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

DEEP LEARNING FOR 3D GEOMETRY ANALYSIS IN MEDICINE

HLUBOKÉ UČENÍ PRO ANALÝZU 3D GEOMETRIE V MEDICÍNĚ

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **Bc. TIBOR KUBÍK**
AUTOR PRÁCE

SUPERVISOR **Ing. MICHAL ŠPANĚL, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2023

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Master's Thesis Assignment

145039

Institut:	Department of Computer Graphics and Multimedia (UPGM)
Student:	**Kubík Tibor, Bc.**
Programme:	Information Technology and Artificial Intelligence
Specialization:	Computer Vision
Title:	**Deep Learning for 3D Geometry Analysis in Medicine**
Category:	Artificial Intelligence
Academic year:	2022/23

Assignment:

1. Get familiar with existing deep learning models suitable for 3D geometry processing - graph neural networks, multi-view convolutional neural networks, etc.
2. Get acquainted with some typical problem of 3D medical data analysis like surface mesh segmentation.
3. For the chosen problem, prepare a dataset for your own experiments.
4. Implement chosen models and experiment with them.
5. Evaluate and compare your results using appropriate metrics. Discuss possible future work.
6. Create a short poster or video presenting your work, its goals and results.

Literature:

- Su, H., Maji, S., Kalogerakis, E. and Learned-Miller, E., "Multi-view Convolutional Neural Networks for 3D Shape Recognition," 2015 IEEE International Conference on Computer Vision (ICCV), 2015, https://arxiv.org/abs/1505.00880.
- Wu, W., Qi, Z. and Li, F., "PointConv: Deep Convolutional Networks on 3D Point Clouds", 2018, https://arxiv.org/abs/1811.07246.
- Hanocka, R., et al. "MeshCNN: a network with an edge." ACM Trans. Graph, 2019, https://arxiv.org/abs/1809.05910.

Requirements for the semestral defence:

- The first three items of the assignment.

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

Supervisor:	**Španěl Michal, Ing., Ph.D.**
Consultant:	Ing. Oldřich Kodym, Ph.D.
Head of Department:	Černocký Jan, prof. Dr. Ing.
Beginning of work:	1.11.2022
Submission deadline:	31.7.2023
Approval date:	3.11.2022

Abstract

Deep neural network architectures designed for traditional signals like regularly sampled images and grids do not straightforwardly translate to irregularly sampled and triangulated surfaces, point clouds, and other geometric representations. As the acquisition tools that produce such 3D data are becoming broadly available, this generalization is increasingly needed for treatment planning in digital medicine. This work aims to examine the application of deep learning techniques for the analysis of triangular meshes. A recurrent multi-view approach is proposed for the task of segmentation of teeth in surface dental scans, an industry-desirable automation. On complex real-world orthodontic cases containing dental irregularities or scanned appliances, the proposed method outperforms both the conventional segmentation algorithm based on 3D Graph-Cut, and non-Euclidean methods that analyze point clouds or directly meshes. It achieves an average weighted IoU score of 0.966 and Hausdorff distance at 95 percentile of 0.382 mm. The results are promising for a deployment in dental planning software, enabling clinicians to streamline their workflow and devote more attention and focus on the treatment itself.

Abstrakt

Architektúry hlbokých neurónových sietí navrhnuté pre tradičné signály, ako sú pravidelne vzorkované obrázky a mriežky, sa nedajú priamo previesť na geometrické reprezentácie s nepravidelným charakterom, ako napríklad triangulované povrchy či mračná bodov. Keďže nástroje, ktoré produkujú tieto 3D dáta sú čoraz dostupnejšie, toto rozšírenie je pre plánovanie zákrokov v digitálnej medicíne stále viac a viac potrebné. Cieľom tejto práce je preskúmať využitie techník hlbokého učenia na analýzu trojuholníkových sietí. Na úlohu automatickej segmentácie zubov v povrchových skenoch čeľustí, automatizácie žiadanej v priemysle, je v tejto práci navrhnutý a vyhodnotený prístup založený na rekurentných viacpohľadových neurónových sieťach. Tento algoritmus prekonáva jednak konvenčný segmentačný algoritmus založený na metóde 3D Graph-Cutu, rovnako ako aj iné neeuklidovské metódy spracovávajúce mračná bodov či priamo štruktúru mešov. Na komplexných ortodontických prípadoch obsahujúcich skeny s krivými zubami či naskenovanými ortodontickými aparátmi dosahuje navrhnutý prístup hodnôt 0,966 na metrike váhovaného IoU a 0,382 mm na metrike 95 percentilu Hausdorffovej vzdialenosti. Výsledky sú sľubné pre nasadenie do softvéru na plánovanie zubných zákrokov, čo by zubným lekárom zefektívnilo pracovný postup a umožnilo im venovať viac pozornosti na samotnú liečbu.

Keywords

mesh segmentation, teeth segmentation, dental scans, geometric deep learning, recurrent multi-view neural networks, ConvLSTM, PointNet, PointNet++, MeshCNN

Kľúčové slová

segmentácia meshov, segmentácia zubov, dentálne skeny, geometric deep learning, rekurentné viacpohľadové neurónové siete, ConvLSTM, PointNet, PointNet++, MeshCNN

Reference

KUBÍK, Tibor. *Deep Learning for 3D Geometry Analysis in Medicine.* Brno, 2023. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Michal Španěl, Ph.D.

