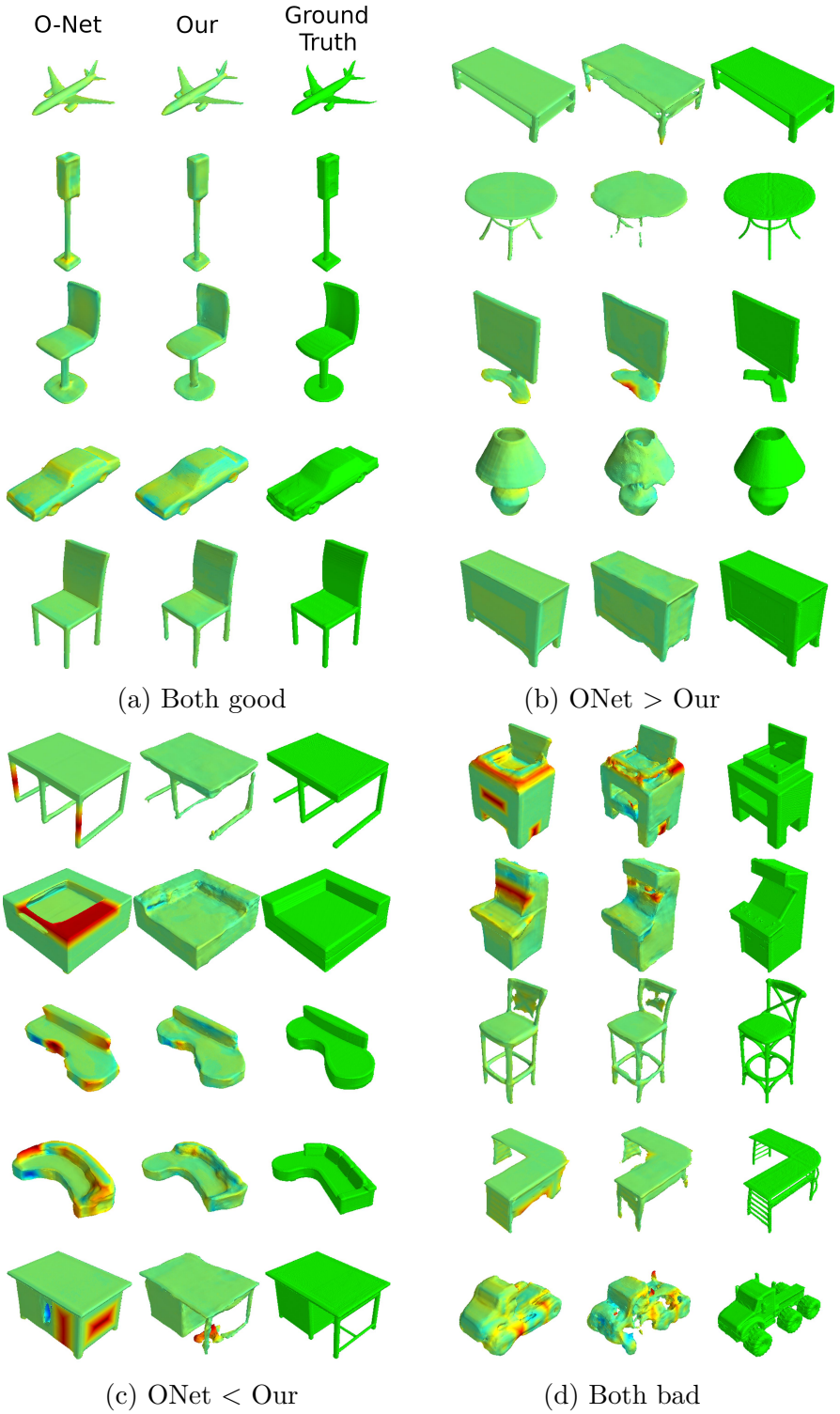
Table 4.2: Per-class Statistic of Reconstruction Results on ShapeNet-16 Dataset, 300 Points, Noise SD=0.05

Over all categories in Shapnet-13, our method outperforms other methods on IoU and Chamfer-L1 metrics by a large gap but only is marginally better than ONet/DMC on normal consistency score. For visual inspection and evaluation, we plot the output meshes of different methods and visualizing surface error in Figure 4.3 and 4.4 where the error is color-coded. To make it easier to compare with the ground truth, we use "jet" colormap to visualize the reconstruction errors. The surfaces reconstructed out of the ground truth is plotted in red, and inside of the ground truth is plotted in blue. We scale the error across each example to maximize the color range, but keep it linear and keep zero-error colored in light-green constantly. The dataset contains 13 different classes of objects. Those class names and per-class statistics of the performance are shown in Table 4.2. Our method outperforms other methods for most shape categories. Importantly, it wins by a large margin in classes that have larger shape variation (like airplanes and vessel) or contain small structure and fine details (like lamps and rifles). This finding can be confirmed by observing the reconstruction results shown in Figure 4.3 and 4.4.

4.3.2 Difference between our method and ONet

Among the methods compared, ONet [16] is the one closest to our method in terms of quantitative metrics. ONet produces some good-looking reconstruction in Figure 4.5 and seems to have advantages for learning from some particular classes of shapes (Table 4.2). We are therefore interested in its behavior and the differences from our PCNN-based occupancy learning. Browsing through output meshes of ONet and our method, we select some representative shapes and group them as a) both working well; b) ONet works better; c) our method performs better; d) both do not appear to work well. See Figure 4.5 for some examples in these four groups. There are at least three noticeable characteristics worth mentioning:

1. First, ONet is good at learning and recovering the classes that have less variations in



(a) Both good (b) ONet > Our

(c) ONet < Our (d) Both bad

Figure 4.5: Comparison of our method with ONet. We collect four groups of test shapes according to if it's reconstructed well by ONet and Our method.

geometries of individual objects within the same category. Shape classes like airplanes, tables, and cars are examples of "less variant" categories, as you can easily imagine a stereotypical version of them - in other words, it is easy to think of a mean/average representative shape (cf. Figure 4.5a, 4.5b and Table 4.2). ONet uses an encoder-decoder network using a PointNet [4] encoder to extract a latent code from the input point cloud. As a result, the categorical and semantic features are captured relatively well, but this also introduces artifacts and problems. ONet tends to remember stereotypical shapes and attempts to fit every input shape to those remembered corpora, resulting in imagined infill on shapes where it is not present (Figure 4.5c). This approach can fail more catastrophically when attempting to reconstruct previously unseen classes (outlined later in Figure 4.7).

2. Second, ONet produces cleaner and more regular mesh than our methods, especially for the shapes that could be approximated by the assembly of primary elements like planes, cubes, circles, and discs (cf. Figure 4.5b). While our outputs, even for the shapes reconstructed more accurately than with ONet, contain more noise and non-smooth surfaces. This observation explains our method's relative weakness on the surface normal consistency metric in Table 4.1. This phenomenon can likely be attributed to the use of RBF-based PCNN (local and grained) to model the occupancy function, while ONet uses a conditional batch normalization [156] network as the decoder. However, we believe sacrificing regularity and smoothness is worthy as our method can more faithfully reconstruct the shape (i.e. in the absence of representative training samples). As shown in Figure 4.5c, individual shapes from within a given category can vary greatly (like the desk with a standard set of 4 legs versus that with horizontal feet or squared versus asymmetrically shaped sofa), thus individual cases can often consist of long curved surfaces and global asymmetries that are not adequately captured by the

encoder. Note also, that high frequency jitter [4] on the reconstructed surface is easy to remove with mesh spectral filtering.

3. Lastly, there are some challenging shapes neither ONet nor our method can recover satisfactorily from 300 input points with noise (Figure 4.5d). Possible reasons may be two-fold. First, these objects are composed of multiple, highly intricate and complex parts so that 300 un-oriented points do not provide enough information about their shape. Meanwhile, the training dataset does not contain enough similar objects for these shape corpora to be adequately learned, so that both networks have trouble recovering/hallucinating the missing information. Despite this, the low resolution input point cloud certainly illustrates their comparative power to hallucinate/infill likely surface characteristics.

4.3.3 Scaling and Generalization

Since our method can reliably reconstruct sufficient fine details with local geometric evidence, we expect it to extend to reconstructing denser sample shapes and gaining accuracy from richer information. Meanwhile, the proposed sampling layer for PCNN and the network architecture are easy to scale-up. We conduct several experiments that have varied number/density of input points and different level of noise. The results are reported in Table 4.3. We show some 3D shapes reconstructed by our networks configured and trained with 300/600/1000 input points in Figure 4.6. From the metrics and visual results, we can see that reconstruction accuracy and quality increase regularly with denser input and lower noise intensity. In these experiments, the network architecture or training parameters are unchanged from the initial experimental setups, except for the size of Gaussian kernels used in each PCNN layers, which are set as reciprocal of the number of points of a particular layer.

[4] The noise is also roughly constant frequency, as the wave length is proportional to the RBF kernel size and interval between RBF centres.

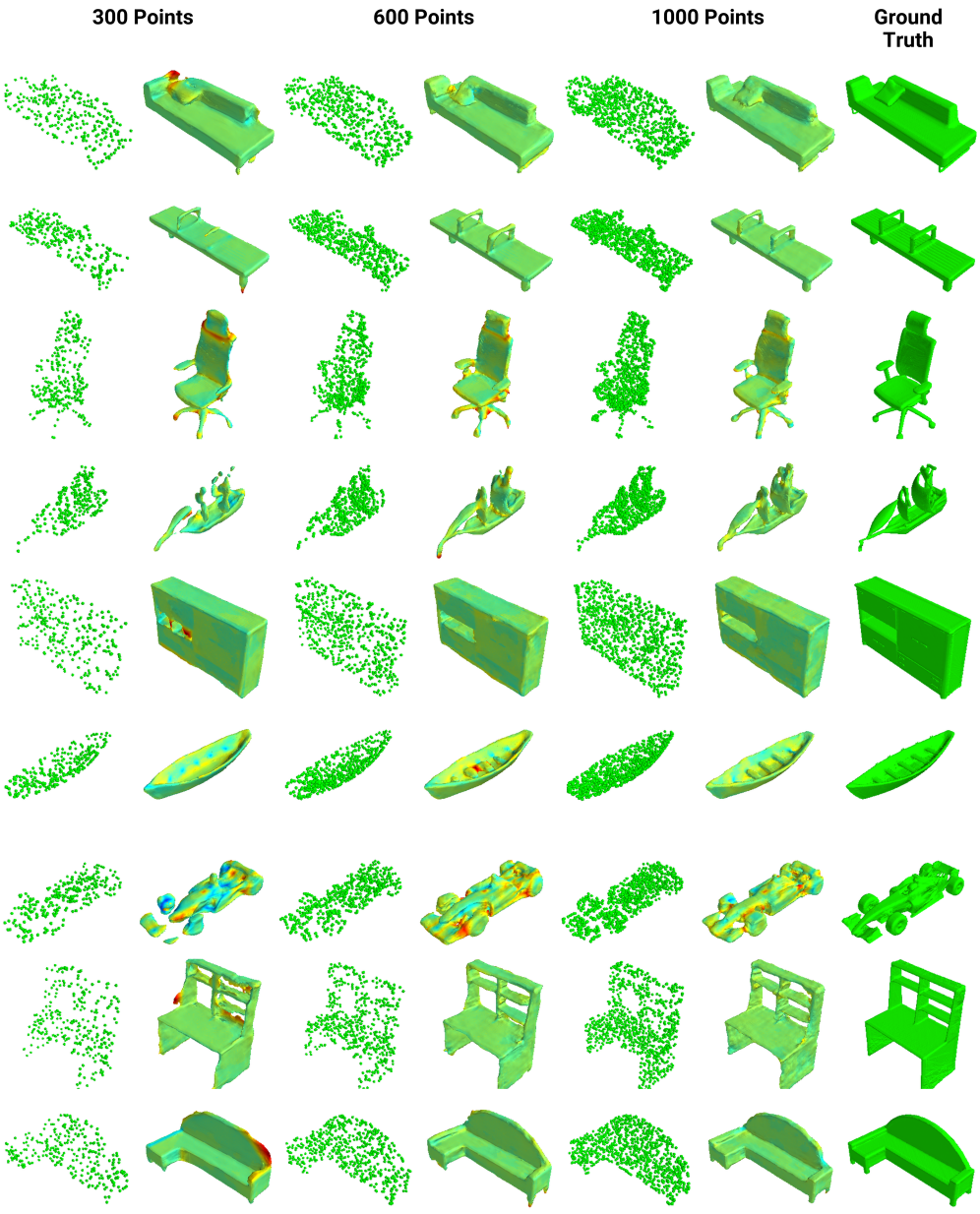


Figure 4.6: This figure shows the shapes generated from different densities of input point clouds (corrupted by normal distribution noise with 0.05 variance), using correspondingly trained networks.

Density	Noise	IoU	Ch.L1	Normal
150	0.025	0.780	0.083	0.886
300	0.05	0.817	0.060	0.905
	0.025	0.829	0.061	0.909
600	0.05	0.846	0.053	0.922
	0.025	0.865	0.048	0.924
1000	0.05	0.870	0.045	0.932

Table 4.3: Performance of our method on ShapeNet-13 Dataset, with different density and noise. The metrics are averages over all shapes in the evaluation set (containing 13 categories as shown in Table 4.2).

To test our method's capacity to generalize to unseen object categories, we apply the 1000-point model trained on ShapeNet-13 to the McGill mesh dataset [157], *without any re-training or fine-tuning*. Point cloud sampling is done in the same way mentioned in 4.3.1. A 1000-point ONet is trained accordingly to make a comparison. The reconstruction results shown in Figure 4.7 demonstrate that our method has remarkable potential for up-scaling and domain generalization.

Our network is capable of recovering gross shape and fine details. In contrast, ONet-1000 suffers from catastrophic failure with such dense input - the reconstructed shape bears little detail or gross features in common with ground truth. It can only hallucinate shape from what is knows, because it relies on the learned low-dimensional shape embedding space that encodes categorical features. Given an unfamiliar input, it fails to associate it with latent embedding. For example, ants have no representation in the embedding space, so the first two ant shapes are mapped to latent codes that has no clear meaning (the gross shape of these ants appear to fall in the vicinity of car or truck) and then decoded.

As is already argued, our network can directly extract local features for each query point. In addition, we speculate the lateral connections allow for aggregated local features to come into play without having to consider whether a learned global arrangement of those features has been observed. In other words, while the co-occurrence of certain local shape features

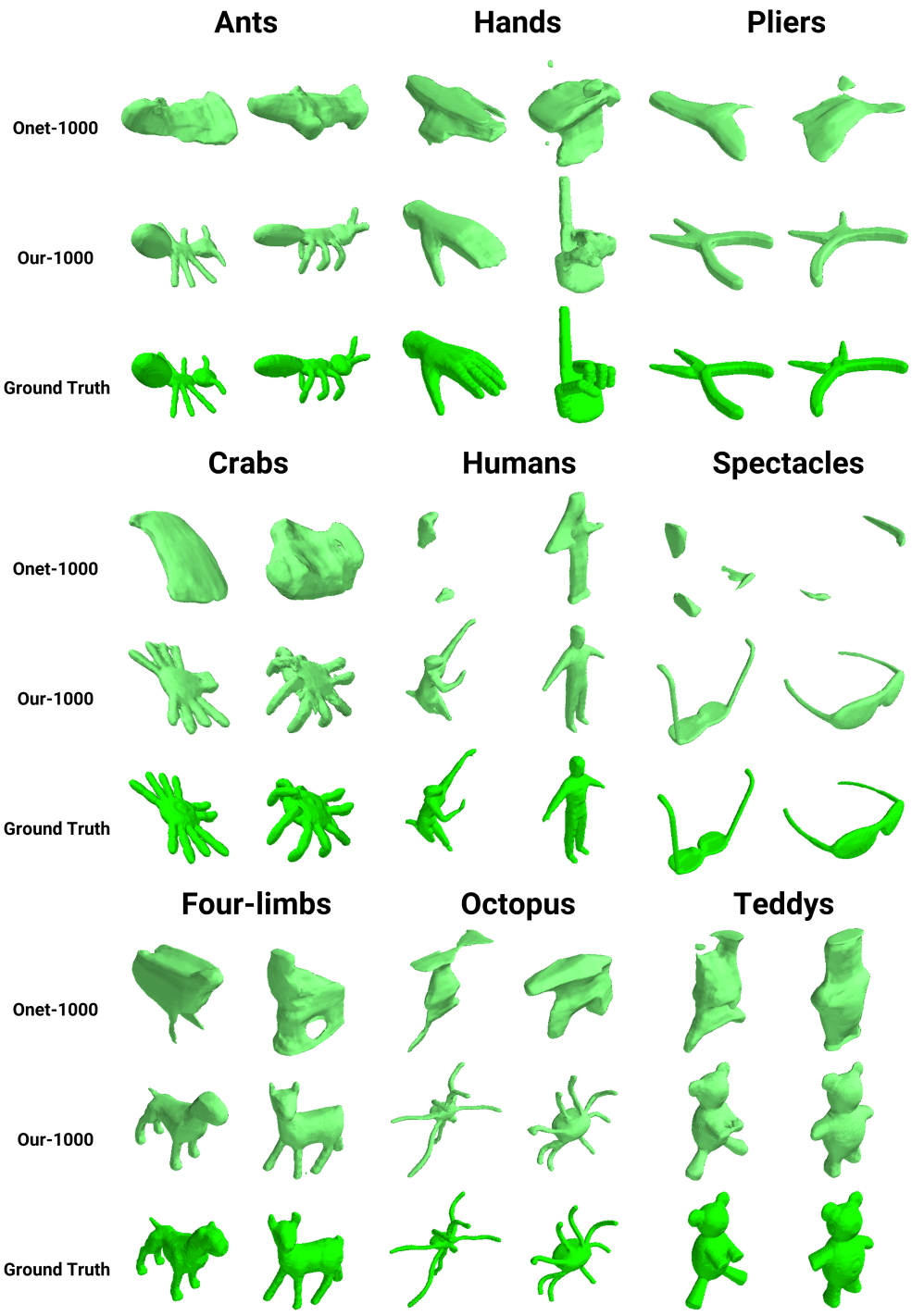


Figure 4.7: Comparison of our method with ONet on unseen classes in McGill dataset [157].

extracted by the network play an important role in reconstruction, it may still retrieve learned local representations. When all of the related neurons are firing it's able to reconstruct the whole shape well. but even when some of those neurons are firing in the same areas, it is able to use those to reconstruct locally, keeping a faithful representation of the point clouds it's actually seeing.

4.4 Rotation Invariance/Robustness of SCFL Surface Reconstruction

Traditional surface reconstruction methods like MLS [158] and the Poisson method [86] are invariant, undergoing geometric translation and rotation. Unfortunately, most recent point cloud deep learning frameworks, including those learning-based ones in Table 4.1 do not have such properties. We intuitively feel the implicit neural methods like ONet rely more on a canonical pose of the input point cloud, as recognizing and fitting stereotypical shape is pose-related. Figure 4.8 shows an experiment that we conducted to deliberately rotate a chair's point cloud into different poses to test ONet and our method. Both networks are trained with the same canonical oriented objects in ShapeNet-13. While they all produce a successful reconstruction in regular orientation, ONet fails to react robustly to the rotations. In summary, our occupancy learning method is more faithful to local/relative shape information. The shape detail is recovered more from local shape evidence than in ONet and other learning-based generation which instead tend to remember and fit to stereotypical shapes of the categories.

Dynamic Plane Convolutional ONet (DPC-ONet) [25] is a recent implicit neural representation method that pushes forward the quality of surface reconstruction by encoding the shape features onto multiple 2D dynamic planes. As reported in [25], DPC-ONet has better generalization towards inputs shapes with unseen orientations, comparing with ONet [16] and

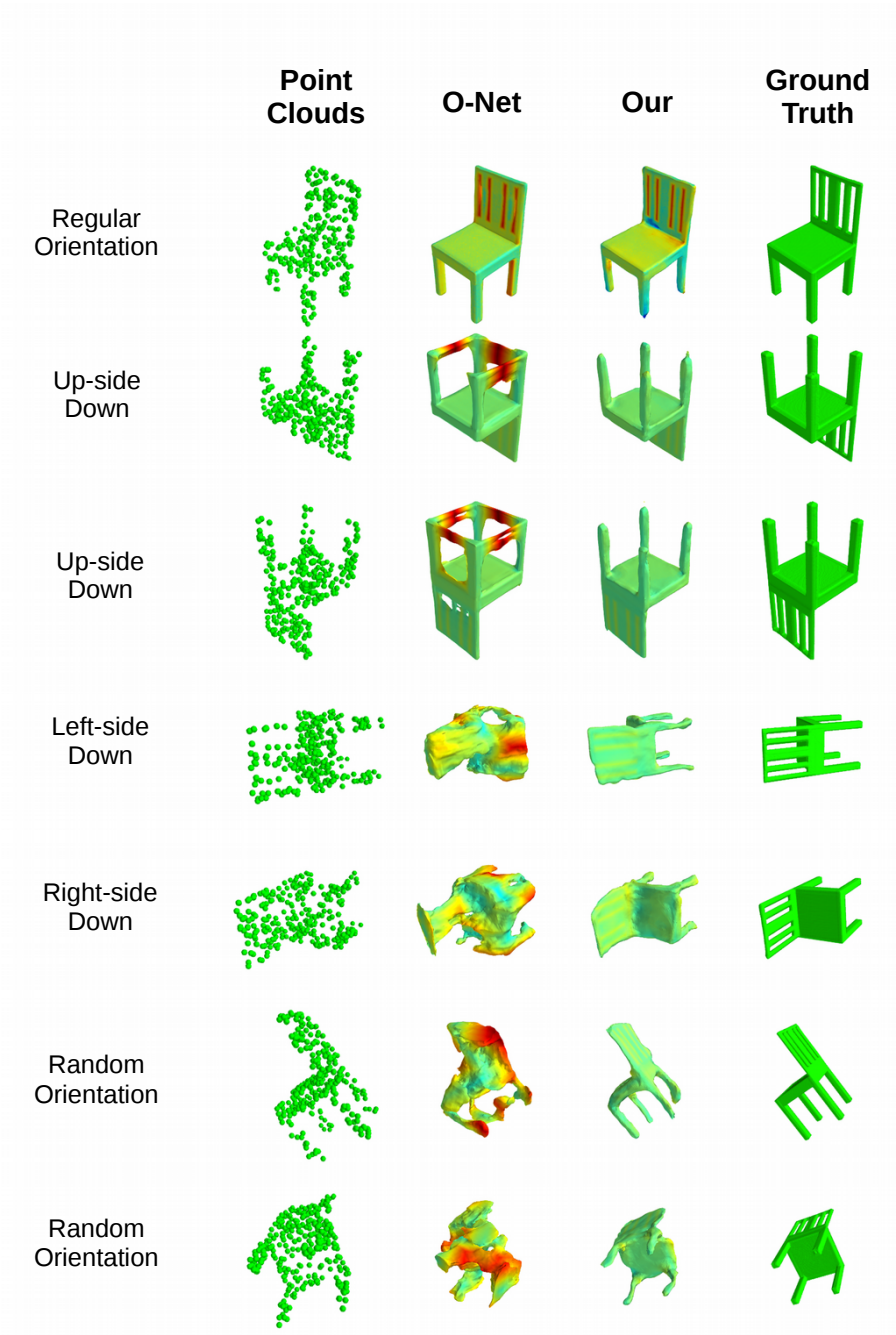


Figure 4.8: The effects of applying geometric rotations to input point clouds. This experiment shows that a simple small rotation can crush ONet entirely, while our model is relatively invariant to rotation.

