



BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

GENERATIVE MODELS FOR 3D SHAPE COMPLETION

GENERATIVNÍ MODELY PRO DOPLNĚNÍ 3D TVARU

MASTER'S THESIS

DIPLOMOVÁ PRÁCE

AUTHOR

AUTOR PRÁCE

Bc. PETER ZDRAVECKÝ

SUPERVISOR

VEDOUCÍ PRÁCE

Ing. TIBOR KUBÍK

BRNO 2024

Master's Thesis Assignment



154507

Institut: Department of Computer Graphics and Multimedia (DCGM)
Student: **Zdravecký Peter, Bc.**
Programme: Information Technology and Artificial Intelligence
Specialization: Computer Vision
Title: **Generative Models for 3D Shape Completion**
Category: Computer vision
Academic year: 2023/24

Assignment:

1. Explore current methods for analyzing 3D shapes using neural networks (multi-view networks, graph networks, point cloud processing networks, etc.).
2. Study generative models to fill missing parts of images and 3D shapes.
3. Prepare a dataset for your experiments.
4. Design a suitable method to automatically fill in the missing parts of a 3D shape.
5. Implement the method using existing libraries for designing and training neural networks.
6. Evaluate the solution qualitatively and quantitatively. Discuss possible extensions.
7. Present your work (its objectives, proposed method and achieved results) in the form of a poster or video.

Literature:

- Chu *et al.*, DiffComplete: Diffusion-based Generative 3D Shape Completion, 2023, <https://arxiv.org/abs/2306.16329>.
- Lyu *et al.*, A Conditional Point Diffusion-Refinement Paradigm for 3D Point Cloud Completion, 2022, <https://arxiv.org/abs/2112.03530>.

Requirements for the semestral defence:

- First 3 points of the assignment, and partially point 4.

Detailed formal requirements can be found at <https://www.fit.vut.cz/study/theses/>

Supervisor: **Kubík Tibor, Ing.**
Head of Department: Černocký Jan, prof. Dr. Ing.
Beginning of work: 1.11.2023
Submission deadline: 17.5.2024
Approval date: 9.11.2023

Abstract

In many real-world scenarios, scanned 3D models contain missing parts due to occlusion, scanning errors, or the incomplete nature of the data itself. The goal of this work is to create an automated process for 3D shape completion using a supervised deep learning-based method. The proposed solution is based on the prior work of DiffComplete, which uses a diffusion-based model operating over distance field representation and handles the task as a generative problem. The results showed a high capability of this model with an 81.6 IoU metric on the custom-prepared test set of furniture objects. The model also demonstrates strong generalization capabilities on shapes that are out of the training distribution (average 70.9 IoU metric). Apart from more detailed data-centric experiments, this work further extends current state-of-the-art in two ways. Firstly, it addresses the most crucial shortcoming, expensive computation, by processing the input in a low-resolution domain. Secondly, it utilizes user input (Region of Interest), which gives the user more control over generation in ambiguous scenarios.

Abstrakt

Naskenované 3D modely často trpia chybami kvôli oklúzii, skenovacím nedostatkom alebo neúplnosti samotného modelu. Cieľom tejto práce je vyvinúť automatizovaný proces na doplnenie chýbajúcich častí 3D tvarov prostredníctvom hlbokého učenia. Navrhované riešenie vychádza z predchádzajúcej práce DiffComplete, ktorá využíva generatívny difúzny proces na vyplnenie chýbajúcich častí 3D tvarov. Úloha sa takto vníma ako generatívny problém. Výsledky preukazujú vysokú účinnosť tohto modelu s IoU skóre dosahujúcim 81,6 na konkrétnej testovacej sade pozostávajúcej z tvarov nábytku. Model navyše úspešne generalizuje aj na tvary, ktoré nie sú zahrnuté v trénovacej sade, dosahujúc priemerné IoU skóre 70,9. Práca okrem popisu dátovo orientovaných experimentov obohacuje súčasnú problematiku vyplňania 3D útvarov dvoma spôsobmi. Po prvé rieši najväčšiu limitáciu, výpočetnú náročnosť, spracovaním vstupu v priestore s nízkym rozlíšením. Po druhé využíva užívateľský vstup (vo forme oblasti záujmu), čo umožňuje užívateľovi lepšie ovládať proces generácie v nejednoznačných situáciách.

Keywords

shape completion, geometric deep learning, generative models, 3D shapes, diffusion models, DiffComplete, region of interest

Klíčové slová

doplnenie 3D tvaru, hlboké neuronové siete, generatívne modely, 3D tvary, difúzne modely, DiffComplete, oblasť záujmu

Reference

ZDRAVECKÝ, Peter. *Generative Models for 3D Shape Completion*. Brno, 2024. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Tibor Kubík

Rozšírený abstrakt

Úvod a cieľ práce

S narastajúcou dostupnosťou nástrojov, ktoré dokážu generovať 3D dáta z fyzických tvarov rastie aj potreba riešiť limitácie sprevádzajúce tieto nástroje. Naskenovanému 3D objektu môžu chýbať niektoré podstatné časti (respektíve obsahuje diery), ktoré nutno doplniť, čo predstavuje jeden z kritických problémov 3D skenovania. Táto problematika sa nazýva “vypĺňanie útvarov”. Existujúce metódy fungujú celkom dobre pri vypĺňaní malých oblastí alebo rovných plôch s málo detailami. Ideálna metóda na vypĺňanie útvarov by mala byť schopná zvládnuť aj väčšie chýbajúce oblasti a komplexnú geometriu, čo je bežný prípad v reálnych situáciach.

V posledných rokoch dosahujú techniky hlbokého učenia výnimočné úspechy v rôznych úlohách, ktoré sa týkajú analýzy 2D obrazových dát. To ukazuje, že ich schopnosť efektívne extrahovať dôležité informácie z vstupných dát by mohla efektívne slúžiť na vývoj automatických systémov, ktoré by spracovávali 3D polygonálne modely. Avšak aplikácia týchto metód na 3D neeuclidovské dáta nie je jednoduchá, pretože vlastnosti polygonálnych modelov sa výrazne líšia od vlastností pravidelne vzorkovaných obrázkov. Preto je nevyhnutné prispôbiť typické operácie, ako je konvolúcia, na nepravidelné povrchy týchto modelov.

Cieľom tejto práce je preskúmať techniky hlbokého učenia na analýzu 3D dát geometricky reprezentovaných, ako polygonálne modely. Praktickým cieľom je ďalej návrh a aplikovanie týchto metód na úlohu automatického doplnenia 3D tvarov (viz obr. 1).



Obrázok 1: Príklad vstupu v podobe neúplného tvaru (a), vygenerovaného výstupu z navrhovaného riešenia (b) a originálneho tvaru (c). Cieľom tejto metódy je doplniť chýbajúce časti neuplného 3D tvaru.

Návrh riešenia

Navrhované riešenie využíva difúzny model s použitím reprezentácie uloženú v pravidelnej mriežke (truncated signed distance field). Úloha sa vníma, ako generatívny problém, teda vytvorenie úplného tvaru z neúplného vstupu. Riešenie je založené na metóde *Diff-Complete* [10]. Cieľom je vylepšiť nedostatky a preskúmať potenciál metódy *Diff-Complete*. Metóda navrhovaná v tejto práci je rozšírená o nový prístup, ktorý spočíva v spracovávaní v priestore s nízkym rozlíšením. Výstupu sa potom zvýši rozlíšenie, aby sa dosiahla jemnejšia geometria s krajšími detailami. Spracovávanie v nízkom rozlíšení by zároveň malo

šetriť výpočetnými prostriedkami a znížiť čas inferencie. Dodatočné rozšírenia prezentovanej metódy umožňujú využiť užívateľský vstup vo forme Oblasti záujmu pre efektívnejšie riadenie procesu vyplňania. Takto možno napraviť veľkú časť zlyhaní základného riešenia pri dopĺňaní chýbajúcich častí modelov.

Experimenty a dosiahnuté výsledky

V rámci výsledkov sa v oboch experimentálnych osiach potvrdil dobrý výkon predstavovaného riešenia. Proces vyplňania tvarov vykazuje pôsobivé výsledky. V rámci dátovej sady (datasetu) s modelmi nábytku Objaverse dosahuje v priemere skóre prieniku nad zjednotením (IoU) s hodnotou 81,62, Chamferovu vzdialenosť (CD) 3,53, a priemerná absolútna chyba (L1) 0,026.

Model navyše ukazuje robustnú schopnosť generalizovania. Na tvaroch mimo trénovej sady dosahuje priemerne 70,90 IoU skóre, 5,28 CD a 0,047 L1 skóre. Integráciou užívateľského vstupu sa podstatne zjednodušil proces vyplňania tvarov, čo viedlo k 84,7 IoU, zníženým hodnotám CD (2,86) a L1 (0,018) na testovacom datasete. Výsledky týkajúce sa tvarov mimo trénovacieho rozloženia, priemerné skóre sú 76,81 IoU, 4,13 CD, 0,033 L1.

Ďalšie experimenty sa sústredili na vyplňanie tvarov vo vysokom rozlíšení. V tejto oblasti sa výrazne preukázali výpočetné nároky, obzvlášť pri veľkosti mriežky $64 \times 64 \times 64$. Napriek týmto prekážkam, základné riešenie stále dokázalo vyplňať tvary, aj keď sa niekedy dosiahli zašumené výsledky, kvôli neúplnej konvergencii počas trénovania modelu. L1 metrika v rámci Objaverse datasetu bola 0,058, čo indikuje dvojnásobnú chybovosť oproti základnému modelu, ktorý spracovával nízke rozlíšenia.

Aby sa tieto nedostatky zmiernili, v experimentoch sa skúmali aj techniky založené na super-rozlíšení a stratégia zahŕňajúca efektívne spracovávanie v nízkom rozlíšení. Experiment so super-rozlíšením bol založený na prístupe využívajúcom dve neurónové siete, čo malo za cieľ zjednodušiť konvergenciu a zvýšiť presnosť, avšak niekedy sa tu narazilo na šum a nepresnosti. L1 chyba sa trochu znížila na 0,053, čo je ale stále dosť oproti výsledkom v nízkom rozlíšení.

Efektívne spracovávanie v nízkom rozlíšení predstavuje nový prístup, ktorý znižuje rozlíšenie vstupu pre spracovávanie a potom mu spätne zvyšuje rozlíšenie. Týmto sa dosiahli hladšie a viac vizuálne pôsobivé výsledky, s L1 skóre o hodnote 0,032. Táto metóda však má problém so zachovávaním jemných detailov a s generalizáciou do nových kategórií, čo zdôrazňuje kritickosť diverzifikácie trénovacieho datasetu.

Z akademického hľadiska táto práca rozširuje základné princípy ustanovené v predošlom výskume na DiffComplete modeli. Hlavne skúma detaily v rámci vyplňania tvarov naprieč rôznymi dátovými sadami a rozlíšeniami. Taktiež analyzuje vplyv užívateľského vstupu vo forme špecifikovania oblasti záujmu, v rámci ktorej má model priorizovať, kde bude dopĺňať. Ďalej popisuje výpočetnú náročnosť spojenú s vyplňaním tvarov vo vysokom rozlíšení. Práca aj navrhuje inovatívne prístupy, akými možno zmierniť problémy spojené s tvarmi s vysokým rozlíšením, ako napríklad techniky super-rozlíšenia a efektívne spracovávanie v nízkom rozlíšení. Schopnosť vyplňať 3D tvary má priame využitie v počítačovej grafike, 3D modelovaní, rozšírenej realite (AR), a podobne. Zo zlepšenia v metódach vyplňania 3D tvarov môžu výrazne profitovať odvetvia, ktorým závisí na presných 3D rekonštrukciách, ako sú zábavný priemysel, architektúra, medicína a ďalšie.

Generative Models for 3D Shape Completion

Declaration

I hereby declare that this Master's thesis was prepared as an original work by the author under the supervision of Ing. Tibor Kubík. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Peter Zdravecký
May 3, 2024

Acknowledgements

I would like to thank my supervisor for his essential support in completing this work. Computational resources were provided by the e-INFRA CZ project (ID:90254), supported by the Ministry of Education, Youth and Sports of the Czech Republic.

Contents

1	Introduction	3
2	3D Data Representations and Deep Learning Connection	4
2.1	Polygonal Mesh	5
2.2	Point Cloud	5
2.3	Voxel Grid	7
2.4	Truncated Signed Distance Field	7
2.5	Multi-View Representation	9
2.6	Neural Implicit Representation	10
3	Operations on 3D Data and Processing	13
3.1	Spatial Transformations	13
3.2	Boolean Operations	14
3.3	Advanced Operations	14
4	Neural 3D Shape Analysis Methods	18
4.1	3D Shapes Generation and Completion	18
4.2	State-of-the-Art in 3D Shape Completion Methods	22
5	Proposed Solution for 3D Shape Completion using Deep Neural Networks	30
5.1	Problem Definition	30
5.2	Dataset Preparation Pipeline: Smashing the Objects	31
5.3	Shape Completion Pipeline: Filling Holes via Diffusion Process	33
6	Implementation Details	39
6.1	Technologies	39
6.2	Dataset Specifications	39
6.3	Specification of Training Configurations	44
7	Conducted Experiments and Achieved Results	46
7.1	Evaluation Metrics	46
7.2	Evaluation Axis 1: Completion Ability of the Proposed Solution	47
7.3	Evaluation Axis 2: Focus on Higher Resolution Results	55
7.4	Post-Processing Enhancements	59
7.5	Summary of Results	60
7.6	Future Work	62
8	Conclusion	64

Bibliography	65
A Contents of the Included Storage Media	73
B Poster	74

Chapter 1

Introduction

With the increasing accessibility of tools that generate 3D data from physical shapes, there is a need for solutions to address the potential drawbacks these tools may present. One of the critical issues is that the scanned 3D model may have missing sections that need repair. The problem of *filling the holes* is called shape completion. The existing method performs well for filling small areas or flat surfaces without extra details. The ideal shape completion method should be able to handle larger missing areas and complex geometry, which is usually the case for real-world shapes.

This work examines the application of deep learning techniques for the shape completion task. The proposed solution uses a diffusion-based model and handles the task as a generative problem to create a complete shape from an incomplete one, using a TSDF representation. The solution is based on the *DiffComplete* [10] method. The aim is to improve the shortcomings and explore the potential capabilities of the DiffComplete method. The method is further extended with a novel approach of processing in low-resolution space followed by an upscaling process to obtain high-resolution results. Processing in a low-resolution space should save computational resources and speed up inference time. The additional enhancement incorporates user input in the form of a Region of Interest to more effectively guide the completion process and rectify failure cases from the baseline solution.

The method was extensively evaluated on multiple datasets, including out of distribution sets. The effectiveness of the proposed solution was evaluated using three metrics: intersection over union, Chamfer distance, and mean absolute error. The baseline solution showed high shape completion capability, with an 81.6 IoU, CD 3.53, and L1 0.026 metric score in the test dataset. The model also demonstrates strong generalization capabilities on shapes that are not part of the training distribution (average 70.9 IoU, CD 5.28, and L1 0.047 metric score). Those scores are further improved by incorporating a Region of Interest. Lastly, the strength of the proposed approach lies in its processing in a low-resolution domain, which enhances the inference speed and reduces the computational demands, given that diffusion models are challenging in this respect.

The text of this thesis is structured as follows. Chapter 2 presents the key aspects of various 3D representations. Chapter 3 deals with operations used in a work with 3D data. Then, Chapter 4 provides an overview of the literature addressing the analysis of 3D shapes, followed by shape generation and completion, with a review of current approaches for shape completion. The proposed methods are then presented in Chapter 5. Chapter 6 focuses on details related to implementation and datasets used, to facilitate the replication of the work. The experiments and results are summarized in Chapter 7.

Chapter 2

3D Data Representations and Deep Learning Connection

Today, 3D data are considered crucial across various domains, such as *computer-aided design (CAD)*, medical imaging, and industries emphasizing visualization, like the gaming industry, animation, or cinema. 3D data can be derived from various sources, for instance, through the digitization of real-world objects or by modeling using specialized software, such as *Blender*. The discretization of the real-world shapes utilizing sensors is another approach to obtain 3D shapes in digital form. However, this method may introduce noise into the result data. Another approach that is used in medical imaging is to capture 3D shapes of patients' bodies (e.g., teeth) as multiple 2D images (*slices*).

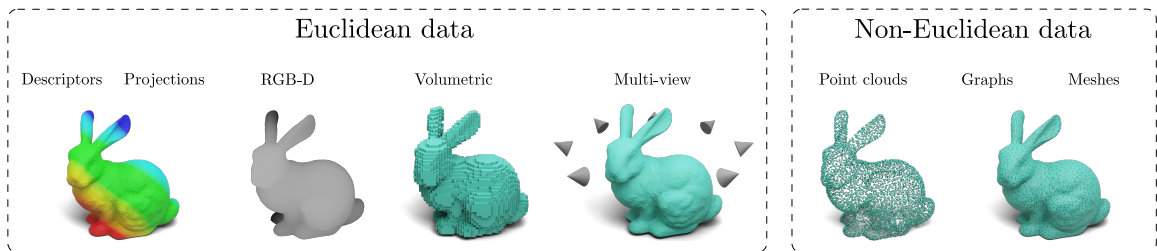


Figure 2.1: Taxonomy of 3D representations. Visual illustration of organizing 3D data representations into Euclidean and non-Euclidean structures.

There is no flawless 3D representation of the data without any constraints, indicating that every representation comes with its own set of uses, benefits, and drawbacks. The representations can be categorized into Euclidean and non-Euclidean. Data points in Euclidean representations are arranged following the principles of Euclidean geometry, typically in a regular grid-like pattern. In contrast, non-Euclidean representations do not adhere to regular grid structures, requiring more complex methods for processing. Ahmed *et al.* [2] conducted a detailed review of the various 3D data representations. Figure 2.1 illustrates a classification of the 3D representation, highlighting the specified categories. Point clouds and polygonal meshes are characterized by non-Euclidean traits. Conversely, voxel grids, truncated signed distance fields, and multi-view representations fall under Euclidean representations because of their regular data structure.

2.1 Polygonal Mesh

A polygonal mesh, denoted as \mathcal{M} , is composed of vertices, edges, and faces that together define the surface of a 3D object:

$$\mathcal{M} = (V, E, F), \quad (2.1)$$

where V denotes the set of vertices, E the set of edges, and F the set of faces. The set of vertices V includes points within a 3D space, each distinguished by a specific coordinate, represented as:

$$V = \{v_i \in \mathbb{R}^3 \mid v_i = (x_i, y_i, z_i)\}. \quad (2.2)$$

Edges E are defined by pairs of vertices that connect to form the linear boundaries of the faces of the mesh. An edge between the vertices v_i and v_j is denoted as:

$$E = \{e_{ij} \mid e_{ij} = (v_i, v_j), v_i, v_j \in V\}. \quad (2.3)$$

The face set F is formed of sequences of edges that enclose parts of the surface of the mesh. Each face f_k is defined as:

$$F = \{f_k \mid f_k = e_{i_1 i_2}, e_{i_2 i_3}, \dots, e_{i_m i_1}, e_{ij} \in E\}. \quad (2.4)$$

For every face f_k , there is a closed loop of edges e_{ij} , with each e_{ij} being a member of the edge set E .

The connectivity of the mesh is demonstrated by the relationships among its vertices, edges, and faces. These relationships are often represented through a connection list or an adjacency matrix. At a local level, the geometry of the mesh reflects a segment of Euclidean space. It adheres to traditional measures such as distance, angles, and flatness. However, meshes exhibit non-Euclidean characteristics on a larger scale.

Mesh processing and analysis are often not ideal, mainly due to the non-uniform nature of the mesh, which differs from the standardized grid layouts designed for traditional deep learning frameworks. One of the most used approaches involves transforming 3D meshes into a **graph-based format**. Mapping mesh vertices to graph vertices and their connections to edges allow one to use *graph neural networks (GNNs)* [73]. These networks enable exploration of the structures and characteristics presented in the 3D data.

One significant advantage of this representation is its ability to provide precise control over the geometry of shapes. This precision is crucial for creating detailed and precise 3D models, which are essential in medical imaging, architectural design, and video game development. However, the complex structure of polygonal meshes poses significant computational challenges, especially for models with intricate details. Storage and manipulation of such models require considerable resources. Furthermore, the irregular nature of the meshes complicates the application of conventional machine learning techniques, as mentioned before.

2.2 Point Cloud

Mathematically, a point cloud \mathcal{P} is defined as a set of points p_i , where each point is a vector in a 3D space, potentially enhanced with additional attributes:

$$\mathcal{P} = \{p_i \mid p_i \in \mathbb{R}^{3+}\}. \quad (2.5)$$

Each point p_i is represented by its coordinates (x_i, y_i, z_i) , indicating its position in the 3D space, along with optional attributes such as normals (n_{xi}, n_{yi}, n_{zi}) and colors (r_i, g_i, b_i) :

$$p_i = (x_i, y_i, z_i, n_{xi}, n_{yi}, n_{zi}, r_i, g_i, b_i, \dots). \quad (2.6)$$

Point clouds lack connectivity or adjacency information between points, making them less suitable for global analyses in Euclidean space. However, on a local scale, subsets of point clouds can exhibit Euclidean characteristics. These subsets, known as neighborhoods, are identified through various approaches, such as fixed-radius searches and k-nearest neighbors (k-NN), among others (see Figure 2.2). In a fixed-radius search, the neighborhood of a point encompasses all points within a certain distance. This approach establishes a Euclidean space where distance metrics are consistent, even with transformations like translations and rotations. In contrast, the k-NN method determines the neighborhood of a point by the k closest points, regardless of their absolute distances. This provides the flexibility to adjust to various point densities.

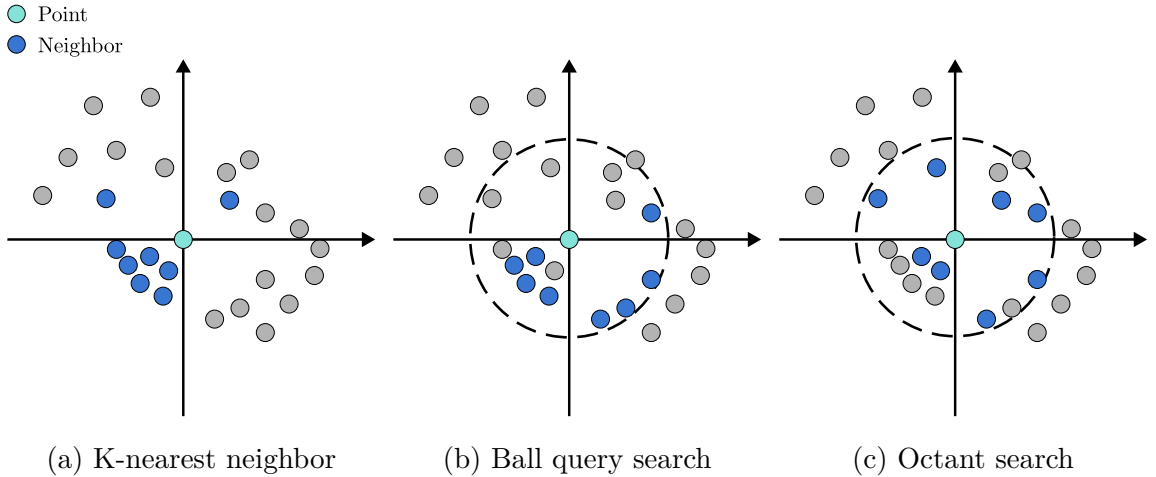


Figure 2.2: Illustration of various point cloud neighborhoods. The resulting points that form the neighborhood are determined by the algorithm employed. The ball query searching algorithm selects points randomly from a spherical area. Recreated from [4].

Point clouds are often produced by scanning physical objects with a sampling density sufficient to preserve the object’s geometry. However, scanning equipment can introduce inaccuracies and noise into the collected data. As a result, additional processing of the point clouds is required to derive valuable information appropriate for further machine learning tasks. Operations such as outlier removal, smoothing, and normal vector calculation require significant computational effort and may not consistently produce accurate results.

This 3D data representation has become a viable choice for modern machine learning solutions. Algorithms tailored for point clouds often aim to capture intricate geometric features within a broader context, maintaining robustness against inherent irregularities. Techniques such as PointNet [51] and its subsequent improvement, PointNet++ [52], illustrate this strategy by directly processing point cloud data. Those techniques are used to identify essential features of both local groupings and the overall configuration.

2.3 Voxel Grid

The voxel grid representation divides the 3D space into a three-dimensional array of voxels, where each voxel is the smallest box-shaped unit that can distinguish parts of a 3D object. Mathematically, a voxel grid is a 3D scalar field \mathcal{G} defined on a discrete domain. Here, each voxel v_{ijk} corresponds to a value at the discrete coordinates (i, j, k) within the grid:

$$\mathcal{G} = \{v_{ijk} \mid v_{ijk} \in \mathbb{R}, (i, j, k) \in \mathbb{Z}^{3+}, 0 \leq i < I, 0 \leq j < J, 0 \leq k < K\}. \quad (2.7)$$

Every voxel v_{ijk} is linked to a specific 3D space position, usually defined by the center point of the voxel. It can contain various types of information, color, density, or occupancy in binary voxel grid ($d \in \{0, 1\}$), for example. I , J and K represent the resolution of the volume grid and v_{ijk} can be denoted as:

$$v_{ijk} = (x_i, y_j, z_k, \alpha_{ijk}, \dots), \quad (2.8)$$

where (x_i, y_j, z_k) denotes the 3D position of the voxel and α_{ijk} includes additional attributes such as density or color.

The structured nature of voxel grids makes them compatible with conventional convolution operations, extended from 2D to 3D, for processing volumetric data. In this context, convolution involves moving a 3D kernel across the input grid to create feature maps that reflect spatial hierarchies, for 3D tasks such as object detection, segmentation, and classification. The 3D convolution operation for a voxel v_{ijk} is mathematically given by:

$$f(v_{ijk}) = \sum_{a=-A}^A \sum_{b=-B}^B \sum_{c=-C}^C w_{abc} \cdot v_{i+a, j+b, k+c}, \quad (2.9)$$

where w_{abc} are the weights of the 3D kernel, with A , B , and C indicating the size of the kernel.

The strength of voxel-based representation lies in the ability to encapsulate a 3D object within a spatial grid. Certain operations, such as calculating the volume or determining the intersection with other objects, are much simpler. Nevertheless, voxel-based representations are limited by high memory demands and computational costs as a result of the three-dimensional nature of the data. This highlights the importance of efficient data structures such as octree or methods like sparse convolutions. Such approaches can help mitigate computational and memory costs and manage volumetric data on a large scale in practical applications.

2.4 Truncated Signed Distance Field

Curless *et al.* [11] introduced the *Signed Distance Function (SDF)* to reconstruct 3D shapes from range images, leading to the *Truncated Signed Distance Field (TSDF)* concept. SDF is defined as a continuous function. The signed distance reflects how far a point is from the nearest object’s surface; it is positive in front of an object (free space) and negative behind it (inside the object). TSDF is presented as a discretization of the signed distances to a volumetric grid. As an example, the following text describes the method of obtaining TSDF from range images [68].