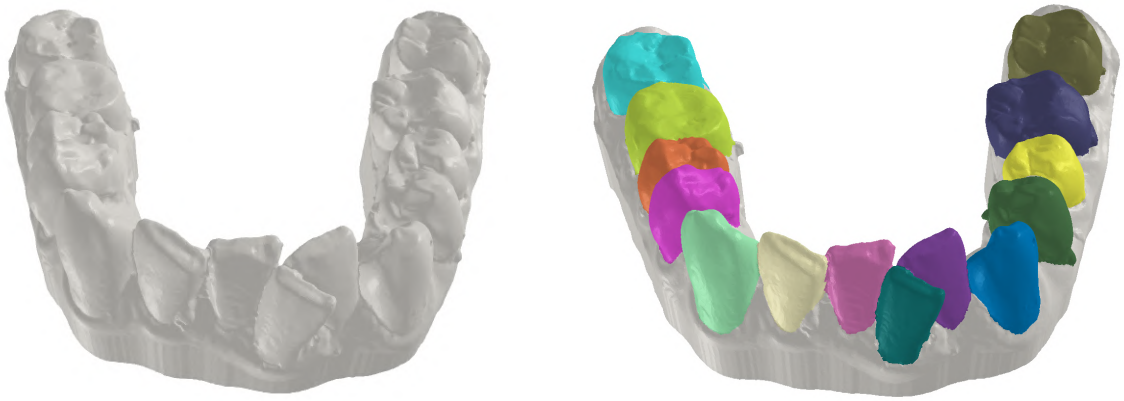
Rozšírený abstrakt

Úvod a cieľ práce

Optické skenery produkujúce 3D povrchové modely čeľustí sú čoraz bežnejšou súčasťou moderných zubných kliník. Umožňujú digitalizovať proces plánovania zubných zákrokov rôzneho charakteru, ako napríklad plánovanie zubných implantátov či nápravu nepravidelností chrupu, čeľustných a tvárových anomálií. Digitálny proces síce znižuje pacientský čas strávený na klinike, no častokrát prináša nové manuálne kroky pre zubných lekárov, ktoré čiastočne degradujú celkovú efektivitu pri tomto druhu plánovania. Jedným z týchto pracných a časovo náročných krokov je manuálne určenie regiónov zubov v 3D skene, keďže táto informácia nie je priamo obsiahnutá pri skenovaní.

Metódy hlbokého učenia dosahujú v posledných rokoch vynikajúce výsledky v mnohých úlohách analýzy 2D obrazových dát. To nasvedčuje tomu, že ich efektívny spôsob extrakcie charakteristických informácií zo vstupných dát by mohol byť vhodným aparátom na tvorbu automatických systémov spracuvávajúcich 3D polygonálne modely. Toto rozšírenie na 3D neeuklidovské dáta však nie je priamočiare, keďže charakter polygonálnych modelov je značne iný ako charakter pravidelne vzorkovaných obrázkov. Typicky používané operácie akou je napríklad konvolúcia je preto nutné prispôsobiť na nepravidelný tvar povrchových 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 automatickej segmentácie zubných regiónov v povrchových skenoch čeľustí ortodontických pacientov (viz obr. 1). Je pritom nutné vziať do úvahy vysokú variabilitu týchto dát, keďže v bežnej klinickej praxi tieto dáta reprezentujú čeľuste so značne pokrivenými zubami, či čeľuste naskenované s ortodontickými aparátmi akými sú napríklad strojčeky. Systém robustný voči týmto rysom by dokázal zubným lekárom značne zefektívniť pracovný postup a umožnil im tak venovať viac času na samotnú liečbu.



(a) Vstupný polygonálny model

(b) Vysegmentované regióny zubov navrhnutou metódou

Obrázok 1: Ukážka vstupného povrchového skenu čeľusti (a) a príslušného výstupu (b) metódy na automatickú segmentáciu zubných regiónov na 3D modeli. Cieľom tejto práce je určenie triedy každého polygónu vstupného mešu do jednej zo šestnástich tried zubov a triedy reprezentujúcej ďasno.

Návrh riešenia

Práca primárne prezentuje prístup založený na rekurentných viacpohľadových neurónových sieťach [41, 66]. Tento prístup je založený na extrakcii príznakov zo sekvencie hĺbkových máp a máp krivosti vyrendrovaných z analyzovaného 3D objektu. Takto definované riešenie obchádza nepravidelný charakter neeuklidovských dát a zároveň prináša do učenia jednoduché aplikovanie overených techník z analýzy 2D obrazových dát [71]. Navrhnutý viacpohľadový systém ďalej obsahuje jednotky ako analýzu spojitých komponent či *"region voting"*. Druhá spomínaná jednotka využíva istotu predikcií na určenie finálnej triedy v prekrývajúcich sa častiach automaticky nagenerovaných regiónov a zároveň umožňuje analyzovať neisté časti získaných výsledkov.

Práca ďalej prezentuje použitie dvoch neeuklidovských architektúr, ktoré priamo využívajú geometrické príznaky na ich vstupe. Prvá z týchto metód je založená na architektúrach PointNet [56] a PointNet++ [57], a teda prevádza vstupný meš na mračno bodov, ktoré je následne analyzované. Druhá využíva ako simplex hranu trojuholníkovej siete a teda redefinuje operáciu konvolúcie a poolingu práve nad hranou. Je teda založená na rodine architektúr vychádzajcich z architektúry MeshCNN [24, 26].

Experimenty a dosiahnuté výsledky

Experimentálna časť tejto práce prezentuje výsledky po dvoch osiach. Najskôr je do hĺbky vyhodnotená hlavná metóda formou ablačnej štúdie. V týchto experimetoch je preukázaný konkrétny prínos jednotlivých komponent systému spolu s ukážkami chýb, ktoré sú pri generovaní segmentačných másk jednotlivými komponentami potlačené. Okrem ablačnej štúdie je s hlavnou metódou experimentované v kontexte najoptimálnejšieho nastavenia viacpohľadovej konfigurácie a taktiež s analýzou neistoty pomocou *"region voting"* komponenty. Metóda nie je robustná voči prípadom s úzkymi medzizubnými priestormi, avšak na komplexných ortodontických prípadoch obsahujúcich skeny s krivými zubami či naskenovanými ortodontickými aparátmi dosahuje navrhnutý prístup hodnôt 0,966 na metrike váhovaného IoU a 0,382 mm na metrike 95 percentilu Hausdorffovej vzdialenosti.