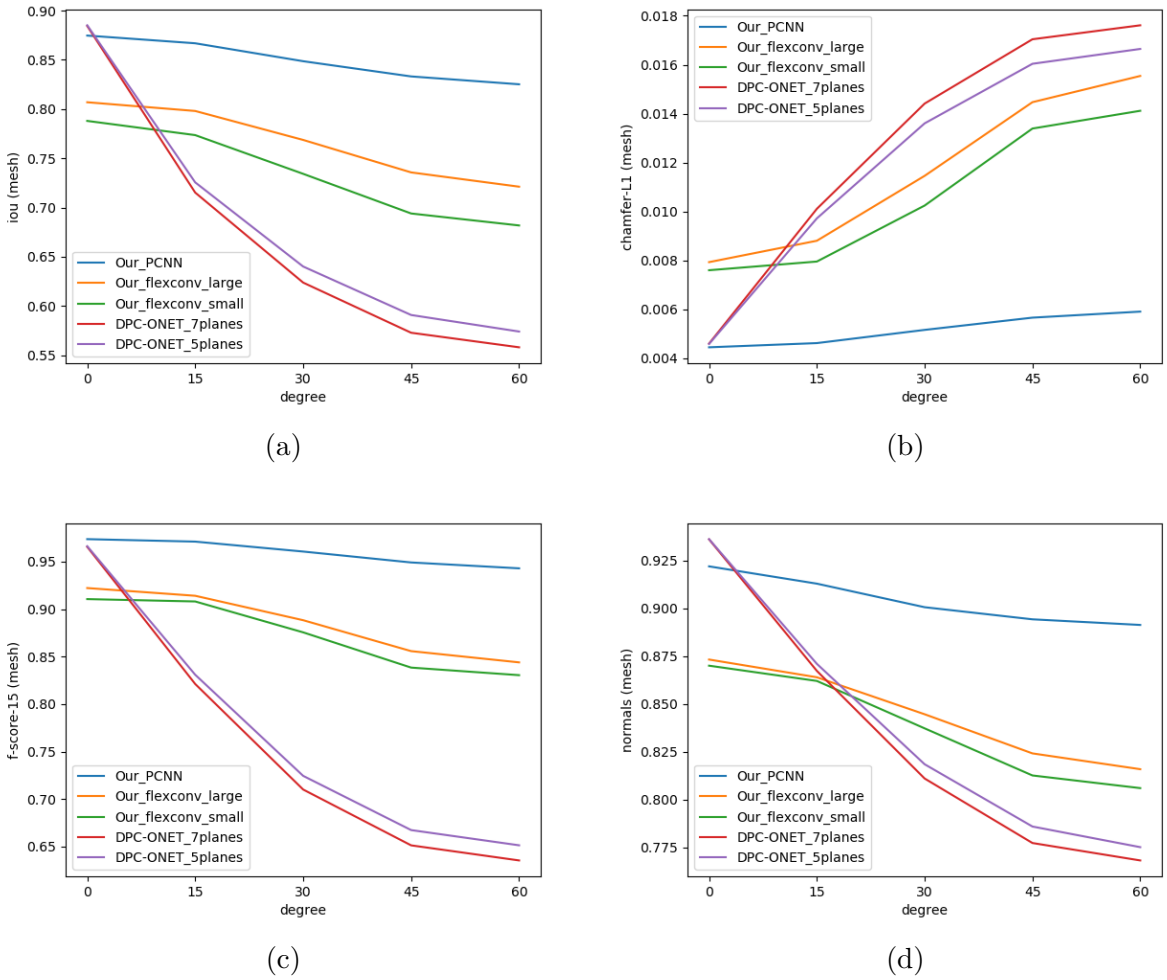


Figure 4.9: Rotation invariance experiment on ShapeNet. We plot four different metrics (IoU, Chamfer-L1 difference, f-score, and normal consistency) to compare with DPC-ONet on test shapes rotated with angles uniformaly sampled from $[0, \Theta_{\max}]$. Flatter curves represent more invariance (less variation with change of pose) on each metric.

ConvONet [17], therefore, we evaluate our method (including one PCNN and two Flex-Conv based versions) with the same experimental setups proposed in [25]. Again, all networks are trained with the same canonical oriented objects in ShapeNet-13, and tested with the same evaluation set. This test applies random rotation to the inputs in the ShapeNet test set along $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ axes with random angles drawn uniformly from $[0, \Theta_{\max}]$ for each test shape. The max rotation angle $\Theta_{\max}$ is taken as 15°, 30°, 45°, and 60° for different amounts of

deviation from the canonical training pose. The result of this experiment is shown in Figure 4.9. We can see the progressive drop of reconstruction quality when the input is rotated, but our method has a better generalization and robustness on irregular orientations. One thing worth noticing when comparing with DPC-ONet [25] is that our networks (PCNN and two Flex-Conv based versions) are trained with a simplest loss, the binary cross-entropy, while DPC-ONet method involves a dynamic plane dependent similarity loss.

4.5 Fast Iso-surface by Space Triangulation

In most existing implicit neural learning based reconstruction methods [15, 11, 10, 25], Multiresolution IsoSurface Extraction (MISE) [16] is used for fast occupancy value inference and triangle mesh generation. Our point convolution architecture requires more computation than the MLP-based architecture networks used in [15, 11, 10]. We propose an iso-surface reconstructing algorithm that is even faster than MISE for our learned occupancy network. This method may reduce the reconstruction time by one order of magnitude. This method, sharing the same idea with MISE that reduces the number of query points where the function value can be inferred. We generate a small sample point set that is much less than the number needed in MISE, and only infer occupancy functions at those points. The occupancy function's value is approximated by interpolation of the sparse sample.

Fundamentally, this is equivalent to the piece-wise linear continuation method [65] which evaluates the function with a triangulation around the iso-surface. The original method [65] considers the general problem of piece-wise approximating level-sets of a continuous function. Without any prior-knowledge about the geometry and topology of the shape, it has to trace the iso-surface by incrementally pivoting tetrahedrons at the edge of previous step constructed mesh. In our case, the input point cloud is itself a geometric approximation of the iso-surface that we initiate from. So our method may extract a set of query points first,

and then evaluate the occupancy function at these points in parallel. The main idea is to build a triangulation of the space which has 1) finer resolution (smaller tetrahedrons) around the iso-surface, 2) coarser (larger tetrahedrons) in space far-away from it, and 3) containing enough elements in the occupied space (inside) of the shape. The third requirement ensures a good sampling of thin structures and sharp edges.

For above requirements, we purpose a scheme for extracting query point set that efficiently samples the function. This is inspired by Eigen-Crust [76], a Delauney Triangulation-based reconstruction method. The query point set is constructed as $\{Q', V(Q')\}$, the union of: 1) Q', random points that's generated by adding Gaussian noise in the input point cloud P, and 2) $V(Q')$, vertices of the Voronoi diagram of Q'. Voronoi diagrams segment space into cells (polyhedrons) corresponding to the points in Q' containing points of $\mathbb{R}^3$ that do not haves a small distance to any other $q \in Q'$. As such, those vertices have same distance to at least three points in Q'. The reason for taking Voronoi vertices as sample points comes from the observation that pole-vertices [5] form the medial axis of the inside and outside space (thus it is likely they can be reliably classified) and the non-pole-vertices are close to the point cloud (making samples more dense around the surface).

After feeding $\{Q', V(Q')\}$ into the network, we will have the occupancy function's value at each point. If the sample points $\{Q', V(Q')\}$ are distributed well, i.e. tightly enclose the object surface, linear interpolation could be enough to approximate the predicted occupancy function. We take the tetrahedron interpolation suggested by [159]. We first construct Delauney triangulation on $\{Q', V(Q')\}$, parsing the free space into tetrahedrons. As depicted in Figure 4.10a, each tetrahedron is defined by it's vertices $A, B, C, D \in \{Q', V(Q')\}$. If the point to be interpolated, q, is inside this tetrahedron T_{ABCD}, the interpolation is:

[5] Poles are the two vertices of one Voronoi cell that are most far away from the centre, and locat in opposite side of it.

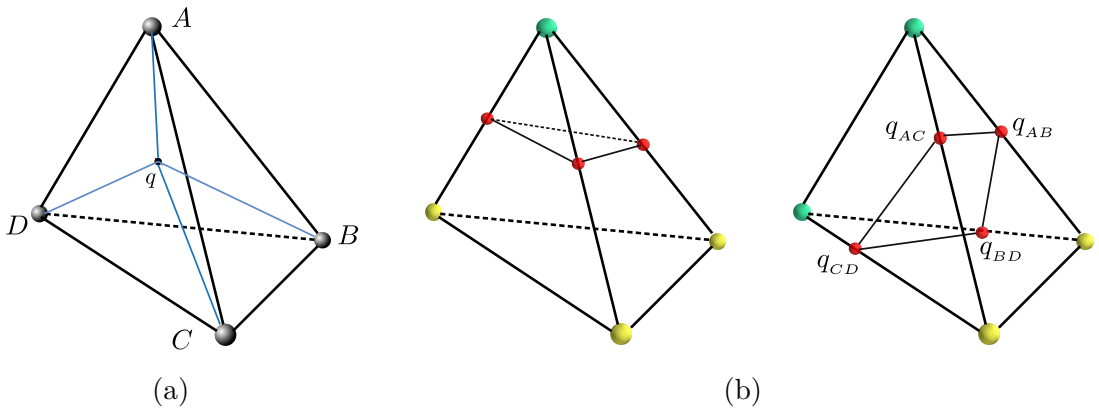


Figure 4.10: Illustration of the proposed tetrahedon interpolation.

$$\hat{g}(q) = \frac{1}{\text{vol}(T_{ABCD})}[\text{vol}(T_{ABCq}) \cdot g(D) + \text{vol}(T_{ABqD}) \cdot g(C) + \text{vol}(T_{AqCD}) \cdot g(B) + \text{vol}(T_{qBCD}) \cdot g(A)],$$

where $\text{vol}(\cdot)$ indicates volume of a tetrahedron.

This interpolation-based method could be twisted to be more efficient: i.e. to directly build a triangle mesh on the tetrahedrons (Figure 4.10b). There are two different situations for tetrahedrons that intersect with the iso-surface. The first (Figure 4.10b left) is when one vertex is located in the other side of the surface. In this situation, only one triangle needs to be added to the mesh. We simply find out the cross-surface on the corresponding three edge with 1D linear interpolation. The second is illustrated in Figure 4.10b right, and we add two triangle faces. We either insert and edge $q_{AB} - q_{BD}$ or $q_{AB} - q_{CD}$ in the way that make the triangles as wide as possible. Hereinafter, the first fast algorithm is referred to as tetra-interpolation whereas the second as tetra-triangulation.

Long inference and reconstruction time is a big issue of implicit function based shape learning, and is roughly proportional to the number of query points considered. Theoretically, our fast algorithm could be directly used with other implicit neural representation methods like DeepSDF [11], IM-Net [15], ONet [16], and DPC-ONet [25].

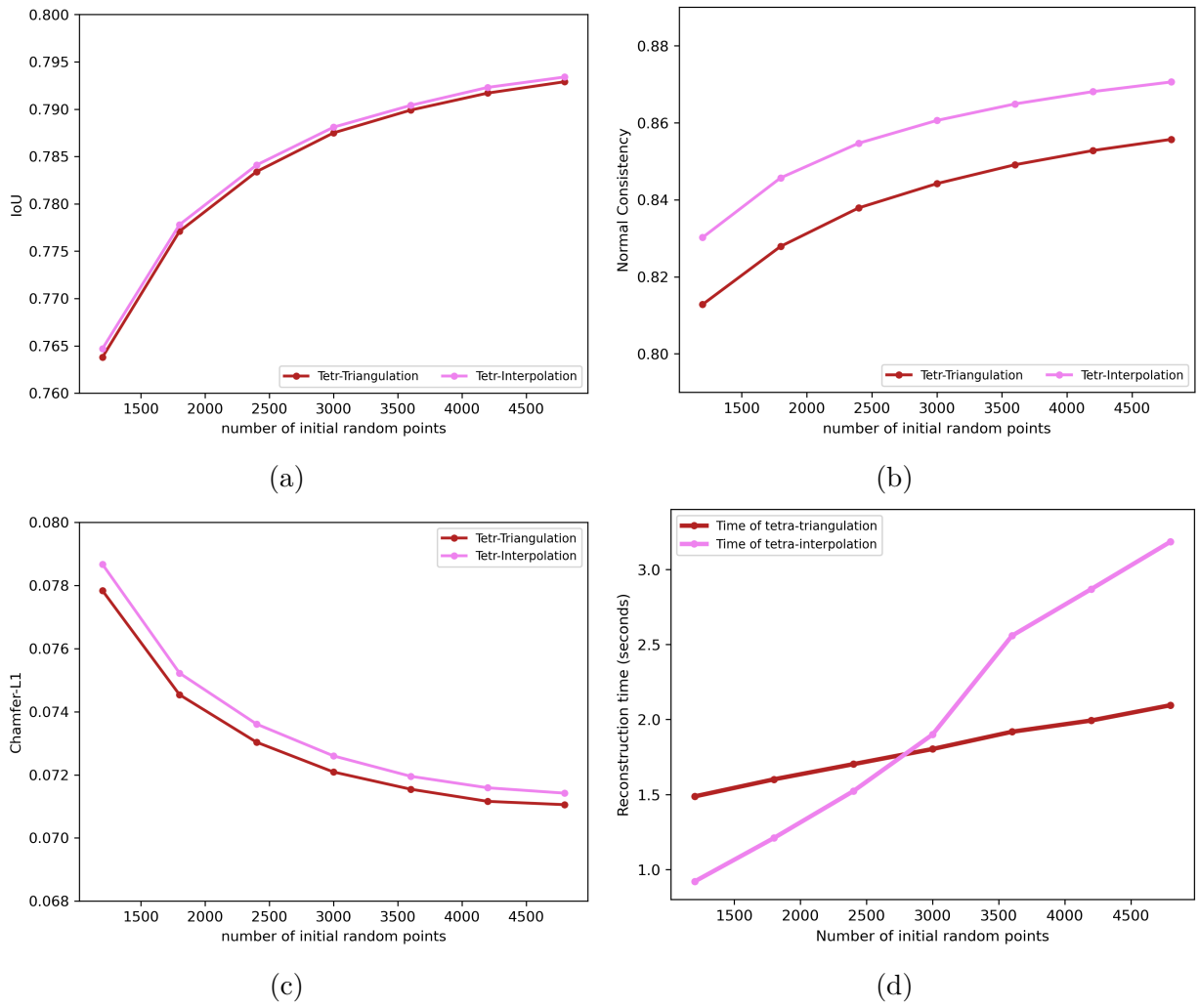


Figure 4.11: The reconstruction quality measurements and inference time of our fast reconstruction algorithms. The metrics are plotted as curves that change with respect to the number of initial random sampling points.

To make a comprehensive study of it's effect of speeding up reconstruction and any assessing trade-offs this incurs on accuracy, we use different numbers of initial sample points Q' and study how this parameter affects the reconstruction accuracy and the average time spent for inferring one instance of shape. The relations are shown in Figure 4.11. From the left graph, we can see that the performance increase correspondingly with more initial sample points. The IoU metric doesn't diverge with different implementations: tetra-triangulation

and tetra-interpolation, whereas they surpass each other for normal-consistency and Chamfer-L1 metric. This may be explained by the fact that tetra-interpolation involves a Marching Cubes procedure implicitly smoothing the surface, which makes a better surface normal but worsens the surface-to-surface distance metric. In Figure 4.11(d), we observe a linear increase of the amount of vertices $V(Q')$ w.r.t. Q', resulting in a roughly linear time complexity (tetra-triangulation is more efficient). Our experiments are conducted on a system of a Xeon CPU working at 4Ghz, and a GTX1080ti GPU deploying the network. Without fast algorithms, the reconstruction of one shape takes 10-20s. In Figure 4.12, we show results of the fast reconstruction algorithms. They are compared with ground-truths and the meshes reconstructed by Octree volumetric sampling and Marching Cubes.

4.6 Summary

In this chapter, we show how to use our SCFL method to learn occupancy functions for point cloud surface reconstruction. Like [26, 25, 39, 160], this work address the deficiencies of prior works [16, 17] that use MLPs to represent 3D shapes as continuous functions. We generalize point-based convolution operations to the continuous domain to produce 3D spatial functions. The target shape implicit representation is learned with query points and the corresponding occupancy states at those points. The spatial pooling used in our method achieves efficient local-global shape feature aggregation without making use of a 3D regular grid. We conduct experiments to compare our method with the well-known established traditional surface reconstruction methods [79, 86] and influential implicit neural representation based methods [130, 131, 16, 25].

As demonstrated in many other studies, learning-based reconstruction methods have advantages over traditional deterministic methods in situations where data acquisition ability is restricted. Non-learning methods have limited ability to work with sparsely

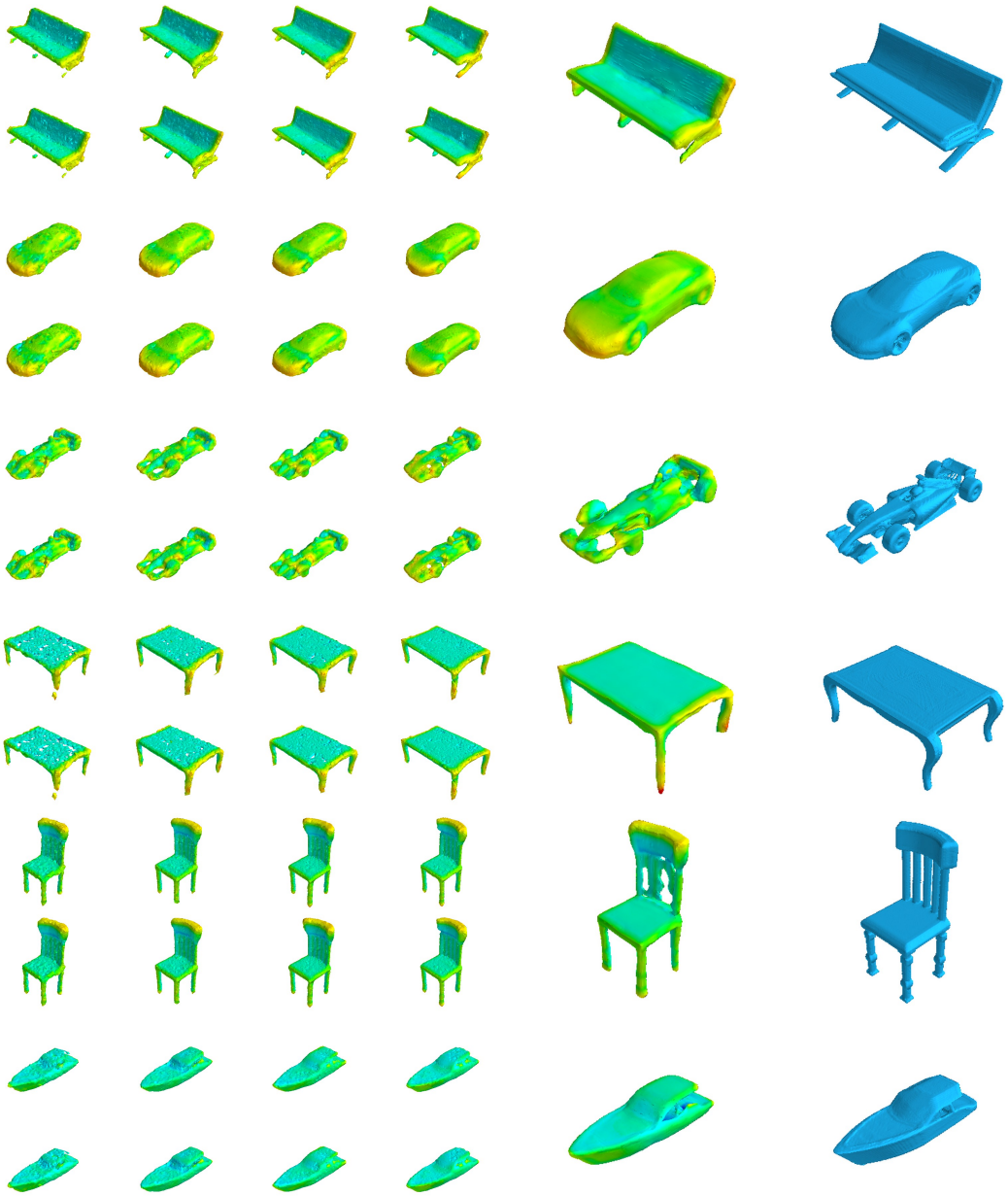


Figure 4.12: Examples of the proposed fast reconstruction algorithm. For each of shape instances, we plot its ground-truth mesh (right-most column) and the reconstruction without fast tetrahedron interpolation (second on the right), along with the meshes generated from the fast algorithm. The first and second row are generated with tetra-triangulation and tetra-interpolation respectively.

sampled point clouds, while learning methods may hallucinate or infill shapes base on the learned prior knowledge about a category of objects. This implies that learning approaches can have utility over non-learning or deterministic approaches. In addition, the proposed occupancy learning method is specifically capable of reconstructing shape faithfully, by combining global shape priors and local features extracted from the arrangement of sparse points. With the success of SCFL in this surface reconstruction study, we are optimistic about generalizing this method to learn other continuous properties of 3D space other than occupancy for future work, for example, the elastic property of materials and tissues (in medical applications where surfaces naturally undergo deformation), helping to model non-linear shape deformation/animation. Our method could also be used to learn the deformation field in non-rigid 3D reconstruction/registration/part correspondence. Eventually, the topic that has received most immediate attention in this dissertation is in the domain of medical imaging problems. Specifically, as covered in the next two chapters, we harness the powerful shape representation ability and differentiability of our SCFL based occupancy network to improve performance in challenging liver image-to-physical registration and tumor-bearing whole-brain segmentation.