The method uses a grid of voxels, each with a center \mathbf{x} and two main attributes: the signed distance $sdf_i(\mathbf{x})$ and the weight $w_i(\mathbf{x})$. This signed distance is calculated as follows:

$$sdf_i(\mathbf{x}) = \text{depth}_i(\text{pic}(\mathbf{x})) - \text{cam}_z(\mathbf{x}), \quad (2.10)$$

where $\text{depth}_i(\text{pic}(\mathbf{x}))$ measures the depth from the camera to the object surface along the viewing ray intersecting \mathbf{x} , while $\text{cam}_z(\mathbf{x})$ is the distance from the voxel to the camera along the optical axis, see Figure 2.3. To optimize computational efficiency and focus resources on areas crucial for representation, the SDF values are then truncated to a limit $\pm t$ prioritizing regions near the object’s surface and disregarding distant areas, leading to $tsdf_i(\mathbf{x})$:

$$tsdf_i(\mathbf{x}) = \max\left(-1, \min\left(1, \frac{sdf_i(\mathbf{x})}{t}\right)\right). \quad (2.11)$$

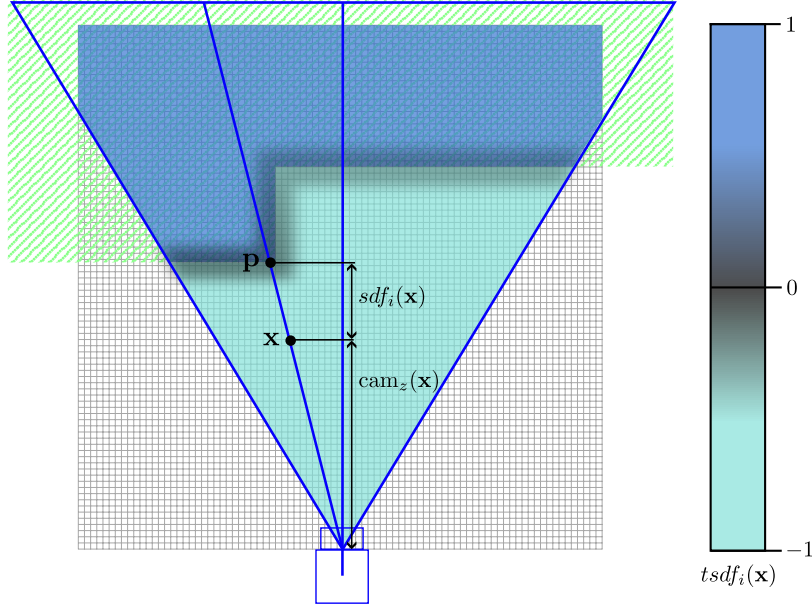


Figure 2.3: Example of a 2D Truncated Signed Distance Field. Figure shows a solid object depicted as a green grid, a camera with its field of view, an optical axis, and a ray, all highlighted in blue. The TSDF grid is shown, where unseen voxels appear white, and other voxels are distinguished by the color bar. The signed distance value of a voxel \mathbf{x} is established based on the depth of the corresponding surface point \mathbf{p} and the distance of the voxel to the camera, denoted as $\text{cam}_z(\mathbf{x})$. Recreated from [68].

The TSDF is constructed by integrating multiple observations into one model to enhance accuracy, combining data from various points of view through a weighted average, iteratively updated for each voxel:

$$TSDf_i(\mathbf{x}) = \frac{W_{i-1}(\mathbf{x})TSDf_{i-1}(\mathbf{x}) + w_i(\mathbf{x})tsdf_i(\mathbf{x})}{W_{i-1}(\mathbf{x}) + w_i(\mathbf{x})}, \quad (2.12)$$

$$W_i(\mathbf{x}) = W_{i-1}(\mathbf{x}) + w_i(\mathbf{x}).$$

where $TSDf_i(\mathbf{x})$ is the updated TSDF value in voxel \mathbf{x} after considering the i -th observation, combining the previous TSDF value $TSDf_{i-1}(\mathbf{x})$ weighted by $W_{i-1}(\mathbf{x})$, with the new observation $tsdf_i(\mathbf{x})$ weighted by $w_i(\mathbf{x})$. $W_i(\mathbf{x})$ updates the total weight by adding the weight of the new observation, ensuring that each measurement contributes according to

its reliability. This iterative process allows for a dynamic and accurate representation of the 3D shape, continuously refining the model as new data are acquired.

Truncated Signed Distance Field is distinguished by the precision in capturing the contours and features of objects within a volumetric space. This precision is particularly advantageous for deep learning tasks that require high spatial accuracy. However, similar to the challenges encountered with voxel-based representation, TSDF has problems with too much memory consumption and the computational effort needed to process a three-dimensional grid. For further processing or visualization, volumetric representations are reconstructed into mesh representations using isosurface extraction at the zero-level set. Methods such as *Marching Cubes (MC)* are traditionally employed for this purpose, but nowadays, techniques that utilize machine learning have been developed to perform this reconstruction, such as *Neural Dual Contouring (NDC)* [8]. An example of a reconstructed 3D shape, using the Marching Cubes [40] algorithm, can be found in Figure 2.4.

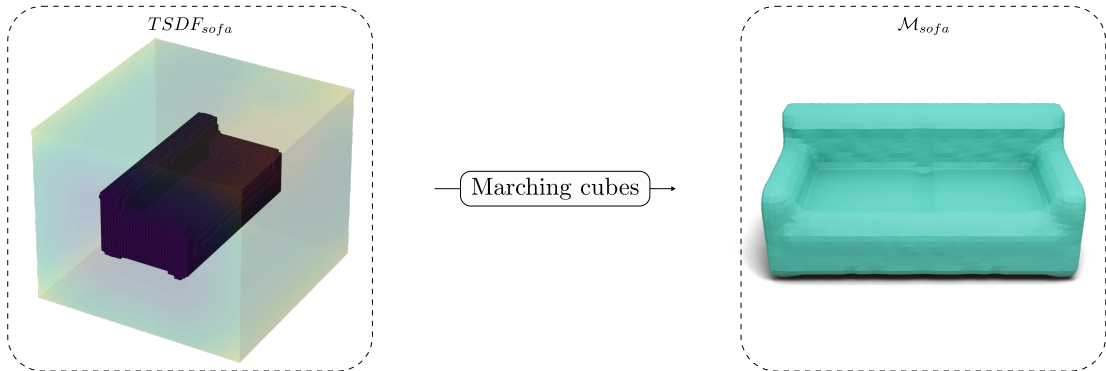


Figure 2.4: Example of a sofa 3D model represented as TSDF in three-dimensional space, utilizing the $64 \times 64 \times 64$ grid. The signed distance field value is indicated by the color of each voxel, with transparent cells representing empty spaces and solid cells indicating the object’s inside for clarity. On the right is a 3D mesh reconstructed using Marching cubes.

2.5 Multi-View Representation

Multi-view representation enhances the capture of three-dimensional shape data by merging multiple 2D images or views from various viewpoints, as illustrated in Figure 2.5. Multi-view representation offers a comprehensive understanding of the object’s structure. This is particularly beneficial for tasks that require fine-grained details and textures, such as object recognition, 3D reconstruction, and photogrammetry.

Mathematically, multi-view representation \mathcal{MV} is defined as a set of 2D images denoted as I_1, I_2, \dots, I_m , with each image I_i representing a unique perspective of the 3D object:

$$\mathcal{MV} = \{I_i \mid I_i \in \mathbb{R}^2; i = 1, 2, \dots, m\}. \quad (2.13)$$

Each image I_i in \mathcal{MV} is associated with a specific camera position and orientation, defined by extrinsic parameters that position the camera in the 3D space and intrinsic parameters that describe the camera’s internal characteristics, such as focal length or lens distortion:

$$I_i = \text{CaptureImage}(O, C_i, P_i), \quad (2.14)$$

where O represents the 3D object, C_i denotes the camera’s extrinsic parameters for view at the index i , and P_i describes the camera’s intrinsic parameters.

Multi-view representation is suitable for the application of powerful 2D image processing techniques, such as *Convolutional Neural Networks (CNNs)*, to analyze 3D objects. Standard CNN architectures can extract features from each view, which are then aggregated to form a cohesive representation of the 3D object:

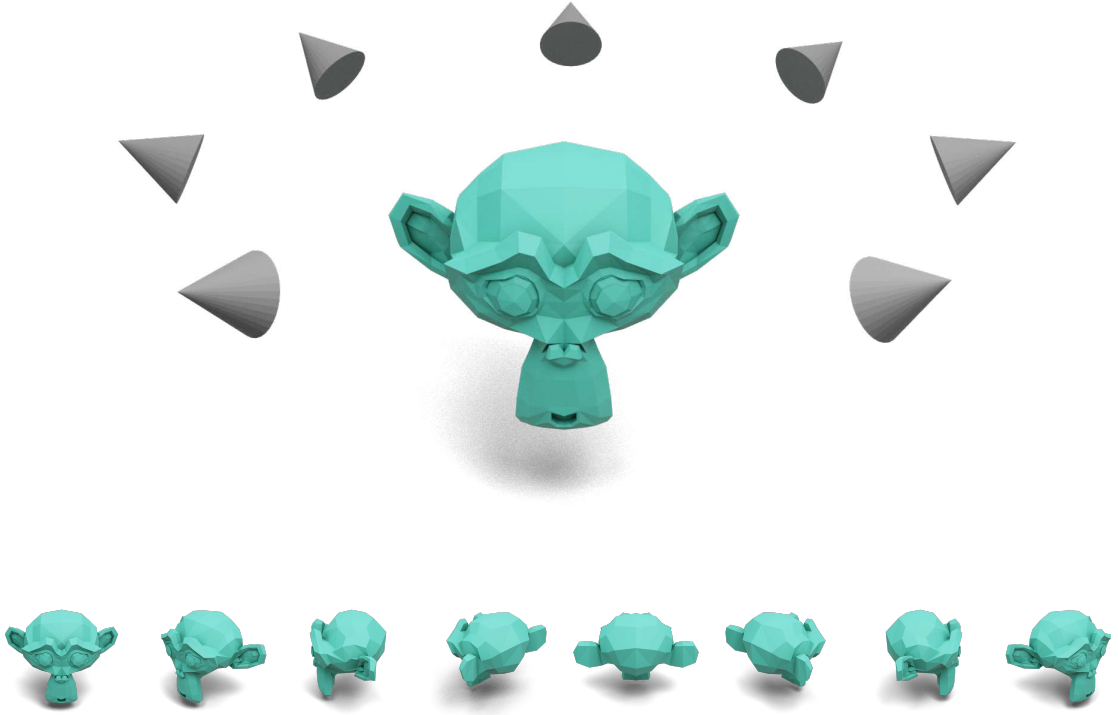


Figure 2.5: Multi-view representation of a 3D model. Notably, the top and bottom views of the model are missing, highlighting a potential limitation of Multi-view representation that requires addressing.

$$F_{3D} = \text{Aggregate}(F(I_1), F(I_2), \dots, F(I_m)), \quad (2.15)$$

where $F(I_i)$ denotes the features extracted from view with index i and $\text{Aggregate}(\cdot)$ represents a function that combines these features into a unified 3D representation.

Despite its advantages, Multi-view representation faces challenges related to occlusion, where parts of the object might be hidden in some (or in the worst case, all) views. To address this, many views are required for accurate representation of complex objects. The computational cost can increase significantly with the number of views, which requires efficient processing techniques to manage the resources effectively.

2.6 Neural Implicit Representation

Neural Implicit Representation offers a novel approach to 3D data representation, leveraging deep learning methods to encode 3D shapes in the weights of neural networks, instead of

explicitly storing geometric details, such as vertices or voxel values. These representations implicitly define the 3D surface or volume through a continuous function learned by a neural network. Mathematically, a neural implicit function \mathcal{F} for a 3D shape can be described as:

$$\mathcal{F} : \mathbb{R}^3 \times \theta \rightarrow \mathbb{R}, \quad \mathcal{F}(x, y, z; \theta) = s, \quad (2.16)$$

where (x, y, z) are the coordinates in a 3D space, θ represent the parameters of the neural network, and a value s is the output of the network. This value typically indicates the presence of the surface at that point. The most used approach within neural implicit representations is the use of Signed Distance Functions or Occupancy Fields, where the network outputs either the signed distance (see Figure 2.6) to the nearest surface or a binary occupancy value indicating whether a point is inside or outside the object:

$$\text{SDF}(x, y, z; \theta) = d, \quad \text{Occupancy}(x, y, z; \theta) = o, \quad (2.17)$$

where d is the signed distance and $o \in \{0, 1\}$ is the occupancy value.

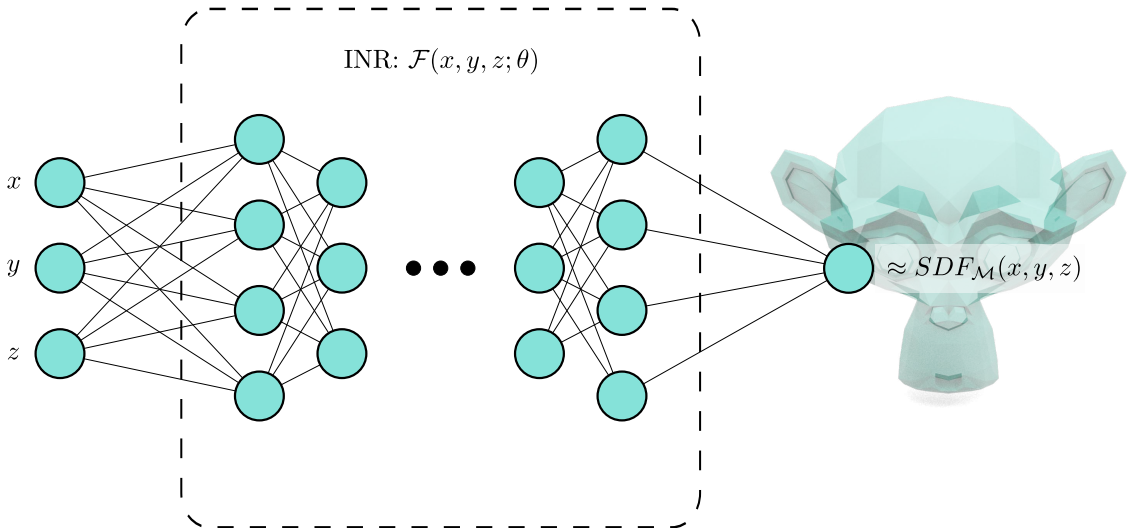


Figure 2.6: Example of neural implicit representation. Figure shows a neural network that takes a position input $x, y, z \in \mathbb{R}^3$ and produces a scalar output in the form of a Signed Distance Function (SDF). Figure recreated from [3].

Training neural networks to serve as implicit representations involves optimizing the network parameters θ so that the neural function \mathcal{F} accurately reflects the approximation of the 3D shape, represented as a mathematically defined implicit function. This is typically achieved by minimizing a loss function that measures the difference between the predicted and actual values (e.g., distance or occupancy) at various points in space.

The flexibility of neural implicit representations allows for high-resolution details without being bound to a fixed grid or topology, enabling smooth and continuous surfaces to be modeled with arbitrary complexity. Complex shapes, high-detail textures, and seamless transitions between features can take advantage of this approach.

In practical applications, Neural Implicit Representations are frequently integrated with coordinate-based neural networks and multilevel feature encoding. These techniques help with the management of large-scale data and intricate geometries. As illustrated in Figure 2.7, an object can be captured in various representations and then encoded as a neural

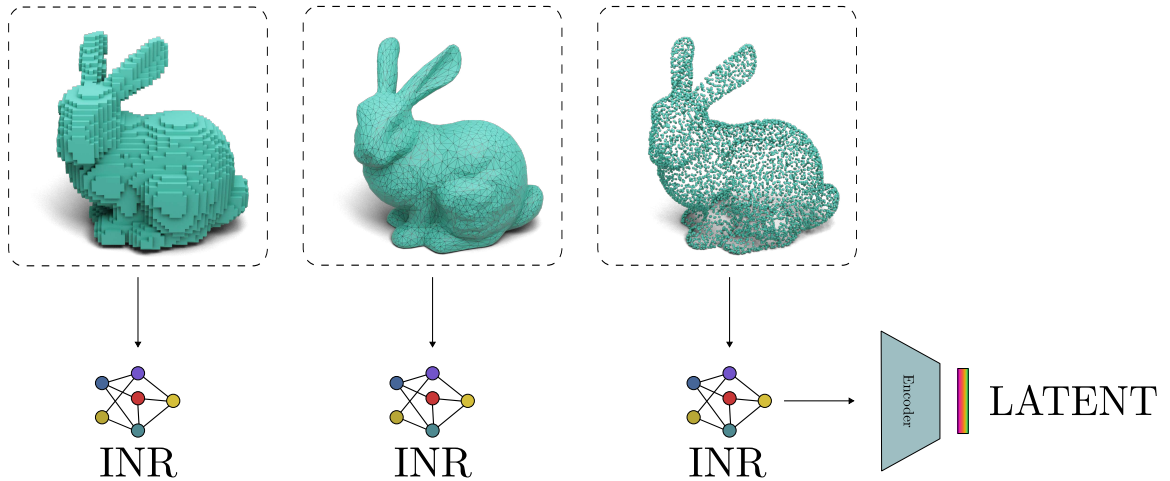


Figure 2.7: Neural implicit representation in latent space. The representation unify the basic representations by trying to encode them into one latent vector that can be used as input to the deep neural network. Figure recreated from [13].

implicit representation. Subsequently, a decoder transforms this into a latent vector suitable for downstream tasks such as classification, segmentation, and more.

Despite their advantages, implicit neural representations can be computationally intensive during training and require careful hyperparameter tuning. Moreover, the black-box nature of neural networks can sometimes lead to unpredictable behaviors or artifacts in captured shapes.

Chapter 3

Operations on 3D Data and Processing

Choosing the right format for various 3D data types and specific application needs is crucial. To efficiently manage these data, effective tools and methods are necessary. These methods include basic operations such as translation, rotation, and scaling. Additional advanced methods can rectify any shortcomings that may have arisen during the data generation process. For example, the scanning method can introduce noise into the model. The following sections present a selection of essential techniques necessary for understanding this thesis.

3.1 Spatial Transformations

The three primary transformations used to manipulate 3D data—translation, rotation, and scaling—are illustrated in Figure 3.1. Transformations are applied through matrices and matrix multiplication enables combining multiple transformations into a single matrix operation.

Translation moves a mesh model within the 3D space \mathbb{R}^3 , using the translation matrix \mathbf{T} . Rotation rotates the model around an axis, performed with the rotation matrix $\mathbf{R}_z(\theta)$. Scaling changes the size of the mesh model using the scaling matrix \mathbf{S} . All matrices are presented in Equation 3.1.

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{R}_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{S} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (3.1)$$

Basic 3D transformations are also used outside of geometric manipulation. In data augmentation, they can be utilized to create various orientations and sizes, which in turn help improve the generalization capability of learning frameworks. Another use case of 3D transformations is data normalization. Models can be aligned to their origin point and models' size can be normalized by scaling to a unit sphere. This normalization unifies the coordinate system for all shapes in datasets and enables algorithms to focus on the object's shape rather than its position or size, improving feature detection and model precision.

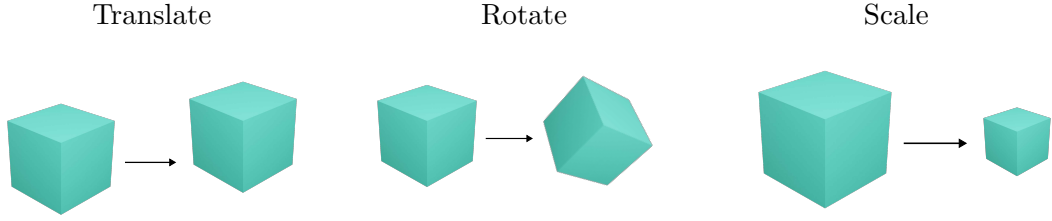


Figure 3.1: Visualization of standard affine 3D transformations. Figure shows an example of three spatial transformations on a 3D cube model: translation, rotation, and scale.

3.2 Boolean Operations

Boolean operations on 3D shapes are important in computational geometry and *Constructive Solid Geometry (CSG)* for creating and modifying 3D shapes using set-theory principles. Landier *et al.* [36] developed robust solutions for polygonal meshes to maintain the integrity of the mesh structure during these operations. To visually understand Boolean operations on 3D shapes, see Figure 3.2.

The task of performing Boolean operations on polygons is complex due to the need to preserve the manifold characteristics of the mesh [55]. These operations may often fail to execute due to an inconsistent or inadequate mesh topology. However, another representation offers an easier solution for Boolean operations, such as converting a mesh to a volumetric representation and applying per-voxel boolean operations. The following math definitions are formulated on binary voxel grids.

The **union** of two meshes combines their geometries into a single mesh. When used on polygonal meshes, union algorithms carefully combine vertices, edges, and faces to maintain the integrity of the initial shapes in the resulting mesh. This process aims to avoid creating non-manifold edges or faces that overlap. Union operation is described mathematically as follows:

$$\mathcal{A} \cup \mathcal{B} = \{x \mid x \in \mathcal{A} \vee x \in \mathcal{B}\}. \quad (3.2)$$

The **intersection** creates a mesh from the shared volume between two or more meshes. Algorithms for this type of operation carefully calculate the intersecting region and construct a mesh that accurately represents the overlap, while maintaining the topological consistency. Here is a mathematical definition of an intersection:

$$\mathcal{A} \cap \mathcal{B} = \{x \mid x \in \mathcal{A} \wedge x \in \mathcal{B}\}. \quad (3.3)$$

The **difference** subtracts one mesh from another, removing the subtracted volume. Algorithms for polygonal meshes carry out this process by removing the intersecting areas and correcting any irregularities in the final mesh, such as holes or isolated vertices, to ensure a coherent and functional mesh.

$$\mathcal{A} \setminus \mathcal{B} = \{x \mid x \in \mathcal{A} \wedge x \notin \mathcal{B}\}. \quad (3.4)$$

3.3 Advanced Operations

In addition to the fundamental transformations employed for spatial manipulation of 3D data and Boolean operations used for combining or differentiating shapes, there are also

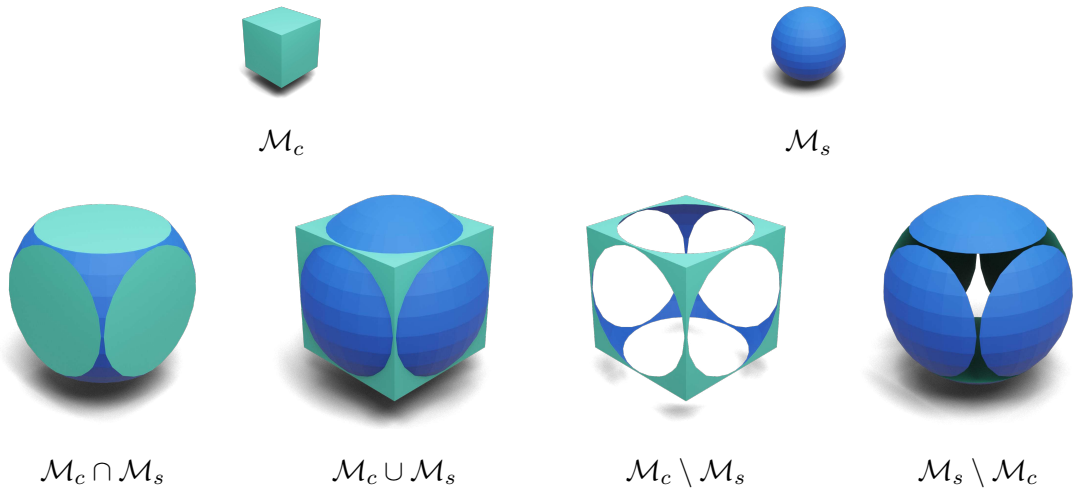


Figure 3.2: Visual examples of Boolean operations. From left to right, operations are: intersection, union, and two differences. Figure recreated from PyMesh documentation https://pymesh.readthedocs.io/en/latest/mesh_boolean.html.

more complex algorithms. This section will highlight and explain some algorithms particularly relevant to this work.

3.3.1 Surface Smoothing

Surface smoothing is a procedure in 3D modeling that seeks to improve models' visual and physical characteristics by reducing their surface noise and imperfections. This process has applications in producing attractive graphics in digital media, generating accurate models for scientific simulations, preparing models for 3D printing, etc.

Laplacian Smoothing

One of the most basic techniques for surface smoothing is Laplacian smoothing [18]. It iteratively modifies the position of each vertex based on the average positions of its neighboring vertices (see Figure 3.3), effectively distributing the vertices more uniformly across the surface. The operation follows the following equation:

$$\mathbf{v}_i^{q+1} = \frac{1}{N} \sum_{j=1}^N \mathbf{v}_j^q, \quad (3.5)$$

where \mathbf{v}_i^{q+1} denotes the new position of vertex i after the smoothing iteration $q + 1$, N represents the number of neighboring vertices, and \mathbf{v}_j^q is the position of neighboring vertex j in the current iteration q . Figure 3.4 demonstrates Laplacian smoothing on a 3D model of the Stanford bunny. Despite its simplicity and computational efficiency, Laplacian smoothing can lead to volume reduction and loss of detail. Enhanced methods such as the improved Laplacian smoothing proposed by Vollmer *et al.* [63] aim to preserve volume and features by repositioning vertices towards their original locations post-smoothing.

To address the limitations of Laplacian smoothing and meet specific requirements, various advanced techniques have been developed:

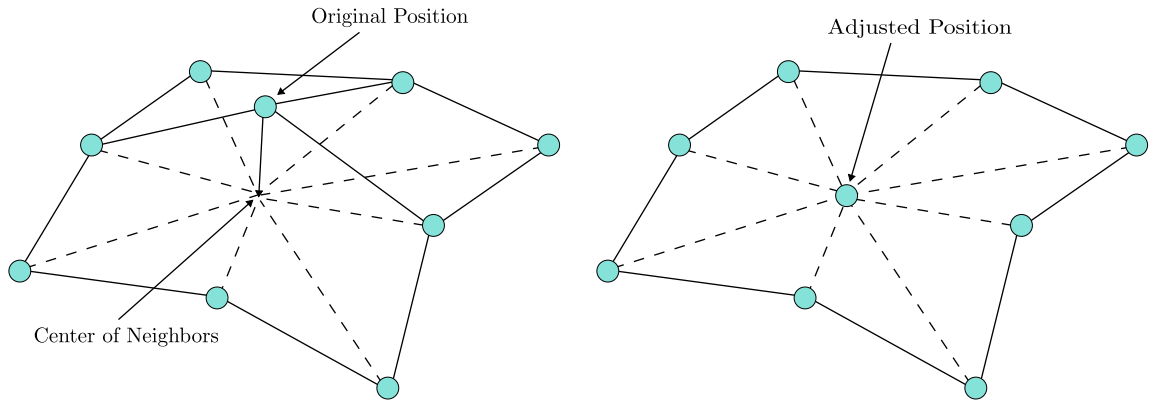


Figure 3.3: Example of mesh smoothing using Laplacian smoothing. Figure shows the vertex that is adjusted based on the centers of the neighbors. Figure recreated from [74].