Vyhodnotenie po druhej osi následne zasadzuje najlepšie dosiahnuté výsledky viacpohľadového výstupu do porovania s inými metódami. Pre prinesenie férových porovnaní je hlavný prístup porovnaný s dvoma vyššie spomínamými neeuklidovskými metódami, ktoré sú vrámci tejto práce natrénované na tom istom datasete. Presnosť metódy je zároveň porovnaná s výstupmi konvenčného segmentačného algoritmu založenom na algoritme Graph-Cut. Hlavná metóda dosahuje najlepšie výsledky v porovnaní so všetkými inými navrhnutými prístupmi a to po kvalitativnej i kvantitatívnej stránke vyhodnotenia. Metóda je najpresnejšia i vrámci experimentu s rôzne nateselovanou geometriou vstupných dát.

Po konzultovaní výstupov s expertmi vo sfére vývoja 3D softvéru na plánovanie zubných zákrokov bude teda metóda otestovaná na väčšej testovacej množine a po preukázaní jej funkčnosti na testovacej množine s ešte väčšou anatomickou a geometrickou variabilitou bude zaintegrovaná do aplikácie a následne nasadená do produkcie.

Deep Learning for 3D Geometry Analysis in Medicine

Declaration

I hereby declare that this Master's thesis was prepared as an original work by the author under the supervision of Ing. Michal Španěl, Ph.D. The supplementary information was provided by Ing. Oldřich Kodym, Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Tibor Kubík
July 24, 2023

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my amazing supervisor, who has been supporting and guiding me for years since my undergraduate studies. I would also like to extend my gratitude to all the great people from TESCAN 3DIM, s.r.o., and especially to Olda for his thoughtful comments and recommendations on this thesis. Finally, I would like to thank my beloved girlfriend, family, and friends. Data used in this work as well as computational resources were provided by TESCAN 3DIM, s.r.o.

Contents

Chapter 1

Introduction

Computer-aided design (*CAD*) has recently been widely promoted in various branches of medicine. One of them is digital orthodontics, a pivotal segment in modern dental health care, where CAD dental systems assist in deletion, extraction, or rearranging of the teeth in a variety of treatment procedures. To do so, orthodontists need to manually segment teeth within the dental scan of the patient's jaw, which is time-consuming and expertise-dependent. Therefore, there is a great need for a reliable framework that would allow for full automation of this step. *Convolutional neural networks* (*CNNs*) have demonstrated their tremendous ability to analyze Euclidean data like images, so they appear to be a reasonable means to build such an automatic system.

When employing CNNs for teeth mesh segmentation, two considerations should be taken into account. The first one stems from the nature of the domain and the data itself. The prevalence of dental anomalies and dental irregularities is substantially high, particularly among orthodontic patients. The patient's arch may be asymmetric or incomplete (either caused by extractions, craniofacial malformations, retention, or tooth agenesis). Occasionally, dental appliances such as braces or wires are present. Positional tooth anomalies are also frequently seen among patients (rotations, teeth transpositions or transmigrations). Second, surface meshes are characterized by the non-regular structure and the ambiguously defined neighborhood. The direct application of operations defined in the Euclidean domain, such as convolution, is therefore more challenging.

This work aims to examine the application of deep learning techniques for the analysis of non-Euclidean data geometrically represented as triangular meshes. Among the plethora of approaches that either bypass the challenges associated with the structural narrative of 3D models or redefine the necessary CNN operations over non-Euclidean data, a method based on multi-view approach is proposed for the task of mesh teeth segmentation. The method is further extended with features such as recurrent unit that provides spatial correlation between viewpoints, deep supervision units, voting process, and connected component analysis.

The method was extensively evaluated on a test dataset of challenging real-world orthodontic cases, by means of the suggested overlap and boundary metrics. The effectiveness of the individual components of the proposed framework was evaluated in an ablation study. The method was compared with a conventional solution based on the Graph-Cut algorithm as well as with custom trained non-Euclidean approaches based on PointNet [56], PointNet++ [57] and SparseMeshCNN [26]. The proposed method outperforms all the mentioned approaches, achieving an average weighted IoU score of 0.966 and a Hausdorff distance at 95 percentile of 0.382 mm. The results show a potential increase in the effectiveness of

digital orthodontic planning. The method's deployment in dental planning software would enable clinicians to streamline their workflow and devote more attention and focus on the treatment itself.

The text of this thesis is structured as follows. First, in Chapter 2, the reader is introduced to the role of 3D data in medicine, with a focus on digital orthodontics. Then, in Chapter 3, key aspects of various 3D representations are presented in order to understand the pitfalls of applying convolution to 3D data. Chapter 4 follows with an overview of the literature that addresses such pitfalls along with a review of current approaches to teeth segmentation. The proposed methods are then presented in Chapter 5 together with the data used in this work. To facilitate the replication of the work, technologies and details of the implementation are introduced in Chapter 6. The experiments and results are summarized in Chapter 7.

Chapter 2

Leveraging 3D Data for Medicine and Dental Treatment Planning

3D imaging and *CAD/CAM* (*Computer-Aided Design/Computer-Aided Manufacturing*) *systems* have emerged as powerful tools in a variety of fields, including medicine. 3D imaging techniques enable to capture the inner three-dimensional anatomical structures, allowing for detailed and accurate visualization of bones, organs, or other tissues. On top of that, they enable to capture intricate geometric details of the surface, which is of significant importance in dental treatment planning. CAD/CAM systems transfer the flow of diagnostic and treatment methods into a digital format, making treatment more comfortable for the patient with reduced chair and laboratory time. The captured images are easier to store and transport [52], and allow better communication with patients thanks to the simple visualization of the treatment outcome. Generally, there are two prevalent forms of 3D data used in digital medicine and dentistry:

- **volumetric data** resulting from *CT*/*CBCT* (*Computed Tomography*/*Cone Beam Computed Tomography*) or *MRI* (*Magnetic Resonance Imaging*) acquisition, and
- **surface data**, which are usually obtained by various reflective surface scanners, or by surface extraction from volumetric data [47].