Chapter 5

Improving Non-Rigid Registration in Image-Guided Surgery with Implicit Neural Representations

5.1 Overview

Non-rigid image-to-physical registration is a crucial component in image-guided liver surgery. To overcome the problems caused by noisy, partial, and sparse intraoperative sampling, we apply our novel occupancy learning method onto mesh-to-point-cloud registration, aligning the preoperative liver image to intraoperative physical samples. We use a point cloud network with the method proposed in the previous chapters to reconstruct occupancy functions from sparse sample points and use this reconstructed liver to guide non-rigid registration. Experiments show this method reduces Target Registration Error (TRE) of rigid and non-rigid baselines by 21.5% and 11.8%. For training this occupancy network, a novel crossover method is proposed to synthesize deformed liver meshes and points. We demonstrate that the system is robust to the imperfectness of generated training data. This method might be useful in

other areas that require soft-tissue registration, where only very sparse data is available during acquisition.

5.2 Introduction

Image-guided surgery is an area that has received a considerable amount of attention in recent decades. In image-guided surgery, intraoperative localization of tumors and subsurface structures are provided to surgeons to enhance surgical precision and update preoperative plans. To this end, a variety of intraoperative medical imaging techniques are typically used to allocate organs and tissues. During open-abdominal live surgery, for instance, a representation of an organ (e.g. liver) can be acquired with an optically tracked stylus by swabbing the visible organ surface [161]. In laparoscopic liver surgery, more specifically, ultrasound imaging instruments and Cone-Beam CT (CBCT) can then be used to acquire a volumetric representation of an intraoperative liver that can drive registrations using nonlinear biomechanical deformation models [162, 163]. Recently, computer vision methods like binocular stereo and visual tracking have been explored as new structure acquisition tools in liver surgery [164, 165]. Across these types of approaches, there exists a trade-off between acquiring accurate, dense, and complete structures and minimizing interventions and invasive steps of the surgery.

In this section, we focus on reconstruction and registration of the liver in image-based surgical applications. Specifically, we address the non-rigid registration of a liver mesh (constructed from preoperative scans) to a point cloud (sampled intraoperatively from liver surface) registration. Such registration is crucial, as any inaccuracies in surgery guidance can be useless or even harmful. Non-rigid registration is very challenging to perform; due to a compromised ability to acquire representations of the liver under the constraints of the live surgical environment. Primarily, there is limited ability to acquire a complete and

dense scan of the organ surface, with a tendency for only partial views, or the acquisition of only very limited point clouds (that are typically very sparse, or non-uniformly sampled w.r.t. the organ as a whole). Moreover, liver and other soft tissues may undergo substantial deformation during the surgery operation itself, compounding the problem of aligning to a known model when again, only sparse data is available during acquisition.

In previous approaches to this challenging problem, Collins et al. [161] use a 2D polynomial function to model the deformation of liver's posterior surface and optimize transformation, rotation, and deformation parameters jointly with the Levenberg–Marquardt (LM) algorithm. Heiselman et al. [22] model laparoscopic deformations with boundary conditions applied to control points selected from surface of liver. Some researchers [162, 163] have suggested that non-rigid liver registration should consider the limitations of the Linear Elastic (LE) assumption and have proposed associated nonlinear biomechanical deformation models. In this work, we use a LE-based deformation model similar to [22], because 1) the data is too sparse for driving complex models, and 2) deformations in this dataset do not seem to be too excessive.

Since clinical validation is challenging to perform, phantoms [22, 166] and animals [162] have typically been used for validating registration methods. Brewer et al. [21] built a dataset (published online as a challenge) by developing a phantom based evaluation system and sampling deformed surfaces with an optically tracked stylus. The points are sampled sparsely from only $<50\%$ coverage of the liver surface. Human factors like swabbing noise are also involved in the data collection, wherein our goal is to explore how our network can be adapted to, and can aide in the goal of non-rigid registration needed for this task.

Our method is thus directly based on the implicit neural representation and surface reconstruction methods discussed in the previous chapters. Our SCFL method (as outlined earlier) uses extended point convolution operations (PCNN [3] and Flex-Convolution [19]) to predict volumetric occupancy functions from sparse input point clouds. This approach

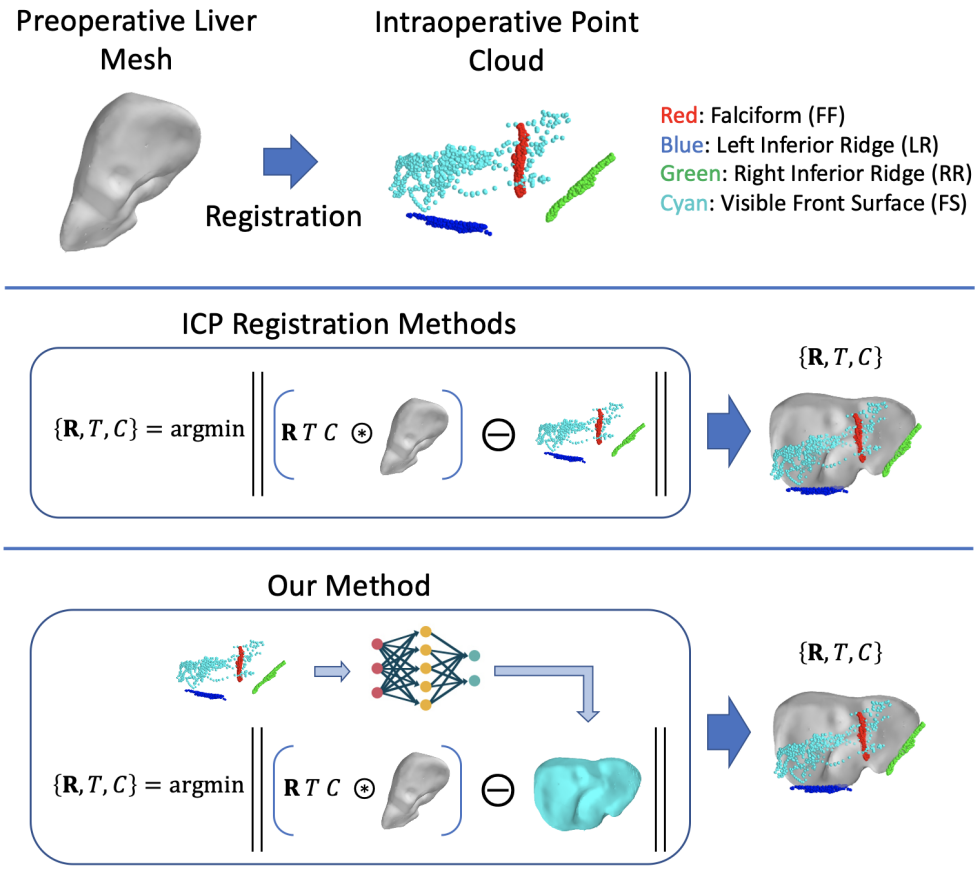


Figure 5.1: System-level diagrams of regular ICP and our method.

works like an *enhanced* RBF surface reconstruction, and achieves better robustness and rotation invariance. In this specific study, we train the network proposed in Chapter 3/4 with synthetic data generated from the liver registration dataset [21] and use it as an intermediate step in mesh-to-point-cloud registration.

Here, we address the issue of partial view / sparsely sampled points [21] and propose a novel occupancy-learning based non-rigid registration. We also propose a novel strategy for formulating / synthesizing more comprehensive training data to assist with our learning methodology. Our method is principled on the learning of an occupancy function representative of the shape of the organ, which has a number of advantages: 1) by optimizing over a learned occupancy function, the method improves registration accuracy and robustness; 2) with the

network computation deployed on GPUs, it is executed efficiently; 3) the framework supports an online workflow for swabbing the liver surface incrementally or interactively to refine the registration.

The structure of our method is illustrated in Figure 5.1. Unlike the regular Iterative Closest Point (ICP) method or its variants which optimize the geometric registration error measured over individual points, our method minimizes the misalignment between a preoperative acquired liver mesh and the occupancy function predicted from an input point cloud.

Our experiments show that our method reduces TRE by >10% from baseline methods [161, 22]. Considering the fact that the synthetic training data is generated based on baseline methods that are error-containing, this performance is even more impressive. The gain of performance is a result of using an occupancy network that efficiently removes noise (in a nonlinear and neighborhood-aware manner) and is thereby able to more effectively localize the deformed surface.

5.3 Related Works

5.3.1 Non-rigid Image-to-Physical Liver Registration

While some early works use rigid or affine transformations in the registration process, many researchers [161, 167] have suggested the limitations of the rigid body assumption in image-guided surgery and explored varied types of deformation models of the liver. Collins et al. [161] use 2D polynomial height functions to model the posterior side as a support surface and optimize transformation, rotation and deformation parameters jointly with the Levenberg–Marquardt (LM) algorithm. Deformations in laparoscopic surgery are usually significantly larger than deformations associated with open surgery, as insufflation (blowing air into the abdominal cavity to make space for laparoscopic surgery operations) will stretch

and compress the liver in unpredictable ways. Method [22] models laparoscopic deformations with boundary conditions applied to control points selected from the posterior support surface as well as falciform, left and right triangular ligaments.

While using independent boundary conditions and the linear elastic Finite Element Model (FEM) is convenient and efficient in non-rigid liver registration, it may cause considerable error when deformation is large. With intraoperative CBCT scans, [163] proposes a co-rotational formulation of elasticity accounting for large deformations. Oktay et al. [167] create meshes of organs and the abdominal wall and simulate insufflation to achieve a biomechanically-realistic deformation model. In this study, a linear elastic based deformation model like [22] is used.

Clinical validation of such registration methods is challenging due to the difficulty in obtaining data representative of the true state of organ deformation. Phantoms [22, 166] and animals [162, 166] are used for validating registration methods. [166] develops a human-to-phantom validation system and uses a resampling strategy to improve registration accuracy. The dataset we used in this work is also created with a phantom validation [21] thus affords a built in measure of ground truth.

5.4 Method

5.4.1 Algorithm Overview

The proposed non-rigid registration is a process that iteratively minimizes the distance of a live acquired mesh to the iso-surface of predicted volumetric occupancy function.

Given a point cloud $P \in \mathbb{R}^{N\times 3}$ as input, the network produces a volumetric function in 3D space, $g(\cdot \mid P) : \mathbb{R}^3 \to \mathbb{R}$, representing the shape from which the point cloud P is sampled. As the output layer of this network is a softmax operation, $g(q \mid P) \in [0, 1]$ represents how likely a query point q locates inside the expected volume. In Figure 5.2, cross-sections of

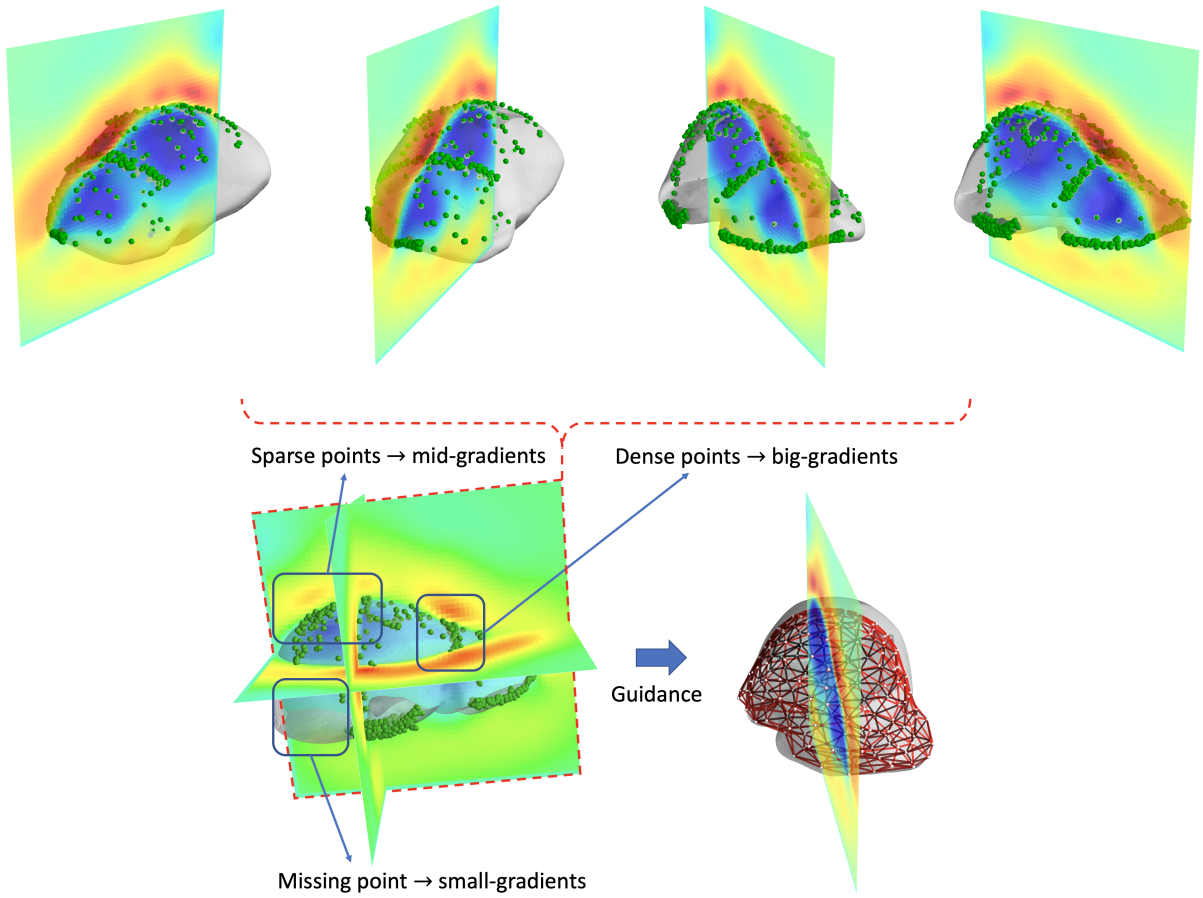


Figure 5.2: Illustrations of our occupancy function-guided mesh (red wired frame) to point cloud (green points) registration.

the predicted occupancy function are plotted, overlaid with the input point cloud and an iso-surface can be generated at $g(\cdot \mid P) = 0.5$. A Jet color-map is used to visualize it, with red indicating outside the expected volume and blue indicating inside. As it is a volumetric, completed and de-noised reconstruction of the liver, it is advantageous to drive registration with respect to the predicted occupancy function instead of with the original sample points. We optimize the following objective:

$$F = \frac{1}{N} \sum_{i=1}^{N} \left[g\left(f_{T,R,C}\left(v_i \right) \mid P \right) - 0.5 \right]^2 + \alpha E \tag{5.1}$$

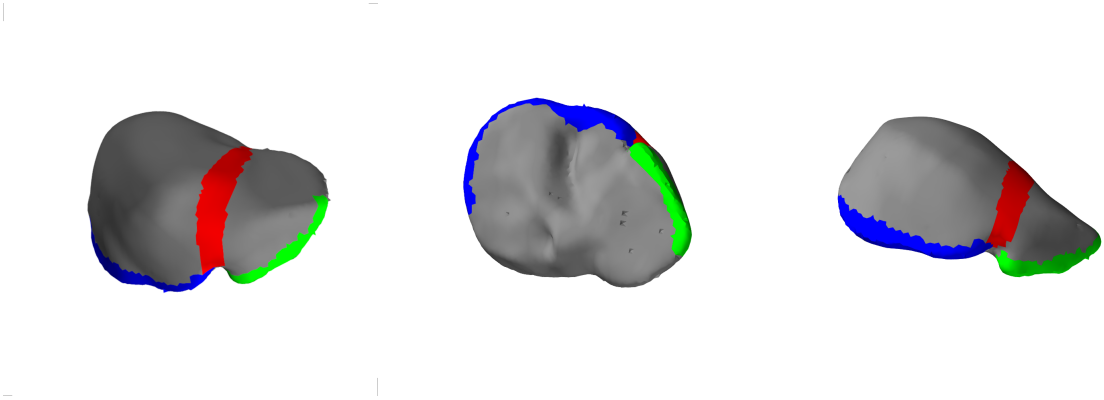
where v_i is the 3D Cartesian coordinates of the ith surface vertex on the preoperative liver mesh, and $P : \{p_1, p_2, \dots p_N\}$ represents the point cloud sampled in the real operating room. In this equation $f_{T,R,C}(q_i)$ is the registration operation and is composed of transformation $T \in \mathbb{R}^3$, rotation $R \in SO(3)$, and non-rigid deformation parameterized with the deformation coefficient C (defined later in section 5.4.2). When the first term of (5.1) decreases, the mesh gradually approaches the 0.5 iso-surface. Thanks to the continuous and smooth RBFs used in the occupancy network, the produced occupancy function is differentiable everywhere, so that we can easily calculate the Jacobian matrix of the residual (w.r.t. $\{T, R, C\}$) and can solve the optimization problem (5.1) with the Levenberg–Marquardt algorithm. E is the total strain energy calculated as $E = d^T K d$ in which K is the stiffness matrix of the tetrahedron liver mesh and d indicates the displacement of all vertices.

What is most interesting about this registration method is the probabilistic nature of the learned occupancy function. As shown in Figure 5.2, the network is more confident about the occupancy state in the area where it is more densely sampled as compared to where sample points are sparse or missing. The function value is polarized in a well-sampled area, and the magnitude of gradient around this area might be several orders greater than the area not sampled. This acts as an automatic weighting of the areas that are trusted, as well as offering a trade off between 1) matching shape and 2) minimizing deformation energy.

5.4.2 Linear Elastic Liver Deformation Model

The non-rigid liver registration and the strain energy term in Equation 5.1 require a physical deformation model for the liver. We build a linear elastic liver deformation model with a similar method used in [22].

First of all, the geometric liver model is taken from the published Image-to-Physical Liver Registration Sparse Data Challenge [21]. The volumetric liver is represented as a fine



(a) front (b) back (c) bottom

Figure 5.3: Our labeled liver mesh.

tetrahedral mesh. We extract a triangular surface mesh from the volumetric mesh and label the surface areas. Because the sample points in dataset [21] are collected from 1) front surface (FS), 2) falciform ligament area (FF), 3) left inferior ridge (LR), and 4) right inferior ridge (RR), we label the extracted liver surface accordingly. The labeled surface mesh is shown in Figure 5.3, with FS, FF, LR, and RR in gray, red, green, and blue respectively.

Heiselman et al. [22] model liver deformations with boundary conditions applied to the posterior support surface, falciform, left and right triangular ligaments. That's because their method is designed specifically for laparoscopic liver surgeries in which force is intensively applied through ligaments to the liver. Instead, the non-rigid deformation in the phantom-based dataset [21] is created by modifying the posterior support conditions, i.e., simulating the open-abdominal liver surgery in which the ligaments are cut off. Because the preoperative CT scan and open-abdominal liver surgery are mostly conducted in the same position, the basic linear model is sufficient to achieve a good approximation for such small deformations. In the static equilibrium state, the model can be described by Navier-Cauchy constitutive equation:

$$\frac{E}{2\left(1+\upsilon\right)\left(1-2\upsilon\right)}\nabla\left(\nabla\cdot\mathbf{u}\right)+\frac{E}{2\left(1+\upsilon\right)}\nabla^{2}\mathbf{u}+\mathbf{F}=0, \tag{5.2}$$

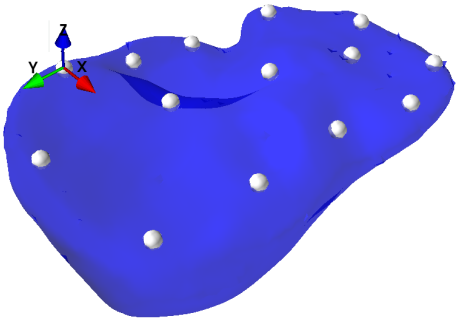


Figure 5.4: Control points.

where $\mathbf{u}$ is the displacement, $\mathbf{F}$ is the applied force, E is Young's modulus, and υ is Poisson's ratio. Using FEM with the tetrahedron mesh, this equation can be solved with a linear system of nodal displacements and force:

$$K\mathbf{u} = \mathbf{F}, \tag{5.3}$$

where K is the global stiffness matrix [6].