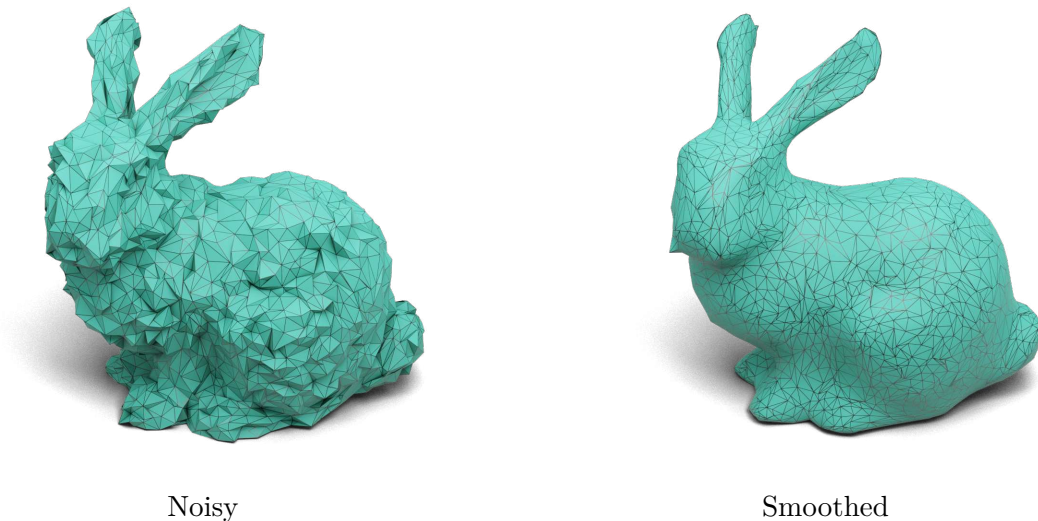


Figure 3.4: Example of Laplacian smoothing on noisy mesh. The smoothed version of the Stanford Bunny was obtained by 3 iterations of the smoothing algorithm.

- **Bilateral Smoothing:** Considers both the spatial separation and the difference in intensity, aiming to maintain edges while decreasing noise.
- **Taubin Smoothing:** Involves a series of iterations that switch between smoothing and shrinking steps to address the volume reduction that occurs with Laplacian smoothing.
- **Humphrey’s Classes Smoothing:** Combines Laplacian smoothing with curvature-based optimization to better retain geometric characteristics.

3.3.2 Hole Filling

Hole filling is used in 3D processing to repair and reconstruct incompletely formed regions, commonly called „holes“ in mesh structures [26] (see Figure 3.5).

It is vital to maintain the completeness of 3D models, especially those created from real-world data collection methods that might have gaps due to occlusion or scanning constraints. Various algorithms [16, 86] exist to address the challenge of hole filling, with strategies ranging from simple triangulation to more sophisticated methods that consider the curvature and topology of the surrounding mesh:

- **Triangulation Based:** Fills holes by creating new faces within the gap, typically employing methods such as Delaunay triangulation to minimize the creation of skinny or poorly shaped triangles.
- **Geometry Based:** Analyze the geometric features surrounding the hole to produce a fill that smoothly continues the existing curvature and surface normals, leading to a more visually coherent result.
- **Patch Based:** Use patches from other parts of the mesh or from a library of shapes to cover the hole in a way that matches the surrounding geometry as closely as possible.

These methods are effective for filling small holes, however, they are ineffective when dealing with larger missing parts of a model. As a result, this thesis seeks to create a machine learning approach that surpasses traditional methods and is suitable for larger-scale scenarios, while adequately maintaining the mesh features.

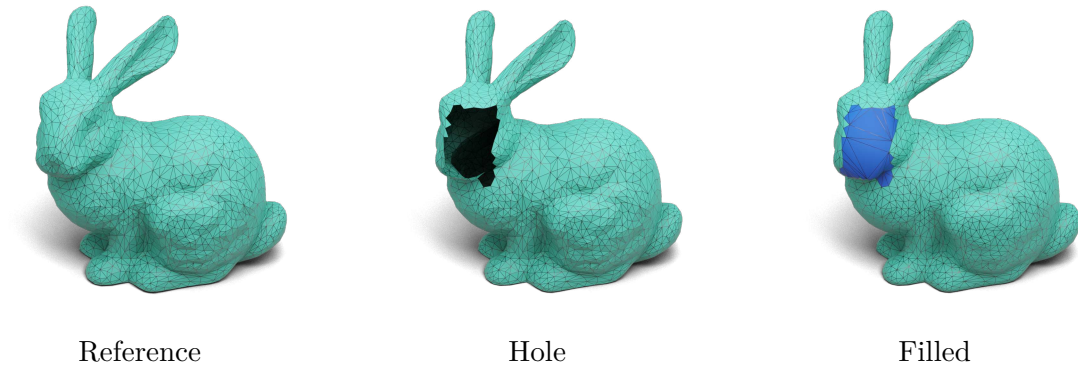


Figure 3.5: Example of mesh hole filling. Given mesh \mathcal{M} with missing geometry (middle image), the goal of the algorithms is to propose a patch (blue surface in rightmost image) that fills the surface in an accurate way.

Chapter 4

Neural 3D Shape Analysis Methods

Within the context of deep learning for 3D shapes, various methodologies align with the different types of 3D data representations. These are grouped into Euclidean methods, involving volumetric and multi-view representations, and non-Euclidean methods, which include point-based, edge-based, face-based, and graph approaches. An illustration of the taxonomy and the reference methods can be found in Figure 4.1. To dive deeper into the deep learning approach, refer to a study on deep geometry learning conducted by Yun *et al.* [75].

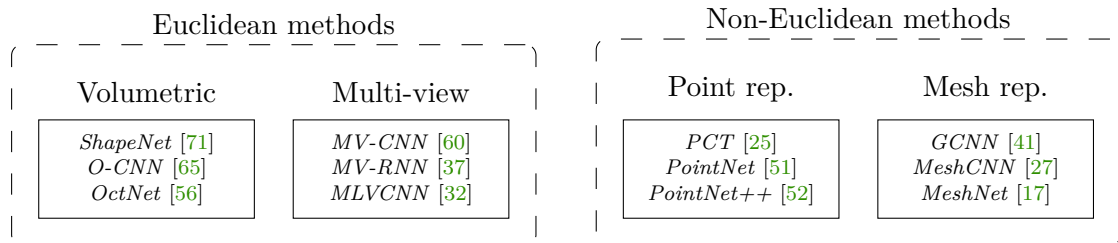


Figure 4.1: Euclidean/non-Euclidean taxonomy of deep learning 3D shape analysis. It is important to note that this is not an exhaustive listing of all papers in the field, but rather a curated collection of the most notable ones in each category.

4.1 3D Shapes Generation and Completion

This section explores the state-of-the-art complex approaches and advanced frameworks, providing information on their capabilities and possible uses. The main focus is on advanced methods that played a key role in expanding the capabilities of 3D shape **generation** and **completion**. For example, point embeddings [45] and conditional generative adversarial networks [82] allowed a more detailed generation of 3D point cloud shapes. On top of improving the visual quality, these techniques also contribute to the overall geometric accuracy of the generated models.

Another notable area of development is the hierarchical and part-based modeling approach. This method, as seen in studies such as the one by Li *et al.* [39], focuses on understanding and generating 3D shapes considering their constituent parts. This approach not

only aids in generating more realistic models, but also provides a deeper understanding of the structural composition of complex shapes.

Research in the field of integration of implicit and explicit representations of shapes, such as the work of Poursaeed *et al.* [50] demonstrates the potential of combining different shape representations to achieve more comprehensive and detailed models.

Shape completion methods have also made significant progress. One notable advancement is the use of deep neural networks, specifically transformer-based networks such as the approach proposed by Yan *et al.* [79] or diffusion-based approaches proposed by Chu *et al.* [10]. These methods have demonstrated improved efficiency and precision in addressing the difficulties posed by incomplete data.

As the field advances, there is a growing emphasis on the development of not only visually accurate but also functionally realistic models. Recent studies [42, 28] have explored the integration of physical properties and realistic texture. These advances are crucial in closing the gap between virtual 3D models and their real-world counterparts, enabling more immersive and practical applications.

In the 3D shape generation and completion field, two main model types are commonly employed: encoder-decoder models and generative models. These models offer distinct advantages for each representation type, including mesh, point cloud, voxel, or implicit representation, as indicated by numerous studies and papers used as references for the models utilized. The taxonomy is illustrated in Figure 4.2.

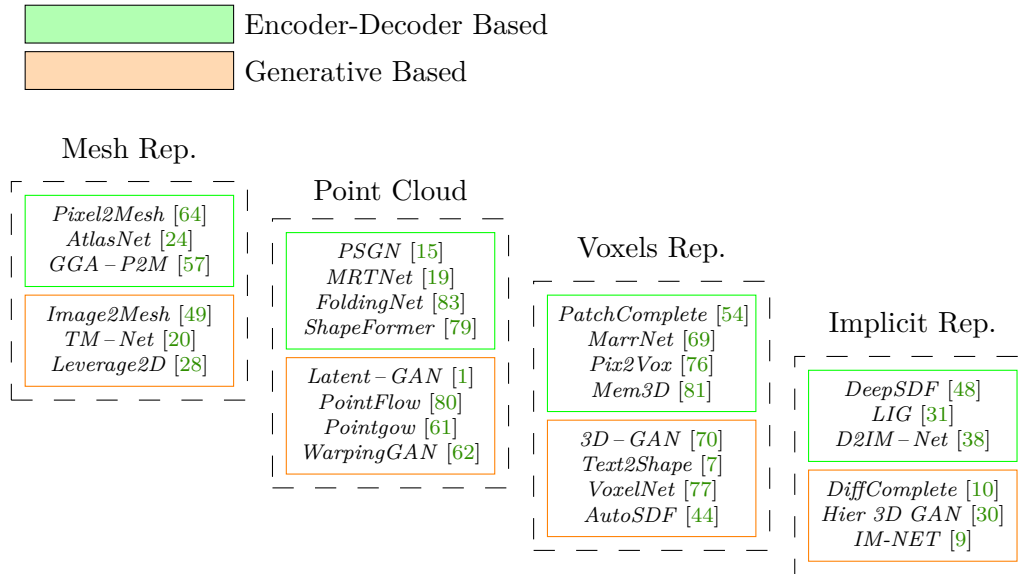


Figure 4.2: Encoder – Decoder/Generative based taxonomy of deep learning 3D shape generation and completion. It is important to note that this is not an exhaustive listing of all papers in the field, but rather a collection of the most notable ones in each category.

To better understand the progress and techniques of 3D shape completion and generation using deep learning, refer to Xu *et al.* [78]. The following sections are focused on explaining the fundamental principles of *diffusion-based* generative models and *Generative Adversarial Networks (GANs)*.

4.1.1 Diffusion Models

Diffusion models have gained significant attention because of their ability to produce high-quality and diverse samples. These models operate by gradually transforming a data distribution by adding and then removing noise, effectively learning the data distribution during this denoising process [6].

At their core, diffusion models consist of two main phases: the forward process and the backward process. In the forward process, noise is incrementally added to the data until it is transformed into a Gaussian distribution. The backward process, which is the generative phase, involves learning to reverse this noise addition to recover the original data from the noise. Section 4.2.1 describes the process in detail based on a specific paper working with 3D models.

Training and Objective Function

Training of diffusion models involves optimizing the parameters of the backward process. Typically, this is done using a variant of the variational lower bound or other objectives that measure the difference between the generated and the original data.

Variants of Diffusion Models

Several variations of Diffusion Models have been developed to enhance performance, address specific challenges, or adapt the model to different types of data:

- *Conditional Diffusion Models* [84]: Incorporate conditioning information, allowing the generation of data that adheres to specific conditions or attributes.
- *Discrete Diffusion Models* [59]: Adapted for data inherently discrete by nature, such as text or categorical data.
- *Continuous Diffusion Models* [34]: Utilize continuous-time dynamics for a more flexible and potentially more efficient diffusion process.
- *Hybrid Models* [35]: Combine elements of diffusion models with other generative approaches, such as *variational autoencoders* or generative adversarial networks, to leverage the strengths of each approach.

Despite their promising results, Diffusion Models face several challenges, including computational efficiency while needing to balance the trade-off between sample quality and diversity.

4.1.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs), introduced by Goodfellow *et al.* [22] have revolutionized the field of generative models, with their ability to generate realistic and high-quality data. GANs consist of two competing neural network models: a generator G that creates data samples and a discriminator D that evaluates them [21].

The foundation of GANs is a minimax game between the generator and the discriminator. The generator network G creates samples from the same distribution as the training data. The discriminator network D tries to distinguish between the real and the fake data produced by G . Through this adversarial process, G learns to produce more realistic data and D becomes better at detecting fakes.

Training and Objective Function

The minimax game that forms the basis of GANs has the value function $f(D, G)$ defined as:

$$\min_G \max_D f(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D_{\theta_D}(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D_{\theta_D}(G_{\theta_G}(z)))] \quad (4.1)$$

where p_{data} represents the data distribution, p_z denotes the noise distribution, x is a sample from the real data, z is a noise sample, θ_D and θ_G are the parameters for D and G respectively. The discriminator D , parameterized by θ_D , aims to maximize the probability of correctly classifying both real and fake data. The generator G , parameterized by θ_G , aims to minimize the probability that D correctly classifies fake data as fake. Ideally, D should output high probabilities for real data and low probabilities for data generated by G . G tries to generate data that D will classify as real.

Training continues until a *Nash equilibrium* is reached, where D and G cannot improve their strategies, given the other player's strategy. This point is known as the *saddle point* in the context of GANs.

Variants of GANs

GANs have been extended into various architectures to improve the basic GAN structure or address specific issues. Even in the generative process or to improve certain aspects of the generation, like image resolution, diversity, or conditional generation. Some noteworthy variants include:

- *Deep Convolutional GANs (DCGANs)* [53]: Particularly useful for image data, DCGANs introduce convolutional layers to the generator and the discriminator, providing more robust feature learning and generation capabilities.
- *Conditional GANs (CGANs)* [43]: Incorporate additional inputs that condition the generation process, allowing for the generation of targeted data. Additional inputs can include class labels or types of images, allowing the creation of specific categories of images.
- *Cycle GANs* [87]: Designed for image-to-image translation tasks without paired examples, using a cycle consistency loss to learn the translation.
- *StackGANs (SGAN)* [29]: Use a multistage generation process for high-resolution image synthesis. A rough image sketch is created first, then refined in the following stages to produce high-resolution details.
- *Progressive Growing GANs (PGGANs)* [33]: Start by generating low-resolution images and progressively increase resolution by adding layers to the networks. Allow for better training stability and higher-quality results.

Despite their achievements, GAN training is frequently difficult due to problems such as mode collapse, vanishing gradients, and lack of convergence. One of the recent innovations in GANs is *auxiliary classifier GANs (AC-GANs)* [47]. AC-GANs enhance the quality and diversity of the samples produced by integrating an additional classifier into the discriminator. Similarly, self-attention GANs employ attention mechanisms to capture global dependencies within the data.

4.2 State-of-the-Art in 3D Shape Completion Methods

The following section outlines the state-of-the-art approaches used for the task of **shape completion**.

4.2.1 DiffComplete

Chu *et al.* [10] introduced *DiffComplete*, a diffusion-based approach to 3D shape completion on range scans represented by implicit shape representation. It balances realism, multi-modality, and high accuracy, distinguishing itself from previous deterministic and probabilistic approaches. Important advances include a hierarchical feature aggregation mechanism for injecting spatially consistent conditional features and an occupancy-aware fusion strategy for handling multiple incomplete shapes. This approach achieves state-of-the-art performance on extensive benchmarks, substantially improving completion accuracy and quality. DiffComplete exhibits strong adaptability to objects from unseen classes, eliminating the need to retrain the model for different applications. This method will be further discussed later, as it is the main building block of this thesis.

Dataset and Problem Formulation

This study focused on generating training data by creating incomplete 3D scans from depth frames. A truncated signed distance field was utilized in a volumetric grid to represent incomplete scans. To represent ground-truth shapes, a *truncated unsigned distance field (TUDF)* was used because of the high number of open meshes in the dataset.

Two volume representations are presented: an incomplete scan denoted c , and a complete 3D shape denoted as x_0 , both utilized in the learning phase. During inference, only c is provided along with x_T , which is generated from the standard Gaussian distribution. The objective is to generate the complete shape x_0 approaching the shape completion problem as a generation problem, using the information from the incomplete scan. The task uses the probabilistic diffusion model, which includes a *forward* and *backward* process.

- In the *forward process* $q(x_{0:T})$, Gaussian noise is gradually added to obscure the ground-truth shape x_0 , into a random noise volume x_T where T is the total number of time stamps.
- The *backward process* $p_\theta(x_{0:T}, c)$ utilizes a shape completion network, with learned parameters θ , to iteratively remove noise from the noise volume x_T .

Since forward and backward processes are controlled by a discrete-time Markov chain across time steps $\{0, \dots, T\}$, the Gaussian transition probabilities they follow can be expressed as shown in Equation 4.2 and Equation 4.3:

$$q(x_{0:T}) = q(x_0) \prod_{t=1}^T q(x_t | x_{t-1}), \quad q(x_t | x_{t-1}) := \mathcal{N}\left(\sqrt{1 - \beta_t}x_{t-1}, \beta_t \mathbf{I}\right), \quad (4.2)$$

$$p_\theta(x_{0:T}, c) = p(x_T) \prod_{t=1}^T p_\theta(x_{t-1} | x_t, c), \quad p_\theta(x_{t-1} | x_t) := \mathcal{N}\left(\mu_\theta(x_t, t, c), \sigma_t^2 \mathbf{I}\right). \quad (4.3)$$

In Equation 4.2, β_t represents a scalar value within the range $[0, 1]$ that determines the variance schedule, effectively controlling the level of noise introduced at each step t of the process. For Equation 4.3, $p(x_T)$ denotes a Gaussian prior at a specific time stamp t , where μ_θ represents the mean predicted by the network. Furthermore, σ_t^2 in this equation refers to the variance. The $\mathcal{N}(0, \mathbf{I})$ denotes a unit Gaussian distribution.

There is a simplification in which the prediction $\mu_\theta(x_t, t, c)$ is altered to predict $\epsilon_\theta(x_t, t, c)$. This new prediction aims to approximate the noise introduced to corrupt x_{t-1} in the forward process, and the variable σ_t is substituted by the pre-defined β_t . The training objective for this simplified approach is shown in Equation 4.4. To maximize the probability of generation $p_\theta(x_0)$ (to obtain the original shape), the mean square error loss function is used:

$$\arg \min_{\theta} E_{t, x_0, \epsilon, c} \left[\|\epsilon - \epsilon_\theta(x_t, t, c)\|^2 \right], \quad \epsilon \in \mathcal{N}(0, \mathbf{I}), \quad (4.4)$$

where ϵ is the noise applied to corrupt x_t into x_{t+1} , and ϵ_θ is the noise predicted by the shape completion network.

Shape Completion Network Overview

To condition the generation process, the technique proposed by ControlNet [85] was used. This involves encoding using separate branches with identical network structures, but without sharing parameters. Complete and incomplete shapes are represented in multiresolution 3D volume space, preserving spatial structures. Adding the feature volumes from both branches achieves a cost-effective and spatially consistent feature aggregation. This method avoids the computational expenses of more complex techniques, like cross-attention.

Network Architecture Details

The network architecture (see Figure 4.3) used employs a dual branch strategy, consisting of one branch to handle complete shapes and another branch to handle incomplete shapes.

- The primary branch, derived from the 3D U-Net [46], is used for improved diffusion. It takes as input a corrupted complete shape x_t . This branch consists of five phases:
 1. projection into a higher-dimensional space,
 2. encoding and downsampling,
 3. integration of non-local information using a middle block with a self-attention layer,
 4. upsampling to restore the feature volume to the original size,
 5. and finally, projection to the original dimensional space.
- The secondary branch processes the incomplete shapes c and mirrors the structure of the primary branch. This branch focuses on efficient feature extraction and utilizes a projection layer after each encoder/middle block to forward multiscale features to the primary branch’s decoder blocks.

Furthermore, incorporating time-conditioned diffusion models enhances the network’s capabilities by converting the time step into an embedding using two MLPs. To effectively merge features of complete and incomplete shapes, the network implements hierarchical

feature aggregation across multiple levels. Before feature aggregation, projection layers align the shape distributions, before combining them with the control branch. Adjusting the level of the network at which the features are aggregated can influence the balance between the precision of the completion and the multi-modality of the results.

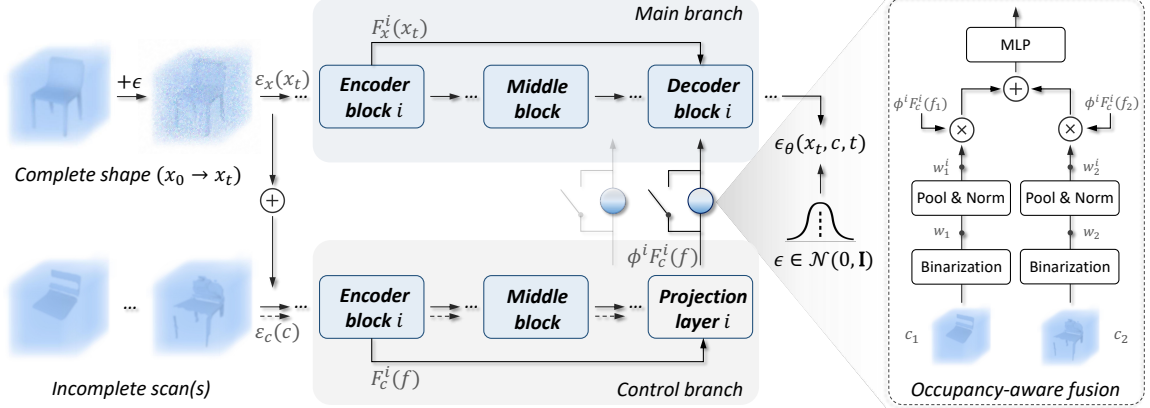


Figure 4.3: Illustration of the network architecture designed for 3D shape processing, featuring a main and a control branch. The main branch, a 3D U-Net structure, handles corrupted complete shapes by initially projecting them into a high-dimensional space, followed by progressive encoding, non-local information integration through self-attention, and subsequent upsampling. At the end, the output is projected back to match the input dimension. The control branch processes incomplete scans by extracting multi-scale features without the upsampling part typically required, instead using a projection layer to pass these features to the main branch. Adapted from [10].

Multiple Incomplete Scans as Inputs

The network has the option to handle multiple incomplete scans simultaneously, enhancing the geometric data for more precise shape completion and making it more suitable for cases where a single scan may not capture the object completely. A new proposed method is used to effectively merge the features from these scans, giving priority to the accuracy of the feature fusion process by aligning the partial shapes initially and using an occupancy-aware technique in the feature space. For a more detailed explanation of this method, refer to the comprehensive explanations provided in the DiffComplete paper.

Training and Inference

The DiffComplete network training process starts with a single incomplete scan as a conditional input. In this initial phase, all network parameters, except the MLP layer responsible for occupancy-based fusion, are trained to achieve the objective described in Equation 4.4. Once the network reaches convergence, the parameters are kept fixed and the MLP layer is fine-tuned by using multiple incomplete scans to improve its performance. To perform inference, an initial point x_T is created by generating a 3D noise volume from a standard Gaussian distribution. The network then reconstructs the complete shape x_0 from x_T in iterative steps T , taking into account partial scans c as conditions.

4.2.2 PatchComplete

Rao *et al.* [54] introduced *PatchComplete*, a novel approach to 3D shape completion. The method leverages multiresolution patch priors for reconstructing complete shapes from partial shapes, even for unseen categories. Unlike traditional methods that rely on category-specific learning, PatchComplete focuses on learning generalized shape priors at the patch level, exploiting the commonalities among different object parts across categories. Learning generalized shapes is motivated by the observation that local geometric structures, such as chair legs or table surfaces, are often shared between different categories of objects. An illustration of the PatchComplete process can be seen in Figure 4.4.

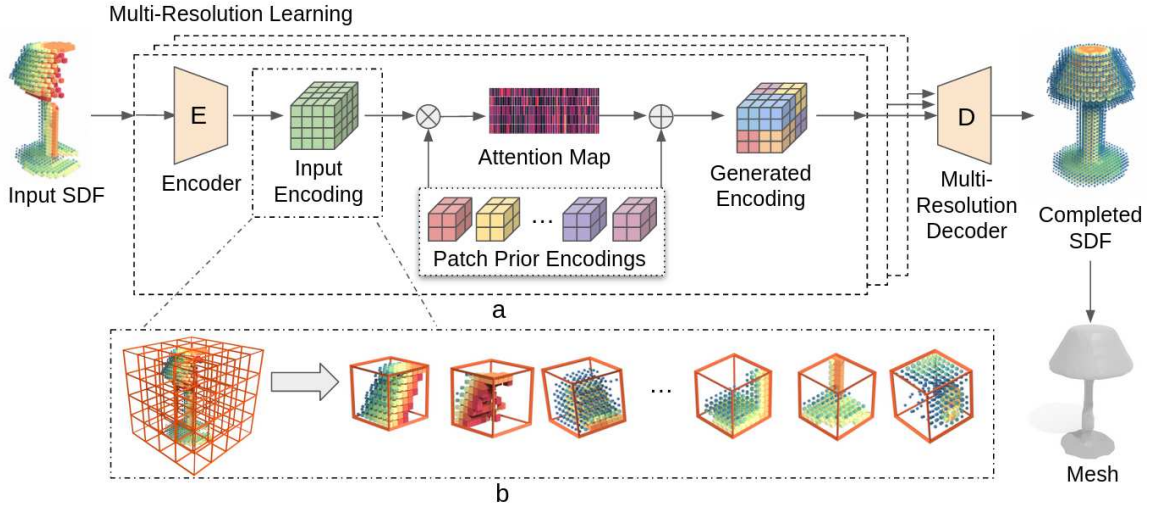


Figure 4.4: Illustration of the multi-resolution learning strategy employed by PatchComplete. The process begins with an input SDF being encoded, which is then used as an input into to attention mechanism together with patch prior encodings to generate a weighted encoding that best matches the input. This is followed by a multi-resolution decoding phase that fuses information across different scales to produce a completed SDF. Finally, the completed SDF is converted into a mesh. Adapted from [54].

Learning Local Patch-Based Shape Priors

Recognizing that local structures can vary in size, PatchComplete learns patch priors at different resolutions (see Figure 4.5). This multi-resolution approach allows for the effective reconstruction of complete shapes by fusing priors across scales, thereby capturing both global shapes and finer details. The fusion process is guided by attention scores that determine the relevance of each prior patch to the input, enabling a coherent assembly of the final shape.

Training

The model is based on 3D convolutional encoders for input scans and patch priors and is trained using a reconstruction loss ℓ_1 . The training process involves two main phases: learning the patch priors and then learning to fuse multi-resolution priors for shape completion. The model is trained on a mixture of synthetic and real-world data to enhance robustness and generalizability.