Since volumetric representations reveal internal anatomical structures, they are commonly used in medical diagnoses related to various areas of the body, including head and neck (diagnosing conditions affecting brain, skull, jaw, and teeth), or chest and abdomen area (diagnoses of lung and liver diseases). Volumetric modalities have also found their way into digital dentistry, such as in dental implant planning or orthodontics. Surface data are predominantly used in digital orthodontics for aligner fabrication and in digital dental prosthesis for the design and fabrication of dental restorations, such as crowns, bridges, and implants. Since the chosen task addressed in this thesis is focused on automating the orthodontic planning process, a section that describes the orthodontic planning procedure follows.

2.1 3D Data in Computer-Aided Orthodontics Treatment

Orthodontics is the branch of dentistry that deals with the diagnosis and treatment of malpositioned teeth and jaws. This usually includes straightening crooked teeth, correcting

irregular bites, and straightening opposing jawbones into alignment [63]. Traditionally, the whole treatment process has been purely analogue. To produce orthodontic appliances (such as archwires or brackets) in the conventional procedure, wax impressions and casts of the patient's teeth have been modeled. Subsequently, these were modeled into target-aligned dental arches [79]. The advent of CAD/CAM systems has transferred this flow into a digital format, bringing all the benefits mentioned above to dentistry diagnoses and treatment planning.

2.1.1 Acquisition of Digital Jaw Surface Models

There are two approaches to obtaining 3D dental surface data. The indirect method refers to the scanning of wax impressions with laboratory scanners, while the direct method refers to the scanning of the teeth and surrounding oral structures directly with *IOS* (intra-oral scanners) devices (see Figure 2.1). Despite the fact that IOS scanners are becoming more and more available, models scanned from cheaper indirect scanning are still more common. This has the effect, among others, that 3D models used in orthodontics typically do not contain information about color and materials.

(a) Indirect scanner of wax impressions **(b)** Intra-oral scanning device

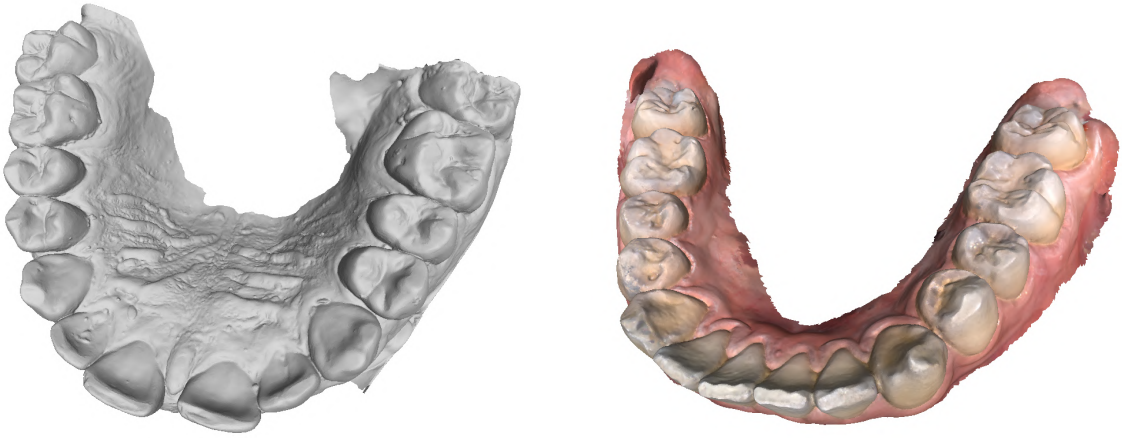
Figure 2.1: Example of hardware devices for scanning prepared wax impressions (a) and for direct scanning of human jaws (b). Indirect scanners are usually cheaper and still prevalently used. Direct scanners typically come with displays for real-time visualization of scanned teeth and produce models with color information. Device pictures are adapted from [32, 54].

From a computational perspective, both scanners produce point clouds that are subsequently triangulated into a 3D mesh.[1] Final digital impression usually lacks information about colors and contains information about geometry only. However, some modern IOS devices might output color information as well. The quality of the resulting meshes then depends on several factors that are generally influenced by two aspects:

[1]There are typically other intermediate steps such as outlier removal, point cloud simplification, estimations of normal vectors, and others.

- **Technical specifications of the scanner.** It influences the resolution of the mesh, quality of interdental spaces in digital impression, the truthfulness of colors (if any), and others.
- **Clinician's proficiency.** The resulting mesh might contain holes, blurred geometry, incomplete scans at the end of the dental arches (completely omitted or insufficiently scanned molars), and others.

Examples of digital impressions are shown in Figure 2.2. Digital impressions are then exported — typically in STL[2] or OBJ[3] format — to dental planning software, where orthodontics treatment is performed.



(a) Digital impression of maxilla without color

(b) Digital impression of mandible with color information

Figure 2.2: Examples of digital impressions obtained by scanning devices. The model in **(a)** was scanned using indirect scanner (no color information), and the one in **(b)** was obtained by IOS scanner. Overall, the quality of these models is high – there are no scanning artifacts and meshes contain more than 300 000 faces each. However, right 2nd Molar was not completely scanned in **(a)**.

2.1.2 Digital Workflow in Orthodontic Treatment

Modern CAD-based orthodontic software typically supports treatment planning using two types of appliances: dental braces and clear aligners. For both, the clinician can plan the final position of the teeth and how the teeth will change during the time of treatment with almost no patient chair time. To do so, **the clinician must first denote which regions of the mesh belong to individual teeth or gingiva**. The software then allows the clinician to manipulate the teeth positions in the 3D scene to get them in the desired arrangement. Based on the difference in the positions of the misaligned and aligned teeth, it is then possible to export a series of 3D models of clear aligners, which the patient progressively replaces. As the process of producing clean aligners contains the action of manual, laborious and time-consuming segmentation of teeth, the whole process could be

[2] https://docs.fileformat.com/cad/stl/
[3] https://docs.fileformat.com/3d/obj/

accelerated by automating the segmentation process. The aligner generation flow is shown in Figure 2.3.