To produce the deformation modes, we evenly select 14 points on the posterior side of the liver mesh as control points (shown in Figure 5.4). Unit offsets in x, y, and z directions (where the surface normal is pre-computed so that the z axis is perpendicular to the posterior surface) are applied individually to each control point to form deformation boundary conditions. The goal is to build independent deformation modes that have one control point displaced 1mm off its original position in x, y, or z directions and all other control points kept at the same position. The global displacement field is a linear combination of these 42 modes:

$$\boldsymbol{d} = \sum_{i=1}^{42} c_i \mathbf{m}_i = C\mathbf{M}, \tag{5.4}$$

[6] In our implementation, the finite element matrix assembly is done using PyMesh `https://github.com/PyMesh/PyMesh`. For example, the global stiffness matrix of a volume mesh `vmesh` can be assembled by the method `pymesh.Assembler(vmesh).assemble("stiffness")`.

Figure 5.5: Our liver deformation modes. The original liver is draw in blue and the deformed in red. Three columns from left to right: offsets in z, x, and y directions.

where $\mathbf{M}$ is a $3N \times 42$ matrix in which the precomputed deformation modes are stored, and C represents the deformation coefficients.

For each $\mathbf{m}_i$, we solve Equation 5.3 by minimizing the total elastic energy, subject to the boundary conditions imposed to the control points. In addition, we also required that no force is applied to the front surface, i.e., we model the deformation by stretching and pressing the posterior surface that supports the liver. Formally, this optimization problem is:

$$\mathbf{m}_i = \underset{\mathbf{u}}{\operatorname{argmin}} \ \mathbf{u}^T K \mathbf{u}, \tag{5.5}$$

subject to

$$\mathbf{u}_{cp} = \mathbf{u}_{bc}^{(i)} \text{and} K_{fs}\mathbf{u} = \mathbf{0}, \tag{5.6}$$

where $\mathbf{u}_{bc}^{(i)}$ represents the boundary condition and K_{fs} is the sub-matrix block that is sliced-out from K corresponding to the front surface nodes. During implementation and experiments, we use quadratic energy minimization API of Libigl (`https://libigl.github.io`) to solve this optimization problem. Some of the produced deformation modes are shown in Figure 5.5 (note that the scale of deformation is enlarged for better viewing).

5.4.3 Training the Liver Occupancy Network

Training an occupancy network requires a complex structure of shape data. Each instance of training example consists of three components: 1) input point cloud P, 2) a set of query points randomly distributed in space around P, and 3) inside-outside labels associated with each query point. To generate appropriate training data, we implement a novel *crossover* method.

Each point cloud in dataset [21] contains four subsets sampled on falciform ligament area (FF), left inferior ridge (LR), right inferior ridge (RR), and front surface (FS) respectively (refer to [168] for comprehensive liver anatomy). We need to randomly take four different point clouds from the dataset and draw one subset from each point cloud to assemble a new point cloud (illustrated in Figure 5.6). To make this possible, the real data is first registered to a liver mesh (provided with the dataset), and the closest points on the mesh are recorded representing where these points are originally sampled. For each synthesized point cloud, we use a random vector as deformation coefficients and apply the displacement computed with (5.4) to the mesh. The closest points on the mesh are deformed accordingly. In addition, Gaussian noise is applied to simulate sampling error before assembly. Query points are then randomly sampled from an enlarged bounding-box and their occupancy is determined

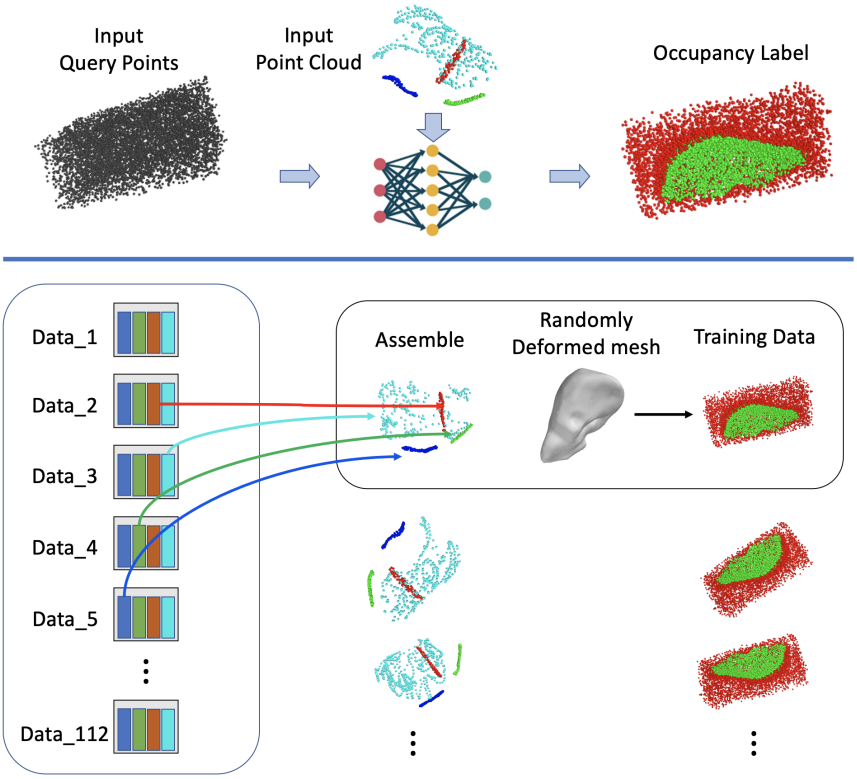


Figure 5.6: The crossover method can synthesize unique and realistic data to train our occupancy network.

according to the deformed mesh.

There are other changes that have to be made to the network. First, the sub-surface labels (FF, LR, RR and FS) are fed into the network together with input points. We treat the label as an additional feature channel associate with each point (so it becomes 4-dimensional). As generally mentioned in section 3.3, our DFS layers allow integration of potentially any other channels of information (like RGB or surface normal) by simply operating it as per-point feature. Second, each point cloud of a given sparse registration dataset [21] (as well as the generated training data) has a slightly different number of sample points, but the network has a fixed-size input layer. To solve this problem, we use a network that takes 1500 input points, which is greater than most of the point clouds. In both training and evaluation, if a point cloud has less than 1500 points, placeholder points (placed far away and labeled with a

digit that's different from FF, LR, RR and FS) are concatenated to it. We randomly discard some points if there are more than 1500 points in total. With this solution, it is convenient to dynamically add extra sample points, suitable for swabbing liver surface incrementally and interactively in the operating room.

5.4.4 Implicit Shape Registration Algorithm

Like many existing methods, our registration has two stages: rigid alignment and non-rigid correction. The first stage ensures a good initial state for non-rigid registration. In our method, the first stage of rigid registration is necessary and crucial. The predicted occupancy is only valid in the space around P (cf. Figure 5.2), as the RBFs and learned shape features in the network have restricted size. If a query point q locates too far away from the input shape, its occupancy and gradient are not reliable. Our registration algorithm is described below:

Input: Position of vertices v of the source mesh $\mathcal{M}$; Target point cloud P;
Output: Parameters R, T, C and registered vertices $v^{(t)}$.
Initialize $\{R, T\}$ by rigid registration $P \rightarrow \hat{P}$ to fit $\mathcal{M}$;
Initialize $C \leftarrow \mathbf{0}$, $t \leftarrow 0$;
repeat
  Updating vertices $v^{(t)} \leftarrow f_{T,R,C}(v)$;
  Computing occupancy function with the network $g\left(v^{(t)}|\hat{P}\right)$;
  Computing residuals $r_i : \left[g\left(v_i^{(t)}|\hat{P}\right) - 0.5\right], i = 1, 2, \ldots, N$;
  Computing strain energy $E = d^T K d$;
  Computing Jacobian $J = \frac{\partial[r_1,\ldots,r_N,\sqrt{\alpha}E]}{\partial[R,T,C]}$;
  Updating registration parameters (LM update):
  $[R, T, C] -= \left(J^T J + \lambda \text{diag}\left(J^T J\right)\right)^{-1} J^T \mathbf{r}$, where $\mathbf{r} : [r_1, \ldots, r_N, \sqrt{\alpha}E]$;
  $t \leftarrow t + 1$
until $\|\Delta r\| < \varepsilon$ *OR* $t = t_{\max}$;

Algorithm 1: Occupancy-guided registration algorithm

5.5 Experiments and Results

The novelty and key point of our method is training an occupancy network in order to more robustly guide registration when only sparse input data is available from a live scan. To test the robustness of our method, we synthesize training data with two baseline methods: ICP rigid registration and non-rigid registration [22]. Each training set contains 1200 deformed examples and the shared validation set has 112 instances generated from the original dataset without crossover. Gaussian noise with a standard deviation of 2mm is applied to every point.

We test our method with networks that are trained with these two training sets. Their TREs are measured and shown in Table 5.1. The one using rigid registration baseline to synthesize training data is named *Our1*. These results show that our method can improve the registration accuracy with reduced sensitivity to how training data is synthesized. The boost of average TRE from baseline-rigid to **Our1** (21.5%) is greater than from baseline-nonrigid to **Our2** (11.8%).

		Baselines			
	Surface Coverage	Rigid	Nonrigid	Our1	Our2
Average TRE (mm)	20%-28%	5.72	6.00	**4.30**	4.42
	28%-36%	5.23	**4.70**	4.73	4.74
	36%-44%	5.70	3.89	3.90	**3.35**
	mean	5.53	4.85	4.34	**4.28**
Median TRE (mm)	20%-28%	5.17	5.90	**3.92**	4.04
	28%-36%	5.01	4.91	**3.65**	3.91
	36%-44%	5.57	3.77	3.44	**3.33**
	mean	5.19	4.76	**3.70**	3.75

Table 5.1: Target registration results for Sparse Data Challenge

There is a noticeable issue in this experiment in that our method performs slightly worse in the data of middle surface coverage (28%-36%) than more or *even* less coverage. We are not

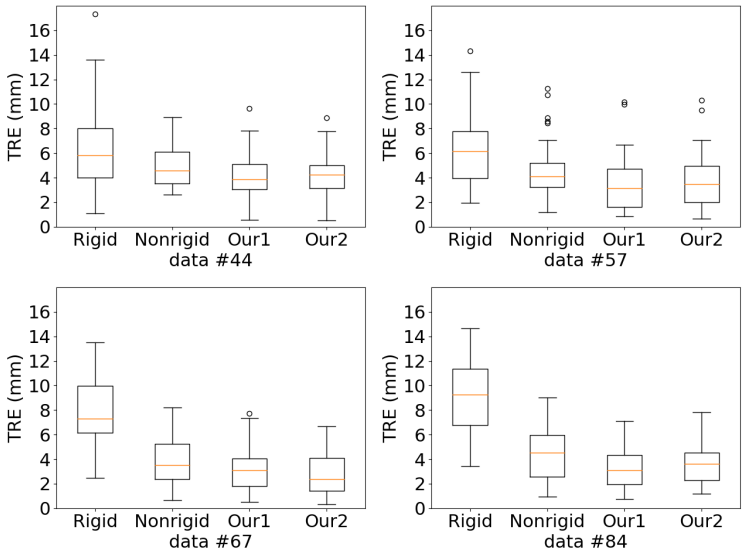


Figure 5.7: TRE statistical box plots of four registration cases.

sure what exactly causes this phenomenon, but it probably implies that the implementation and hyper-parameters of our method could be further optimized.

In the Sparse Data Challenge [21], the positioning of targets in four cases (#44, #57, #67 and #84) are available to participants for validation. Their error statics are plotted in Figure 5.7. We show registration results of the four methods for data#44 in Figure 5.8, for visual inspection. Our method out-performs the baseline methods constantly, with an acceptable variation of error and predictable behavior.

5.6 Summary

In this chapter, we introduce a novel methodology that improves non-rigid image-to-physical registration with deep learning point cloud reconstruction. To better understand the message behind this improvement, it's helpful to recognize two different roles the occupancy network plays. First, in the area where sample points are densely represented, the network works like a noise-removal module, producing a shape volumetric occupancy function that has a smooth

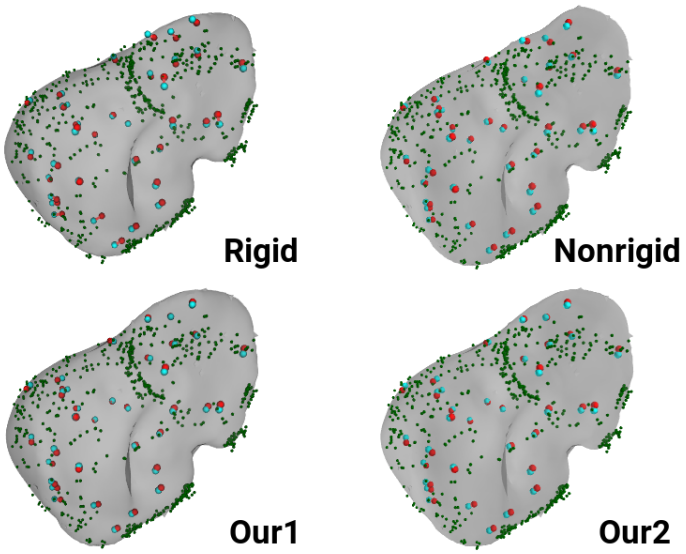


Figure 5.8: Registration results of the 44th point data (corresponding to the first box plot in Figure 5.7). The registered mesh is overlaid with sampled points (dark green dots), registered targets (red points), and ground-truth targets (cyan points).

iso-surface. Second, in the area where sample points are missing or sparse, the network is inferring the shape and deformation based on the global information carried within the point cloud based on the prior knowledge learned from data. Importantly, this occupancy function is probabilistic, so that there is no need to weight these two roles by hand. The magnitude of occupancy function and its gradient is self-modulated during training.

This method is a successful attempt and promising direction that generalizes and utilizes the rich outcome of recent 3D shape deep learning studies, in order to render more tractable solutions for reconstruction and registration of non-rigid bodies undergoing deformation. To this end, solutions such as the proposed, which leverage learning-based reconstruction, have significant practical importance and benefit in applications such as image-guided surgery, and more generally, in applications in which only very sparse data is available during acquisition, from which accurate reconstruction is then desired.

Chapter 6

Learning Tumor-Induced Deformations to Improve Tumor-Bearing Brain MR Segmentation

6.1 Overview

In this chapter, we apply SCFL method to develop a novel framework that extends atlas-based whole-brain segmentation methods to tumor-bearing MR images. Given a patient brain MR image where the tumor is initially segmented, we use a point-cloud deep learning method to predict a displacement field, which is meant to be the deformation (inverse) caused by the growth (mass-effect and cell-infiltration) of the tumor. This is then used to warp and modify the brain atlas to represent the change so that existing atlas-based healthy-brain segmentation methods could be applied to these pathological images. To show the practicality of our method, we implement a pipeline with nnU-Net [169] MRI tumor initial segmentation

and SAMSEG [24], an atlas-based whole-brain segmentation method. To train and validate the deformation network, we synthesize pathological images by simulating artificial tumors in healthy images with TumorSim [24]. This method is evaluated with both real and synthesized data. These experiments show that segmentation accuracy can be improved by learning tumor-induced deformation before applying standard full brain segmentation.

6.2 Introduction and Related Works

Brain tumors are among the most fatal cancers in the human body because a large portion of them are malignant or borderline-malignant like glioblastomas and astrocytomas [170], and even benign tumors like gliomas and meningiomas may damage vital brain functions by compressing or eroding the surrounding tissue. Magnetic Resonance Imaging (MRI) is the most widely used neural imaging technique in brain cancer diagnosis and treatment. Accurate and robust segmentation of the tumor body, peritumoral edema, brain tissues affected by the tumor [171], and organ-at-risk [172] is essential for diagnosis, survival prediction, treatment planning, and many other medical practices.

Semantic whole brain MRI segmentation refers to the task of segmenting the image into regions according to tissue types (White Matter (WM), Grey Matter (GM), and Cerebrospinal Fluid(CSF)) or anatomical structures (cortex, hippocampus, thalamus, lentiform nucleus). For fine anatomic segmentation, atlas-based generative approaches provide the most reliable and robust results [173]. This strategy relies on aligning a probabilistic atlas to the input images to provide a segmentation prior. However, directly using atlas-based segmentation methods in tumor-bearing patient images is problematic because the tumor's mass-effect may break the spatial information of the brain anatomy represented in the population-based atlas. There are two common solutions to this problem. First, in some atlas-based approaches, brain (including the tumor) segmentation and registration problems are solved jointly [174,

175, 172]. Scheufele et al. [175] introduce a joint image registration and biophysical inversion framework. Agn et al. [172] combine a generative model for whole-brain segmentation with a spatial regularization model of tumor shape, and *simultaneously* segment the tumor and organs-at-risk. However, given an accurate tumor segmentation, finding a solution with the reaction-diffusion model that grows a tumor in the atlas is still highly complex and challenging because of its inherent non-linearity, ill-posedness, and ill-conditioned nature [176]. A second possible solution is warping the input image (toward a non-pathological state) with the estimated displacement to eliminate the tumor-induced deformation [177]; or warping (growing a tumor in) the brain atlas to reflect the tumor and spatial changes in the patient image [178, 179, 180, 181]. The latter is sometimes referred to as "Seeded-Atlas" [181]. However, these approaches require a good tumor segmentation or even a well-positioned seed manually placed by experts to initialize the tumor growth. Our method follows the same principle with the hope of being more robust to those problems by learning from the synthesized deformation fields. Cuadra et al. [178] interpolate the tumor displacement with a radius velocity inside the tumor mask, and a Demons [182] non-rigid registration outside. Bauer et al. [179] proposed a Markov random field (MRF) based tumor growth model to simulate a tumor on the atlas that approximates the observed tumor on the patient image. Many other methods [180, 181] use a finite element method (FEM) to solve the deformation's mechanical properties.

Recent years have witnessed an influx of discriminative neural-network approaches (mostly based on Fully-Convolutional Networks) for brain tumor segmentation [183, 184, 185, 186], and they show remarkable performance in the BraTS [187] challenge [7]. However, joint generative model-based whole-brain segmentation couldn't benefit from these, as they require a generative tumor model based on Restricted Boltzmann Machines (RBMs) and VAEs

[7] a challenge that attempts to automatically segment brain tumors where the intracranial gliomas are significant and have infiltration into other brain tissues.

as employed in [171, 172], to represent the tumor shape distribution. Meanwhile, given an accurate tumor segmentation, finding a solution with a reaction–diffusion model that grows a tumor in the atlas is still extremely complex because of its inherent non-linearity, ill-posedeness, and ill-conditioned nature [176]. This problem is studied under the terminology *patient-specific biophysical inversion* [188, 189] and *tumor-growth calibration* [176, 189].

Due to the difficulties above, we propose to train a regression model that *directly* predicts the tumor-induced deformation so that the probabilistic atlas can be warped and modified to represent the changed geometric structure. Our method is inspired by recent CNN-based volumetric brain image-to-image registration methods [23, 190]. The proposed method is similar to [177], in terms of trying to find out the inversion of the tumor growth. Like VoxelMorph methods [190], our network is a regression model that directly produces a vector field. The point cloud network method proposed in [3, 151] effectively maps a sparse point set to a continuous function in 3D space. As demonstrated in the previous chapter, the network was effectively adapted to learn the occupancy function of deformed livers in open-abdominal liver surgery. In this study, we use a similar point cloud network to learn tumor-induced deformations to aid in whole-brain segmentation. This novel solution robustly combines a discriminative tumor segmentation with an atlas-based brain semantic segmentation. To train our network, a novel training data synthesis method is implemented based on TumorSim [24]. In addition, we introduce new evaluation methods to validate the proposed method.