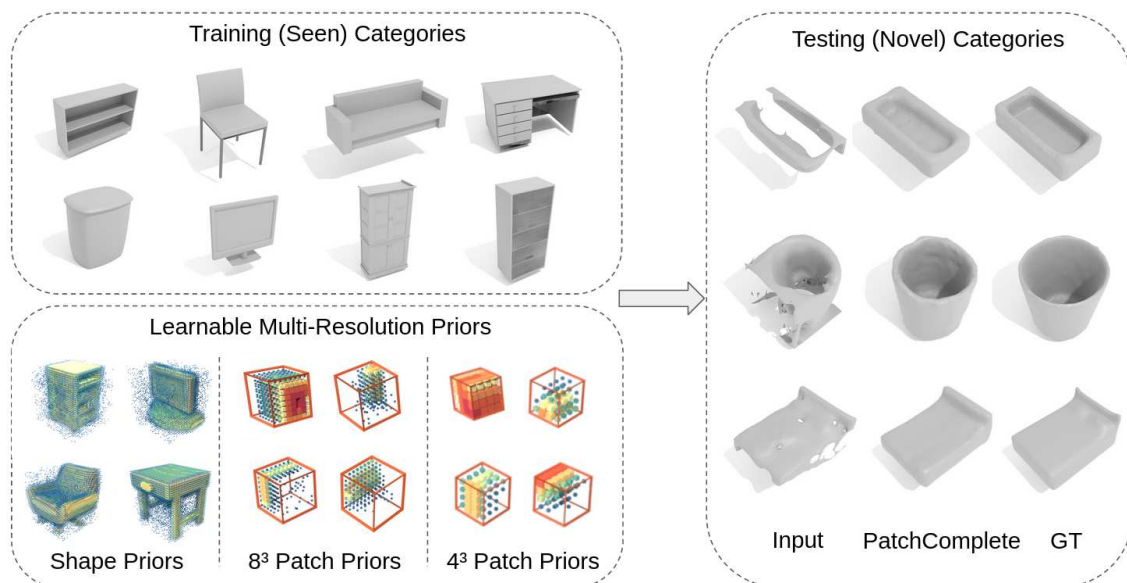


Figure 4.5: Visualization of PatchComplete leveraging the concept of transferable knowledge across different categories for 3D shape completion. During the training phase, on the left, a variety of objects from seen categories are used to establish learnable multi-resolution shape priors. These priors consist of different resolutions: 8^3 patch priors and 4^3 patch priors, to capture the geometry at varying scales. On the right, PatchComplete is tested on novel categories that are not included in the training set. The sequence demonstrates the original incomplete input, the reconstruction made by PatchComplete, and the ground truth for direct comparison. Adapted from [54].

4.2.3 3D-EPN

Dai *et al.* [12] introduce a novel approach to 3D shape completion, named *3D-Encoder-Predictor Network (3D-EPN)*. The approach innovatively integrates volumetric deep neural networks with 3D shape synthesis techniques. This method advances by inferring not only complete shapes from partial 3D scans at a low resolution but also by enhancing the inferred shapes with high-resolution details. Through a multi-resolution 3D shape synthesis process, leveraging a comprehensive shape database. The illustration of the process overview is shown in Figure 4.6.

Method Overview

The 3D-EPN framework distinguishes itself through its two-phase strategic process. Initially, it employs a 3D encoder-predictor architecture that effectively processes partial scans to generate a coarse, yet complete, volumetric representation. The volumetric representation is then refined using a patch-based 3D shape synthesis method, which introduces fine-scale details. Details are based on geometric constraints from similar shapes within a dedicated database. This method ensures the preservation of the global structure while incorporating local details. The network is trained on a mixture of synthetic and real-world data to improve its robustness and generalizability.

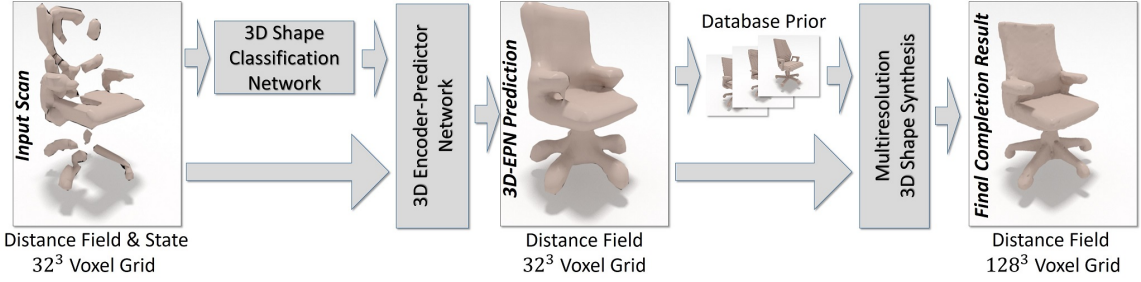


Figure 4.6: Overview of the shape completion process for 3D-EPN. The process begins with a partial 3D scan input, represented as a truncated signed distance field within a $32 \times 32 \times 32$ voxel grid. The scan is processed by a 3D shape classification network before entering the 3D-EPN. The predicted distance field, which retains the resolution of the $32 \times 32 \times 32$ grid, serves as the basis for further refinement. The database priors are then utilized in a multi-resolution 3D shape synthesis step, where the low-resolution prediction is correlated with higher-resolution database models to enhance detail. The final result is a completed shape represented as a distance field within a $128 \times 128 \times 128$ voxel grid, which has improved resolution and detail, reflecting the step of 3D-EPN prediction and database-informed synthesis. Adapted from [12].

3D Encoder-Predictor Network Architecture

Central to the 3D-EPN framework is its network architecture (see Figure 4.7). The first component is the 3D encoder, which compresses the input partial scan into a latent space. This is achieved by processing the input through a series of 3D convolutional layers. The transformation is followed by two fully connected layers, resulting in a condensed representation that includes both geometric data from the scan and semantic classifications from a 3D-CNN shape classifier. The second component is the predictor network, which employs 3D up-convolutions to expand the latent representation into a full-sized output of the estimated distance field values. Skip connections connect the corresponding layers of the encoder and predictor. This allows for the transfer and integration of the local structure from the input to the output, ensuring detailed and accurate shape predictions.

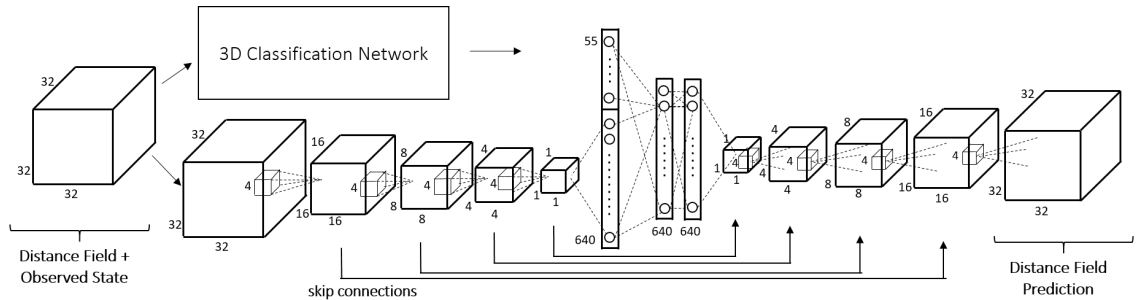


Figure 4.7: Visualization of the dual-component architecture of the 3D-EPN. The combination of the 3D convolutions, fully connected layers, shape classifier network and predictor network. Adapted from [12].

Shape Synthesis

The novelty of the 3D-EPN approach lies in its shape synthesis step, which significantly increases the resolution and detail of the initial predicted shape. Correlating low-resolution output with high-resolution models from a shape database. The method adeptly synthesizes fine-scale details, ensuring that the resultant mesh accurately reflects the intended global structure and local geometries. This process effectively leverages the amount of geometric information available in the database to enrich the reconstructed shapes.

4.2.4 ShapeFormer

ShapeFormer, introduced by Yan et al. [79], is at the forefront of transformer-based networks, used for shape completion. Designed to compute the distribution of completions of objects from partially and potentially noisy point clouds. Unlike previous methods, ShapeFormer takes advantage of a compact 3D representation known as a *Vector Quantized Deep Implicit Function (VQDIF)*. Representation effectively uses spatial sparsity to encapsulate a 3D shape through a succinct sequence of discrete variables. The illustration of the process overview is shown in Figure 4.8.

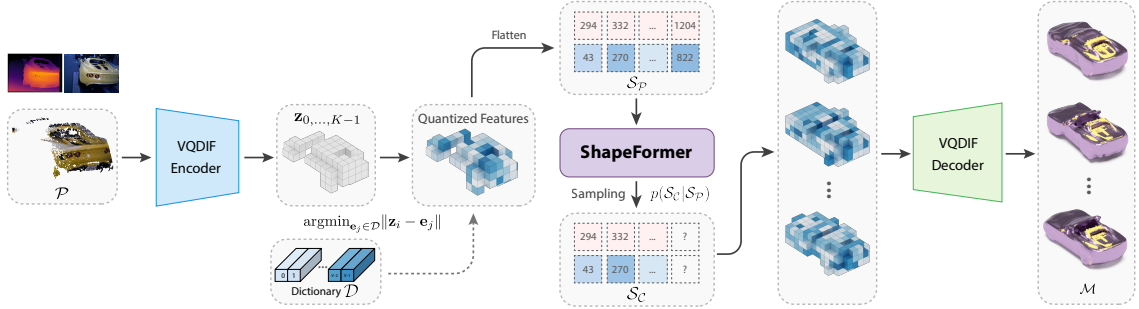


Figure 4.8: Overview of the shape completion process for ShapeFormer. The initial input, potentially a depth image represented as a point cloud P , is transformed by the VQDIF Encoder into a sparse sequence of features. This sequence is then compactly expressed as a series of 2-tuples, each tuple containing a spatial coordinate and a corresponding feature index from a pre-defined dictionary. These tuples are denoted as S_P and visually differentiated by dashed lines. ShapeFormer operates on S_P , predicting the likely complete sequence S_C by modeling the conditional probability distribution. Subsequent autoregressive sampling facilitates the generation of a predicted complete sequence. The VQDIF decoder then translates S_C back into a dense form, generating the surface reconstruction M . Adapted from [79].

Method overview

ShapeFormer addresses the challenge of 3D shape completion by harnessing the autoregressive capabilities of transformers to learn distributions over possible completions. The method incorporates local codes within a sequence of discrete, vector-quantized features. Thereby considerably reducing the size of the representation while retaining essential structural details.

Compact Sequence Encoding for 3D Shapes

VQDIF supports ShapeFormer by allowing the encoding of complex 3D shapes into compact sequences. Each sequence element comprises a 2-tuple representing the position and content of non-empty local features. Subsequently, transformers are applied to capture the global structure and dependencies within these sequences. This compact representation drastically reduces the sequence length from cubic to quadratic in terms of feature resolution, which is a significant breakthrough for the application of generative models in the 3D domain.

Chapter 5

Proposed Solution for 3D Shape Completion using Deep Neural Networks

To address the problem of automatic shape completion, a deep learning-based supervised solution is proposed. The first section of this chapter formally defines the problem to be solved. Subsequently, the process of generating the dataset for the shape completion task is introduced as a key element of supervised methods. After an in-depth explanation of the process of obtaining the dataset, the proposed method is described in detail.

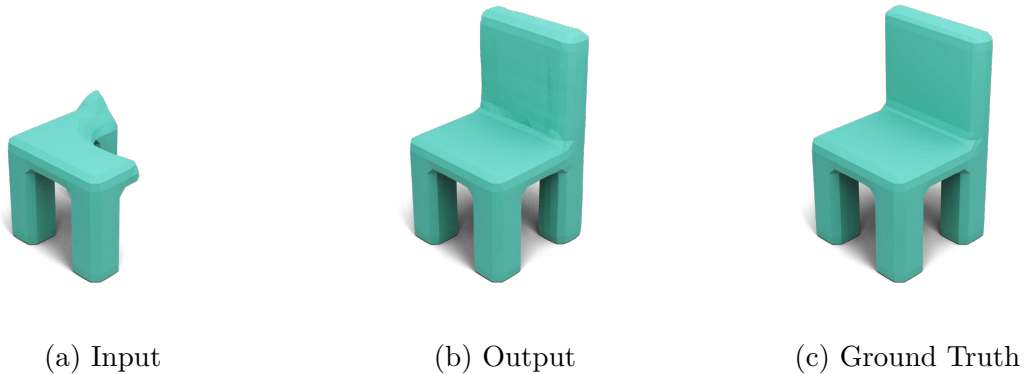


Figure 5.1: Example of an input in the form of an incomplete shape (a), generated output from the proposed solution (b), and a ground truth mesh (c). The goal of this method is to complete the missing parts of the given shape.

5.1 Problem Definition

Given the 3D shape represented as a triangular mesh $\mathcal{M} = (V, E, F)$, which represents the partial shape of a real-world object with one or more holes caused by incomplete scanning or object breakage, the shape completion problem is formulated as follows:

$$\mathcal{F}(\mathcal{M}_P) = \mathcal{M}_C, \quad (5.1)$$

where the function \mathcal{F} maps a partial (incomplete) shape denoted by \mathcal{M}_P to a complete shape \mathcal{M}_C . The mapping process is designed to extend the sets V , E , F of the partial mesh with elements that represent missing patches introduced by the function \mathcal{F} . Mathematically, this is formulated as:

$$\begin{aligned} V_C &= V_P + V_{\text{fill}}, \\ E_C &= E_P + E_{\text{fill}}, \\ F_C &= F_P + F_{\text{fill}}, \end{aligned} \tag{5.2}$$

where *fill* represents the elements newly introduced to the partial shape by the completion process.

The proposed solution operates with a TSDF representation within a voxel grid, utilizing a conversion function from \mathcal{M} to TSDF and vice versa.

5.2 Dataset Preparation Pipeline: Smashing the Objects

For the shape completion problem, the dataset should consist of partial (incomplete) shapes P and their corresponding complete shapes C . Complete shapes, in the form of polygonal mesh, were obtained from datasets such as Objaverse [14], ShapeNet [5], or ModelNet [72]. Although the datasets only contain labels, based on self-supervised learning principles, the corresponding inputs can be generated.

5.2.1 Process of Acquiring Incomplete Shapes

The objective is to generate a dataset D comprising pairs (x, y) , where y are obtained from mesh samples sourced from the datasets mentioned above. The corresponding incomplete input x is acquired in the form of a distance field, along with a ground truth distance field y , using Algorithm 1.

Algorithm 1 Creating Complete and Incomplete TSDF counterpart for a given mesh \mathcal{M}

```

1:  $\mathcal{M}_C \leftarrow \text{LoadMesh}(\text{ModelPath})$ 
2:  $\mathcal{M}_C \leftarrow \text{NormalizeMesh}(\mathcal{M}_C)$  ▷ Scale to unit cube and center
3:  $\text{TSDF}_C \leftarrow \text{MeshToTSDF}(\mathcal{M}_C)$ 
4:  $\text{Save}(\text{TSDF}_C, \text{„complete\_shape“})$ 
5:  $\mathcal{M}_P \leftarrow \mathcal{M}_C.\text{copy}()$ 
6: for  $i \in \{1, \dots, \text{NumHoles}\}$  do
7:    $\mathcal{M}_o \leftarrow \text{RandomPrimitive}()$ 
8:    $\mathcal{M}_o \leftarrow \text{TransformPrimitive}(\mathcal{M}_o)$  ▷ Scale, position, rotate
9:    $\mathcal{M}_P \leftarrow \mathcal{M}_P \setminus \mathcal{M}_o$  ▷ Boolean Difference
10: end for
11:  $\text{TSDF}_P \leftarrow \text{MeshToTSDF}(\mathcal{M}_P)$ 
12:  $\text{Save}(\text{TSDF}_P, \text{„incomplete\_shape“})$ 

```

The incomplete shapes are generated from complete 3D models through a series of steps, which simulate scenarios in which occlusions during scanning might lead to the absence of certain parts of the model. This absence could also be caused by the inability to capture specific sections of the model or from intrinsic attributes of the model itself, such as a missing

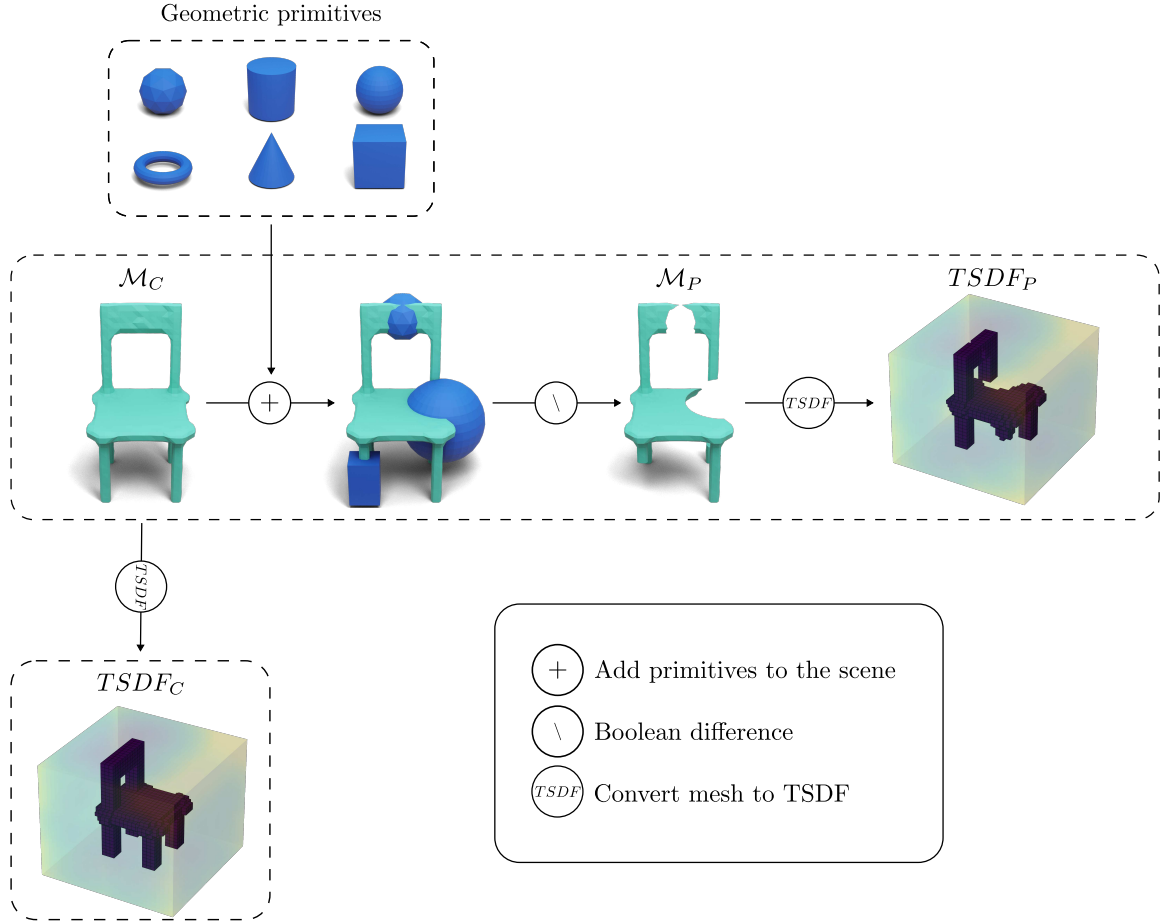


Figure 5.2: Visualization of generating an incomplete shape from a complete one: Beginning with the complete model \mathcal{M}_C , the initial step involves storing the ground truth as a TSDF ($TSDF_C$). Following this, a randomized process is employed to select and transform geometric primitives, which are then introduced into the scene such that they intersect with \mathcal{M}_C . Subsequently, a Boolean difference of the mesh with each geometric primitive gives an incomplete model \mathcal{M}_P . Finally, partial mesh \mathcal{M}_P is converted into a TSDF representation $TSDF_P$ and stored.

tooth in a dental scan. The pipeline to transform a given input \mathcal{M}_C into an incomplete shape \mathcal{M}_P is illustrated in Figure 5.2.

The methodology to create incomplete shapes from complete 3D models, as interpreted in Algorithm 1, involves a series of steps designed to replicate real-world scenarios of occlusion or inherent missing parts in the models. Initially, the model represented as a polygonal mesh is loaded and subsequently normalized by centering at the origin and scaling to a unit cube, a prerequisite for converting the mesh to a TSDF representation. After preparation, the model undergoes processing to generate its incomplete counterpart through Boolean difference operations. Since not all meshes are suitable for these operations (e.g. may contain self-intersecting triangles), an initial evaluation is carried out to determine if using Boolean differences is feasible. Following the validation of the suitability of the mesh, the TSDF representation of the complete shape (denoted $TSDF_C$) is stored. Then, the original mesh is duplicated for subsequent modification. The selection and application of geometric

primitives to create holes within the mesh are controlled by randomized processes, including the choice of primitive type and the number of holes to introduce. These primitives are randomly positioned and rotated to ensure that they intersect with the original mesh \mathcal{M}_C . The scale of primitives (in other words, holes) can be adjusted as needed. The Boolean difference operation is then applied, producing the incomplete mesh \mathcal{M}_P , which is subsequently converted to its TSDF representation, $TSDF_P$. Volumes of $TSDF_C$ and $TSDF_P$ are calculated to quantify the volume missing from the incomplete shape.

5.3 Shape Completion Pipeline: Filling Holes via Diffusion Process

The proposed solution for the shape completion task uses a diffusion-based model. The detailed description of the diffusion process and other details, such as the objective function, are provided in Section 4.2.1. This section will therefore concentrate on the architecture modeling the backward process, which builds on the Diffcomplete network proposed by Chu *et al.* [10]. The absence of code for the original architecture and some ambiguities in the paper required adaptations, resulting in a structure that might diverge from the original Diffcomplete. In particular, the adapted architecture incorporates enhancements aimed at addressing computational challenges associated with high-resolution output. The foundational concept for this architecture was inspired by the 3D U-Net architecture¹ [46]. The specifics of the input and output channels for each architectural block are shown in Table 5.1.

Pipeline Overview

The process of obtaining the complete shape from an incomplete one begins with the transformation of the given mesh \mathcal{M}_P into a TSDF representation. Once the partial shape has been acquired, it serves as a condition for the diffusion process. Subsequently, an iterative backward diffusion process is used, denoted as $p_\theta(TSDF_C^t, TSDF_P)$, which is modeled by the proposed network. Upon completion of the iterative process, the results $TSDF_C^0$ ($TSDF_C$) are reconstructed into the mesh representation \mathcal{M}_C using marching cubes. The visualization of this process is shown in Figure 5.3.

Diffusion Process

In the training phase, two types of input are introduced: $TSDF_P$, which denotes an incomplete scan, and $TSDF_C^0$, which signifies a complete 3D shape. During the training phase, a forward process is used to add noise to $TSDF_C^0$, obtaining $TSDF_C^t$, where t is chosen from a uniform distribution, and the noise is added by a linear noise scheduler.

During the inference phase, a randomized 3D noise volume of the standard Gaussian distribution is used as input $TSDF_C^t$. The trained completion network is then used for T iterations to produce $TSDF_C$ from $TSDF_C^t$, conditioned on the partial shape $TSDF_P$. To accelerate the inference process, the technique of subsampling a set of timestamps $[1, \dots, T/10]$ is employed, as discussed in [58]. This sampling technique is applied during each inference instance, such as during the validation or testing phase. The EMA is not used for inference due to unsatisfactory results.

¹<https://github.com/openai/improved-diffusion>

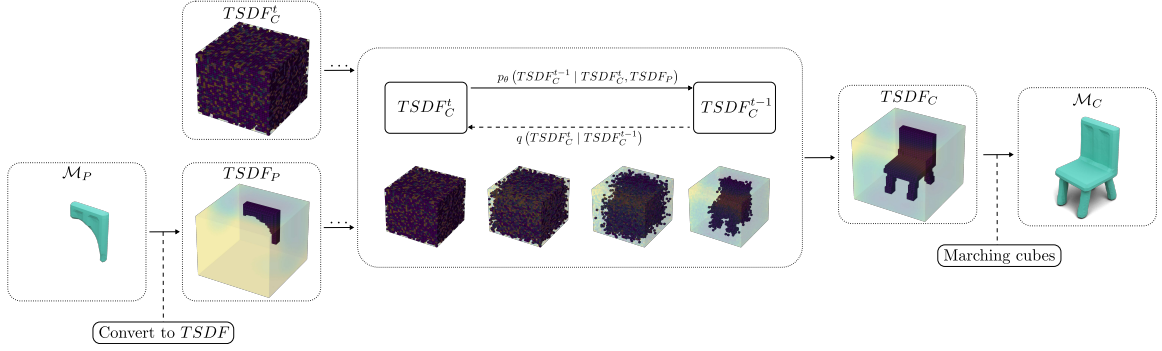


Figure 5.3: Overview of the shape completion process for the proposed method. Given an input \mathcal{M}_P the process starts with conversion to TSDF representation. Following this, a conditioned diffusion process is used to obtain the complete shape. Then the marching cubes algorithm is used to derive a mesh from the TSDF, with an optional final smoothing step.

Proposed Architecture

The proposed architecture contains the following high-level components:

- **Main Branch and Control Branch:** Central components of the architecture that process the input and condition in high-dimensional space.
- **Pre-processing Blocks:** Transform input into a higher-dimensional space before it undergoes processing by the main or control branch.
- **Downsampling and Upsampling Phases:** Essential when the condition (incomplete shape) possesses a lower resolution than the input/output, to synchronize the resolutions.
- **Post-processing Block:** Reverts the output to the same dimensional space as the input.

The architecture is illustrated in Figure 5.4. The process begins with the input $TSDF_C^t$, which signifies the complete shape at a specific timestamp t during the training forward phase. The condition, represented as $TSDF_P$, corresponds to the partial shape.

Preprocess and Downscale Phase

The initial stage for both inputs involves a **preprocessing phase**, where two 3D convolutions with a kernel size of 3 are applied. Convolution aligns the distribution of the given input. The preprocessing phase can provide additional information necessary for the generation process. By projecting the 3D values into a multidimensional space, potentially more informative values are obtained. Subsequent convolutions will also use a kernel size 3 unless otherwise stated. The first convolution transforms the input channels to 32, and the subsequent one increases them to 64. An optional **downscale phase** may be used for the input $TSDF_C^t$ to align its spatial resolution, to match the condition $TSDF_P$, if necessary. This alignment is achieved through a down-sample operation. The operation uses a 3D convolution with a stride of 2. Once both the input and the condition are prepared in a higher-dimensional space, they are ready for processing by the main and control

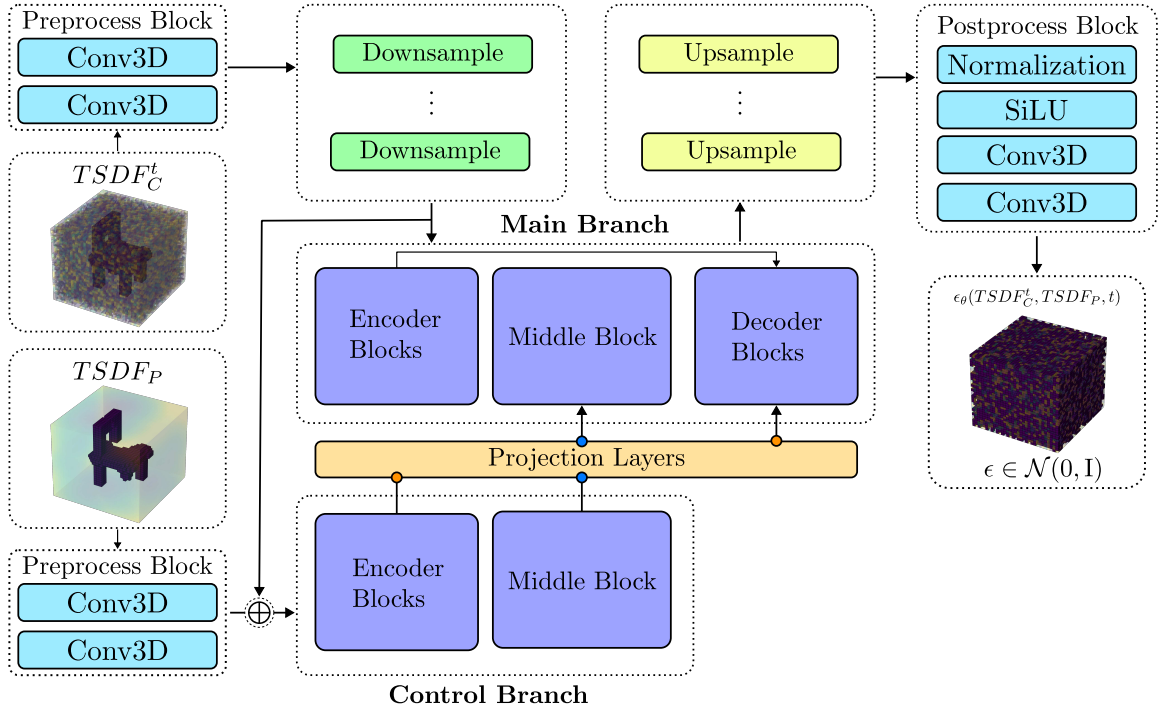


Figure 5.4: Visualization of the proposed architecture for shape completion task. The input, denoted as $TSDF_C^t$, represents the complete shape, with added noise at time t , while the condition input is marked as $TSDF_P$. Both the input and the condition undergo initial processing by a preprocessing block. If necessary, the input is then downsampled to align the spatial resolution of the condition. Subsequently, the input and condition are merged through tensor addition and directed to the conditional network. The main branch processes the input, whereas the control branch handles the condition. The features of the control branch are integrated into the main branch, helping the network shape the output based on the condition. If required, the output of the main branch is upsampled to the spatial resolution of $TSDF_C^t$. A post-processing block is then applied to revert the output to a lower-dimensional space. This architectural framework aims to reconstruct the noise $\epsilon_\theta(TSDF_C^t, TSDF_P, t)$ introduced during the forward process.

branches. As illustrated in Figure 5.4, the condition is combined with the input in this higher-dimensional space through a tensor addition. It is important to note that, while the main and control branches share a similar architectural structure, they do not share parameters.