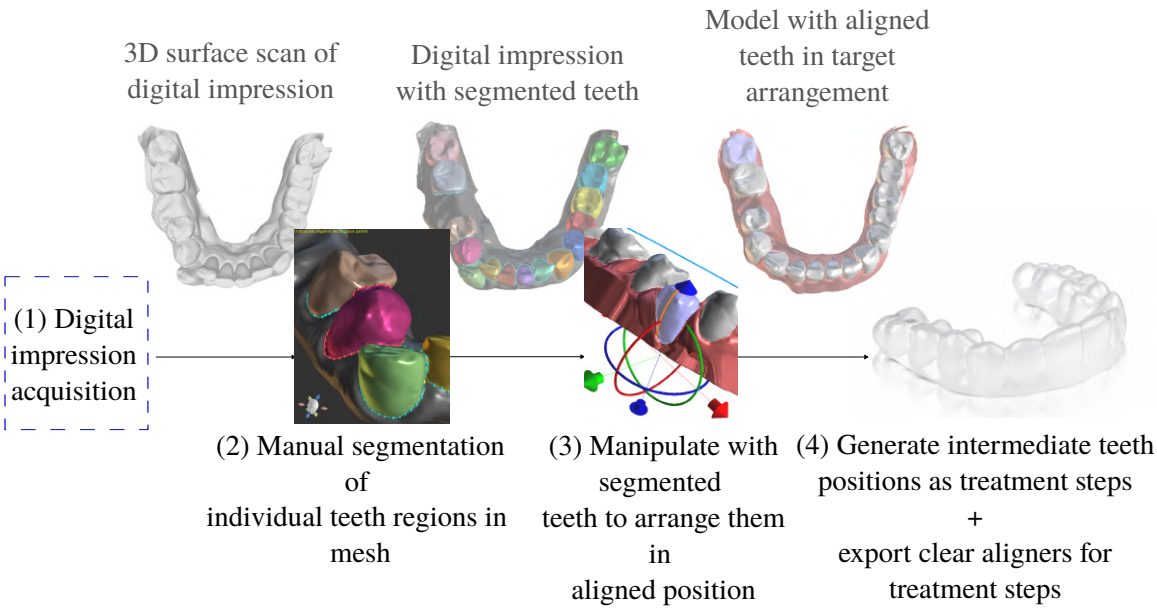


Figure 2.3: Flow of clear aligner generation in dental planning software. In order to manipulate individual teeth to attain the ideal arrangement, the tooth regions initially need to be manually segmented by a clinician.

Chapter 3

Key Properties of 3D Data and Their Relevance for Deep Learning

Three-dimensional data are the center and main commodity in fields such as computer graphics, computational geometry, computational fabrication, and computer vision. They serve as a valuable asset that provides rich information about the geometry of real-world objects and scenes. In addition to advances in acquisition, representation and analysis of surfaces of natural entities (i.e. *surface models*), 3D data might also enable one to reveal the inner structure of scanned objects. To decrease computational complexity and thus facilitate data processing and analysis, 3D data are usually represented in a discretized way. Optimally, the representation should meet several requirements, such as generality, compactness, unambiguity, and processing efficiency. However, it is difficult to have a single representation that meets all the listed requirements. As an example, storing a non-rigid model with an extensive amount of fine details completely, accurately, and in a memory-efficient way, while providing efficient processing techniques, is an ambitious task. For that reason, 3D data are represented in different forms. Each aims to prioritize different requirements, finds usage in different use-cases, and varies in structure and geometric properties. Figure 3.1 shows some of the core representations of the 3D data. There exist various tax-

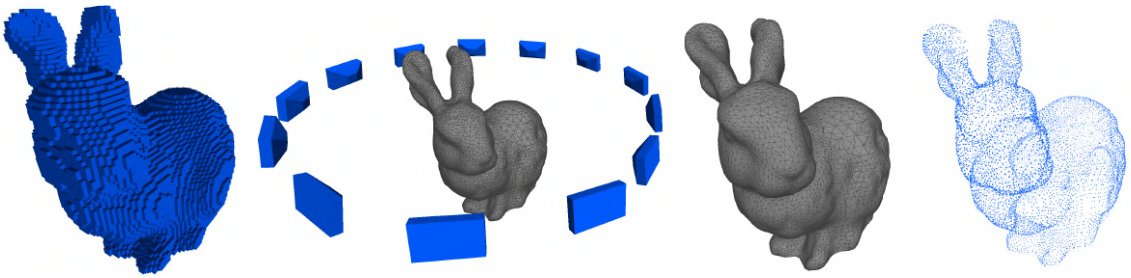


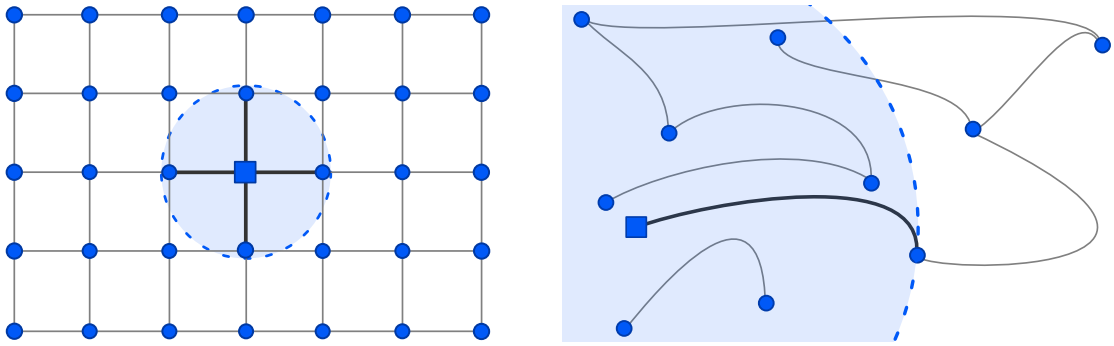
Figure 3.1: Different representations of the same 3D object. *The Stanford Bunny* represented by (from left to right) *voxel grid* with resolution of 107 × 107 × 88 voxels, *multi-view representation* with 13 rendered images from various viewpoints (blue cameras), *triangular mesh* with 18 000 faces and *point cloud* with 13 115 points. The two leftmost representations are examples of Euclidean data, in contrast mesh and point cloud are both cases of non-Euclidean data.

onomies offering their comprehensive overview. Since this work deals with the application of deep learning techniques for the analysis of 3D data, the division into Euclidean and non-

Euclidean data is the most relevant one. For neural networks, the premise is the underlying grid structure of Euclidean data, which allows for a global parametrization and a common system of coordinates. In addition, in Euclidean data, functional proximity coincides with physical proximity, which introduces an important inductive bias.