6.3 Method

Figure 6.1 shows a graphical overview of our method. Given an initial tumor segmentation, we sample a number of points from the boundary (surface) of the tumor segment and feed that into a point-cloud network [151] that predicts tumor growth. It produces a 3D vector field that models the brain's deformation caused by the growth of the tumor (Section 6.3.1). We

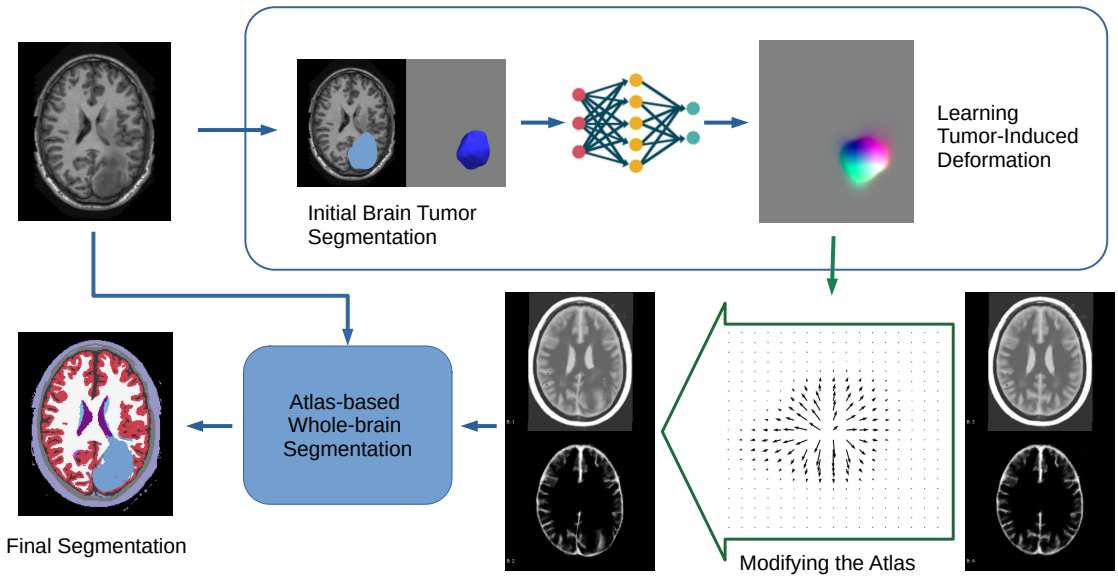


Figure 6.1: A graphical abstract of the proposed segmentation method.

use TumorSim [24], an open-source package that generates pathological *pseudo*-ground-truth from healthy brain templates, to generate displacement data for training (Section 6.3.2). Once trained, the network produces the tumor-induced mass-effect deformation as a spatial transformation $\phi : \mathbb{R}^3 \rightarrow \mathbb{R}^3$. As indicated in Figure 6.1, the population-based probabilistic atlas is modified by warping with the predicted transformation to reflect the geometric change in the patient images. As explained in section 6.2, normal atlas-based segmentation utilizes labels and associated priors to achieve the segmentation, however, those atlases typically don't have labels for tumors because they are based on healthy brains. In this case, the tumor segmentation information that wouldn't been initially presented in the atlas is then overlaid onto it by appending a "tumorous" label and corresponding probability (Section 6.3.3). Finally, the patient image is segmented with a generative whole-brain segmentation method using the modified atlas. This pipeline uses a pre-trained nnU-Net model [169] for initial tumor segmentation and SamSeg [24] for the final brain segmentation.

6.3.1 Deformation Network

In most glioma models [24, 191, 192], the tumor behavior consists of two components: proliferation and infiltration. We opted to learn a combined overall deformation with the network because the purpose is not an accurate tumor characterization, but a transformation that is good enough to improve atlas-based segmentation. Meanwhile, as indicated by [24], there is a correlation between cancer cell infiltration and edema (tissue swelling due to trapped fluid). So we take the tumor core and edema from the initial tumor label and transform that to a point cloud $P : (P_{TC}, P_{ED})$ consisting of two groups of points: P_{TC} representing tumor core and P_{ED} representing edema. A similar network and the same form of input as [193] is used, i.e., we associate a two-bit one-hot feature indicating TC/ED for each point.

Let ϕ denote the predicted deformation: we implement two different setups for learning ϕ with a point cloud network $\text{def}\,(\cdot|P)$: 1) that directly regresses on the displacement field (DISP), $\phi_{DISP} : q \mapsto q + \tau\text{def}\,(q|P)$ and 2) we represent diffeomorphism with a Stationary Velocity Field (SVF), $\phi^{(1)}_{SVF} : \boldsymbol{q} \mapsto \exp\left(\tau\text{def}\,(\boldsymbol{q}|P)\right)$, where the SVF is integrated over $[0, 1]$ to obtain the transformation (exponential map of SVF approximated in a certain resolution). When warping atlases, the scalar τ that adjusts the magnitude of transform can be determined either empirically by trials (as a constant) or interactively by a human expert (as a per-patient parameter), providing an easy intervention that can be used to compensate for the incorrect overall size of the initial tumor segment and the scale of predicted deformation, thus optimizing the whole-brain segmentation.

6.3.2 Synthesizing Tumor Images for Training

We use TumorSim [24] to grow artificial tumors in healthy brain models of BrainWeb [194]. The input brain structure is modelled with a labeled tetrahedral volumetric mesh, where tissue mechanical properties (for biophysical deformation) are associated with each mesh

element, and the WM tractography (for cancer cell infiltration) is captured with a diffusion tensor image (DTI). Given an initial tumor seed and the required tumor-growth parameters, the pathological brain is produced by solving the reaction-diffusion system. The initial seeds are sampled from ground-truth tumor masks in the BraTS dataset [187] with randomization and augmentation: 1) seed marks are flipped from one hemisphere to another to double the number of instances; 2) random erosion and dilation are applied to approximate the longitudinal earlier and later stages of the tumor. For each simulation, we draw random tumor-growth parameters: initial mass-effect pressure and tensor infiltration multipliers of GM and WM. In TumorSim, the initial tumor surface pressure direction is randomized by Von-Mises-Fisher distribution to further increase the output variability. [8]

TumorSim produces as output a ground truth tumor segment $T : (T_{TC}, T_{ED})$ and dense deformation fields of mass-effect and infiltration, which are then combined to form the overall deformation $D \in \mathbb{R}^{\mathrm{nx}\times\mathrm{ny}\times\mathrm{nz}\times 3}$. We first turn this ground-truth tumor and edema volume to a triangle mesh using Marching Cubes [66]. The training point clouds P_{TC} and P_{ED} are uniformly sampled from the mesh surfaces ∂T_{TC} and ∂T_{ED}, respectively. Then, a set of query points $Q : \{q_1, q_2, \cdots, q_J\}$ are randomly drawn from the inside and peripheral tumor space. The deformation vectors $D(q_j)$ of those query points are interpolated from D. In the SVF diffeomorphism learning, the displacement field D is first transformed [9] to SVF before interpolation.

Let the tuple $\left(P_i, Q_i, \left\{D_i(q_j^{(i)}) | j = [1, \ldots, J]\right\}\right)$ indicate the i-th instance of our training data, the batch loss used to train the network is:

$$\mathcal{L}_B(\theta) = \frac{1}{B}\sum_{i=1}^{B}\sum_{j=1}^{J}\mathcal{L}\left(\mathrm{def}_\theta\left(q_j^{(i)}|P_i\right), D_i\left(q_j^{(i)}\right)\right), \tag{6.1}$$

[8] As pointed by [176, 195], tumor location carries important information for low and high-grade gliomas and high-risk tumor sites can be statistically identified. So the seed can not be purely random.

[9] We use Scaling-and-Squaring implemented in MIRTK (https://mirtk.github.io) to transform between DISP and SVF, specifically, `calculate-logarithmic-map` and `calculate-exponential-map`.

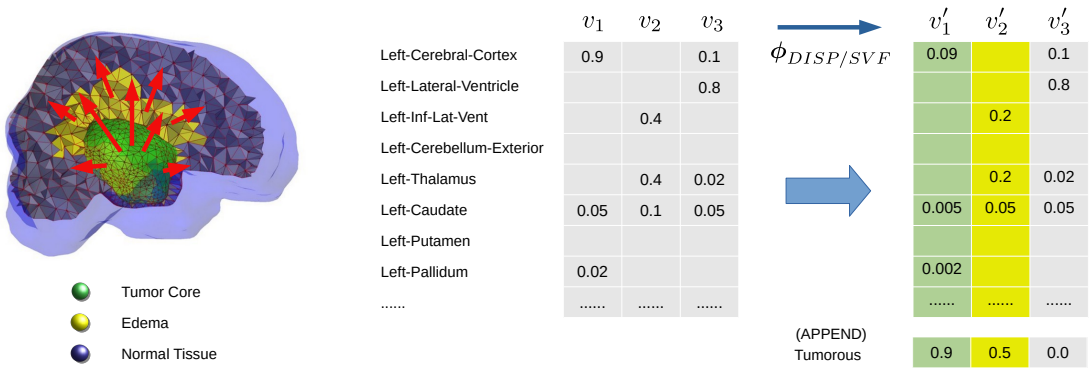


Figure 6.2: Modifying the probabilistic atlas.

where θ indicates the network's trainable parameters and B is batch size. For simplicity and robustness, we use mean absolute error as the per-point loss $\mathcal{L}$ and train the network with the Adam [196] optimizer.

6.3.3 Modifying the Atlas

After the deformation network is trained, it infers the tumor-induced deformation from a tumor segment. The initial tumor segment and the inferred deformation provide strong clues to help whole-brain segmentation. Like seeded-atlas approaches [179, 180, 174], we incorporate the tumor shape information into the population-based healthy atlas to reflect the structural changes of a specific patient. The probabilistic atlas used in SAMSEG [197, 172] is a tetrahedron mesh where each vertex v_i is associated with a vector $\boldsymbol{\alpha}_i$ that represents the prior of each anatomical label. As shown in Figure 6.2, the atlas is modified in two aspects: geometric and probabilistic. First, the atlas A is deformed (by interpolation) so that $A \circ \phi_{DISP/SVF}$ appears the same as if the vertices are transformed by the predicted deformation $\phi_{DISP/SVF} : v_i \mapsto v'_i$ (directly transforming the vertices' coordinates results in bad triangles and thus breaks the mesh's local deformation property). Secondly, we append the tumor label to the prior's vector. In the case that transformed vertices v'_i fall in the tumor segment T_{TC} or T_{ED} (indicated with green and yellow in Figure 6.2), a "tumorous"

probability is appended, and the original label priors are equally rescaled to keep their sum as 1, i.e., $\boldsymbol{\alpha}_i = [(1-\alpha_T)\,\boldsymbol{\alpha}'_i, \alpha_T]^{\mathrm{T}}$. The hyper-parameters α_{TC} and α_{ED} could be determined based on how we trust the initial tumor segmentation. In our implementation, they are tentatively set as 0.9 and 0.5 so that edema segments can be further refined.

6.3.4 Overall Algorithm

Overall, the algorithm of processing one tumorous image with a pretrained deformation network is abstractly described as follows:

Input: A population-based probabilistic atlas $\mathcal{A}$; Input multi-model patient scan $\mathcal{M}$, Scale factor τ.

1 Find ϕ_{Affine}, the affine transform from $\mathcal{M}$ to $\mathcal{A}$ (Talairach space [10]);
2 Run nnU-Net on $\mathcal{M}$ to produce initial tumor segments T_{TC}, T_{ED};
3 Perform marching cubes to turn the volumetric segments T_{TC}, T_{ED} into surface meshes;
4 Sample a point cloud $P : (P_{TC}, P_{ED})$ from ∂T_{TC} and ∂T_{ED}, and apply affine transformation $P \leftarrow \phi_{\text{Affine}}(P)$;
5 Produce the dense deformation by evaluation def $(\cdot|P)$ at the 3D grid in the Talairach space $\boldsymbol{q} \leftarrow \langle\mathcal{A}\rangle_{\text{range}} * \text{meshgrid}\,(0:nx, 0:ny, 0:nz)$
if *Is DISP* **then**
$\phi \leftarrow \tau\text{def}\,(\boldsymbol{q}|P)$
else
$\phi \leftarrow \exp\,(\tau\text{def}\,(\boldsymbol{q}|P))$
endif;
6 Modify the atlas $\mathcal{A}$ with T_{TC}, T_{ED}, and $\phi_{DISP/SVF}$
$\mathcal{A}' \leftarrow [\phi \circ \mathcal{A},\ \ \mathcal{A}_{T_{TC},T_{ED}}]$;
7 Run SAMSEG with $\mathcal{M}$, $\mathcal{A}'$ and ϕ_{Affine}.

Output: $\mathcal{S}$,the semantic label of each voxel of $\mathcal{M}$ preduced by SAMSEG.

Algorithm 2: The detailed segmentation pipeline

6.4 Experiments

To train the deformation network, we generate 1000 simulated deformation examples, and split them into training, validation and testing sets. For each instance of tumor-deformation pair, we create DISP and SVF version of points for training the two different versions of network.

Validation of medical image segmentation with comparable metrics is a difficult task because of the lack of reliable labeled data [24, 187]. In this study, though, it's imaginable that even if experts are recruited to produce such ground truth, their manual segments might show significant variations and disagree for tumor-bearing scans. We solve this problem with novel evaluation methods. First, we validate the network by showing that the degree of brain inter-hemisphere symmetry is improved when transforming the tumor-bearing images with $\phi^{-1}_{DISP/SVF}$, the inverse of the predicted tumor-growth (Section 6.4.1). Second, we synthesize pathological ground-truth by fusing well-labeled healthy brain images (MindBoggle-101 [198]) and tumor scans (BraTS [187]), and conducting quantitative evaluations against these (Section 6.4.2). In addition, a regular segmentation test is performed with real tumor images for subjective evaluation (Section 6.4.3).

6.4.1 Inter-Hemisphere Symmetry Validation

In this proof-of-concept experiment, we naively assume that a healthy brain's left and right cerebral hemispheres are roughly symmetric, and the growth of a tumor tends to break the symmetry. Therefore the degree-of-symmetry of a pathological brain can be increased by finding the inverse of the tumor-induced deformation. We randomly take 47 multi-model scans $\{\mathcal{M}_i\}_{i=1,2,\cdots,47}$ from the BraTS2020 training set and produce their deformation ϕ_i. Their segmentations $\mathcal{S}_i$ are produced by SAMSEG using an atlas that has no specific tumor

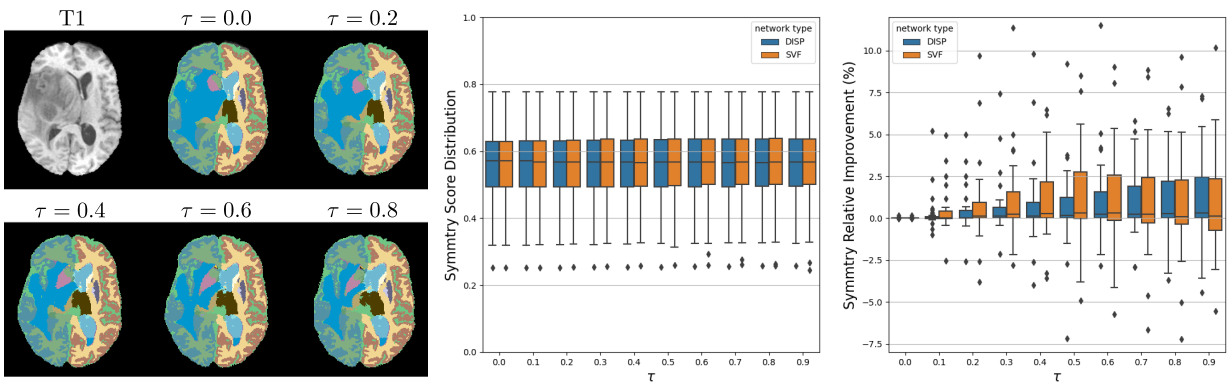


Figure 6.3: The measurement of left-right symmetry when using different τ

information, i.e., uniform tumor prior over the brain area [11]. A mirrored segmentation $\mathcal{S}_i^*$ is created by flipping $\mathcal{S}_i$ and swapping the left-right labels. We use the weighted mean Dice score [199] of each segment as the symmetry measurement: Dice $(\mathcal{S}_i^*, \mathcal{S}_i)$. Similarly, we compute Dice $\left(\left(\phi_i^{-1} \circ \mathcal{S}_i\right)^*, \phi_i^{-1} \circ \mathcal{S}_i\right)$ as the symmetry score for when tumor-induced deformation is eliminated by inverse transformation. We run tests with different scale factors $0 \leq \tau < 1$. The score's distribution and its relative improvement are shown in Figure 6.3. Both DISP or SVF deformations have a clear positive effect on the symmetry comparing with the original brain structure ($\tau = 0$). Meanwhile, we notice that the original brain symmetry score has a large deviation. It is worth mentioning that some tissues are not precisely symmetric, and the segmentation is not 100% accurate. The improvement of symmetry shown in Figure 6.3 is a sign that some useful information about the brain geometric change is captured by the deformation network.

6.4.2 Synthesized Images Evaluation

To quantitatively evaluate the proposed method, we synthesize pathological images associated with ground-truth anatomical labels. In Mindboggle-101 [198], OASIS-TRT-20 is a subset of

[11] This atlas is generated from the SAMSEG atlas comes with FreeSurfer (https://surfer.nmr.mgh.harvard.edu). We remove the non-brain labels like skull and optic-chiasm, and re-distribute the label priors because BraTS data are skull-stripped.

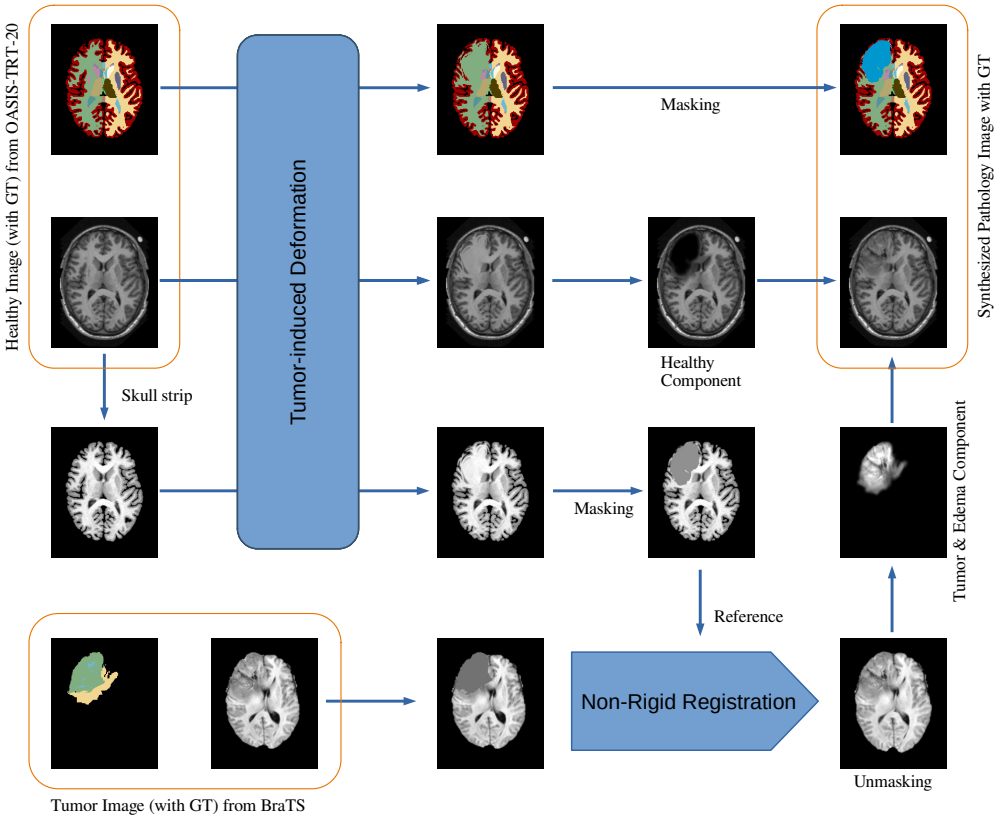


Figure 6.4: An illustration of the proposed method for synthesizing pathological brain images and ground-truth labels.

20 health subjects whose subcortical volumes are manually labeled. We use these images to synthesize pathological images by deforming and smoothly fusing tumor image patches (from BraTS) into them. The high-quality labels are deformed accordingly as the ground-truth. The idea of this data synthesis procedure is sketched in the Figure 6.4. The output image is composed by fitting and inserting real tumor image patches (retrieved from BraTS dataset that best matches the simulated tumor) into the deformed healthy brain images. For this purpose, the patient image is aligned to the deformed image space. Before registration, both the moving and reference images are masked to eliminate the different appearances in the tumor area. The registered tumor patch is extracted and placed (with soft boundary) onto the deformed image where the artificial tumor should be. Some results of this image synthesis

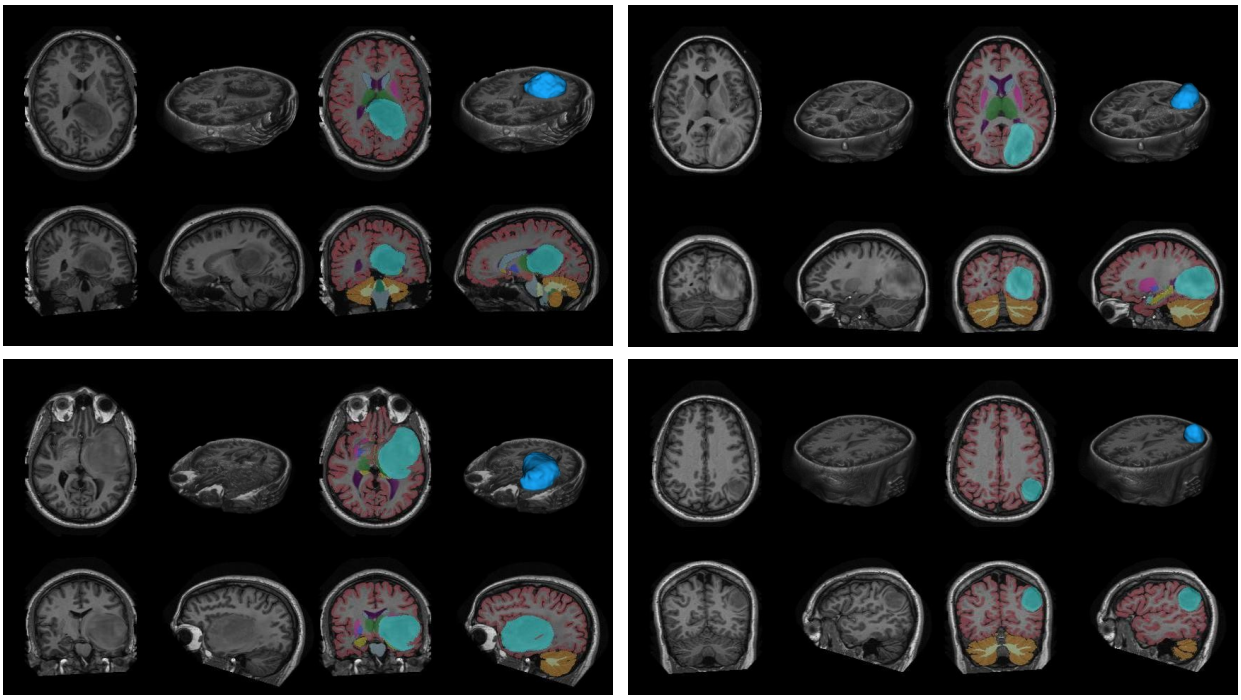


Figure 6.5: Examples of the synthesized test images.