Main and Control Branch

Both the main and control branches are composed of multiple encoder blocks, succeeded by a middle block, with decoder blocks following in the main branch only, as the control branch does not necessitate decoder blocks. A detailed illustration of these branches is provided in Figure 5.5.

The architecture incorporates four encoder blocks, a single middle block, and four decoder blocks:

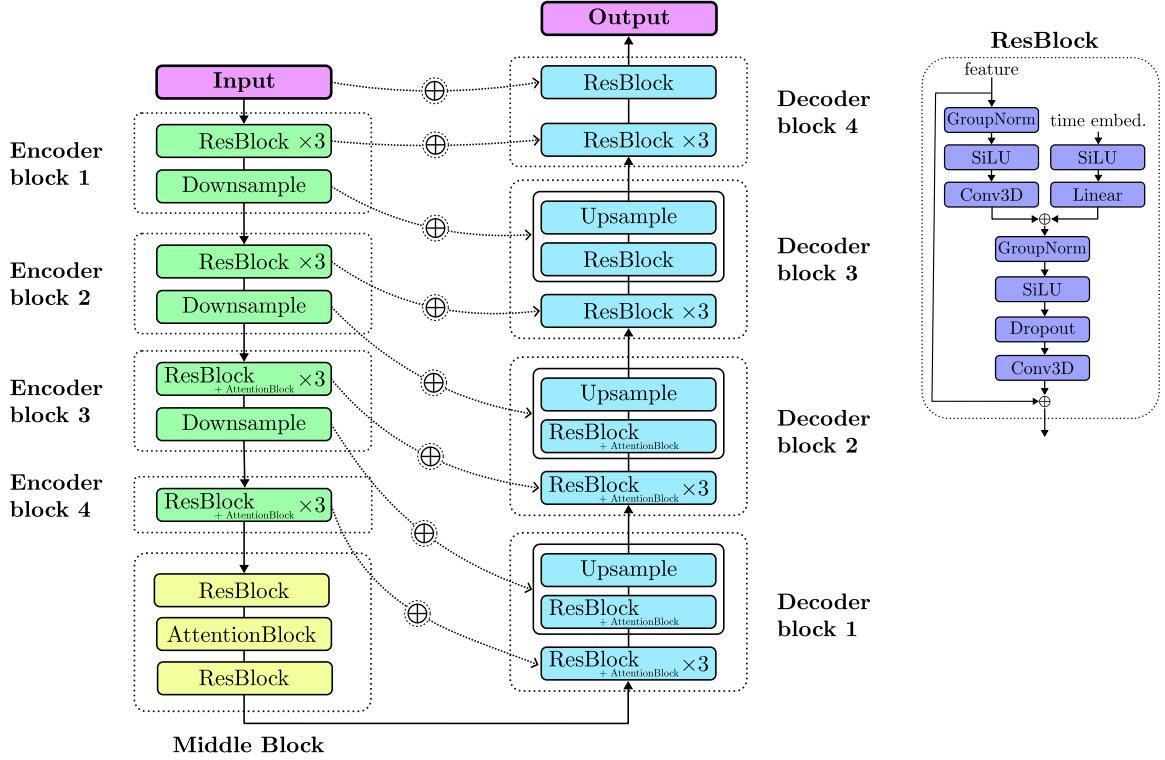


Figure 5.5: Detailed view of main/control branch architecture. The architecture is composed of encoder blocks, a middle block, and decoder blocks, with the *ResBlock* serving as the core element for processing features and time embeddings. The structure includes precisely four encoder blocks, one middle block, and four decoder blocks. Skip connections, represented as dotted arrows, transfer features from encoder blocks to decoder blocks. The aggregation of features occurs by concatenating the output of a previous decoder block with the corresponding encoder block. In particular, the final two encoder blocks and the initial two decoder blocks are enhanced with attention layers after each *ResBlock*, prioritizing the model’s focus on relevant features.

- **Encoder blocks** are comprised of three *ResBlocks* and a *Downsample block*, except for the final encoder block, which omits the downsample block. The purpose of the encoder block is to gradually reduce the spatial dimensions of the input while increasing the depth of features, facilitating the extraction of hierarchical representations.
- **Middle block** contains a *ResBlock*, an *Attention block*, and another *ResBlock*. The middle block is strategically designed to enhance the representation of features by incorporating attention mechanisms, which enable the model to focus on relevant parts of the input.
- **Decoder blocks** consist of four *ResBlocks*, followed by an *Upsample block*. Upsampling is achieved through 3D interpolation using the nearest-neighbor approach, followed by a 3D convolution. The last two encoder blocks and the initial two decoder blocks are augmented with an *Attention layer* after each *ResBlock*. The decoder blocks are responsible for reconstructing the original input from the representations generated by the encoder.

In alignment with the conventions of U-Net-type architectures, skip connections are employed. The visualizations of those connections are shown in Figure 5.5 for each block. The inputs for the decoder block, along with the skip connections, are combined through the concatenation along the channel dimension.

Condition Feature Aggregation

The control branch consists of encoder blocks and a middle block. The features extracted from these components are integrated into the main branch through a two-step aggregation process. Initially, a projection layer is applied to each block’s output, utilizing a 3D convolution with a kernel size of 1 for this transformation. Subsequently, these projected features are merged with the main branch’s skip connections through a tensor addition. The selection and number of control branch connections incorporated can influence the architecture’s ability of multimodality.

This aggregation and skip connection mechanism within the control and main branches can be mathematically represented as follows:

$$d^i = [D_x^{i-1}(x_t), F_x^i(TSDF_C^t) + \phi^i(F_c^i(TSDF_P))], \quad (5.3)$$

where d^i signifies the input to the current decoder block, with $[\cdot, \cdot]$ denoting the concatenation operation and ϕ^i representing the projection layer. D_x^{i-1} refers to the output from the preceding decoder block, while F_x^i / F_c^i indicates the skip connection from the main/control branch corresponding to the decoder block.

Upsample and Postprocess Phase

The output of the main branch undergoes the **upsampling phase** through the upsample blocks, to align with the spatial resolution of the condition, effectively mirroring the earlier downsampling process. Following this, the **postprocessing phase** comes into play, incorporating a normalization layer, a *SiLU* layer, and a pair of 3D convolution layers, to project shape from high-dimensional space back to the 3D space.

Block	Input Channels	Output Channels
Preprocess	C	32
	32	64
Encoder 1	64	128
Encoder 2	128	128
Encoder 3	128	128
Encoder 4	128	128
Middle Block	128	128
Decoder 1	256	128
Decoder 2	256	128
Decoder 3	256	128
Decoder 4	192	64
Postprocess	64	32
	32	C

Table 5.1: Channel specifications for architectural blocks. The specified input and output channels are exclusively related to the entry point of each block and do not detail the internal dynamics where skip connections are utilized, as illustrated in Figure 5.5. For detailed information on the input and output parameters of each layer, refer to the provided implementation on GitHub.

Chapter 6

Implementation Details

This chapter offers an overview of the parts implemented and obtained in this thesis, covering the technologies and datasets utilized, as well as a description of the training and evaluation parameters. The goal is to ensure adherence to the principles of reproducible research, offering readers all the necessary details and resources to replicate the results presented here accurately. **The code containing the implementation and documentation is publicly available on GitHub <https://github.com/Monnte/shape-completion/>.**

6.1 Technologies

The proposed solution was implemented using the Python programming language. The PyTorch¹ framework was used to construct and train neural networks. Data loading and processing were facilitated by the Numpy² package, enabling CPU vectorization. Trimesh³ was used to handle 3D shapes. A critical tool was the mesh2sdf package⁴, used in the study by Wang *et al.* [67] to generate data in the TSDF format. Additionally, the objaverse⁵ package provided a framework for downloading models from this dataset. The codebase was derived from Improved Diffusion, an open-source repository on GitHub⁶. For the evaluation of the results, part of the code was adapted from the publicly accessible PatchComplete repository on GitHub⁷.

6.2 Dataset Specifications

Three datasets of 3D models were chosen for training and evaluation. The Objaverse [14] dataset served as the primary source of training, while ShapeNet [5] and ModelNet [72] were used to evaluate performance on known and unknown out of distribution categories. The training dataset, designed to mirror the findings of the DiffComplete paper, consisted mainly of 3D furniture models. To examine out of distribution scenarios involving unknown categories, animals and vehicles from the Objaverse dataset were also explored.

¹<https://pytorch.org/>

²<https://numpy.org/>

³<https://trimesh.org/>

⁴<https://github.com/wang-ps/mesh2sdf>

⁵<https://objaverse.allenai.org/>

⁶<https://github.com/openai/improved-diffusion>

⁷<https://github.com/yuchenrao/PatchComplete>

The datasets were divided into training, validation, and testing sets using a split ratio of 70/10/20.

The generation of incomplete counterparts of the original shapes was carried out as described in Section 5.2. The numbers of complete shapes, and the number determining how many incomplete shapes should be generated for one complete shape, will be provided in subsequent sections. Up to 3 geometric primitives with sizes ranging from (0.5, 0.9) were allowed to be subtracted from each original mesh. Due to computational constraints, further experiments and evaluations primarily used $32 \times 32 \times 32$ grid resolution. Figure 6.1 offers a visual comparison of 3D shapes reconstructed from TSDF representations at different resolutions.

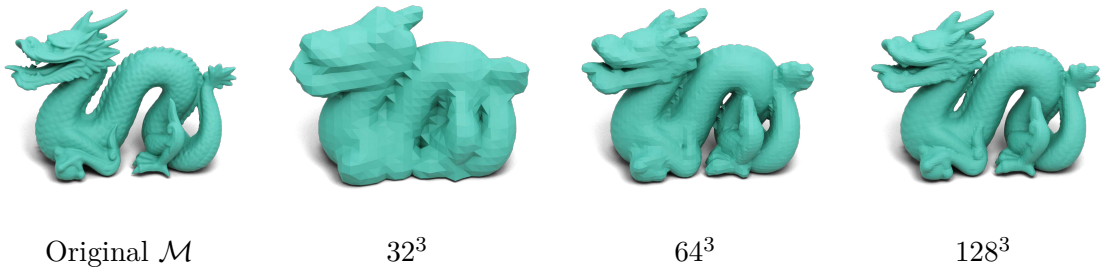


Figure 6.1: Visual comparison of a Stanford dragon 3D model captured in TSDF representation at different grid resolutions. The models are visualized using mesh reconstruction from distance field via Marching Cubes.

The resolution of the grid $32 \times 32 \times 32$ might not adequately capture all the intricacies of a model. Higher resolutions are necessary for more detailed representations, though they come with significant computational costs. Generated incomplete models miss 10% to 90% of their original volumes. A visual comparison of a single model and its various incomplete counterparts, each with different degrees of missing volume, is shown in Figure 6.2.

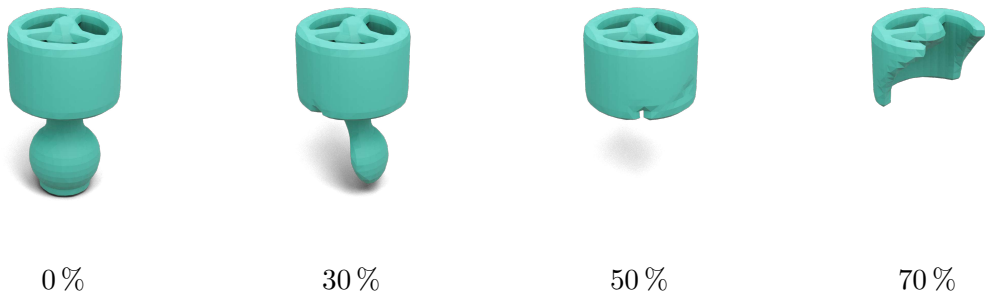


Figure 6.2: Visualization of a 3D shape with missing volume. Varying degrees of missing volume emphasize how much information is provided for shape completion.

6.2.1 Objaverse

The Objaverse 1.0 dataset was selected as the primary dataset for this thesis. Compared to other 3D datasets, Objaverse has exceptional scale (more than 800 000 high-quality 3D

models created by a community of more than 100 000 artists), diversity (covers a broad array of categories) and detailed annotations.

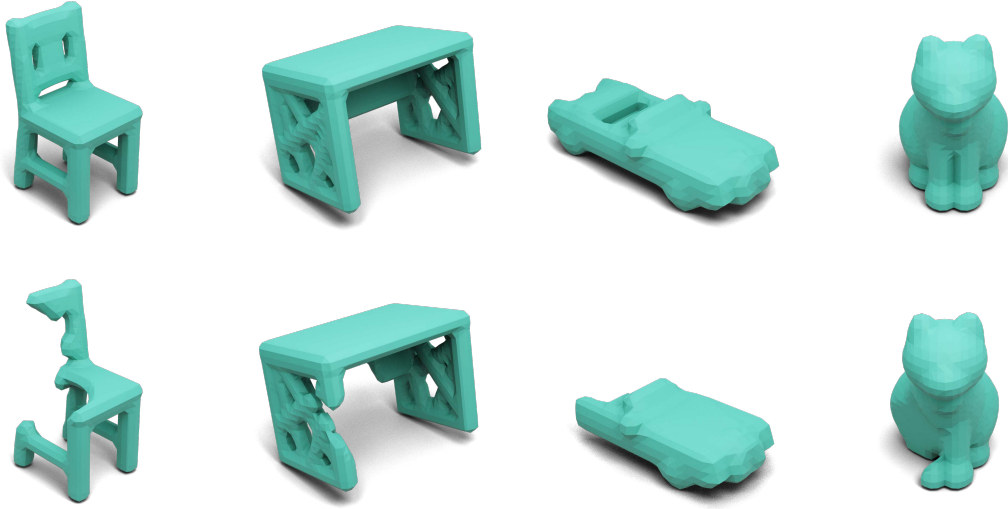


Figure 6.3: Selection of models from the Objaverse dataset, highlighting examples from various categories. From left to right: the first two models represent the *furniture* category, the third model is from the *vehicles* category, and the fourth illustrate the *animals-pets* category. The top row displays complete models, while the bottom row shows their respective incomplete versions. Each model is reconstructed from the TSDF representation on a $32 \times 32 \times 32$ grid.

For training purposes, the category named *furniture* was utilized, with the tags: *chair*, *lamp*, *bathtub*, *chandelier*, *bench*, *bed*, *table*, *sofa*, and *toilet*. The number of complete shapes corresponding to each tag is in Table 6.1. For out of distribution testing, two additional categories were employed: *cars-vehicles* and *animals-pets*. Within the category of vehicles, the tags *car*, *truck*, *bus*, and *airplane* were used, and within the category of animals, the tags *cat* and *dog* were utilized. Selected shapes from the datasets are visualized in Figure 6.3. The parameter specifying the number of incomplete counterparts to be created was set to 5 for *furniture*, 10 for *animals-pets*, and 5 for *vehicles*. The distribution of the furniture dataset, in terms of missing volume, is illustrated in Figure 6.4. The other datasets follow a very similar distribution.

As illustrated in Figure 6.4, the missing volume in the training, validation, and testing dataset splits were expected to exhibit a consistent distribution pattern. However, the representation of the 40 % and 50 % bars for the training and testing splits, respectively, contradicts this expectation, revealing an inconsistency in the distribution. This inconsistency arises from the failure in generating the predefined number of incomplete counterparts, where the process exceeds the maximum allowed attempts.

6.2.2 ShapeNet

For out of distribution testing of known or unknown categories, ShapeNet was used, offering a diverse and extensive collection of 3D models organized under the WordNet taxonomy.

An overview of the categories within ShapeNet, together with the complete shape count for each category, is presented in Table 6.2. A selection of models from the datasets is shown

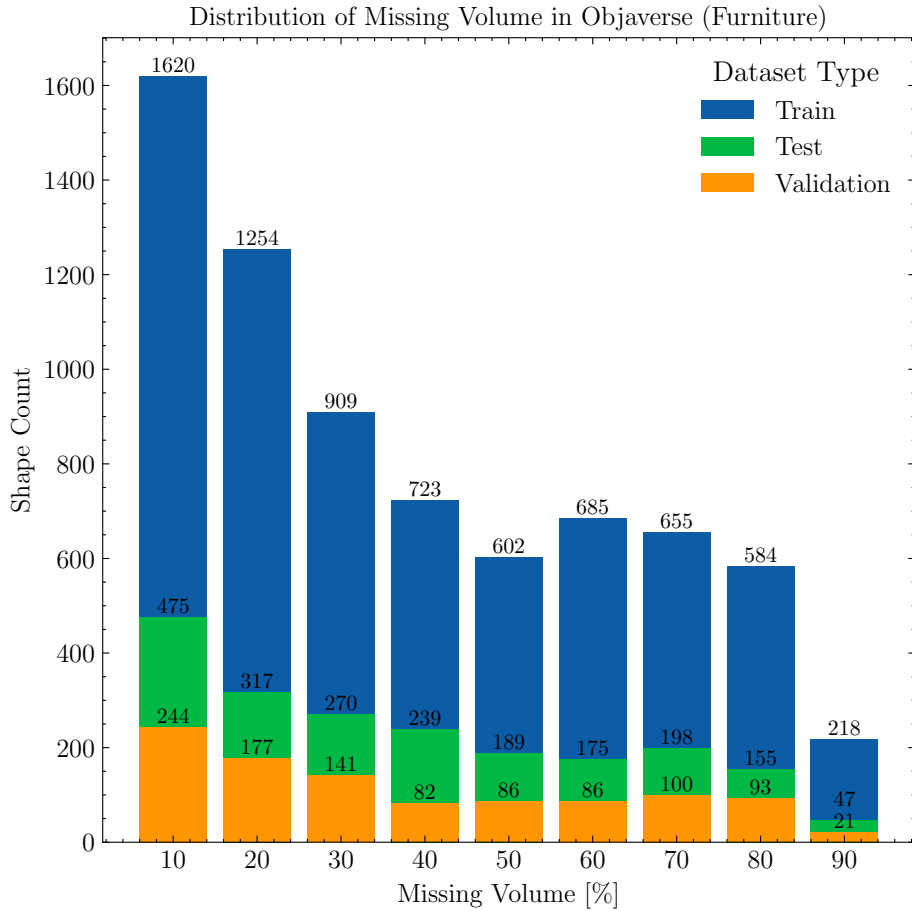


Figure 6.4: Distribution of missing volume for training shapes across Objaverse furniture dataset. The bar chart shows the count of incomplete shapes based on the missing volume.

Table 6.1: Enumeration of complete shape counts within the Objaverse datasets. Table shows the counts of complete shapes for the given tags in the datasets.

Furniture		Animals-Pets		Vehicles	
Tag	Count	Tag	Count	Tag	Count
Sofa	193	Table	645	Cat	272
Chair	513	Bed	109	Dog	201
Lamp	452	Bench	110		
Chandelier	27	Toilet	37		
Bathtub	4				
Total			2090		473
					539

in Figure 6.5. The parameter specifying the number of incomplete counterparts generated for each complete shape from the dataset was set to 5.

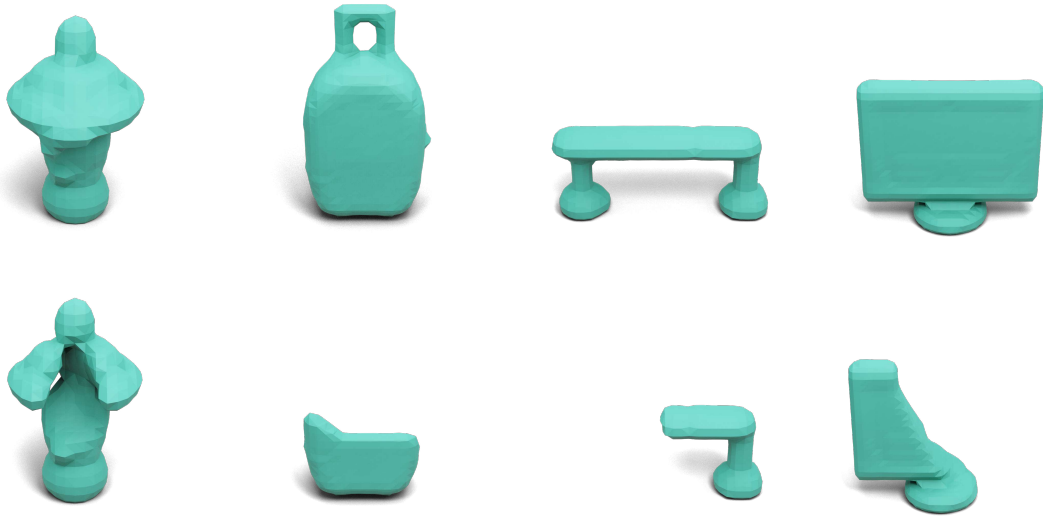


Figure 6.5: Ground truth models from the ShapeNet dataset (top row) alongside their incomplete counterparts (bottom row). Each model is reconstructed from the TSDF representation on a $32 \times 32 \times 32$ grid.

Table 6.2: Enumeration of complete shape counts within the ShapeNet dataset. Table shows the counts of complete shapes for the given tags in the dataset.

Tag	Count	Tag	Count	Tag	Count	Tag	Count
Bag	83	Dishwasher	93	Cabinet	551	Laptop	429
Stove	218	Chair	585	Pot	471	Table	591
Lamp	557	Bathtub	501	Microwave	152	Piano	239
Bench	557	Faucet	503	Washer	169	Trash Bin	343
Keyboard	64	Display	531	Bookshelf	421	Basket	113
Guitar	495	Bed	233	File Cabinet	297	Printer	166
Sofa	567	Bowl	184				
Total							9113

6.2.3 ModelNet

Another dataset used for out of distribution testing of known or unknown categories was ModelNet. ModelNet provides a comprehensive collection and structured categorization of 3D CAD models, making it an ideal benchmark to evaluate the performance of a shape completion task. ModelNet covers a wide range of common objects found in everyday environments.

Shape categories in ModelNet with their respective counts are in Table 6.3. Some selected model examples from the ModelNet dataset are shown in Figure 6.6. The parameter specifying the number of incomplete counterparts generated for each complete shape from the dataset was set to 5.

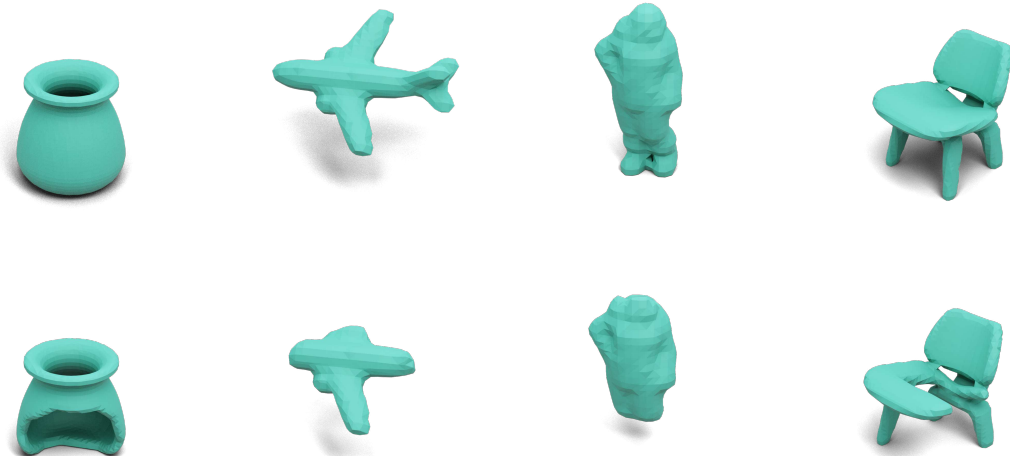


Figure 6.6: Ground truth models from the ModelNet dataset (top row) alongside their incomplete counterparts (bottom row). Each model is reconstructed from the TSDF representation on a $32 \times 32 \times 32$ grid.

Table 6.3: Enumeration of complete shape counts within the ModelNet dataset. Table shows the counts of complete shapes for the given tags in the dataset.

Tag	Count	Tag	Count	Tag	Count	Tag	Count
Airplane	17	Bookshelf	234	Cup	26	Mantel	332
Bathtub	39	Bottle	165	Curtain	66	Monitor	52
Bed	117	Bowl	42	Desk	52	Night Stand	94
Bench	44	Car	9	Door	12	Person	13
Chair	301	Cone	77	Dresser	59	Piano	27
Flower Pot	53	Glass Box	10	Guitar	5	Plant	107
Keyboard	29	Lamp	29	Laptop	14	Radio	9
Range Hood	168	Sink	38	Sofa	244	Stairs	19
Stool	17	Table	148	Tent	24	Toilet	207
TV Stand	90	Vase	237	Wardrobe	12	Xbox	8
Total							3246

6.3 Specification of Training Configurations

Instead of employing training paradigms such as pretraining by class, or pretraining on all classes through a generative task without any conditional input and then introducing conditions, an alternative approach is adopted. The network is trained from scratch, as mentioned in the DiffComplete [10] paper, the most effective approach for accurate shape completion.

To facilitate the reproducibility of this study, this section outlines the configurations of the training parameters for the most effective setup of each proposed approach. The approaches used include *CompleteBase*, *CompleteBase* enhanced with Attention layers, and *CompleteBase* enhanced with downsample and upsample phases. Each approach is de-

scribed in Chapter 7, where the experiments were performed. All important parameters are summarized in Table 6.4.

Table 6.4: Summary of training configurations for proposed approaches. Table shows hyperparameters for the best-performing configurations.