The aforementioned facts allow the use of deep learning techniques that are known from the image domain, namely the application of convolutional neural networks (CNNs). The following sections gradually demonstrate that the representations classified as non-Euclidean do not satisfy the properties of Euclidean geometry mentioned earlier. Therefore, it is important to understand the structure and relationships between both Euclidean and non-Euclidean representations in order to extend deep learning techniques to the analysis of non-rigid objects with irregular structure.

Note that from the plethora of 3D representations, only those important for proper comprehension of the remaining chapters of this text are described in detail, although there exist many others [53, 61].



(a) Regular data **(b)** Irregular data

Figure 3.2: Comparison of structured (e. g. image) **and unstructured 2D data** (e. g. undirected graph)**.** (a) Determination of neighborhood samples is trivial, as all samples lie on a uniform grid. (b) In unstructured data, the neighborhood is determined ambiguously. In given graph, many nodes are closer to the square node in terms of Euclidean distance (all within the blue circle), but the actual neighborhood is specified by adjacent vertices (connected by an edge).

3.1 Euclidean-Structured Discrete Data

Data emanating from the Euclidean domain are characterized by having a *regular spatial structure*. These data include characters, words, sentences, or sounds in $\mathbb{R}$, images lying in $\mathbb{R}^2$, and videos or volumetric data as modalities lying in $\mathbb{R}^3$. See Figure 3.3 for examples. Let's briefly examine the importance of regular spatial structure of Euclidean data on images. Formally, an image $\mathcal{I}$ is a function

$$\mathcal{I} : \Omega \rightarrow \{0, 1, ..., 255\}^c \tag{3.1}$$

where $\Omega = \{0; m-1\} \times \{0; n-1\}$ are image pixels, m and n are the number of rows and columns (image resolution) and c is the number of channels (usually $c \in \{1, 3\}$). In terms of data representation, image pixels are organized into a regular matrix. This implies that there is a clear notion of using vectors to denote the pixel's position. That is, $\mathcal{I}(i)$

is the intensity of the image at the i-th position in the image $\mathcal{I}$, where $i \in \Omega$. When considering the grid structure and the vector notion, the i-th position in $\mathcal{I}$ can be referred to as the vector (x_0, y_0). The image value at another pixel d units along the x direction is then calculated as $\mathcal{I}(x_0 + d, y_0)$ which is a vector $(d, 0)$ added to (x_0, y_0). This property is immensely useful when defining filter operations like *convolution*. Convolution operation is a key element in CNNs as it exploits the *translation equivariance* of images [4]. Let

$$f, k : \Omega \subset \mathbb{R}^2 \to \mathbb{R} \tag{3.2}$$

be two functions that are defined on the image $\mathcal{I}$ and let k be the convolution kernel with a local spatial support. Convolution, denoted by $*$, is calculated by the following formula:

$$(f * k)(x) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} k[s, t] \cdot f[i - s, j - t], \tag{3.3}$$

such that $x = [i, j]$. The vector notion for pixel position can be generalized to define a **planar translation** of the image-based function by a vector $v \in \mathbb{R}^2$, which is calculated as follows:

$$\tau_v(f(x)) = f(x - v). \tag{3.4}$$

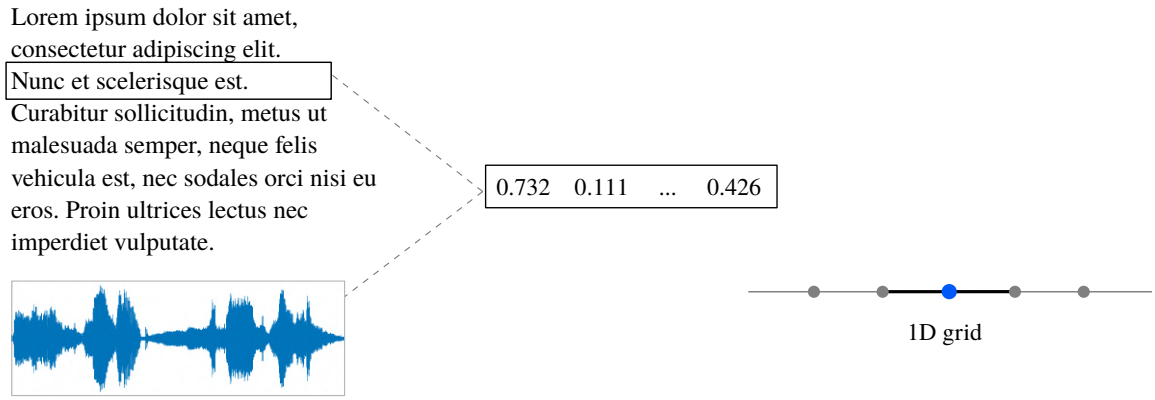
Putting these together, the translation equivariance simply means that the planar translation commutes with convolution:

$$\tau_v(f * k) = \tau_v(f) * k. \tag{3.5}$$

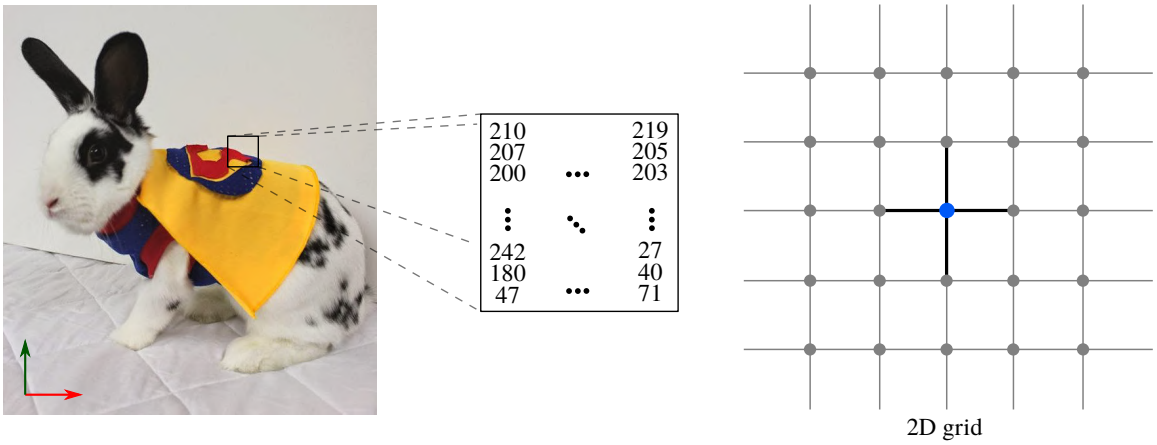
Demonstrated convolution property — the translation equivariance τ_v — is a key factor contributing to the effectiveness of CNNs [8]. It enables the trainable weights sharing, resulting in a significant reduction in the amount of network parameters. This reduction facilitates generalization and reduces overfitting [19]. The dependence of CNNs on the spatial structure of Euclidean data, particularly the concept of *locality*, is evident. The receptive field of a CNN inherently reflects the grid-like spatial structure present in the data, emphasizing the importance of the regular spatial arrangement in this context.

3.1.1 Voxel Grids

Voxel grids, also known as volumetric data, are formally described as a set $\mathcal{G}$ of samples (x, y, z, v), representing the value v of some property of the data, at location (x, y, z) within $\mathbb{R}^3$. Typical examples of recorded properties include physical quantities such as density, pressure, and temperature. The special case of volumetric data are *binary volumes*, where $v \in \{0, 1\}$ with 0 indicating *void voxels* (background) and 1 indicating *solid voxels* (the object). $\mathcal{G}$ is typically *isotropic* as it contains samples retrieved at regularly spaced intervals along three orthogonal axes. Alternatively, a *rectilinear grid* is employed. Since the set of samples is defined on a regular grid, a 3D array is typically used to store the values. Therefore, volumetric data are representative of a modality lying in Euclidean $\mathbb{R}^3$. It implies that the definition of the voxel neighborhood can be derived by a trivial extension from 1D or 2D, defining a convolution in 3D. In the basic setup, for each voxel, feature vector f_v with the value and dimensionality of v are used as input to the neural network (*voxel-wise features*), similarly to pixel values in images.



(a) Text and speech as examples of 1D data



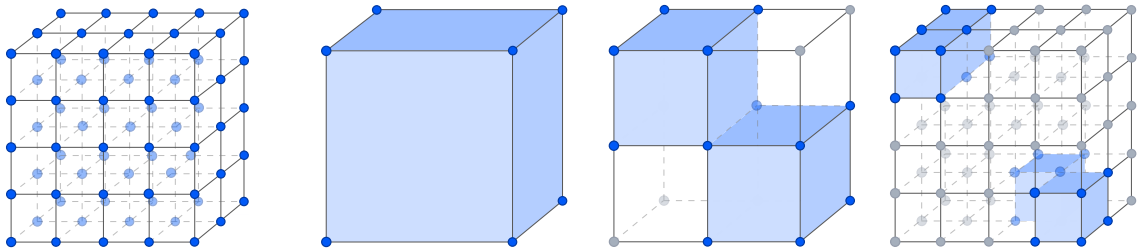
(b) Image as an example of 2D data

Figure 3.3: Examples of Euclidean data in $\mathbb{R}$ and $\mathbb{R}^2$. (a) Typical representatives of 1D Euclidean data are text or speech. The processed data are organized in a 1D matrix, so for each sample/token, it is trivial to state the neighboring elements as well as the distances to any other samples/tokens. **(b)** Image of a bunny serves as an example of 2D Euclidean data. Pixels form a regular grid, which again enables simple determination of neighboring pixels or distances to other pixels using vectors.

3.1.2 Octrees

Storing volumetric data of significant size without employing any auxiliary acceleration structure can lead to memory inefficiency. A common choice is a tree data structure called *octree*. In a voxel octree, volume decomposition starts with a single large voxel. Such a voxel completely encompasses the space occupied by the grid. Then, each internal node has exactly eight children, subdividing the respective part of the space. If a particular subdivision does not contain any further voxels, or there is exactly one, the recursion in given branch stops. Through this approach, large empty chunks of space do not unnecessarily take up memory. This three-dimensional analogue of a *quad-tree* finds usage in the nearest neighbor search [15] or computer graphics [48]. The neighborhood of a voxel stored in an octree is determined based on its position within the octree hierarchy, for example, by exploring its adjacent nodes. Stored values can serve as the input features for neural

networks in a voxel-wise manner, in the same fashion as in voxel grids. See Figure 3.4 for a visualization of the space decomposition by both voxel grids and octrees.



(a) Voxel grid **(b)** Space decomposition of an octree (three levels)

Figure 3.4: Visual examples of voxel grid and voxel octree. (a) Voxel grid in terms of data structure forms a 3D matrix. **(b)** Octrees are of tree structure, which reduces the exploration of blank regions of the space (blue boxes illustrate occupied areas of the space at given tree level). Please note that for presentation purposes, the illustrated decomposition in **(b)** divides the space in four parts only.

3.1.3 Multi-View

Multi-view representation enables to represent 3D non-Euclidean data by a series of 2D Euclidean renderings. That is, the 3D shape is formed by a set $\mathcal{MV}$ of functions $\mathcal{I}$ (defined in Equation 3.1), which are simply c-channel images with the analyzed/processed object rendered under distinct viewpoints. Content of individual elements of $\mathcal{MV}$ is parametrized by several settings called *viewpoint settings*:

- **extrinsic camera parameters** – camera positions and rotations with regards to the world coordinate system,
- **projection type** – perspective or orthographic with corresponding additional settings like view angle in perspective projection, and
- **rendering configuration** – lightning, shading, selection of reflectance model, etc.