	Tumor side			Non-Tumor side			All sides		
	sub	cortex	all	sub	cortex	all	sub	cortex	all
SAMSEG	0.704	0.794	0.783	0.727	0.814	0.804	0.716	0.805	0.794
Our(τ=0)	0.722	0.827	0.815	**0.789**	0.845	0.838	0.760	0.836	0.827
Our-DISP	**0.741**	**0.833**	**0.821**	0.777	**0.848**	**0.839**	0.760	**0.841**	**0.831**
Our-SVF	0.736	0.829	0.818	0.780	0.847	0.838	**0.760**	0.838	0.828

Table 6.1: The synthesized test set segmentation accuracy (weighted-mean-Dice).

procedure are show in Figure 6.5.

Our DISP and SVF models are tested with a fixed scale factor $\tau = 0.5$, and we compare performance with SAMSEG and a no-deformation baseline ($\tau = 0.0$) that the atlas is only modified with the pre-segmented tumor (as shown in Table 6.1). Our method, including the baseline, outperforms plain SAMSEG. We speculate that the GMM estimation in SAMSEG benefits from the accurate tumor prior. Compared with the baseline, we can see the tumor side segmentation, especially the subcortical tissues (indicated as *sub*), are further improved with the learned deformation.

6.4.3 Test with Real Data

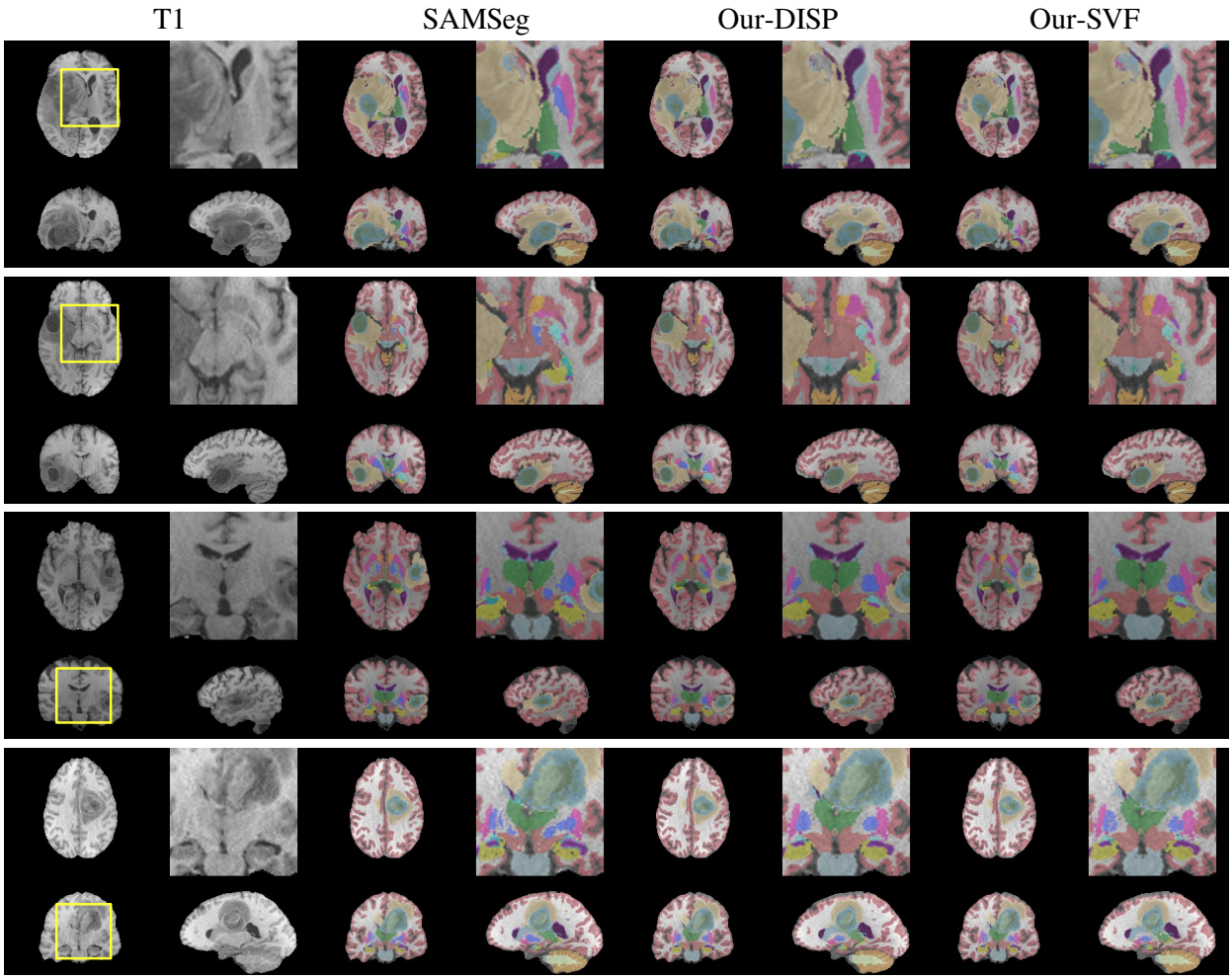


Figure 6.6: Results of testing our method in BraTS dataset, compared with SAMSEG. Note, this experiment is not a strict *test* because nnU-Net is trained with the same BraTS dataset. So the initial tumor label is quite accurate and could be instead treated as if it came from a high-quality manual segmentation.

Lastly, we test our method with the real patient scans found in BraTS. The result is in accordance with previous experiments and shows that modifying the atlas with our method has a visible positive effect on the segmentation. Figure 6.6 show four comparisons ($\tau = 0.7$). The subcortical brain structures, especially tissues that are close or adjacent to the tumor or edema, tend to be better segmented. Note the segmentation outputs are different in

DISP/SVF setups. Different τ also plays a role in segmentation in the area close to the lesion (cf. Section 6.4.2). Therefore, an advocated workflow of using the proposed method in a medical scenario would be to run a batch of segmentations in a tumor-bearing scan with different τ. Experts can interactively pick one to work with or make a further refinement.

6.5 Summary

We propose a novel method that directly estimates tumor-induced deformation (inverse problem of the tumor growth) from one single MR scan. This way, the highly entangled image-to-atlas registration and tumor-growth calibration problems are avoided. Acting as an intermediate process, this learned mass-effect deformation improves anatomical/semantic segmentation of tumor-bearing brain images. Moreover, this framework is not specific to any particular type of tumor or segmentation methods, and offers the flexibility to work with any existing atlas-based generative approaches.

Chapter 7

Conclusion

7.1 Summary and Conclusion

This dissertation aims to study new frameworks and methods for effective INR learning for point clouds and explore applications that benefit from its advancement. In so, it proposes a successful occupancy learning framework with a novel DFS operation that enables associations between features localized around query points and input points which aims to enhance local feature representations. Three applications are implemented with this new method: 3D shape reconstruction of generic 3D models from sparsely sampled point clouds; implicit occupancy function based image-registration of the liver from sparse samples captured during surgery; and in learning deformations induced by aggressive tumor growth to guide atlas-based full brain image segmentation. The outcomes of this dissertation include: a comprehensive review on related works; a novel network architecture and INR-based algorithm for efficiently learning implicit shape representations from point cloud data, a successful surface reconstruction model for generic object shapes (assessed against ShapeNet-13); a methodology for solving complex problems where access to adequate labeled training data is limited, and new approaches for tackling challenging medical 3D registration and segmentation problems using our network

to predict shape in the presence of deformation, have shown to give improved performance.

Literature Review and Identification of INR's Limitation In chapter 2 of this thesis, a broad range of related topics are reviewed, including coverage of basic knowledge about 3D shape acquisition and representation, influential surface reconstruction methods, machine learning approaches of 3D point clouds, and existing INR works which attempt to implicitly represent 3D shapes. Together, these contextualize the general background of our research. The core of our work lies at the intersection of 1) surface reconstruction from point clouds, 2) INR of 3D shapes, and 3) convolutional deep learning of point clouds.

From this initial review, we notice that most existing 3D shape INR methods use an encoder-decoder model where an MLP decodes the low-dimensional shape embedding $\mathbf{Z}_i$ (a global shape representation typically produced with pretrained networks or auto-encoders of 3D shapes) to a continuous function that represents a target 3D shape, i.e., $\mathbf{Z}_i \mapsto MLP_{\mathbf{Z}_i}(\cdot)$. This MLP itself is a mapping (conditioned on $\mathbf{Z}_i$) from $\mathbb{R}^3$ to $\mathbb{R}$ (assuming scalar output), responding to an input "query" by returning the function value at this 3D query point. Assuming general object shapes can be equivalently represented as point clouds, voxel volumes, multiview images, implicit functions, or eventually low-dimensional latent embeddings, other INR methods [11, 16, 15, 14, 25] have thus far performed some useful domain transformation learning and show the potential to achieve impressive performance in shape generative tasks.

Specifically, we have seen global embedding based INR methods do well for transforming 3D shapes from one form to another, generating arbitrary shapes, faking an object from 2D images, or hallucinating common structure to inpaint missing parts [141, 7, 8]. From this perspective, our network is essentially a mapping from the point cloud domain to an implicit function domain, with point convolution operations. However, our research (originating from surface reconstruction given only sparse point cloud samples), intends to recover the objective shape from a vastly imperfect observation, and thus has a slightly different emphasis

than other generative 3D shape tasks. On one hand, we expect the network to learn shape prior so that it becomes able to recover shapes from corrupted input scans that traditional non-learning methods can not deal with. On the other hand, we require the trained models to reconstruct such observed shapes robustly and faithfully, especially given their likely nuances and deviation from known shapes.

In the following, we attempt to delve more deeply into the specific features of our proposed architectures and why they are effective.

Conditioning-via-Sampling and the SCFL Framework Due to an encoder-decoder architecture and low-dimensional shape embedding utilized in other INR 3D shape learning methods, they are usually incapable of adequately capturing local geometric detail [16]. There are already several solutions that have been proposed to mitigate this problem: 1) combining convolutional voxel-grid of features and implicit representations to fit complex 3D structures [17, 25, 147, 142]; 2) incorporating high-frequency functions into INR to fit the data containing fine-grained local details [39, 26]; 3) crafting sophisticated mathematical/geometrical formulations of novel network structures, or special loss functions to improve the learned local shape priors to mitigate any global embedding problems [148, 149, 12, 43].

Different from the solutions above, we propose a new DFS-based architecture "conditioning-via-sampling" (detailed in Chapter 3) that unifies both the encoder and decoder in order to completely avoid the low-dimensional global shape embedding problem. The way we map points, as a swarm of 3D coordinates, into a reconstructed implicit function is fundamentally different from other INR approaches which use conditioning-via-concatenation [15, 11, 147, 25] and conditioning-via-hypernetwork [146, 16, 39, 145].

Our method is inspired by PCNN [3], which uses an extension operation to map input points to continuous feature spaces within hidden continuous functions, so that convolution operations become possible. We define multiple DFS operations (PCNN-based and Flex-

Convolution-based) that sample feature vectors for query points within these continuous functions. By stacking DFS layers, we build a multi-scale hierarchical network to learn implicit occupancy functions from sparse and noisy point clouds.

The surface reconstruction system implemented with this network achieves competitive performance in reconstruction of generic object shapes. Moreover, our method demonstrates several interesting properties: faithfulness for recovering uncommon characteristics, robustness to rotation, flexibility to handle different levels of sparsity in the input point clouds, and significantly better generalization in the presence of unseen shape categories. The reliability and robustness of our SCFL framework make it a good option for many real-world problems, especially in scenarios where high-quality 3D acquisition is restricted or tasks where robustness is vitally important, like in the medical tasks studied in this dissertation.

Solving Complex Problems by Synthesizing Training Data In our surface reconstruction work, we are motivated by the effectiveness and convenience of using training data (query point and function value pairs) generated from shapes represented as meshes. This framework was leveraged in real-world medical tasks in Chapter 5 and 6, which were extended to feed more complex and specialized human knowledge into the network. In liver registration (chapter 5) we synthesize point clouds sampled from the liver surface to make the network learn the stylus sampling pattern and spectrum of a physical liver's deformation. In tumor-bearing brain segmentation (chapter 6) we synthesize tumor growths and record volumetric deformation fields, which are used to train the network to capture correspondence between tumor's position/shape and deformation caused by its mass-effect and infiltration.

Occupancy Function Guided Liver Image-to-Physical Registration Because our DFS operator is defined with continuous convolution and trained with arbitrarily sampled query points, the occupancy functions produced by our network have well-behaved derivatives

(i.e. are globally continuous, piece-wise differentiable). In Chapter 5, we introduce a novel methodology that improves non-rigid image-to-physical registration with the PCNN-based occupancy network. We train the network to produce liver occupancy functions from sparsely sampled points and use this reconstructed liver to guide non-rigid registration. Comparing with rigid and non-rigid ICP baselines, this method sees significant reduction in registration errors.

Learning Tumor-Induced Deformation In our final study, we investigate tumor-bearing brain segmentation and propose a novel method that directly estimates tumor-induced deformation with our SCFL network. In this way, highly entangled image-to-atlas registration and tumor-growth calibration problems are decoupled. The predicted deformation is then used to warp and modify the brain atlas to represent the change so that existing atlas-based healthy-brain segmentation methods can be applied directly to these pathological images. Importantly, this framework is not specific to any particular tumor detection/segmentation method (compatible with manual delineation of tumors) or atlas-based segmentation back-ends. Again, this learned mass-effect deformation improves anatomical/organ-at-risk segmentation of tumor-bearing brain MR images.

7.2 Features of Our DFS based SCFL

Aside from the properties such as permutation and transformation invariance that can be seen in many point cloud deep learning methods, our SCFL method has several distinct features. We emphasize that this framework maximizes the flexibility of point data representations.

First, as an inheritance from the point-convolution-based operator PCNN and Flex-Convolution, our SCFL network can be configured to take as input any dimensional points and any dimensional input feature vector. For example, in Section 3.3.1 we illustrate the

PCNN operators with 2D points. If needed, we can use time as the 4th dimension and represent dynamic 3D objects (like 3D animation) as 4D point clouds. There is no significant difficulty to generalize this method to 2D or 4D point sets.

Second, our method supports flexible training. From its architecture (Figure 3.2), we can see the network has two streams, the point cloud main stream and the query point side stream. They are only entangled at the second restriction operator of the dual-restriction block. The two streams' learnable parameters can be pre-trained and fine-tuned separately to apply domain transfer learning. For example, if we have only a small number of *labeled* data for training point cloud part segmentation, we can pre-train the main stream with a large number of occupancy training data generated from unlabeled 3D meshes (detailed in Chapter 4).

Third, it also supports flexible inference. Since the main stream is independent of the query points, one can compute and clamp it first. Then a large number of query points (for example, voxelization of a volume [151]) or a set of moving points (like optimizing the mesh vertices position) can be fed to the side stream recurrently. Without having to compute the main stream for every batch of query points, inference can be achieved more efficiently.

Lastly and most importantly, this method preserves the *positional property* associated with the feature (and the learned target function) throughout the entire stage of the network, c.f. fully convolutional networks for a counterpart in image domain. Importantly, each feature map is an implicit continuous function in the 3D space. So we can *locate* the continuous function and sample it anywhere in $\mathbb{R}^3$, and compute its local derivative.

7.3 Discussions from Multiple Perspectives

With the success achieved by the proposed SCFL method, it is rational to wonder what the key characteristics might be that have enabled it to work so well. While a concrete in-depth

answer to this question is restricted by our finite knowledge and empirical study, we attempt to speculate about possible reasons for performance observed from several view points.

Network Enriched RBF Learning From a traditional surface reconstruction perspective, our occupancy network could be viewed as a deep-learning enhanced RBF surface reconstruction method (cf. section 2.2.2) which is equipped with powerful shape priors learned from the dataset. As such, our method appears to keep the advantages of both learning based surface reconstruction methods and non-learning optimization methods.

As reviewed in section 2.2.2, RBFs are popular for interpolating scattered data and is used for surface reconstruction [200, 83, 84]. These methods typically fit the input point cloud to the zero-level-set of a scalar implicit function that are modeled as overlaid RBFs. For each input point cloud, the coefficients associated with each RBF are optimized to minimize distance from the iso-surface to all observed input points. As shown in section 3.3, the Gaussian function is the fundamental building block of our PCNN-based occupancy network, because the learned continuous feature field is essentially modeled by overlaid Gaussians. The extension operation maps points to sum of Gaussian functions that are shifted to each point (Equation 3.3), and the convolution kernels are composed as $3 \times 3 \times 3$ grids of Gaussian functions each scaled with a learnable weight (see Figure 3.4 for a visual illustration).

Our occupancy learning approach enhances a traditional RBF-based reconstruction method in two aspects. First, by allowing for a hierarchical distribution of RBF (Gaussian) that model local shape geometry, the learned mixtures of these localized RBF models enhance performance over traditional RBF reconstructions (like that in [200, 83, 84]). That's because these hierarchical distributed RBFs may overlay and combine with each other to represent complex shape characteristics. Moreover, as we move deeper into the layers of the network, the RBF-based features become more complex than simple Gaussians, because it allows for non-linear mixing and combinations, enhancing its representation ability over simple linear

system of RBFs.

Second, as a learning model, our network may benefit from seeing exemplars in the training phase and reconstructed shapes with the help of learned shape priors, rather than simply interpolating surface over each instance of point clouds like in traditional RBF-based methods. Notice, in comparison to traditional optimization based methods, our networks do not need optimization to regress occupancy functions for new inputs after training. Instead, the occupancy value is predicted by a single feed-forward run. Because the network is already adjusted to a global "low-energy" state during training, so it generally produces shapes that have small *"reconstruction"* loss when presented with any new point clouds.

Built-in Regularization From learning theory and probability perspectives, a classification model with greater representation ability (complex model) needs larger training data to prevent over-fitting and achieve satisfactory generalization to unseen data. Deep learning models typically imply a hypothesis class with tremendously high complexity, which makes it necessary to apply regularization (dropout, weight-decay, data augmentation, etc.).

While the point convolutional shape features of our method is a powerful representation of 3D shapes, the network appears to be less complex and less likely for over-fitting than other learning-based surface reconstruction models. This is convincingly expressed with the network's robustness for shapes from unseen categories (section 4.3.3) and rotation equivariant property (section 4.4). We suspect that the proposed condition-via-sampling framework largely simplifies occupancy learning over "condition-via-concatenation" and "condition-via-hypernetwork" formulations. The localized feature sampling of each query acts like a built-in regularization term for this learning problem.

When comparing our method with the encoder-decoder configuration used in "condition-via-concatenation" or "condition-via-hypernetwork" approaches, we argue that the other methods require a model having much greater expressive or complex hypothesis set than our

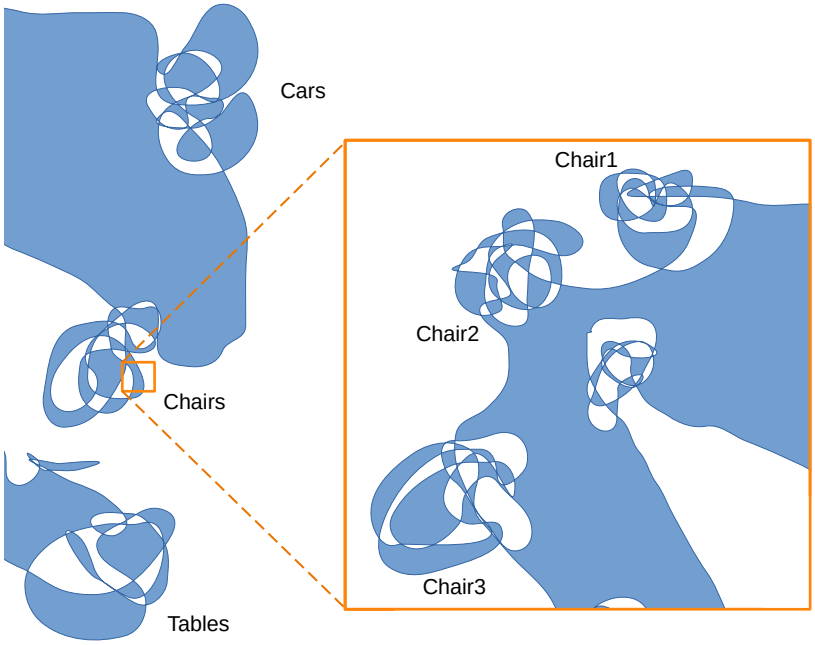


Figure 7.1: The decoder MLP in "condition-via-concatenation" framework is expected to have complex (binary) decision boundary in the product space $\mathbb{R}^{\dim E} \times \mathbb{R}^3$ (collapsed into 2D plane for visualization). This sketch aims to intuitively illustrate the complex manifolds within particular instance of shapes, and with related class embeddings map to proximal locations in the manifold.

method. To highlight this, let's first restrict our vision to the decoder MLP in condition-via-concatenation framework as shown in Figure 2.4. The latent code $Z_i \in \mathbb{R}^{\dim E}$ in global shape embedding space is concatenated with the coordinates of the query point $q \in \mathbb{R}^3$ to be the input [12], and the decoder MLP maps this concatenated vector to a binary output that indicates occupancy states, i.e. $\mathbb{R}^{\dim E} \times \mathbb{R}^3 \to \{0, 1\}$. In general, it seems reasonable to assume that the decision boundary of this model should fit the manifold represented by all training shapes. Taking a car$\mapsto Z_1$ and a chair$\mapsto Z_2$ as example, we can imagine there should be a piece of decision boundary depicting the "car" shape locating around $[Z_1, \cdot, \cdot, \cdot]^T$ and a similar situation for the "chair" shape around $[Z_2, \cdot, \cdot, \cdot]^T$. As there are multiple categories of object shapes embedded in the latent space with $\dim E \gg 3$, what supervises the network

[12] All input shapes are scaled and shifted into an unit cubic space centered at $(0, 0, 0)$.

is likely a manifold that has a lot of detail curled up in the vicinities of these individual shapes $[Z_i, \cdot, \cdot, \cdot]^T, i = 1, 2....$ This is abstractly depicted in Figure 7.1 with the $\mathbb{R}^{\dim E+3}$ embedded space imprecisely been collapsed onto the 2D plane. This shape representation is highly entangled, the network may only learn averaged stereotypical representations for the object classes, and new inputs tend to be attracted into these "embedding sinks" of known shapes, thus prompting a reconstruction that may blend between nearby shapes in a global manner. Moreover, when feeding the network with a new input from an unseen category (like those animals in Figure 4.7), it may well be mapping into regions well outside the learned manifold (a kind of "no-mans-land" in embedded space), and the reconstruction model fails catastrophically, because it has no good representation to draw from in the learned embedding space.