Hyperparameter	Completion	Super Resolution	Completion LR Conditon
architecture design	<i>CompleteBase</i>	<i>CompleteBase</i> + Attention	<i>CompleteBase</i> + DOWN/UP
iteration count	200 000	2 000 000	2 000 000
learning rate	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$
diffusion steps	1 000	1 000	1 000
ema rate	0.9999	0.9999	0.9999
noise scheduler	Linear	Linear	Linear
sampler scheduler	Uniform	Uniform	Uniform
optimizer	AdamW	AdamW	AdamW
betas for optimizer	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)
weight decay	0.0	0.0	0.0
dropout	0.0	0.0	0.0
loss function	Rescaled MSE	Rescaled MSE	Rescaled MSE
expected input resolution	$32 \times 32 \times 32$	$64 \times 64 \times 64$	$64 \times 64 \times 64$
expected condition resolution	$32 \times 32 \times 32$	$64 \times 64 \times 64$	$32 \times 32 \times 32$
batch size	32	8	32
evaluation timestep respacing	DDIM 100	DDIM 100	DDIM 100
FiLM-like conditioning	✓	✓	✓

Chapter 7

Conducted Experiments and Achieved Results

The experiments first focused on the shape completion ability of the proposed method. Subsequently, the network’s interpretation of various conditions is explored by attempting to complete the same model using diverse inputs with different missing parts. Another experiment evaluates the network’s ability to complete shapes beyond the training distribution. To address the identified failure cases in the baseline approach, an additional input in the form of a *Region of Interest (RoI)* is introduced. All of these components constitute the *first axis of experimentation*, presented in Section 7.2.

The *second axis of experiments*, detailed in Section 7.3, focuses on experiments involving the spatial resolution of the input and output. This set of experiments tests the baseline approach in a higher spatial dimension. Subsequent experiments aim to improve results by separating the shape completion task from the super-resolution task, employing another network for super-resolution. The final experiment focuses on processing the input and condition in a lower resolution space, followed by an upscaling phase for the input to obtain results in a high-resolution space. The goal of these experiments is to identify a computationally sustainable and sufficiently accurate method for completing the shapes.

For the experiments conducted in this work, an NVIDIA A100 graphics card with 80GB of memory was utilized to ensure enough memory and computing power. The time to train each setup is provided in further sections. The inference times for the proposed method range from 3 to 10 seconds.

7.1 Evaluation Metrics

To ensure a comprehensive and unbiased quantitative evaluation, three metrics are utilized: **Intersection over Union (IoU)**, **Chamfer Distance (CD)**, and **Mean Absolute Error (L1 Loss)**. Each metric works with distinct data representations. A detailed explanation of each metric is presented in the following sections.

Intersection over Union (IoU)

IoU measures the overlap between two binary volumes, presented in a voxel grid: the predicted volume \mathcal{V}_p and the ground truth volume \mathcal{V}_{gt} . Volume is obtained by thresholding the representation $TSDf$, where all negative values are set to 1 and the others are set to 0.

IoU is defined as the size of the intersection divided by the size of the union of the two volumes:

$$IoU(\mathcal{V}_p, \mathcal{V}_{gt}) = \frac{|\mathcal{V}_p \cap \mathcal{V}_{gt}|}{|\mathcal{V}_p \cup \mathcal{V}_{gt}|}, \quad (7.1)$$

where $|\mathcal{V}_p \cap \mathcal{V}_{gt}|$ represents the number of voxels common to both \mathcal{V}_p and \mathcal{V}_{gt} , and $|\mathcal{V}_p \cup \mathcal{V}_{gt}|$ is the total number of unique voxels present in either \mathcal{V}_p or \mathcal{V}_{gt} .

Chamfer Distance (CD)

The Chamfer Distance is a measure of similarity between two point clouds, defined as the average distance between each point in one cloud and its nearest neighbor in the other cloud. For point clouds $\mathcal{P}_1 = \{x_i \in \mathbb{R}^3\}_{i=1}^n$ and $\mathcal{P}_2 = \{x_j \in \mathbb{R}^3\}_{j=1}^m$, the Chamfer Distance is calculated as follows:

$$CD(\mathcal{P}_1, \mathcal{P}_2) = \frac{1}{2n} \sum_{i=1}^n \|x_i - NN(x_i, \mathcal{P}_2)\| + \frac{1}{2m} \sum_{j=1}^m \|x_j - NN(x_j, \mathcal{P}_1)\|, \quad (7.2)$$

where $NN(x, \mathcal{P}) = \arg \min_{x' \in \mathcal{P}} \|x - x'\|$ represents the nearest neighbor function that finds the point x' in the point cloud \mathcal{P} that is closest to the point x . This metric emphasizes the average error across the surface of the shapes.

Mean Absolute Error (L1 Loss)

Mean Absolute Error, also known as L1 loss, quantifies the average magnitude of errors in a set of predictions without considering their direction. It is a measure of how close the predictions are to the ground truth data, where all individual differences are weighted equally in the average:

$$\mathcal{L}_1(P, T) = \frac{1}{N} \sum_{i=1}^N |P_i - GT_i|, \quad (7.3)$$

where P_i is the predicted value, GT_i is the true value of the voxel, and N is the total number of voxels. The result is the average of the absolute differences between the predicted and actual values. This metric is applied to the raw output of the process (TSDF) rather than being converted to another representation.

7.2 Evaluation Axis 1: Completion Ability of the Proposed Solution

The outcomes of the proposed solution for shape completion are discussed in this section, with an emphasis on its completion capability. The focus was primarily on basic shape completion scenarios, as well as out of distribution cases involving both known and unknown categories. An examination of the influence of imbalances within the training dataset was performed. Particularly the missing volume’s influence and its effects on shape completion were analyzed. Failures in shape completion were identified, and strategies were outlined to address them. The section concludes with the proposal of a solution to a common failure in shape completion, which involves integrating user input through a specified region of interest to improve the completion process.

7.2.1 Baseline Evaluation

Initially, the efficacy of the proposed method was examined in an attempt to replicate the results of DiffComplete using a specific setup. This involved the use of the default architecture described in Section 5.3, omitting the upscaling and downscaling phases and the incorporation of attention layers in the decoder and encoder blocks. This configuration was named *CompletionBase*, with its hyperparameters detailed in Table 6.4. The duration of the training was approximately 3 days. The progression of loss over the training iterations is visualized in Figure 7.1.

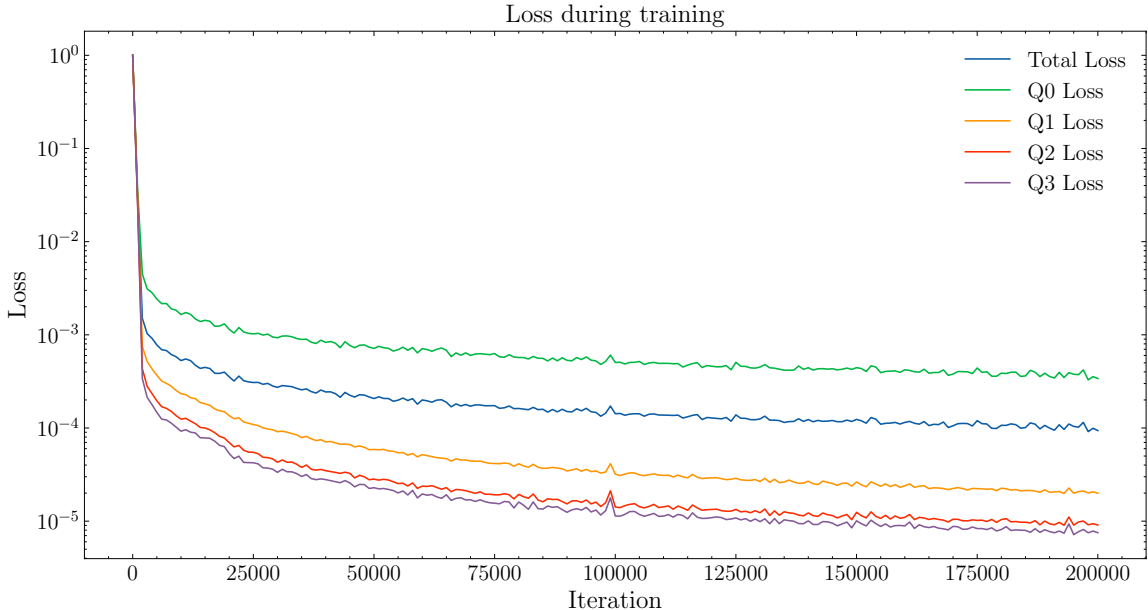


Figure 7.1: Training loss across diffusion timestamps. The Figure shows training loss over numerous iterations. Losses are categorized into quartiles. These quartiles correspond to timestamps of noise addition within the 0–1000 range of diffusion steps. Quartiles help visualize how the model’s loss varies at different points in the diffusion timeline.

A qualitative evaluation demonstrated the effectiveness of the technique in producing smooth mesh results after reconstruction. **In most cases, the predictions were accurate and aligned with the ground truth data.** However, when a shape lacked substantial volume, the network had to reconstruct the shape using minimal available features of the shape; therefore, in some cases, the output did not match the ground truth shape. **Here, the multi-modality of the network proved advantageous, enabling the generation of multiple predictions from which the most suitable could be selected.** The qualitative results are illustrated in Figure 7.2. Regarding quantitative metrics, the proposed solution demonstrated robust performance in both the quality of the results and the precision of the shape completions, as shown in Table 7.1. To further evaluate the baseline solution’s capabilities, a subsequent experiment was conducted to empirically assess how the network would manage the same object with varying missing sections and to determine whether the network could generalize or complete unknown categories.

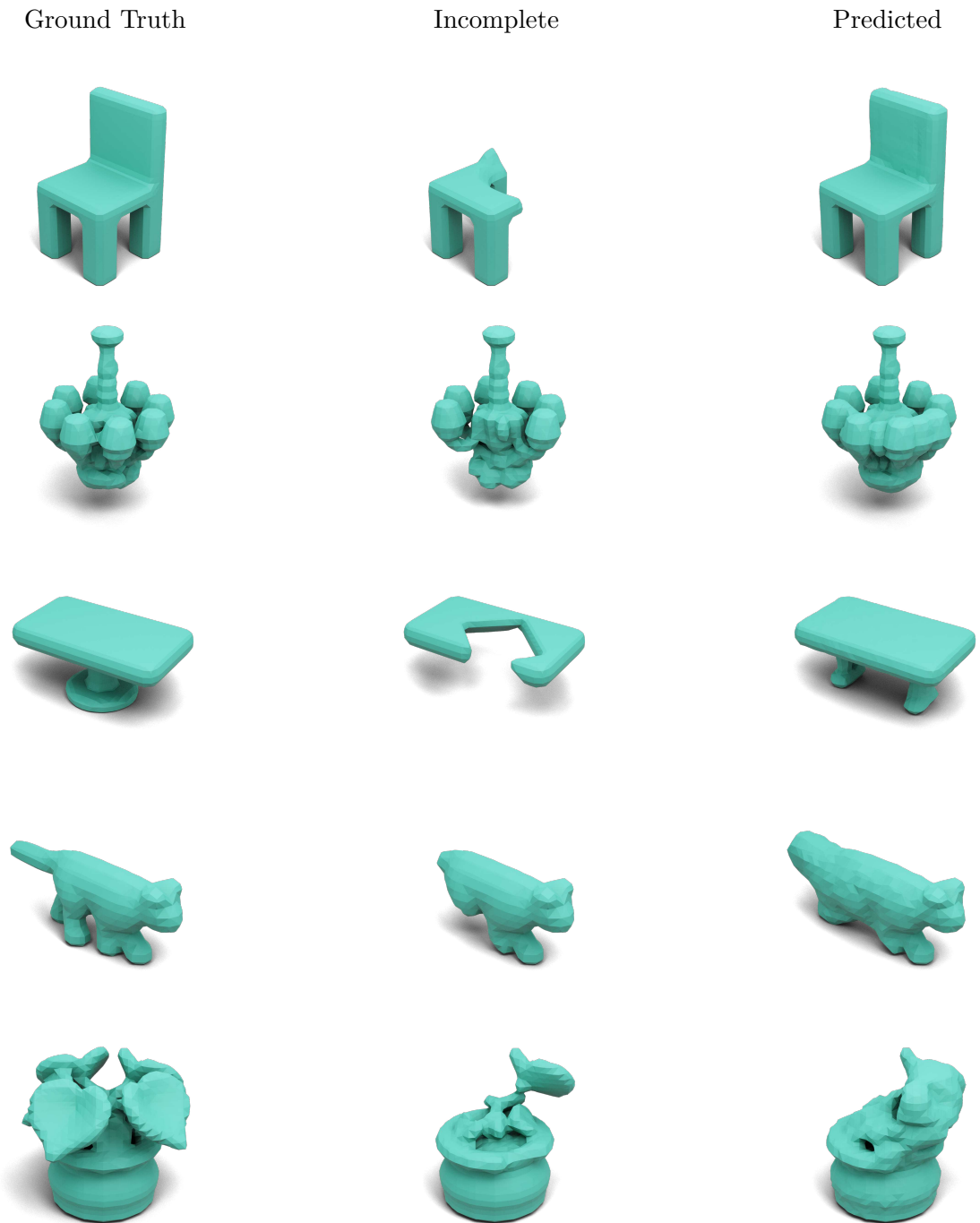


Figure 7.2: Visualization of qualitative results for shape completion baseline method. The figure illustrates the completed versions of incomplete shapes, alongside their corresponding ground truth. The initial three shapes were selected from the Objaverse *furniture* dataset. The next two shapes, a plant from ModelNet and a cat from the Objaverse *animals* datasets, represent out of distribution examples.

Table 7.1: Quantitative results for the shape completion baseline method. The shapes from Objaverse – Furniture belong to the same distribution as the training set, and the hold-out set is used for evaluation purposes. The results for all test datasets are presented, with the Intersection over Union (IoU) and Chamfer Distance (CD) metrics scaled by ($\times 10^2$).

Baseline Method $32 \times 32 \times 32$			
Dataset	Metrics		
	$IoU \uparrow (\times 10^2)$	$CD \downarrow (\times 10^2)$	$\mathcal{L}_1 \downarrow$
<u>Objaverse – Furniture</u>	81.62	3.53	0.026
Objaverse – Vehicles	76.05	4.21	0.035
Objaverse – Animals	70.46	5.48	0.052
ModelNet	63.34	5.93	0.055
ShapeNet	73.93	5.52	0.048

7.2.2 Network Perception of the Condition

To demonstrate the baseline solution’s capacity for shape completion, emphasis was placed on whether the network could identify objects even when their most distinctive features were absent. A chair model served as the basis for generating various incomplete shapes.

1. The initial test involved removing all legs of the chair to determine if the network could reconstruct the shape solely based on the remaining features of the chair. In this scenario, the network failed to accurately complete the chair, as the input shape did not provide any indication of missing parts.
2. Consequently, the next input was modified to include a larger missing section in the seat area, which suggested to the network the presence of a missing component. The results of this test were partially successful; the network performed the completion, but incorrectly, failing to identify the intended output as a chair.
3. To facilitate the task, the following input was adjusted to omit three legs, with a hint provided for the absent fourth leg. In this instance, the network completed the shape, though the result does not precisely match the ground truth. A subsequent attempt resulted in a shape that aligned with the ground truth, leveraging the network’s multi-modality.
4. The final test involved three absent legs and one intact leg, where the network correctly matched the ground truth shape within two attempts.

Each scenario was visually represented in Figure 7.3, with two predicted outputs for each input, using different noise volumes as input. These observations suggest that **the network struggles with shape completion in the absence of clear indications of missing parts, and relies heavily on the symmetry and repetitive elements** of the provided shapes. Moreover, for accurate completion, the shape should not be missing a significant part of the volume. Given the network’s reliance on shape symmetry, it may also have the potential to complete unknown categories. Therefore, the subsequent experiment focused on the completion of out of distribution shapes.

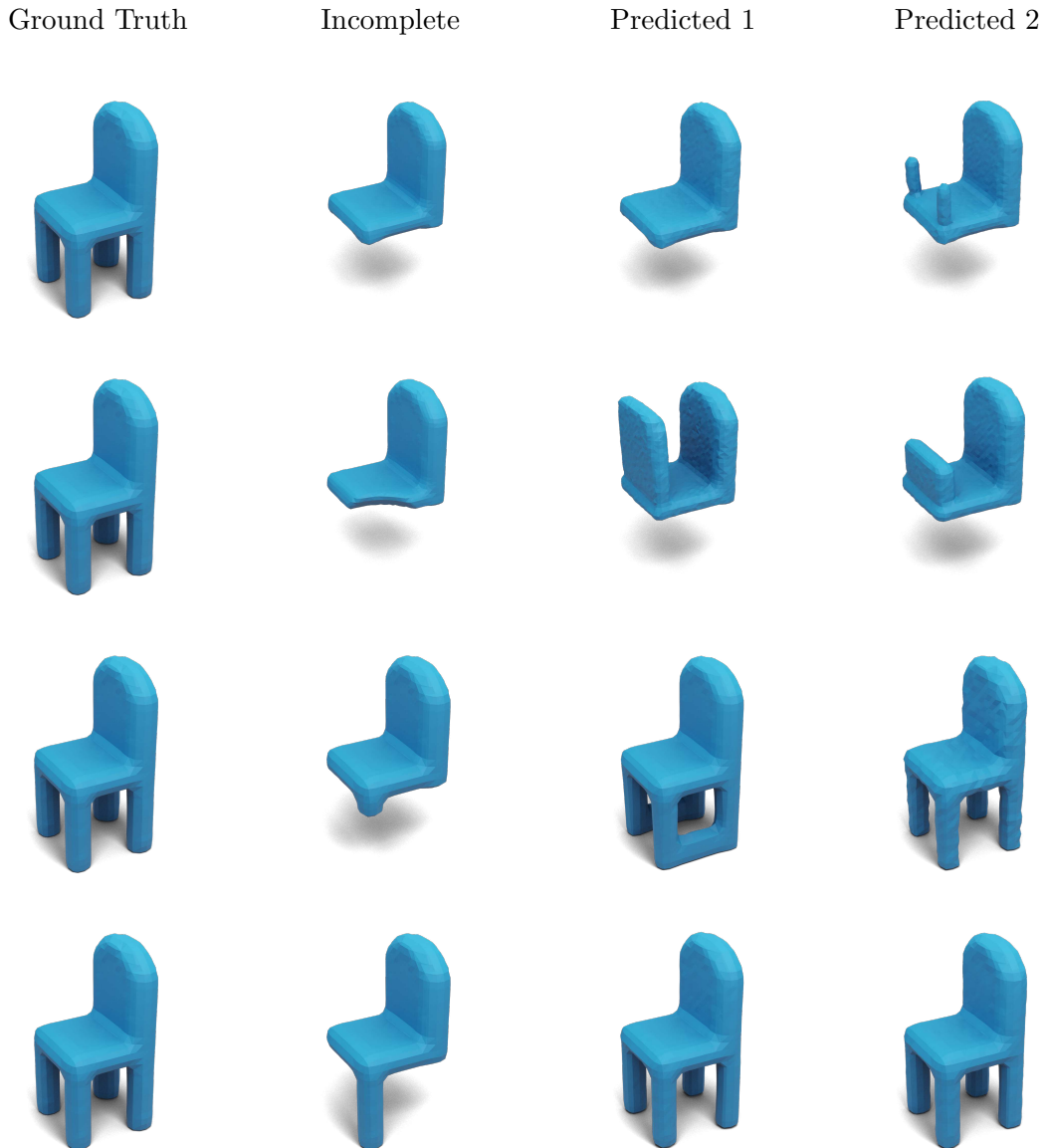


Figure 7.3: Visualization of the network’s perception in the input condition experiment Four cases are displayed: initially, the input condition lacks any legs. In the second case, a larger hole is introduced to signify a missing element. The third case features indications of missing legs. Finally, the presence of an intact leg is demonstrated, leading to a successful and accurate shape completion. For each incomplete shape, two predictions are generated using different noise volumes as input.

7.2.3 Out of Distribution Testing

The proposed solution demonstrates its effectiveness by utilizing the symmetry and repetitive elements of objects, suggesting that it could also complete shapes outside the training distribution. Observations from the last two rows of Figure 7.2 indicate that the **network is capable of partially completing even categories that were not previously known to it**. Quantitative results from additional datasets confirm the strong capacity of the network to generalize to out of distribution data, although within certain constraints, as shown

in Table 7.1. In interpreting the results, there is a decrease in the IoU when compared to known categories, but it is not too drastic, which might seem positive. However, the CD and L1 metrics indicate that while some missing volume is generated, **it does not align well with the geometry of the desired outcomes**. This problem is related to the low variability of the shape categories in the training dataset. Enhancing the dataset by incorporating a wider range of categories could improve the network’s ability to generalize more effectively.

7.2.4 Ability of Filling Smaller Holes

The diversity of the dataset significantly influences the network’s ability to generalize and complete unknown categories. A subsequent experiment aimed to investigate the effect of dataset imbalance, particularly in terms of missing volume in shapes, on the network’s shape completion performance. In this experiment, the network was trained exclusively on shapes with 60% to 90% missing volume and then evaluated on cases with only 10% to 50% missing volume. **The primary findings indicated that the imbalances in the training dataset markedly affect the completion results**. Specifically, the **network tended to overestimate missing parts**, reconstructing more than what was actually absent in the model. This tendency would presumably be mirrored in the reverse scenario. Table 7.2 shows the quantitative results for each dataset, accompanied by a comparison with the results obtained when the network was trained on a complete dataset.

Table 7.2: Quantitative analysis of shape completion with varied missing volumes vs. whole dataset. This table presents the outcomes of an experiment using the baseline method for shape completion, evaluated on shapes with 10% to 50% missing volume, compared to results achieved when the network was trained on the whole dataset.

Baseline Method 60% – 90% Missing Volume vs. Whole Dataset			
Dataset	Metrics (Missing Volume / Whole Dataset)		
	$IoU \uparrow (\times 10^2)$	$CD \downarrow (\times 10^2)$	$\mathcal{L}_1 \downarrow$
Objaverse – Furniture	75.65 / 85.30	4.41 / 3.25	0.038 / 0.022
Objaverse – Vehicles	73.51 / 79.93	4.61 / 3.64	0.038 / 0.026
Objaverse – Animals	71.76 / 78.46	5.51 / 4.39	0.054 / 0.040
ModelNet	65.22 / 75.20	5.70 / 4.11	0.051 / 0.030
ShapeNet	73.82 / 81.79	5.51 / 4.67	0.053 / 0.038

7.2.5 Analysis of Incorrect Completions

The problem of overfilling holes or not adequately filling the entire missing area does not arise solely from the imbalance of the training dataset. As illustrated in Figure 7.4, several types of failure scenarios have been identified for the proposed solution:

1. The first type of failure occurs when the **network partially fills the missing area of the shape, leaving the rest untouched**. This can also happen in instances where there is no indication of a missing part, as discussed above.
2. The second type of failure is observed in unknown categories, where the **network overcompensates for a small missing section, leading to excessive completion**.

3. The third scenario involves the network's failure to accurately reconstruct the shape, despite the potential to leverage repetitive elements and symmetry.
4. The final type of failure is largely attributed to the training dataset's nature and the rotation of objects within it. For example, a bench might be incorrectly completed as a chair shape due to the network's reliance on the orientation of the presented incomplete shape. This particular failure could be mitigated by rotating the input object or incorporating such variations into the training dataset.

To address the other types of failure, a method was employed that involved user input, specifically providing a region of interest.

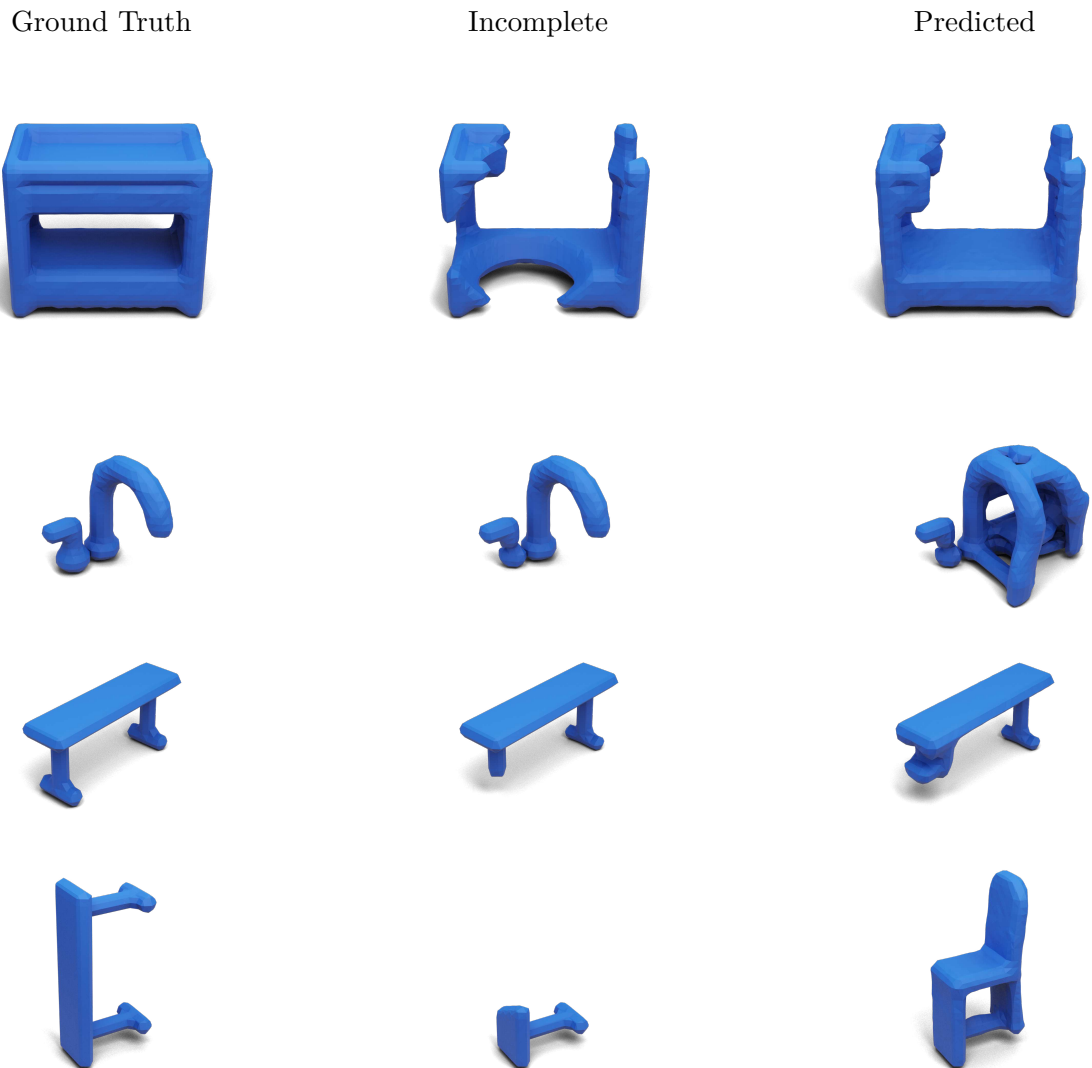


Figure 7.4: Visualization of fail cases in test scenarios. The figure highlights typical failure scenarios in shape completion, such as (from top-down) partial filling, overcompensation in unknown categories, inaccuracies despite symmetrical cues, and orientation-related errors stemming from the dataset's nature.

7.2.6 Enhancement via Region of Interest

To address the network’s shortcomings in achieving accurate shape completion, a novel approach was introduced, leveraging user input to utilize the **Region of Interest (RoI)**. This technique aims to direct the network towards the precise area that requires completion, thus facilitating more accurate reconstructions. Although this method diverges from the automatic shape completion technique, it is still valuable to explore the potential for improved results, given the minimal user input required to draw the RoI into the 3D scene.

In addition to standard input, the RoI was incorporated as a second channel, providing explicit information on the location of missing parts. The RoI for training purposes was constructed by identifying each voxel within the incomplete shape that differed from the corresponding ground truth shape, thus indicating missing volume. Then a bounding box was generated around these voxels to define the RoI.

More detailed technical information is available in code implementation. Figure 7.5 illustrates how the RoI method has successfully corrected previously unsuccessful cases from Figure 7.4. Furthermore, qualitative evaluations, as shown in Table 7.3, confirm that integrating **RoI significantly improves the performance of the shape completion process.**

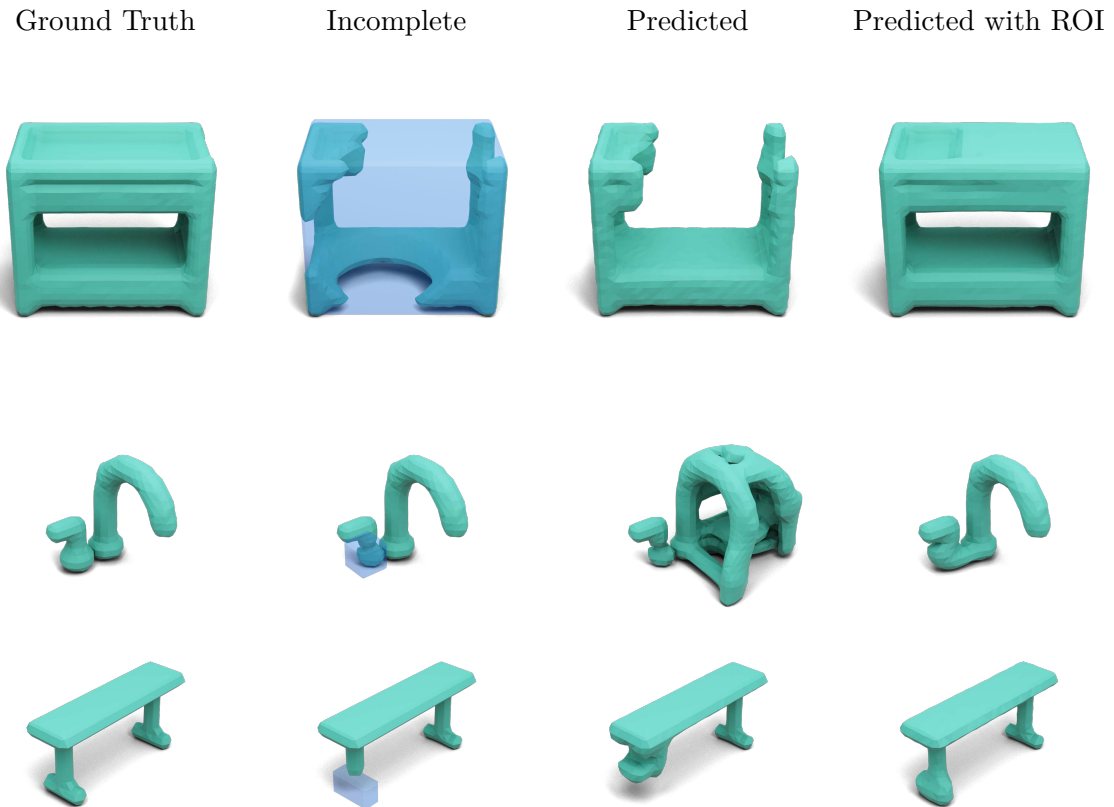


Figure 7.5: Shape completion results using region of interest. The figure demonstrates the impact of employing a region of interest (RoI) strategy to address the challenges of failure cases. The RoI is depicted as transparent blue boxes on incomplete shapes, providing visual cues for the areas that are targeted for completion.

Table 7.3: Quantitative results for the shape completion baseline method enhanced with region of interest. The results for all test datasets are presented, with the Intersection over Union (IoU) and Chamfer Distance (CD) metrics scaled by ($\times 10^2$).

Baseline method enhanced with RoI			
Dataset	Metrics		
	$IoU \uparrow (\times 10^2)$	$CD \downarrow (\times 10^2)$	$\mathcal{L}_1 \downarrow$
Objaverse – Furniture	84.77	2.86	0.018
Objaverse – Vehicles	81.73	3.24	0.023
Objaverse – Animals	76.48	4.32	0.037
ModelNet	69.11	5.11	0.043
ShapeNet	79.93	3.87	0.029

7.3 Evaluation Axis 2: Focus on Higher Resolution Results

This section describes experiments aimed at achieving higher-resolution output. Initially, the experiment involved uniform scaling of both the condition and the input to a higher resolution of $64 \times 64 \times 64$, up from $32 \times 32 \times 32$. Following this, the strategy of engaging an additional network to split the task to shape completion and super-resolution was tested. The concluding experiment explored the processing of the condition in a lower resolution space. This involved downscaling the input to the size of the condition and, once processed, upscaling it back to its original resolution.

7.3.1 High-Resolution Challenges

Due to the effectiveness of the suggested solution in completing shapes in low-resolution scenarios, it is expected to exhibit similar performance in higher-resolution settings. The primary challenge in executing shape completion lies in significant computational and memory demands. Consequently, when using a grid size $64 \times 64 \times 64$, the memory of an NVIDIA A100 graphics card could only accommodate a batch size of 8. During 2 000 000 iterations, the evaluation datasets exhibited considerable **noisy progression** without a clear trend towards improvement. **Training the network configuration that processed the inputs in $64 \times 64 \times 64$ resolution, along with other similar configurations, took approximately 30 days for each**, which is considerably long given the computational resources allocated for the training.

Although the networks did not achieve complete convergence, the results were reasonably acceptable. Figure 7.6 illustrates the visual results, showing the network’s ability to complete the shape, although significant noise is present. In certain instances, the network generated extremely noisy results. The quantitative results, documented in Table 7.4, corroborate the visual findings. To tackle the issue of network convergence, a subsequent experiment was designed to divide the tasks of completion and upscaling (super-resolution) between two networks.

7.3.2 Shape Completion utilizing Super Resolution

In deep learning, a common approach is to divide the task among several networks, which is frequently more successful than attempting to handle all aspects within a single network. To address incomplete convergence and to improve results, two separate networks were

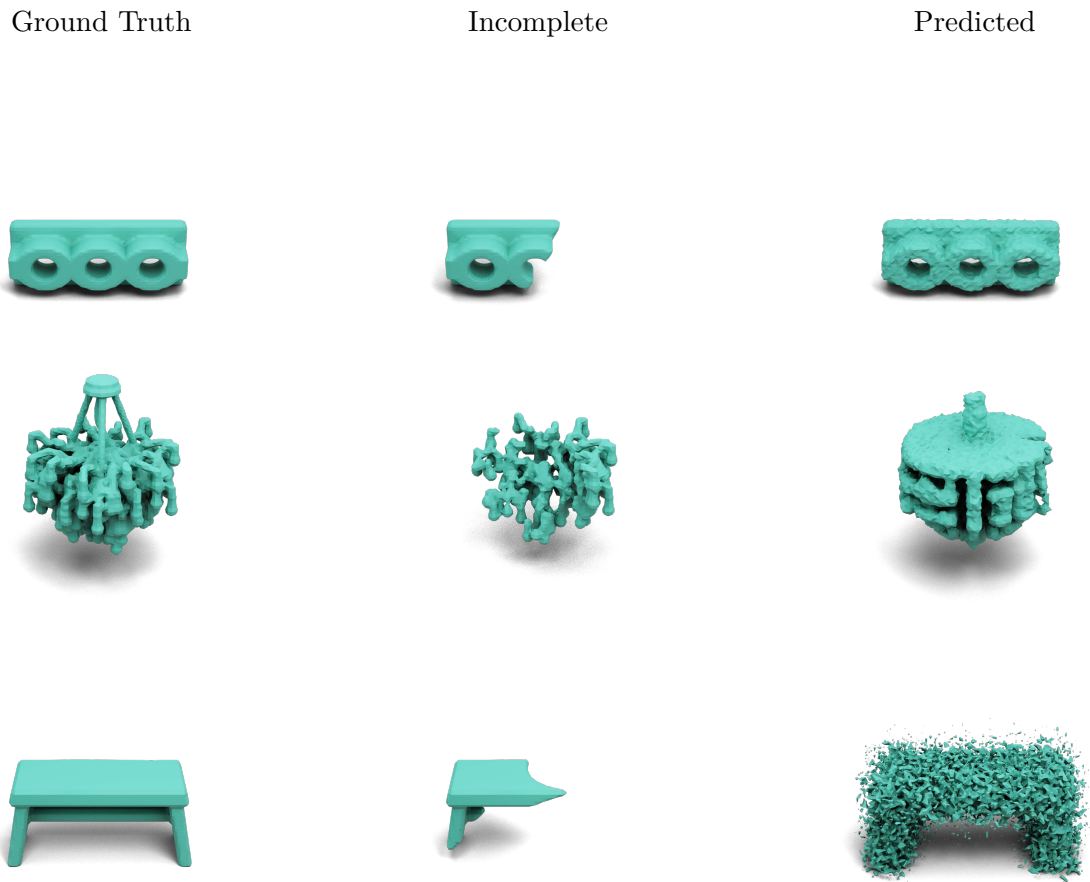


Figure 7.6: Visualization of qualitative results for baseline method in higher resolution space. The Figure illustrates the results obtained using the baseline method in a higher resolution space, highlighting the noisy outcomes.

Table 7.4: Quantitative results for the shape completion baseline method for $64 \times 64 \times 64$ voxel grid size. The results for all test datasets are presented, with the Intersection over Union (IoU) and Chamfer Distance (CD) metrics scaled by $(\times 10^2)$.

Baseline Method $64 \times 64 \times 64$			
Dataset	Metrics		
	$IoU \uparrow (\times 10^2)$	$CD \downarrow (\times 10^2)$	$\mathcal{L}_1 \downarrow$
<u>Objaverse – Furniture</u>	60.50	5.04	0.058

utilized: one for **completing shapes** and another for **super-resolution**. The expectation was that, by splitting the task, each network would achieve convergence more easily.

The architecture selected for shape completion was *CompleteBase*, which has shown promising results. For the super-resolution component, an improved architecture was adopted that incorporates *CompleteBase* and Attention Layers in the final two encoder and decoder blocks, as illustrated in Figure 5.5. The training began with pretraining the shape completion network, followed by freezing its weights, and then proceeding to train the super-resolution network. The output from the shape completion process was upscaled to a higher resolution and then inputted into the super-resolution network.

The qualitative results of this iterative two-step procedure are shown in Figure 7.7. Similarly to the basic completion task at higher resolutions, this method also struggled with full convergence. Consequently, the results were sometimes noisy, or the final shape **completely failed to replicate**. The quantitative data presented in Table 7.5 show that this method produced better outcomes than the baseline approach to complete the task at higher resolutions. However, the results were still unsatisfactory and could not reliably be applied in practical scenarios.

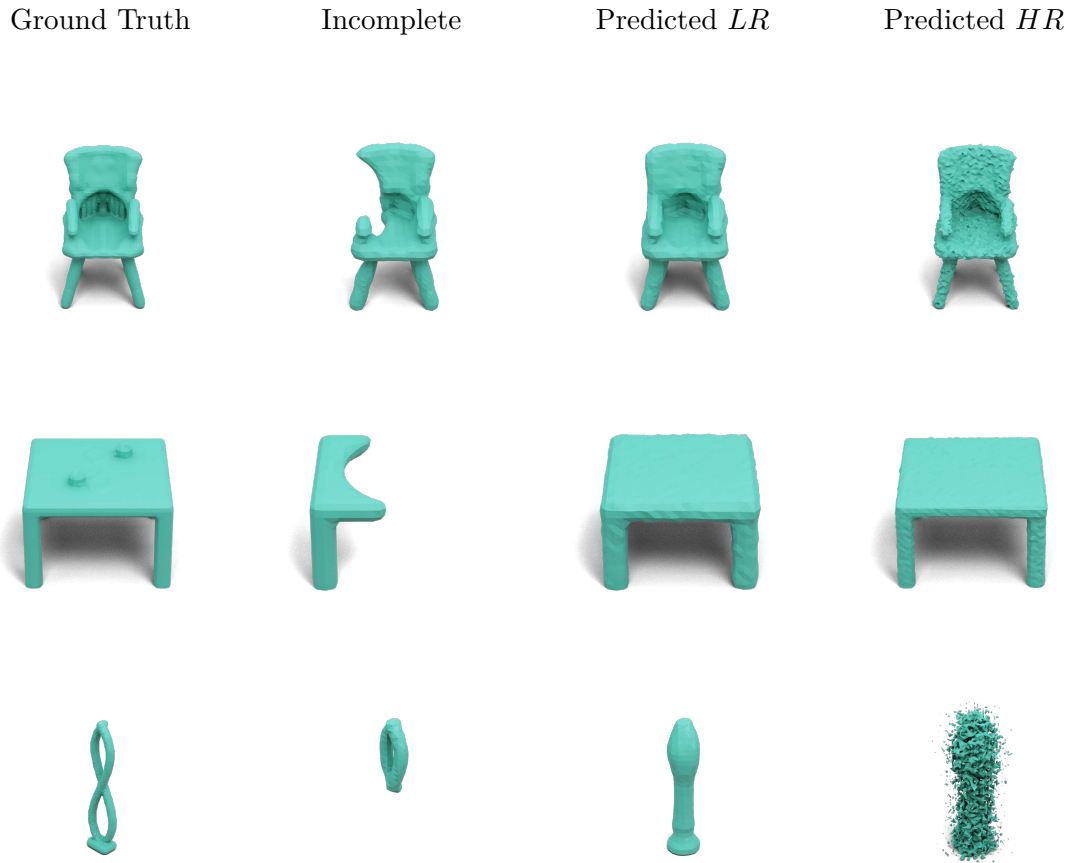


Figure 7.7: Visualization of qualitative results for method utilizing super resolution. The Figure illustrates the outcomes of the proposed two-step method, showcasing instances of noisy outputs and shape that are not completely finalized.

Table 7.5: Quantitative results for shape completion followed by super resolution. The results for all test datasets are presented, with the Intersection over Union (IoU) and Chamfer Distance (CD) metrics scaled by ($\times 10^2$).

Baseline \rightarrow Super Resolution			
Dataset	Metrics		
	$IoU \uparrow (\times 10^2)$	$CD \downarrow (\times 10^2)$	$\mathcal{L}_1 \downarrow$
Objaverse – Furniture	63.98	4.68	0.053
ShapeNet	61.15	6.43	0.071
ModelNet	49.69	6.02	0.067

7.3.3 Efficient Lower Resolution Processing

To address the challenges of insufficient memory for larger batch sizes and incomplete network convergence, an alternative approach was proposed. The fundamental idea is to receive input at a higher resolution, downscale it to match a condition at a lower resolution, handle it effectively at this lower resolution, and then upscale it again to a higher resolution. This method employs the *CompleteBase* architecture, incorporating downscaling and upscaling phases, as depicted in Figure 5.4.

Qualitative analysis revealed success. The network had an excellent ability to produce **smooth** results and **accurately complete** shapes from partial input, as demonstrated in Figure 7.8. However, as expected, the **absence of fine geometric details** in the condition meant that the **network’s generalization capability was not as robust**. In certain cases, the network did not consider the characteristics of the condition, resulting in the generation of different shapes.

This approach is highly dependent on the diversity of the training dataset. Given the limited detail in the conditional input, the network struggles to generalize to new categories. Expanding the dataset with a wider range of categories may mitigate this challenge. Another possible solution might be to utilize the approach of splitting the task between two networks. As illustrated in the last row of Figure 7.8, the low-resolution predictions closely align with the ground truth, potentially serving as inputs for super-resolution rather than generating entirely unrelated shapes. Quantitative outcomes, presented in Table 7.6, further corroborate the challenge of poor generalization, particularly with data that deviate from the training distribution.

Table 7.6: Quantitative results for shape completion leveraging processing the input in lower resolution. The results for all test datasets are presented, with the Intersection over Union (IoU) and Chamfer Distance (CD) metrics scaled by ($\times 10^2$).

Low resolution processing			
Dataset	Metrics		
	$IoU \uparrow (\times 10^2)$	$CD \downarrow (\times 10^2)$	$\mathcal{L}_1 \downarrow$
Objaverse – Furniture	73.09	3.84	0.032
ShapeNet	68.15	5.73	0.052
ModelNet	53.37	5.82	0.056

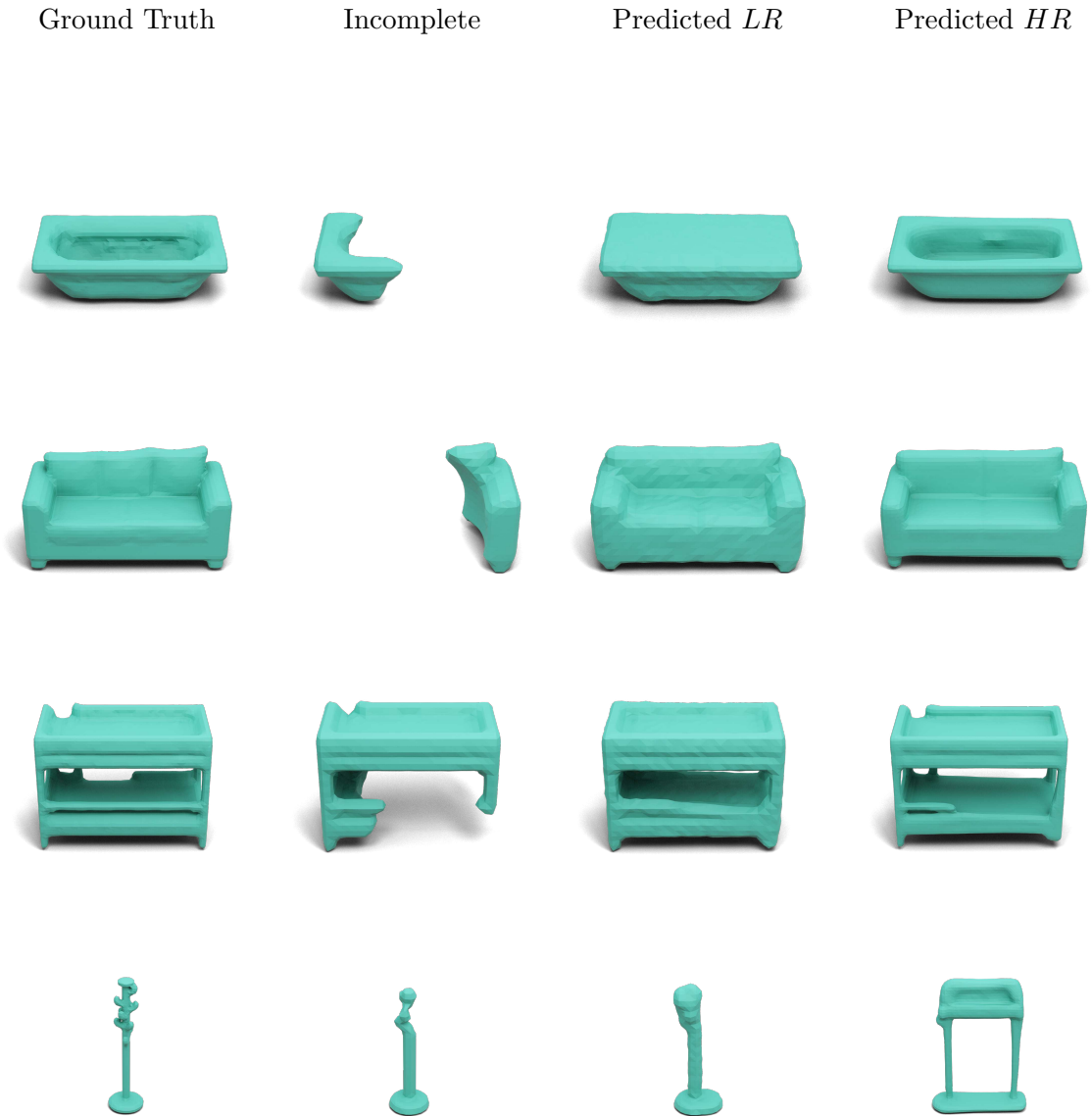


Figure 7.8: Visualization of qualitative results for method utilizing low-resolution processing. The Figure illustrates scenarios where the method successfully completes shapes smoothly and instances where it fails to accurately match the desired shape based on the condition. For comparative analysis, the Figure also displays the predicted shapes at low resolution, followed by their high-resolution counterparts achieved using this method, all derived from the same low-resolution condition.

7.4 Post-Processing Enhancements

Post-processing techniques, notably Laplacian smoothing, can mitigate the issue of noisy outcomes. To demonstrate the potential for enhancing results, two post-processing steps are outlined. The initial step addresses the noisy results through the application of Laplacian smoothing, which is a widely used method to smooth the surfaces of the mesh by averaging the positions of the mesh vertices based on their local neighborhoods (see Section 3.3.1).

The second step involves using a technique introduced by Chen *et al.* [8], which employs a deep neural network to facilitate conversion from TSDF representations to mesh. This technique, called Neural Dual Contouring (NDC), can provide potentially more accurate and detailed results than traditional methods such as Marching Cubes. Explaining the principles of NDC is, however, out of the scope of this thesis.

Figure 7.9 shows the influence of these post-processing methods, showing how Laplacian smoothing can decrease noise, and compares the results of the conventional Marching Cubes reconstruction with those obtained through NDC. This comparative visualization underscores the potential improvements in mesh quality and detail that can be achieved through these advanced post-processing techniques.



Figure 7.9: Visualization of post-processing methods. The Figure presents a comparison of results for a noisy mesh before and after 3 iterations of Laplacian smoothing. It also compares the results of reconstructing a couch using the Marching Cubes and the Neural Dual Contouring methods.

7.5 Summary of Results

Collectively, the results presented on both evaluation axes appear consistent with the good performance of the proposed solution.

The shape completion process demonstrated impressive results, notably reaching an IoU scores of 81.62, CD 3.53, and L1 0.026 when evaluated on the Objaverse furniture dataset. These results closely align with those reported in the Diffcomplete paper, on which this work is based. Despite the similarities, an exact duplication of the shape completion process was not achieved. This discrepancy is illustrated in Figure 7.10, where the diffusion process diverges from that described in the Diffcomplete paper. In particular, at step 80, significant noise persists, contrasting with the near-final shape clarity depicted in the Diffcomplete study. The difference might be due to the varying characteristics of the training datasets.

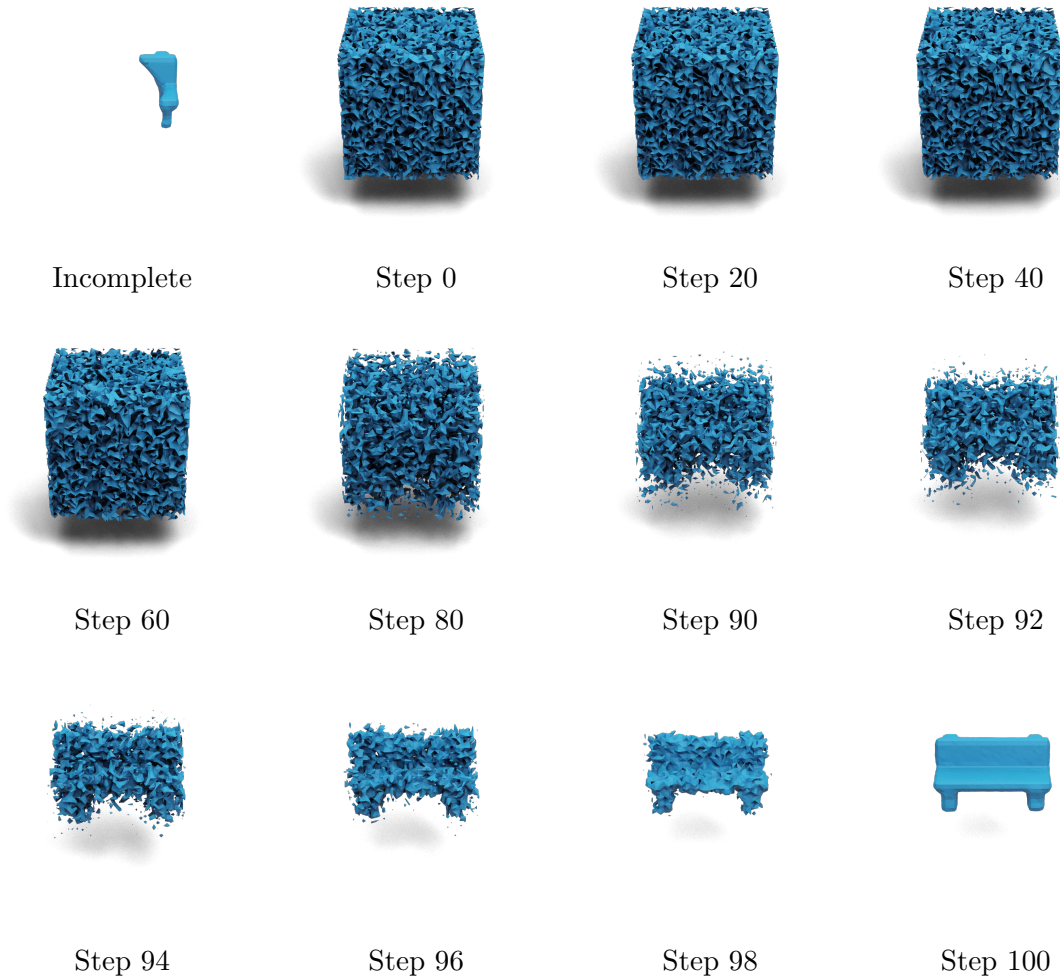


Figure 7.10: Visualization of denoising process. The denoising process gradually converts noise into the completed shape.

Furthermore, the model showed robust generalization abilities on shapes absent from the training distribution, averaging 70.9 IoU, CD 5.28, and L1 0.047 metric scores. The integration of user input greatly facilitated shape completion, resulting in an 84.7 IoU score and a reduced CD 2.86 and L1 0.018 on the test dataset. Regarding the datasets not part of the training distribution, the averaging scores were 76.81 for IoU, 4.13 for CD, and 0.033 for L1.

Subsequent experiments concentrated on high-resolution shape completion, where computational challenges were notably pronounced, especially with a grid size of $64 \times 64 \times 64$. Despite these obstacles, the baseline method demonstrated its ability to complete shapes, although sometimes producing noisy results as a result of incomplete convergence. The L1 metric, registering at 0.058 on the Objaverse dataset, indicates an error rate more than twice that observed in baseline for low-resolution.

To mitigate these issues, the experiments explored super-resolution techniques and a strategy involving efficient processing at lower resolutions. The super-resolution experiment utilized a dual-network approach, aiming to simplify convergence and improve accuracy, yet it occasionally encountered noise and precision issues. The L1 metric slightly lowered to 0.053, which is still a big error considered a low-resolution solution.

Efficient lower-resolution processing offered a novel approach by downscaling the input for processing and subsequently upscaling it, which achieved smoother and visually appealing results, with an L1 score of 0.032. However, this method faced challenges in preserving fine details and generalizing to new categories, emphasizing the critical role of training dataset diversity.

From an academic perspective, this work extends the foundational principles established in prior research of the DiffComplete framework by exploring the intricacies of shape completion across different resolutions and datasets, and also utilization of user interaction in the form of region of interest input. Moreover, the exploration of computational challenges associated with high-resolution shape completion and the innovative approaches proposed to mitigate these issues, such as super-resolution techniques and efficient lower-resolution processing. The ability to accurately complete 3D shapes has direct applications in computer graphics, 3D modeling, augmented reality, and others. Industries that depend on accurate 3D reconstructions, like entertainment, architecture, and medical imaging, have the potential to gain substantial advantages from improvements in shape-completion methods.