In contrast, the MLP in our network (referring to the architecture shown in Figure 3.2) takes as inputs the point features $\psi(q)$ sampled around the surface and maps that into the occupancy states. This shape feature is defined based on the hidden continuous feature functions of our DFS layers. If this 3D shape representation is trained well, the geometric structure of local surface geometry and global categorical shape priors are captured by the features learned hierarchically in the network. The MLP classifier encourages the sampled feature vectors being distinctive for inside and outside query points.

To fully express this idea, it is helpful to conduct a thought experiment about our network. Let's assume the MLP component in our network (Figure 3.2) is progressively simplified to keep less depth and less neurons in each layer (to be trained after initialization). Despite the performance degradation, the network should still work to some extent (at least for the simplest surface with a noise-free dense set of samples). In an extreme situation when it is trimmed down to an un-weighted sum, the occupancy of q is nothing but a vote from the multi-scale features $\psi_q^1, \psi_q^2, ..., \psi_q^n$ produced in each DFS layer (cf. Figure 4.2, an over-simplified hyper-plane decision boundary). Training the network will directly make $\psi(q)$ as

polarized as possible for different sides of the surface. We thus speculate, that adding an MLP between $\psi(q)$ and $\{0, 1\}$ will in theory make the feature more powerful and distinctive because it enables more complex combinations and interactions between scales. In total, the MLP classifier in our configuration, we believe, fits a decision boundary that's regularized over various scales of reconstructions, allowing for more smoothing of localized shape, rather than smoothing between more global embeddings/exemplars. As result, the whole occupancy network requires less training and maintains good generalization, as well as adaptation more faithful to new, unseen formations that may exhibit similar local features to formations previously seen in training. In some sense, this DFS feature may serve to better deconstruct shape into more localised 3D features (akin to the 2D log-gabor-like patterns extracted by traditional CNN networks from natural images), then re-aggregating them back into more complex (global) arrangements representative of the overall shapes (inferred from their sparsely sampled point clouds).

Network-to-Geometry Isomorphism Ultimately, if one entity (object/shape/prior) is effectively modeled or parameterized with another entity, it means there is an isomorphism between these two spaces. For example, a Gaussian Mixture Model (GMM) with two peaks may only work well if the data are really distributed as two clusters. Because the structure (two peaks) of this particular GMM is isomorphic to any two-cluster data, optimizing the centres and variance of Gaussian can model these data. In our case, the network configuration appears to exhibit more isomorphic properties (to the 3D shape space) in a comprehensive way than other configurations. It doesn't matter whether input samples are represented as volumetric grids, point clouds, surface meshes, or implicit functions. When trying to explain our network's robustness regarding rotations or unseen object classes that break other methods catastrophically, we speculate that the location-preserving DFS operation, spatial pooling, and feature aggregation (the multi-resolution hierarchical UNet structure) play

important roles. Because real-world stimulus can be considered as hierarchical arrangement of properties emerging at different scales, the network should be configured as stacked DFS layers with features aggregated in each resolution.

In this way, it is our belief that the success of reconstruction is mainly due to the linkages and associations between low-level and aggregated low-level local features presented. When corresponding neurons are triggered across scales, the network appears to be able to use that to rebuild shape rather than looking at learned embeddings (an averaged/blended exemplar, or eigen-shape - that acts as a surrogate for a class, yet may not be representative of a realistic sample/observed shape). Eventually, the network understands a probabilistic mapping between certain combinations of these features and a set of arbitrary query-label pairs. The ability to rebuild the shape like a chair and all of these different objects, we speculate, may well be due to the network experiencing a form of resonance happening across all of those scales. Moreover, it models co-occurrence of the features in each object category, and so reconstructs that type of shape. The ability to adapt to local variations (which results in apparent improvements in robustness) in shape is because the resonance is happening across *local* associations and low-level features in similar areas. When all of the related neurons are firing the network seems able to reconstruct the whole shape well. But even when some of those neurons are firing in the same areas, it's able to use those to reconstruct locally. This keeps a faithful representation of the point clouds the network is actually seeing (for instance, in its ability to retain asymmetries in some sampled point distributions).

7.4 Future Research Directions

Achieving Theoretically Guaranteed Invariant Properties As stated in section 4.4, our network achieves rotation equivariant shape representations to a certain degree. Recently, Deng et al. propose Vector Neurons (VN) [144] as a point cloud framework that preserves

SO(3) equivariance. They construct rotation-equivariant learnable layers and build networks (with PointNet [4] and DGCNN [20] backbones) for point cloud shape classification, part segmentation, and INR. Their INR model is implemented based on the encoder-decoder architecture where the encoder computes global shape latent code with a VN-based PointNet and the ResNet-based decoder constructed to produce occupancy probability (conditioning-via-concatenation). It would be interesting to see if this SO(3) equivariant operation can be extended (just like PCNN and FlexConvolution have been modified as in section 3) to form DFS layer and achieve conditioning-via-sampling. This work may result in a network with enhanced representation for local fine details and guaranteed rotation equivariance. Moreover, for more distant future considerations, we may explore and formulate models having more complex invariant properties like the diffeomorphism between feature spaces and the output space which may result in a guaranteed invariant topology for those embedded manifolds (decision boundary, contour lines, or level sets).

Integrating INR with FEM Normally, shape INR models ensure output fields with well-behaved and closed-form derivatives, offering a new toolbox for solving problems that are well described by differential equations, such as Poisson equations and the dynamics of fluid flow. Sitzmann et al. [26] introduce an INR model (Sinusoidal Representation Networks or SIREN) in which both the target implicit function and its higher-order derivatives can be supervised or constrained, and they demonstrate multiple applications with this method including solving the Poisson equation and representing shapes with SDFs. Kochkov et al. [201] accelerate the computation of fluid dynamics by using a learned solver to replace the traditional iterative solver for Navier-Stokes equations. These works indicate a promising direction for solving complex problems with INR integrated systems.

In our liver non-rigid registration work (chapter 5), we represent the liver with tetrahedron elements and solve its deformation templates with linear elastic equations and quadratic

minimization solver. For modeling the growth of brain tumors (chapter 6), we use the reaction-diffusion model (Cartesian finite elements) implemented in TumorSim [24] to generate the pathological training and testing images. For both tasks, the models are oversimplified for describing these intractable tissue deformations. In these problems, it would be beneficial if some complex geometry and biological properties can be effectively captured with an INR network and then integrated into a properly defined finite element system. Moreover, a general framework may be established from such geometry application and extended to other domains involving continuous fields.

Learning Novel Implicit Representations As briefly mentioned in section 2.5.2, there are some studies that attempt to learn implicit shape representations other than SDFs or occupancy functions: sign-agnostic SDFs [12, 13], neural-pull [43], on-surface priors [44], differentiable poisson [148], medial fields [46], and hybrid representations [131, 202]. It's worthy and easy to try using our network to learn these functions, because our SCFL framework and conditioning-via-sampling scheme are mostly comparable with these novel implicit representations. However, we notice most representations can only satisfy shapes that have determined inside and outside. There is no good implicit representation can handle well a large size (number of points) point cloud scan of large scale scene. Because such large size data can always be normally parsed into a set of surface patches, we typically instead consider patch-centric representations. With an emphasis on reconstruction tasks, there are two potential directions to work on: 1) determining cross-surface line segments, and 2) learning Gaussian curvature.

For the first direction, we may try to use two arbitrary points that determine a line segment as an instance of query. A long line segment may span across the surface multiple times, but we should restrict the length of it so that it's safe to consider only two cases, either cross (its two endpoints having different polarities) or not (having same polarity). To make

this representation suitable for surface patches that have no obvious inside or outside, the implicit representation should be formulated so that the values polarize between the two sides of the target surface, but their absolute polarities do not matter for the representation. There are some mathematical structures that retain such properties (like dot product) that can be investigated for their ability to infer an inside-outside agnostic implicit surface representation.

For the second direction, we are interested in Gaussian curvature, because its inherently intrinsic to the smooth 2D manifold surface. It's invariant with respect to rotation and transformation. We can start from a simple setup: the network is trained to produce the correct curvature (training shape) at the correct location (on the target surface) and keep zero in other off-surface space (so that it's inside-outside agnostic). Then in reconstruction and testing, we initialize a surface mesh with the input points and evolve the surface with guidance from the Gaussian curvature representation of the surface, i.e., inferring the curvature with the trained network and comparing it with the current surface's curvature, and then modify it to minimize the differences between them.

In light of the above, it would seem that there are many potential directions that could be followed - either from a more theoretical point perspective like future experimentation with the structure of how and what the network can be configured to learn, and from more practical perspectives relating to how to optimize implementations for particular applications.

In addition, there are many applications outside the realm of medical image/shape analysis that can be explored using these tools as a basis, for instance, in more generalized point cloud editing/manipulation or synthesis.

In closing, it is the author's hope that some of these potential avenues can be investigated and realized in the near future, and that the tools and strategies developed through this dissertation can be extended to find utility and application in other domains which would benefit from inference and interpretation of unstructured point cloud data.

Bibliography

[1] Angel X Chang et al. "Shapenet: An information-rich 3d model repository". In: *arXiv preprint arXiv:1512.03012* (2015).

[2] Yuhui Yuan and Jingdong Wang. "OCNet: Object Context Network for Scene Parsing". In: *arXiv preprint arXiv:1809.00916* (2018).

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

[4] Charles R Qi et al. "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.

[5] Aron Monszpart et al. "RAPter: rebuilding man-made scenes with regular arrangements of planes." In: *ACM Trans. Graph.* 34.4 (2015), pp. 103–1.

[6] Charles R Qi et al. "Frustum pointnets for 3d object detection from rgb-d data". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 918–927.

[7] Nikolas Lamb, Sean Banerjee, and Natasha Kholgade Banerjee. "Deepmend: Learning occupancy functions to represent shape for repair". In: *European Conference on Computer Vision*. Springer. 2022, pp. 433–450.

[8] Nikolas Lamb, Sean Banerjee, and Natasha Kholgade Banerjeee. “Deepjoin: Learning a joint occupancy, signed distance, and normal field function for shape repair”. In: *ACM Transactions on Graphics (TOG)* 41.6 (2022), pp. 1–10.

[9] Ronghan Chen, Yang Cong, and Jiahua Dong. “Unsupervised Dense Deformation Embedding Network for Template-Free Shape Correspondence”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 8361–8370.

[10] Matan Atzmon et al. “Controlling Neural Level Sets”. In: *arXiv preprint arXiv:1905.11911* (2019).

[11] Jeong Joon Park et al. “DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 165–174.

[12] Matan Atzmon and Yaron Lipman. “Sal: Sign agnostic learning of shapes from raw data”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2565–2574.

[13] Matan Atzmon and Yaron Lipman. “SALD: Sign agnostic learning with derivatives”. In: *arXiv preprint arXiv:2006.05400* (2020).

[14] Vincent Sitzmann et al. “Metasdf: Meta-learning signed distance functions”. In: *arXiv preprint arXiv:2006.09662* (2020).

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

[16] Lars Mescheder et al. “Occupancy networks: Learning 3d reconstruction in function space”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4460–4470.

[17] Songyou Peng et al. “Convolutional occupancy networks”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*. Springer. 2020, pp. 523–540.

[18] Jiapeng Tang et al. “SA-ConvONet: Sign-Agnostic Optimization of Convolutional Occupancy Networks”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6504–6513.

[19] Fabian Groh, Patrick Wieschollek, and Hendrik PA Lensch. “Flex-Convolution: Million-scale point-cloud learning beyond grid-worlds”. In: *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part I 14*. Springer. 2019, pp. 105–122.

[20] Yue Wang et al. “Dynamic graph cnn for learning on point clouds”. In: *arXiv preprint arXiv:1801.07829* (2018).

[21] E. Lee Brewer et al. “The image-to-physical liver registration sparse data challenge”. In: *Medical Imaging 2019: Image-Guided Procedures, Robotic Interventions, and Modeling*. Vol. 10951. International Society for Optics and Photonics. 2019.

[22] Jon S Heiselman et al. “Characterization and correction of intraoperative soft tissue deformation in image-guided laparoscopic liver surgery”. In: *Journal of Medical Imaging* 5.2 (2017).

[23] Guha Balakrishnan et al. “VoxelMorph: a learning framework for deformable medical image registration”. In: *IEEE transactions on medical imaging* 38.8 (2019), pp. 1788–1800.

[24] Marcel Prastawa, Elizabeth Bullitt, and Guido Gerig. “Simulation of brain tumors in MR images for evaluation of segmentation efficacy”. In: *Medical image analysis* 13.2 (2009), pp. 297–311.

[25] Stefan Lionar et al. “Dynamic plane convolutional occupancy networks”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pp. 1829–1838.

[26] Vincent Sitzmann et al. “Implicit neural representations with periodic activation functions”. In: *Advances in neural information processing systems* 33 (2020), pp. 7462–7473.

[27] Yulan Guo et al. “Deep learning for 3d point clouds: A survey”. In: *IEEE transactions on pattern analysis and machine intelligence* 43.12 (2020), pp. 4338–4364.

[28] Laith Alzubaidi et al. “Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions”. In: *Journal of big Data* 8 (2021), pp. 1–74.

[29] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. “3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3145–3152.

[30] Yangyan Li et al. “Pointcnn: Convolution on x-transformed points”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 820–830.

[31] Yiru Shen et al. “Mining point cloud local structures by kernel correlation and graph pooling”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4548–4557.

[32] Jiaxin Li, Ben M Chen, and Gim Hee Lee. “So-net: Self-organizing network for point cloud analysis”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9397–9406.

[33] Gusi Te et al. “Rgcnn: Regularized graph cnn for point cloud segmentation”. In: *2018 ACM Multimedia Conference on Multimedia Conference*. ACM. 2018, pp. 746–754.

[34] Wenxuan Wu, Zhongang Qi, and Li Fuxin. "Pointconv: Deep convolutional networks on 3d point clouds". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9621–9630.

[35] Yongcheng Liu et al. "Relation-Shape Convolutional Neural Network for Point Cloud Analysis". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8895–8904.

[36] Roman Klokov and Victor Lempitsky. "Escape from cells: Deep kd-networks for the recognition of 3d point cloud models". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 863–872.

[37] Daniele Grattarola and Pierre Vandergheynst. "Generalised implicit neural representations". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 30446–30458.

[38] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. "Scene representation networks: Continuous 3d-structure-aware neural scene representations". In: *Advances in Neural Information Processing Systems* 32 (2019).

[39] Ben Mildenhall et al. "Nerf: Representing scenes as neural radiance fields for view synthesis". In: *Communications of the ACM* 65.1 (2021), pp. 99–106.

[40] Vincent Sitzmann et al. "Light field networks: Neural scene representations with single-evaluation rendering". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 19313–19325.

[41] Michael Niemeyer et al. "Occupancy flow: 4d reconstruction by learning particle dynamics". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 5379–5389.

[42] Junsheng Zhou et al. “Learning consistency-aware unsigned distance functions progressively from raw point clouds”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 16481–16494.

[43] Baorui Ma et al. “Neural-pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces”. In: *arXiv preprint arXiv:2011.13495* (2020).

[44] Baorui Ma, Yu-Shen Liu, and Zhizhong Han. “Reconstructing surfaces for sparse point clouds with on-surface priors”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 6315–6325.

[45] Michael Oechsle, Songyou Peng, and Andreas Geiger. “Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5589–5599.

[46] Daniel Rebain et al. “Deep medial fields”. In: *arXiv preprint arXiv:2106.03804* (2021).

[47] Francis Engelmann et al. “Exploring spatial context for 3d semantic segmentation of point clouds”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 716–724.

[48] Mingyang Jiang et al. “Pointsift: A sift-like network module for 3d point cloud semantic segmentation”. In: *arXiv preprint arXiv:1807.00652* (2018).

[49] Xin-Yang Zheng et al. “SDF-StyleGAN: Implicit SDF-Based StyleGAN for 3D Shape Generation”. In: *arXiv preprint arXiv:2206.12055* (2022).

[50] Marian Kleineberg, Matthias Fey, and Frank Weichert. “Adversarial Generation of Continuous Implicit Shape Representations”. In: *arXiv preprint arXiv:2002.00349* (2020).

[51] Chun-Liang Li et al. “Point Cloud GAN”. In: *arXiv preprint arXiv:1810.05795* (2018).

[52] Matthew Berger et al. “A survey of surface reconstruction from point clouds”. In: *Computer Graphics Forum*. Vol. 36. 1. Wiley Online Library. 2017, pp. 301–329.

[53] Frédéric Cazals and Joachim Giesen. “Delaunay triangulation based surface reconstruction: ideas and algorithms”. PhD thesis. INRIA, 2004.

[54] Cheng Chun You et al. “A survey on surface reconstruction techniques for structured and unstructured data”. In: *2020 IEEE Conference on Open Systems (ICOS)*. IEEE. 2020, pp. 37–42.

[55] Ruud M. Bolle and Baba C. Vemuri. “On three-dimensional surface reconstruction methods”. In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 13.01 (1991), pp. 1–13.

[56] Raphael Sulzer et al. “A Survey and Benchmark of Automatic Surface Reconstruction from Point Clouds”. In: *arXiv preprint arXiv:2301.13656* (2023).

[57] Nicholas M Patrikalakis and Takashi Maekawa. *Shape interrogation for computer aided design and manufacturing*. Vol. 15. Springer, 2002.

[58] Michael S Floater and Kai Hormann. “Surface parameterization: a tutorial and survey”. In: *Advances in multiresolution for geometric modelling* (2005), pp. 157–186.

[59] Pascal Laube, Matthias O Franz, and Georg Umlauf. “Deep learning parametrization for B-spline curve approximation”. In: *2018 International Conference on 3D Vision (3DV)*. IEEE. 2018, pp. 691–699.

[60] Jun Gao et al. “Deepspline: Data-driven reconstruction of parametric curves and surfaces”. In: *arXiv preprint arXiv:1901.03781* (2019).

[61] Fenqiang Zhao et al. “Fast Spherical Mapping of Cortical Surface Meshes Using Deep Unsupervised Learning”. In: *Medical Image Computing and Computer Assisted*

Intervention–MICCAI 2022: 25th International Conference, Singapore, September 18–22, 2022, Proceedings, Part VI. Springer. 2022, pp. 163–173.

[62] Marc Alexa et al. "Point set surfaces". In: *Proceedings Visualization, 2001. VIS'01.* IEEE. 2001, pp. 21–29.

[63] Riccardo Roveri et al. "A network architecture for point cloud classification via automatic depth images generation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4176–4184.

[64] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[65] Eugene L Allgower and Stefan Gnutzmann. "An algorithm for piecewise linear approximation of implicitly defined two-dimensional surfaces". In: *SIAM journal on numerical analysis* 24.2 (1987), pp. 452–469.

[66] William E Lorensen and Harvey E Cline. "Marching cubes: A high resolution 3D surface construction algorithm". In: *ACM siggraph computer graphics*. Vol. 21. 4. ACM. 1987, pp. 163–169.

[67] Greg Turk and Marc Levoy. "Zippered polygon meshes from range images". In: *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. ACM. 1994, pp. 311–318.

[68] Frédéric Cazals—Joachim Giesen. "Delaunay Triangulation Based Surface Reconstruction: Ideas and Algorithms". In: (2004).

[69] Hugues Hoppe et al. *Surface reconstruction from unorganized points*. Vol. 26. 2. ACM, 1992.

[70] Meenakshisundaram Gopi, Shankar Krishnan, and Cláudio T Silva. "Surface reconstruction based on lower dimensional localized Delaunay triangulation". In: *Computer Graphics Forum*. Vol. 19. 3. Wiley Online Library. 2000, pp. 467–478.

[71] David Cohen-Steiner and Frank Da. "A greedy Delaunay-based surface reconstruction algorithm". In: *The visual computer* 20.1 (2004), pp. 4–16.

[72] Nina Amenta, Marshall Bern, and David Eppstein. "The crust and the β-skeleton: Combinatorial curve reconstruction". In: *Graphical models and image processing* 60.2 (1998), pp. 125–135.