7.6 Future Work

Although the proposed solution exhibits strong results in shape completion, there is substantial room for improvement, particularly in addressing the high computational costs. Processing in a low-resolution space has been proposed as a way to mitigate these expenses. An additional enhancement could involve utilizing a condition in higher resolution, which could also be downscaled and processed in lower resolution. This approach has the potential to lead to significantly better results, capturing finer details and more accurately matching the initial condition of the incomplete shape.

Together with these strategies, the exploration of efficient 3D network modules, such as SparseConv [23] or Octree-based [66] layers, could offer a viable way to handle high-resolution 3D shapes without incurring prohibitive computational costs. Adopting these modules could help overcome the limitations currently faced by dense 3D CNN architectures, particularly the cubic increase in computational costs with volume size, thereby complementing the proposed low-resolution processing improvements.

Further research could focus on the diffusion process. The current diffusion process uses 1000 diffusion steps. Expanding this to evaluate the impact of a higher number of diffusion steps, such as 4000, could be beneficial. This increase might allow the model to progressively refine its understanding of shapes, potentially leading to more accurate and detailed completions.

Different sampling schedulers or noise schedulers could also be introduced to assess their impact on training efficiency and the quality of results. By experimenting with various sampling rates and noise schedules, the model may achieve a better balance between training time and completion accuracy.

Moreover, future work should continue to address the challenges of model generalizability to unseen object classes. The quality and diversity of training data are crucial factors that affect performance. Therefore, increasing the training dataset with a wider variety of shapes and classes, especially those significantly diverging from the training set, could be highly beneficial.

Chapter 8

Conclusion

This work aimed to create an automated process for the shape completion task using a supervised method based on deep learning. The proposed method is based on a diffusion process and deals with a shape completion problem as a generative task with conditional input, using a TSDF representation. DiffComplete [10] served as the basis for the proposed solution, which was enhanced by processing in a low-resolution space and integrating additional user input to identify the region of interest where the shape misses parts.

The experimental part of the work showed high capability of the proposed solution to complete incomplete shapes. The baseline results yielded an 81.6 IoU, CD 3.53, and L1 0.026 metric score on the test dataset. Furthermore, the model showed robust generalization abilities on shapes absent from the training distribution, averaging 70.9 IoU, CD 5.28, and L1 0.047 metric scores. The integration of user input greatly facilitated shape completion, resulting in an 84.7 IoU score and a reduced CD 2.86 and L1 0.018 on the test dataset. Regarding the datasets not part of the training distribution, the averaging scores were 76.81 for IoU, 4.13 for CD, and 0.033 for L1.

The method worked well when generating low-resolution outputs, but a more detailed geometry required a higher-resolution grid. With a cubical increase in computational power for an input, it was hard to obtain the resources to train such a big model. In this work, the trained models in the higher resolution did not fully converge. Using the novel approach, processing the data in a low-resolution space with subsequent upscale to the higher resolution showed success in obtaining smooth and precise results.

Although the proposed solution shows strong results in shape completion, there is substantial room for improvement. Exploring efficient 3D network modules, such as SparseConv [23] or Octree-based [66] layers, could offer a viable way to handle high-resolution 3D shapes without incurring prohibitive computational costs. Further research could focus on the diffusion process by experimenting with the number of iteration steps. Different sampling schedulers or noise schedulers could also be introduced to assess their impact on training efficiency and the quality of results. There is also an intention to modify the method, making it more suitable for medical data related to cranial implants. It should be noted that no literature has yet tried a similar approach.

Bibliography

- [1] ACHLIOPTAS, P.; DIAMANTI, O.; MITLIAGKAS, I. and GUIBAS, L. Learning representations and generative models for 3d point clouds. In: PMLR. *International conference on machine learning*. 2018, p. 40–49.
- [2] AHMED, E.; SAINT, A.; SHABAYEK, A. E. R.; CHERENKOVA, K.; DAS, R. et al. Deep Learning Advances on Different 3D Data Representations: A Survey. *CoRR*, 2018, abs/1808.01462. Available at: <http://arxiv.org/abs/1808.01462>.
- [3] ALBLAS, D.; BRUNE, C.; YEUNG, K. K. and WOLTERINK, J. M. Going off-grid: continuous implicit neural representations for 3D vascular modeling. In: Springer. *International Workshop on Statistical Atlases and Computational Models of the Heart*. 2022, p. 79–90.
- [4] CAI, G.; JIANG, Z.; WANG, Z.; HUANG, S.; CHEN, K. et al. Spatial Aggregation Net: Point Cloud Semantic Segmentation Based on Multi-Directional Convolution. *Sensors*, october 2019, vol. 19, p. 4329.
- [5] CHANG, A. X.; FUNKHOUSER, T.; GUIBAS, L.; HANRAHAN, P.; HUANG, Q. et al. *ShapeNet: An Information-Rich 3D Model Repository*. arXiv:1512.03012 [cs.GR]. Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.
- [6] CHANG, Z.; KOULIERIS, G. A. and SHUM, H. P. H. *On the Design Fundamentals of Diffusion Models: A Survey*. 2023.
- [7] CHEN, K.; CHOY, C. B.; SAVVA, M.; CHANG, A. X.; FUNKHOUSER, T. et al. Text2shape: Generating shapes from natural language by learning joint embeddings. In: Springer. *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*. 2019, p. 100–116.
- [8] CHEN, Z.; TAGLIASACCHI, A.; FUNKHOUSER, T. and ZHANG, H. Neural dual contouring. *ACM Transactions on Graphics*. Association for Computing Machinery (ACM), july 2022, vol. 41, no. 4, p. 1–13. ISSN 1557-7368. Available at: <http://dx.doi.org/10.1145/3528223.3530108>.
- [9] CHEN, Z. and ZHANG, H. Learning implicit fields for generative shape modeling. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, p. 5939–5948.
- [10] CHU, R.; XIE, E.; MO, S.; LI, Z.; NIESSNER, M. et al. Diffcomplete: Diffusion-based generative 3d shape completion. *ArXiv preprint arXiv:2306.16329*, 2023.

- [11] CURLESS, B. and LEVOY, M. A volumetric method for building complex models from range images. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 1996, p. 303–312.
- [12] DAI, A.; QI, C. R. and NIESSNER, M. *Shape Completion using 3D-Encoder-Predictor CNNs and Shape Synthesis*. 2017.
- [13] DE LUIGI, L.; CARDACE, A.; SPEZIALETTI, R.; RAMIREZ, P. Z.; SALTI, S. et al. Deep learning on implicit neural representations of shapes. *ArXiv preprint arXiv:2302.05438*, 2023.
- [14] DEITKE, M.; SCHWENK, D.; SALVADOR, J.; WEIHS, L.; MICHEL, O. et al. Objaverse: A Universe of Annotated 3D Objects. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, p. 13142–13153.
- [15] FAN, H.; SU, H. and GUIBAS, L. J. A point set generation network for 3d object reconstruction from a single image. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, p. 605–613.
- [16] FENG, C.; LIANG, J.; REN, M.; QIAO, G.; LU, W. et al. A Fast Hole-Filling Method for Triangular Mesh in Additive Repair. *Applied Sciences*, 2020, vol. 10, no. 3. ISSN 2076-3417. Available at: <https://www.mdpi.com/2076-3417/10/3/969>.
- [17] FENG, Y.; FENG, Y.; YOU, H.; ZHAO, X. and GAO, Y. Meshnet: Mesh neural network for 3d shape representation. In: *Proceedings of the AAAI conference on artificial intelligence*. 2019, vol. 33, no. 01, p. 8279–8286.
- [18] FIELD, D. A. Laplacian smoothing and Delaunay triangulations. *Communications in applied numerical methods*. Wiley Online Library, 1988, vol. 4, no. 6, p. 709–712.
- [19] GADELHA, M.; WANG, R. and MAJI, S. Multiresolution tree networks for 3d point cloud processing. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, p. 103–118.
- [20] GAO, L.; WU, T.; YUAN, Y.-J.; LIN, M.-X.; LAI, Y.-K. et al. Tm-net: Deep generative networks for textured meshes. *ACM Transactions on Graphics (TOG)*. ACM New York, NY, USA, 2021, vol. 40, no. 6, p. 1–15.
- [21] GM, H.; GOURISARIA, M. K.; PANDEY, M. and RAUTARAY, S. S. A comprehensive survey and analysis of generative models in machine learning. *Computer Science Review*, 2020, vol. 38, p. 100285. ISSN 1574-0137. Available at: <https://www.sciencedirect.com/science/article/pii/S1574013720303853>.
- [22] GOODFELLOW, I. J.; POUGET ABADIE, J.; MIRZA, M.; XU, B.; WARDE FARLEY, D. et al. *Generative Adversarial Networks*. 2014.
- [23] GRAHAM, B.; ENGELCKE, M. and VAN DER MAATEN, L. 3d semantic segmentation with submanifold sparse convolutional networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, p. 9224–9232.
- [24] GROUEIX, T.; FISHER, M.; KIM, V. G.; RUSSELL, B. C. and AUBRY, M. A papier-mâché approach to learning 3d surface generation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, p. 216–224.

- [25] GUO, M.-H.; CAI, J.-X.; LIU, Z.-N.; MU, T.-J.; MARTIN, R. R. et al. Pct: Point cloud transformer. *Computational Visual Media*. Springer, 2021, vol. 7, p. 187–199.
- [26] GUO, X.; XIAO, J. and WANG, Y. A survey on algorithms of hole filling in 3D surface reconstruction. *The Visual Computer*. Springer, 2018, vol. 34, p. 93–103.
- [27] HANOCKA, R.; HERTZ, A.; FISH, N.; GIRYES, R.; FLEISHMAN, S. et al. MeshCNN: a network with an edge. *ACM Trans. Graph.*, 2019, vol. 38, no. 4, p. 90:1–90:12. Available at: <https://doi.org/10.1145/3306346.3322959>.
- [28] HENDERSON, P.; TSIMINAKI, V. and LAMPERT, C. H. Leveraging 2d data to learn textured 3d mesh generation. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, p. 7498–7507.
- [29] HUANG, X.; LI, Y.; POURSAEED, O.; HOPCROFT, J. and BELONGIE, S. *Stacked Generative Adversarial Networks*. 2017.
- [30] JIANG, C.; MARCUS, P. et al. Hierarchical detail enhancing mesh-based shape generation with 3d generative adversarial network. *ArXiv preprint arXiv:1709.07581*, 2017.
- [31] JIANG, C.; SUD, A.; MAKADIA, A.; HUANG, J.; NIESSNER, M. et al. Local implicit grid representations for 3d scenes. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, p. 6001–6010.
- [32] JIANG, J.; BAO, D.; CHEN, Z.; ZHAO, X. and GAO, Y. MLVCNN: Multi-Loop-View Convolutional Neural Network for 3D Shape Retrieval. In: *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI Press, 2019. AAAI’19/IAAI’19/EAAI’19. ISBN 978-1-57735-809-1. Available at: <https://doi.org/10.1609/aaai.v33i01.33018513>.
- [33] KARRAS, T.; AILA, T.; LAINE, S. and LEHTINEN, J. *Progressive Growing of GANs for Improved Quality, Stability, and Variation*. 2018.
- [34] KARRAS, T.; AITTALA, M.; AILA, T. and LAINE, S. Elucidating the Design Space of Diffusion-Based Generative Models. In: KOYEJO, S.; MOHAMED, S.; AGARWAL, A.; BELGRAVE, D.; CHO, K. et al., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2022, vol. 35, p. 26565–26577. Available at: https://proceedings.neurips.cc/paper_files/paper/2022/file/a98846e9d9cc01cfb87eb694d946ce6b-Paper-Conference.pdf.
- [35] KINGMA, D.; SALIMANS, T.; POOLE, B. and HO, J. Variational Diffusion Models. In: RANZATO, M.; BEYGEZIMER, A.; DAUPHIN, Y.; LIANG, P. and VAUGHAN, J. W., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2021, vol. 34, p. 21696–21707. Available at: https://proceedings.neurips.cc/paper_files/paper/2021/file/b578f2a52a0229873fefc2a4b06377fa-Paper.pdf.
- [36] LANDIER, S. Boolean operations on arbitrary polygonal and polyhedral meshes. *Computer-Aided Design*. Elsevier, 2017, vol. 85, p. 138–153.

- [37] LE, T.; BUI, G. and DUAN, Y. A multi-view recurrent neural network for 3D mesh segmentation. *Computers & Graphics*, 2017, vol. 66, p. 103–112. ISSN 0097-8493. Available at: <https://doi.org/10.1016/j.cag.2017.05.011>. Shape Modeling International 2017.
- [38] LI, M. and ZHANG, H. D2im-net: Learning detail disentangled implicit fields from single images. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, p. 10246–10255.
- [39] LI, S.; LIU, M. and WALDER, C. EditVAE: Unsupervised parts-aware controllable 3D point cloud shape generation. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2022, vol. 36, no. 2, p. 1386–1394.
- [40] LORENSEN, W. E. and CLINE, H. E. Marching cubes: A high resolution 3D surface construction algorithm. In: *Seminal graphics: pioneering efforts that shaped the field*. 1998, p. 347–353.
- [41] MASCI, J.; BOSCAINI, D.; BRONSTEIN, M. M. and VANDERGHEYNST, P. Geodesic Convolutional Neural Networks on Riemannian Manifolds. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*. December 2015.
- [42] MEZGHANNI, M.; BOULKENAFED, M.; LIEUTIER, A. and OVSJANIKOV, M. Physically-aware generative network for 3d shape modeling. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, p. 9330–9341.
- [43] MIRZA, M. and OSINDERO, S. *Conditional Generative Adversarial Nets*. 2014.
- [44] MITTAL, P.; CHENG, Y.-C.; SINGH, M. and TULSIANI, S. Autosdf: Shape priors for 3d completion, reconstruction and generation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, p. 306–315.
- [45] MO, K.; WANG, H.; YAN, X. and GUIBAS, L. PT2PC: Learning to generate 3D point cloud shapes from part tree conditions. In: Springer. *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*. 2020, p. 683–701.
- [46] NICHOL, A. Q. and DHARIWAL, P. Improved denoising diffusion probabilistic models. In: PMLR. *International Conference on Machine Learning*. 2021, p. 8162–8171.
- [47] ODENA, A.; OLAH, C. and SHLENS, J. Conditional Image Synthesis with Auxiliary Classifier GANs. In: PRECUP, D. and TEH, Y. W., ed. *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 06–11 Aug 2017, vol. 70, p. 2642–2651. Proceedings of Machine Learning Research. Available at: <https://proceedings.mlr.press/v70/odena17a.html>.
- [48] PARK, J. J.; FLORENCE, P.; STRAUB, J.; NEWCOMBE, R. and LOVEGROVE, S. DeepSDF: Learning continuous signed distance functions for shape representation. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, p. 165–174.

- [49] PONTES, J. K.; KONG, C.; SRIDHARAN, S.; LUCEY, S.; ERIKSSON, A. et al. Image2mesh: A learning framework for single image 3d reconstruction. In: Springer. *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part I 14*. 2019, p. 365–381.
- [50] POURSAEED, O.; FISHER, M.; AIGERMAN, N. and KIM, V. G. Coupling explicit and implicit surface representations for generative 3d modeling. In: Springer. *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*. 2020, p. 667–683.
- [51] QI, C. R.; SU, H.; MO, K. and GUIBAS, L. J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21–26, 2017*. IEEE Computer Society, 2017, p. 77–85. Available at: <https://doi.org/10.1109/CVPR.2017.16>.
- [52] QI, C. R.; YI, L.; SU, H. and GUIBAS, L. J. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, USA*. 2017, p. 5099–5108. Available at: <https://proceedings.neurips.cc/paper/2017/hash/d8bf84be3800d12f74d8b05e9b89836f-Abstract.html>.
- [53] RADFORD, A.; METZ, L. and CHINTALA, S. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016.
- [54] RAO, Y.; NIE, Y. and DAI, A. Patchcomplete: Learning multi-resolution patch priors for 3d shape completion on unseen categories. *Advances in Neural Information Processing Systems*, 2022, vol. 35, p. 34436–34450.
- [55] REQUICHA, A. A. and VOELCKER, H. B. Boolean operations in solid modeling: Boundary evaluation and merging algorithms. *Proceedings of the IEEE*. IEEE, 1985, vol. 73, no. 1, p. 30–44.
- [56] RIEGLER, G.; OSMAN ULUSOY, A. and GEIGER, A. Octnet: Learning deep 3d representations at high resolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, p. 3577–3586.
- [57] SHI, Y.; NI, B.; LIU, J.; RONG, D.; QIAN, Y. et al. Geometric granularity aware pixel-to-mesh. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, p. 13097–13106.
- [58] SONG, J.; MENG, C. and ERMON, S. *Denoising Diffusion Implicit Models*. 2022.
- [59] SONG, Y. and ERMON, S. Generative Modeling by Estimating Gradients of the Data Distribution. In: WALLACH, H.; LAROCHELLE, H.; BEYGELZIMER, A.; ALCHÉ BUC, F. d'; FOX, E. et al., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019, vol. 32. Available at: https://proceedings.neurips.cc/paper_files/paper/2019/file/3001ef257407d5a371a96dcd947c7d93-Paper.pdf.

- [60] SU, H.; MAJI, S.; KALOGERAKIS, E. and LEARNED-MILLER, E. G. Multi-view Convolutional Neural Networks for 3D Shape Recognition. In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. 2015, p. 945–953. Available at: <https://doi.org/10.1109/ICCV.2015.114>.
- [61] SUN, Y.; WANG, Y.; LIU, Z.; SIEGEL, J. and SARMA, S. Pointgrow: Autoregressively learned point cloud generation with self-attention. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020, p. 61–70.
- [62] TANG, Y.; QIAN, Y.; ZHANG, Q.; ZENG, Y.; HOU, J. et al. WarpingGAN: Warping multiple uniform priors for adversarial 3D point cloud generation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, p. 6397–6405.
- [63] VOLLMER, J.; MENCL, R. and MUELLER, H. Improved laplacian smoothing of noisy surface meshes. In: Wiley Online Library. *Computer graphics forum*. 1999, vol. 18, no. 3, p. 131–138.
- [64] WANG, N.; ZHANG, Y.; LI, Z.; FU, Y.; LIU, W. et al. Pixel2mesh: Generating 3d mesh models from single rgb images. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, p. 52–67.
- [65] WANG, P.-S.; LIU, Y.; GUO, Y.-X.; SUN, C.-Y. and TONG, X. O-CNN. *ACM Transactions on Graphics*. Association for Computing Machinery (ACM), jul 2017, vol. 36, no. 4, p. 1–11. Available at: <https://doi.org/10.1145%2F3072959.3073608>.
- [66] WANG, P.-S.; LIU, Y.; GUO, Y.-X.; SUN, C.-Y. and TONG, X. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions On Graphics (TOG)*. ACM New York, NY, USA, 2017, vol. 36, no. 4, p. 1–11.
- [67] WANG, P.-S.; LIU, Y. and TONG, X. Dual Octree Graph Networks for Learning Adaptive Volumetric Shape Representations. *ACM Transactions on Graphics (SIGGRAPH)*, 2022, vol. 41, no. 4.
- [68] WERNER, D.; AL HAMADI, A. and WERNER, P. Truncated Signed Distance Function: Experiments on Voxel Size. In: October 2014, vol. 8815, p. 357–364. ISBN 978-3-319-11754-6.
- [69] WU, J.; WANG, Y.; XUE, T.; SUN, X.; FREEMAN, B. et al. Marrnet: 3d shape reconstruction via 2.5 d sketches. *Advances in neural information processing systems*, 2017, vol. 30.
- [70] WU, J.; ZHANG, C.; XUE, T.; FREEMAN, B. and TENENBAUM, J. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *Advances in neural information processing systems*, 2016, vol. 29.
- [71] WU, Z.; SONG, S.; KHOSLA, A.; YU, F.; ZHANG, L. et al. 3D ShapeNets: A deep representation for volumetric shapes. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2015, p. 1912–1920. Available at: <https://doi.org/10.1109/CVPR.2015.7298801>.

- [72] WU, Z.; SONG, S.; KHOSLA, A.; YU, F.; ZHANG, L. et al. 3d shapenets: A deep representation for volumetric shapes. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, p. 1912–1920.
- [73] WU, Z.; PAN, S.; CHEN, F.; LONG, G.; ZHANG, C. et al. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*. IEEE, 2020, vol. 32, no. 1, p. 4–24.
- [74] XIAO, L.; YANG, G.; YANG, K. and MEI, G. Efficient Parallel Algorithms for 3D Laplacian Smoothing on the GPU. *Applied Sciences*, december 2019.
- [75] XIAO, Y.-P.; LAI, Y.-K.; ZHANG, F.-L.; LI, C. and GAO, L. *A Survey on Deep Geometry Learning: From a Representation Perspective*. 2020.
- [76] XIE, H.; YAO, H.; SUN, X.; ZHOU, S. and ZHANG, S. Pix2vox: Context-aware 3d reconstruction from single and multi-view images. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, p. 2690–2698.
- [77] XIE, J.; ZHENG, Z.; GAO, R.; WANG, W.; ZHU, S.-C. et al. Generative VoxelNet: learning energy-based models for 3D shape synthesis and analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. IEEE, 2020, vol. 44, no. 5, p. 2468–2484.
- [78] XU, Q.-C.; MU, T.-J. and YANG, Y.-L. A survey of deep learning-based 3D shape generation. *Computational Visual Media*. Springer, 2023, vol. 9, no. 3, p. 407–442.
- [79] YAN, X.; LIN, L.; MITRA, N. J.; LISCHINSKI, D.; COHEN OR, D. et al. Shapeformer: Transformer-based shape completion via sparse representation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, p. 6239–6249.
- [80] YANG, G.; HUANG, X.; HAO, Z.; LIU, M.-Y.; BELONGIE, S. et al. Pointflow: 3d point cloud generation with continuous normalizing flows. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, p. 4541–4550.
- [81] YANG, S.; XU, M.; XIE, H.; PERRY, S. and XIA, J. Single-view 3D object reconstruction from shape priors in memory. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, p. 3152–3161.
- [82] YANG, X.; WU, Y.; ZHANG, K. and JIN, C. CPCGAN: A controllable 3D point cloud generative adversarial network with semantic label generating. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2021, vol. 35, no. 4, p. 3154–3162.
- [83] YANG, Y.; FENG, C.; SHEN, Y. and TIAN, D. Foldingnet: Point cloud auto-encoder via deep grid deformation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, p. 206–215.
- [84] ZHANG, L.; RAO, A. and AGRAWALA, M. Adding Conditional Control to Text-to-Image Diffusion Models. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. October 2023, p. 3836–3847.

- [85] ZHANG, L.; RAO, A. and AGRAWALA, M. Adding conditional control to text-to-image diffusion models. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, p. 3836–3847.
- [86] ZHAO, W.; GAO, S. and LIN, H. A robust hole-filling algorithm for triangular mesh. *The Visual Computer*. Springer, 2007, vol. 23, p. 987–997.
- [87] ZHU, J.-Y.; PARK, T.; ISOLA, P. and EFROS, A. A. Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct 2017.

Appendix A

Contents of the Included Storage Media

- `dataset_processing/` Folder containing files related to dataset generation.
- `datasets/` Folder with dataset for training, validation, and testing.¹
- `model/` Folder containing the implementation of the model.
- `evaluation/` Folder containing files related to the evaluation of the model.
- `scripts/` Folder containing scripts for training, testing, and sampling.
- `pretrained-models/` Folder with pretrained models.
- `latex/` Folder with L^AT_EX source files.
- `figs/` Folder containing figures showing examples.
- `LICENCE` Project licence.
- `README.md` README file for the project.
- `requirements.txt` Python libraries dependencies.
- `poster.pdf` Poster.
- `thesis.pdf` Thesis report file.
- `thesis-print.pdf` Thesis report file for print.

¹Due to the extensive size of the datasets, only a few samples are provided, and the `README.md` provides the steps for generating the datasets from scratch.

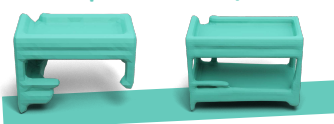
Appendix B

Poster

GENERATIVE MODELS FOR 3D SHAPE COMPLETION

Author: Peter Zdravecký
Supervisor: Tibor Kubík

Incomplete
Completed



Motivation and Proposed Method

The goal is to automatically complete 3D shapes based on the incomplete input using deep learning techniques. In many real world scenarios, scanned 3D models contain missing parts due to occlusion, scanning errors or the incomplete nature of the data itself.

The proposed solution is to use a diffusion-based model and handle the task as a generative problem to create a complete shape from the incomplete one.

Forward process:

$$q(x_{0:T}) = q(x_0) \prod_{t=1}^T q(x_t | x_{t-1}), \quad q(x_t | x_{t-1}) := \mathcal{N}(\sqrt{1 - \beta_t} x_{t-1}, \beta_t \mathbf{I}).$$

Backward process:

$$p_\theta(x_{0:T}, c) = p(x_T) \prod_{t=1}^T p_\theta(x_{t-1} | x_t, c), \quad p_\theta(x_{t-1} | x_t) := \mathcal{N}(\mu_\theta(x_t, t, c), \sigma_\theta^2 \mathbf{I}).$$

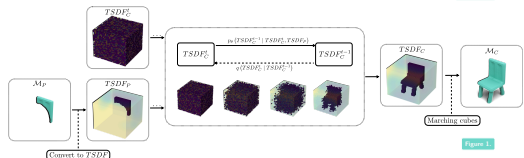


Figure 1

Backward process is modeled using a two-branch architecture utilizing 3D UNet, to handle the input and condition.

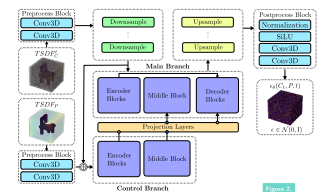


Figure 2

1. Preprocess the input to a higher-dimensional space.
2. Downsample the input to spatial resolution of condition.
3. Process input/condition using two-branch 3D UNet.
4. Upsample output back to the original resolution.
5. Cast output to a lower-dimensional space.

Results and Conclusion

Experimental results show high capability of this model in shape completion task with high score of IoU for chosen datasets. The model possesses a strong ability to make use of the repetitive shape parts to adapt to data out of the training distribution. To enhance the generative process, the Region of Interest can be utilized to define the area of the missing parts. Additional experiments focused on generating results in higher resolution. A method was proposed for this purpose that uses low-resolution processing followed by upscaling process.

Input
Completed
Ground Truth

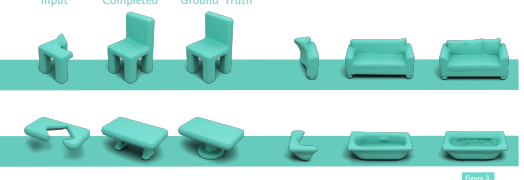


Figure 3

The produced results from the automated shape completion are very promising for real world use. However, the inference time takes approximately 3-5 seconds, therefore shape completion in real time is currently impossible.

Dataset	Metrics		
	IoU ↑ ($\times 10^2$)	CD ↓ ($\times 10^2$)	\mathcal{L}_1 ↓
Objaverse - Furniture	81.62	3.53	0.026
Objaverse - Vehicles	76.05	4.21	0.035
Objaverse - Animals	70.46	5.48	0.052
ModelNet	63.34	5.93	0.055
ShapeNet	73.93	5.52	0.048

Quantitative results on multiple datasets.

Table 1

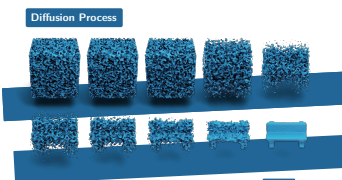


Figure 4

Inference captured in different timestamp of backward diffusion process.