[73] Nina Amenta and Marshall Bern. "Surface reconstruction by Voronoi filtering". In: *Discrete & Computational Geometry* 22.4 (1999), pp. 481–504.

[74] Jean-Daniel Boissonnat. "Geometric structures for three-dimensional shape representation". In: *ACM Transactions on Graphics (TOG)* 3.4 (1984), pp. 266–286.

[75] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. "The power crust". In: *Proceedings of the sixth ACM symposium on Solid modeling and applications*. ACM. 2001, pp. 249–266.

[76] Ravikrishna Kolluri, Jonathan Richard Shewchuk, and James F O'Brien. "Spectral surface reconstruction from noisy point clouds". In: *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. ACM. 2004, pp. 11–21.

[77] Herbert Edelsbrunner and Ernst P Mücke. "Three-dimensional alpha shapes". In: *ACM Transactions on Graphics (TOG)* 13.1 (1994), pp. 43–72.

[78] Marek Teichmann and Michael Capps. "Surface reconstruction with anisotropic density-scaled alpha shapes". In: *Proceedings Visualization'98 (Cat. No. 98CB36276)*. IEEE. 1998, pp. 67–72.

[79] Fausto Bernardini et al. “The ball-pivoting algorithm for surface reconstruction”. In: *IEEE transactions on visualization and computer graphics* 5.4 (1999), pp. 349–359.

[80] Xiaolong Xu and Koichi Harada. “Automatic surface reconstruction with alpha-shape method”. In: *The visual computer* 19.7-8 (2003), pp. 431–443.

[81] David Levin. “Mesh-independent surface interpolation”. In: *Geometric modeling for scientific visualization*. Springer, 2004, pp. 37–49.

[82] Gaël Guennebaud and Markus Gross. “Algebraic point set surfaces”. In: *ACM Transactions on Graphics (TOG)*. Vol. 26. 3. ACM. 2007, p. 23.

[83] Greg Turk and James F. O’Brien. “Variational Implicit Surfaces”. In: (1999).

[84] Jonathan C Carr et al. “Reconstruction and representation of 3D objects with radial basis functions”. In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM. 2001, pp. 67–76.

[85] Michael Kazhdan. “Reconstruction of solid models from oriented point sets”. In: *Proceedings of the third Eurographics symposium on Geometry processing*. Eurographics Association. 2005, p. 73.

[86] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. “Poisson surface reconstruction”. In: *Proceedings of the fourth Eurographics symposium on Geometry processing*. Vol. 7. 2006.

[87] Josiah Manson, Guergana Petrova, and Scott Schaefer. “Streaming surface reconstruction using wavelets”. In: *Computer Graphics Forum*. Vol. 27. 5. Wiley Online Library. 2008, pp. 1411–1420.

[88] Michael Kazhdan and Hugues Hoppe. “Screened poisson surface reconstruction”. In: *ACM Transactions on Graphics (ToG)* 32.3 (2013), p. 29.

[89] Fatih Calakli and Gabriel Taubin. “SSD: Smooth signed distance surface reconstruction”. In: *Computer Graphics Forum*. Vol. 30. 7. Wiley Online Library. 2011, pp. 1993–2002.

[90] Wentao Yuan et al. “Pcn: Point completion network”. In: *2018 International Conference on 3D Vision (3DV)*. IEEE. 2018, pp. 728–737.

[91] Pedro Hermosilla, Tobias Ritschel, and Timo Ropinski. “Total denoising: Unsupervised learning of 3D point cloud cleaning”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 52–60.

[92] Lequan Yu et al. “Pu-net: Point cloud upsampling network”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2790–2799.

[93] Wang Yifan et al. “Patch-based Progressive 3D Point Set Upsampling”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5958–5967.

[94] Weixin Lu et al. “Deepvcp: An end-to-end deep neural network for point cloud registration”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 12–21.

[95] Donghoon Lee et al. “DeepPRO: Deep partial point cloud registration of objects”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5683–5692.

[96] Lizhe Qi et al. “DeepMatch: Toward Lightweight in Point Cloud Registration”. In: *Frontiers in Neurorobotics* 16 (2022).

[97] Francis Williams et al. “Deep geometric prior for surface reconstruction”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10130–10139.

[98] Charlie Nash and Christopher KI Williams. "The shape variational autoencoder: A deep generative model of part-segmented 3D objects". In: *Computer Graphics Forum*. Vol. 36. 5. Wiley Online Library. 2017, pp. 1–12.

[99] Kangxue Yin et al. "P2P-NET: bidirectional point displacement net for shape transform". In: *ACM Transactions on Graphics (TOG)* 37.4 (2018), p. 152.

[100] Panos Achlioptas et al. "Learning representations and generative models for 3d point clouds". In: *arXiv preprint arXiv:1707.02392* (2017).

[101] Charles Ruizhongtai Qi et al. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space". In: *Advances in neural information processing systems*. 2017, pp. 5099–5108.

[102] Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. "Deep learning with sets and point clouds". In: *arXiv preprint arXiv:1611.04500* (2016).

[103] Weikai Chen et al. "Deep rbfnet: Point cloud feature learning using radial basis functions". In: *arXiv preprint arXiv:1812.04302* (2018).

[104] David S Broomhead and David Lowe. *Radial basis functions, multi-variable functional interpolation and adaptive networks*. Tech. rep. Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.

[105] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[106] Jon Louis Bentley. "Multidimensional binary search trees used for associative searching". In: *Commun. ACM* 18 (1975), pp. 509–517.

[107] Yaoqing Yang et al. "Foldingnet: Point cloud auto-encoder via deep grid deformation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 206–215.

[108] Nanyang Wang et al. “Pixel2mesh: Generating 3d mesh models from single rgb images”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 52–67.

[109] Jun Gao et al. “Learning deformable tetrahedral meshes for 3d reconstruction”. In: *Advances In Neural Information Processing Systems* 33 (2020), pp. 9936–9947.

[110] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. “Pointwise convolutional neural networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 984–993.

[111] Yingxue Zhang and Michael Rabbat. “A Graph-CNN for 3D Point Cloud Classification”. In: *arXiv preprint arXiv:1812.01711* (2018).

[112] Pedro Hermosilla et al. “Monte carlo convolution for learning on non-uniformly sampled point clouds”. In: *ACM Transactions on Graphics (TOG)* 37.6 (2018), pp. 1–12.

[113] Jiancheng Yang et al. “Modeling point clouds with self-attention and gumbel subset sampling”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 3323–3332.

[114] Xu Yan et al. “Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 5589–5598.

[115] Shy Shalom et al. *Cone carving for surface reconstruction*. Vol. 29. 6. ACM, 2010.

[116] Simon Fuhrmann and Michael Goesele. “Fusion of depth maps with multiple scales”. In: *ACM Transactions on Graphics (TOG)*. Vol. 30. 6. ACM. 2011, p. 148.

[117] Andrea Tagliasacchi et al. “Vase: Volume-aware surface evolution for surface reconstruction from incomplete point clouds”. In: *Computer Graphics Forum*. Vol. 30. 5. Wiley Online Library. 2011, pp. 1563–1571.

[118] Jianxiong Xiao and Yasutaka Furukawa. “Reconstructing the world’s museums”. In: *International journal of computer vision* 110.3 (2014), pp. 243–258.

[119] Yizhou Yu. “Surface reconstruction from unorganized points using self-organizing neural networks”. In: *IEEE Visualization*. Vol. 99. Citeseer. 1999, pp. 61–64.

[120] IV Ivrissimtzis, W-K Jeong, and H-P Seidel. “Using growing cell structures for surface reconstruction”. In: *2003 Shape Modeling International.* IEEE. 2003, pp. 78–86.

[121] Xue-mei Wu, Gui-xian Li, and Wei-min Zhao. “Sparseness points cloud data surface reconstruction based on radial basis function neural network (RBFNN) and simulated annealing arithmetic”. In: *2007 International Conference on Computational Intelligence and Security Workshops (CISW 2007)*. IEEE. 2007, pp. 877–880.

[122] Akemi Gálvez et al. “Particle swarm optimization for Bézier surface reconstruction”. In: *International Conference on Computational Science*. Springer. 2008, pp. 116–125.

[123] Ioannis T Rekanos. “Shape reconstruction of a perfectly conducting scatterer using differential evolution and particle swarm optimization”. In: *IEEE Transactions on Geoscience and Remote Sensing* 46.7 (2008), pp. 1967–1974.

[124] Hui Huang et al. “Consolidation of unorganized point clouds for surface reconstruction”. In: *ACM transactions on graphics (TOG)* 28.5 (2009), p. 176.

[125] Hui Huang et al. “Edge-aware point set resampling”. In: *ACM transactions on graphics (TOG)* 32.1 (2013), p. 9.

[126] Shihao Wu et al. “Deep points consolidation”. In: *ACM Transactions on Graphics (ToG)* 34.6 (2015), p. 176.

[127] Lequan Yu et al. “EC-Net: an Edge-aware Point set Consolidation Network”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 386–402.

[128] Riccardo Roveri et al. “Pointpronets: Consolidation of point clouds with convolutional neural networks”. In: *Computer Graphics Forum*. Vol. 37. 2. Wiley Online Library. 2018, pp. 87–99.

[129] Tong He et al. “GeoNet: Deep Geodesic Networks for Point Cloud Analysis”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6888–6897.

[130] Christopher B Choy et al. “3d-r2n2: A unified approach for single and multi-view 3d object reconstruction”. In: *European conference on computer vision*. Springer. 2016, pp. 628–644.

[131] Yiyi Liao, Simon Donne, and Andreas Geiger. “Deep marching cubes: Learning explicit surface representations”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2916–2925.

[132] Alexandre Boulch and Renaud Marlet. “Deep learning for robust normal estimation in unstructured point clouds”. In: *Computer Graphics Forum*. Vol. 35. 5. Wiley Online Library. 2016, pp. 281–290.

[133] Nicholas Sharp and Maks Ovsjanikov. “Pointtrinet: Learned triangulation of 3d point sets”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*. Springer. 2020, pp. 762–778.

[134] Minghua Liu, Xiaoshuai Zhang, and Hao Su. “Meshing point clouds with predicted intrinsic-extrinsic ratio guidance”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*. Springer. 2020, pp. 68–84.

[135] Raphael Sulzer et al. “Scalable Surface Reconstruction with Delaunay-Graph Neural Networks”. In: *Computer Graphics Forum*. Vol. 40. 5. Wiley Online Library. 2021, pp. 157–167.

[136] Thibault Groueix et al. “AtlasNet: A Papier-M\ˆ ach\’e Approach to Learning 3D Surface Generation”. In: *arXiv preprint arXiv:1802.05384* (2018).

[137] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. “Deep image prior”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9446–9454.

[138] Harm De Vries et al. “Modulating early visual processing by language”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 6594–6604.

[139] Guy Gafni et al. “Dynamic neural radiance fields for monocular 4d facial avatar reconstruction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8649–8658.

[140] Sida Peng et al. “Animatable implicit neural representations for creating realistic avatars from videos”. In: *arXiv preprint arXiv:2203.08133* (2022).

[141] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. “Implicit functions in feature space for 3d shape reconstruction and completion”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 6970–6981.

[142] Chiyu Jiang 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.

[143] Chen-Hsuan Lin, Chaoyang Wang, and Simon Lucey. “Sdf-srn: Learning signed distance 3d object reconstruction from static images”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 11453–11464.

[144] Congyue Deng et al. “Vector neurons: A general framework for so (3)-equivariant networks”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 12200–12209.

[145] Yinbo Chen and Xiaolong Wang. “Transformers as Meta-learners for Implicit Neural Representations”. In: *Computer Vision – ECCV 2022*. Ed. by Shai Avidan et al. Cham: Springer Nature Switzerland, 2022, pp. 170–187.

[146] David Ha, Andrew Dai, and Quoc V. Le. “Hypernetworks”. In: *International Conference on Learning Representations*. 2017.

[147] Rohan Chabra et al. “Deep local shapes: Learning local sdf priors for detailed 3d reconstruction”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*. Springer. 2020, pp. 608–625.

[148] Songyou Peng et al. “Shape as points: A differentiable poisson solver”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 13032–13044.

[149] Alexandre Boulch and Renaud Marlet. “Poco: Point convolution for surface reconstruction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 6302–6314.

[150] Francis Bach. “Breaking the curse of dimensionality with convex neural networks”. In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 629–681.

[151] Meng Jia and Matthew Kyan. “Learning occupancy function from point clouds for surface reconstruction”. In: *arXiv preprint arXiv:2010.11378* (2020).

[152] Paul Bromiley. “Products and convolutions of Gaussian probability density functions”. In: *Tina-Vision Memo* 3.4 (2003), p. 1.

[153] 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.

[154] Thomas Lewiner et al. "Efficient implementation of marching cubes' cases with topological guarantees". In: *Journal of graphics tools* 8.2 (2003), pp. 1–15.

[155] David Stutz and Andreas Geiger. "Learning 3d shape completion from laser scan data with weak supervision". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1955–1964.

[156] Vincent Dumoulin et al. "Adversarially learned inference". In: *arXiv preprint arXiv:1606.00704* (2016).

[157] Kaleem Siddiqi et al. "Retrieving articulated 3-D models using medial surfaces". In: *Machine vision and applications* 19.4 (2008), pp. 261–275.

[158] Shachar Fleishman, Daniel Cohen-Or, and Cláudio T Silva. "Robust moving least-squares fitting with sharp features". In: *ACM transactions on graphics (TOG)*. Vol. 24. 3. ACM. 2005, pp. 544–552.

[159] Peter Hemingway. "n-Simplex interpolation". In: *HP Laboratories Bristol, HPL-2002* 320 (2002), pp. 1–8.

[160] Yitong Xia et al. "Sinerf: Sinusoidal neural radiance fields for joint pose estimation and scene reconstruction". In: *arXiv preprint arXiv:2210.04553* (2022).

[161] D Caleb Rucker et al. "A mechanics-based nonrigid registration method for liver surgery using sparse intraoperative data". In: *IEEE transactions on medical imaging* 33.1 (2013), pp. 147–158.

[162] Ozan Oktay et al. “Biomechanically driven registration of pre-to intra-operative 3d images for laparoscopic surgery”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2013, pp. 1–9.

[163] Igor Peterlik et al. “Fast elastic registration of soft tissues under large deformations”. In: *Medical image analysis* 45 (2018), pp. 24–40.

[164] Ping-Lin Chang et al. “Robust real-time visual odometry for stereo endoscopy using dense quadrifocal tracking”. In: *International Conference on Information Processing in Computer-Assisted Interventions*. Springer. 2014, pp. 11–20.

[165] Andrew D Speers et al. “Fast and accurate vision-based stereo reconstruction and motion estimation for image-guided liver surgery”. In: *Healthcare technology letters* 5.5 (2018), pp. 208–214.

[166] Jarrod A Collins et al. “Improving registration robustness for image-guided liver surgery in a novel human-to-phantom data framework”. In: *IEEE transactions on medical imaging* 36.7 (2017), pp. 1502–1510.

[167] Jon S Heiselman, William R Jarnagin, and Michael I Miga. “Intraoperative correction of liver deformation using sparse surface and vascular features via linearized iterative boundary reconstruction”. In: *IEEE Transactions on Medical Imaging* 39.6 (2020), pp. 2223–2234.

[168] Sherif RZ Abdel-Misih and Mark Bloomston. “Liver anatomy”. In: *Surgical Clinics* 90.4 (2010), pp. 643–653.

[169] Fabian Isensee et al. “nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation”. In: *Nature methods* 18.2 (2021), pp. 203–211.

[170] Stefan Bauer et al. “A survey of MRI-based medical image analysis for brain tumor studies”. In: *Physics in Medicine & Biology* 58.13 (2013), R97.

[171] Sveinn Pálsson et al. “Predicting survival of glioblastoma from automatic whole-brain and tumor segmentation of MR images”. In: *arXiv preprint arXiv:2109.12334* (2021).

[172] Mikael Agn et al. “A modality-adaptive method for segmenting brain tumors and organs-at-risk in radiation therapy planning”. In: *Medical image analysis* 54 (2019), pp. 220–237.

[173] Donatas Sederevičius et al. “Reliability and sensitivity of two whole-brain segmentation approaches included in FreeSurfer–ASEG and SAMSEG”. In: *NeuroImage* 237 (2021), p. 118113.

[174] Ali Gooya et al. “GLISTR: glioma image segmentation and registration”. In: *IEEE transactions on medical imaging* 31.10 (2012), pp. 1941–1954.

[175] Klaudius Scheufele et al. “Coupling brain-tumor biophysical models and diffeomorphic image registration”. In: *Computer methods in applied mechanics and engineering* 347 (2019), pp. 533–567.

[176] Klaudius Scheufele, Shashank Subramanian, and George Biros. “Fully automatic calibration of tumor-growth models using a single mpMRI scan”. In: *IEEE Transactions on Medical Imaging* 40.1 (2020), pp. 193–204.

[177] Stelios K Kyriacou et al. “Nonlinear elastic registration of brain images with tumor pathology using a biomechanical model [MRI]”. In: *IEEE transactions on medical imaging* 18.7 (1999), pp. 580–592.

[178] Meritxell Bach Cuadra et al. “Atlas-based segmentation of pathological MR brain images using a model of lesion growth”. In: *IEEE transactions on medical imaging* 23.10 (2004), pp. 1301–1314.

[179] Stefan Bauer et al. “Atlas-based segmentation of brain tumor images using a Markov random field-based tumor growth model and non-rigid registration”. In: *2010 annual international conference of the IEEE engineering in medicine and biology*. IEEE. 2010, pp. 4080–4083.

[180] Ashraf Mohamed et al. “Deformable registration of brain tumor images via a statistical model of tumor-induced deformation”. In: *Medical image analysis* 10.5 (2006), pp. 752–763.

[181] Claudio Pollo et al. “Segmentation of brain structures in presence of a space-occupying lesion”. In: *Neuroimage* 24.4 (2005), pp. 990–996.

[182] J-P Thirion. “Image matching as a diffusion process: an analogy with Maxwell’s demons”. In: *Medical image analysis* 2.3 (1998), pp. 243–260.

[183] Amir Gholami et al. “A novel domain adaptation framework for medical image segmentation”. In: *International MICCAI Brainlesion Workshop*. Springer. 2018, pp. 289–298.

[184] Mobeen Ur Rehman et al. “BrainSeg-net: Brain tumor MR image segmentation via enhanced encoder–decoder network”. In: *Diagnostics* 11.2 (2021), p. 169.

[185] Guotai Wang et al. “Automatic brain tumor segmentation based on cascaded convolutional neural networks with uncertainty estimation”. In: *Frontiers in computational neuroscience* 13 (2019), p. 56.

[186] Fabian Isensee et al. “nnU-net for brain tumor segmentation”. In: *International MICCAI Brainlesion Workshop*. Springer. 2020, pp. 118–132.

[187] Bjoern H Menze et al. “The multimodal brain tumor image segmentation benchmark (BRATS)”. In: *IEEE transactions on medical imaging* 34.10 (2014), pp. 1993–2024.

[188] Shashank Subramanian et al. “Where did the tumor start? An inverse solver with sparse localization for tumor growth models”. In: *Inverse problems* 36.4 (2020), p. 045006.

[189] Shashank Subramanian et al. “Multiatlas calibration of biophysical brain tumor growth models with mass effect”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2020, pp. 551–560.

[190] Adrian V Dalca et al. “Unsupervised learning of probabilistic diffeomorphic registration for images and surfaces”. In: *Medical image analysis* 57 (2019), pp. 226–236.

[191] Hana LP Harpold, Ellsworth C Alvord Jr, and Kristin R Swanson. “The evolution of mathematical modeling of glioma proliferation and invasion”. In: *Journal of Neuropathology & Experimental Neurology* 66.1 (2007), pp. 1–9.

[192] Jana Lipková et al. “Personalized radiotherapy design for glioblastoma: Integrating mathematical tumor models, multimodal scans, and bayesian inference”. In: *IEEE transactions on medical imaging* 38.8 (2019), pp. 1875–1884.

[193] Meng Jia and Matthew Kyan. “Improving Intraoperative Liver Registration in Image-Guided Surgery with Learning-Based Reconstruction”. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 1230–1234.

[194] Chris A Cocosco et al. “Brainweb: Online interface to a 3D MRI simulated brain database”. In: *NeuroImage*. Citeseer. 1997.

[195] Michel Bilello et al. “Population-based MRI atlases of spatial distribution are specific to patient and tumor characteristics in glioblastoma”. In: *NeuroImage: Clinical* 12 (2016), pp. 34–40.

[196] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[197] Oula Puonti, Juan Eugenio Iglesias, and Koen Van Leemput. “Fast and sequence-adaptive whole-brain segmentation using parametric Bayesian modeling”. In: *NeuroImage* 143 (2016), pp. 235–249.

[198] Arno Klein and Jason Tourville. “101 labeled brain images and a consistent human cortical labeling protocol”. In: *Frontiers in neuroscience* 6 (2012), p. 171.

[199] Lee R Dice. “Measures of the amount of ecologic association between species”. In: *Ecology* 26.3 (1945), pp. 297–302.

[200] Shigeru Muraki. “Volumetric shape description of range data using “Blobby Model””. In: *SIGGRAPH Comput. Graph.* 25.4 (1991), pp. 227–235.

[201] Dmitrii Kochkov et al. “Machine learning–accelerated computational fluid dynamics”. In: *Proceedings of the National Academy of Sciences* 118.21 (2021).

[202] Tianchang Shen et al. “Deep Marching Tetrahedra: a Hybrid Representation for High-Resolution 3D Shape Synthesis”. In: *arXiv preprint arXiv:2111.04276* (2021).