In addition to viewpoint settings, *viewpoint number* is an important variable in defining a multi-view representation. Its value is equal to $|\mathcal{MV}|$. To fully capture the 3D shape, both properties are generally difficult to set. The optimal value of $|\mathcal{MV}|$ remains an open question, as computational costs are directly related to its value. Regarding viewpoint settings, unwise camera setup might result in a situation where the entire 3D shape properties are not completely captured, resulting in undesirable loss of information. The configuration of the multi-view rendering should always be re-considered with regard to the inherent geometric characteristics of the data and the intended purpose of the generated output.

In addition to surface rendering, 2D images such as depth maps or normal maps can be obtained. Analyzed 3D data are usually a priori uniformly scaled to fit the viewing volume and translated into the origin of the coordinate system. After the transformation to $|\mathcal{MV}|$, the 3D shape can be analyzed using conventional 2D CNNs. The input features are the pixel values of the images. These pixel values may encode the visual and geometric properties of the analyzed mesh, depending on the type of map employed. Examples of multi-view representation are depicted in Figure 3.5. In the example, two different sets of

a multi-view representation are presented – one consisting of depth maps and one of surface renderings.

3.2 Non-Euclidean Data

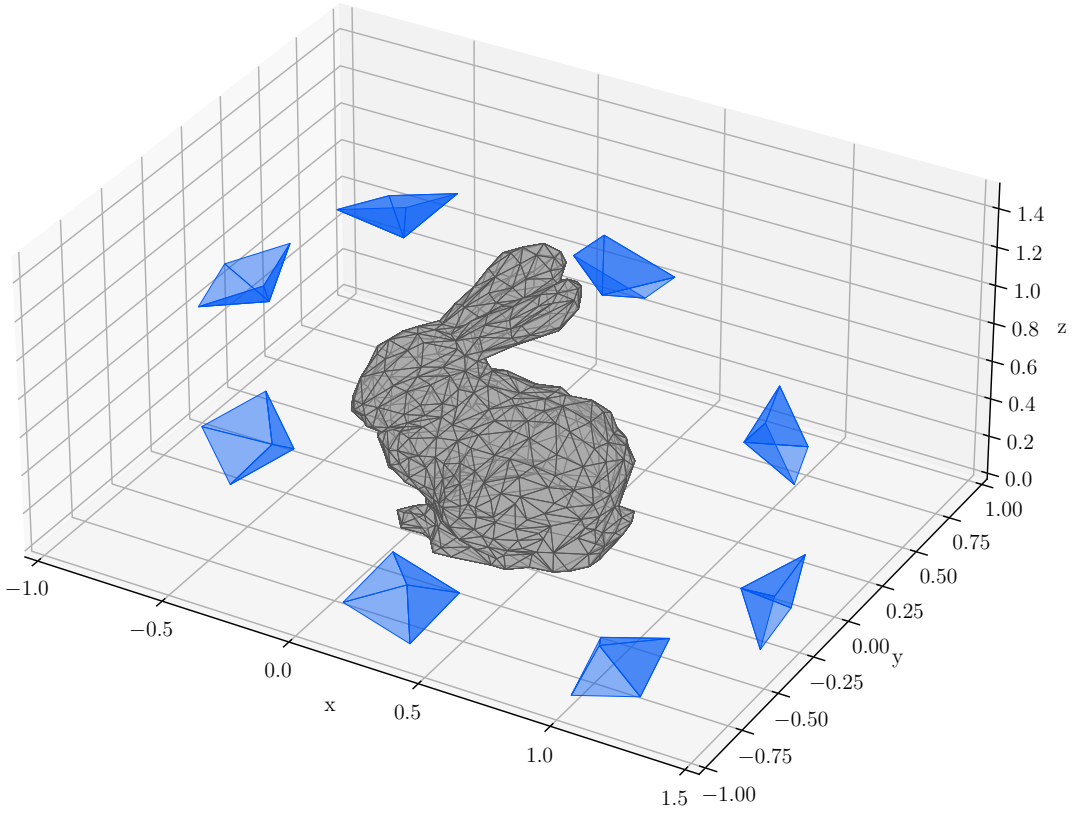
When considering data representation, non-Euclidean data are unorganized and irregularly distributed within its domain. The analysis is to be done on a **complex curved surface** and not a flat structure like an image. As an example, a discrete graph in arbitrary dimension can be constructed as a sample set of a surface. Yet, the nature of curved surfaces in the context of convolutional neural network processing does not allow for a simple extension of the concepts used when processing Euclidean data [4].

A notable concern arises from the lack of shift equivariance in the convolution operator (as well as other similar filter operators) in contrast to the Euclidean domain (the property and its importance is demonstrated in Section 3.1). The reason is that **translation on a curved surface is path-dependent** as distance and direction of movement are not invariant under translation. A more detailed explanation of this phenomenon can be found in the literature on differential geometry [9]. The absence of a translation-equivariant filter operator brings difficulties in effective weight sharing within neural networks. Another consideration is the **ambiguously defined neighborhood**. In images, functional proximity coincides with physical proximity, which introduces an important inductive bias into learning. However, for example in graphs, the Euclidean distance (or other distance metrics such as the Manhattan distance) does not necessarily encode information about node neighbors, but connectivity does, as shown in Figure 3.2. To provide an additional example, the determination of a sample's neighborhood within a point cloud relies on proximity queries, which can take various forms. The consequence of these considerations is that the established 2D image processing techniques (or in general techniques applicable to Euclidean data) like the operation of convolution are not always directly and straightforwardly applicable to non-Euclidean data. The techniques by which this generalization is secured in the context of deep learning analysis are the focus of Chapter 4. The remaining part of this section will proceed to examine several prominent non-Euclidean representations.

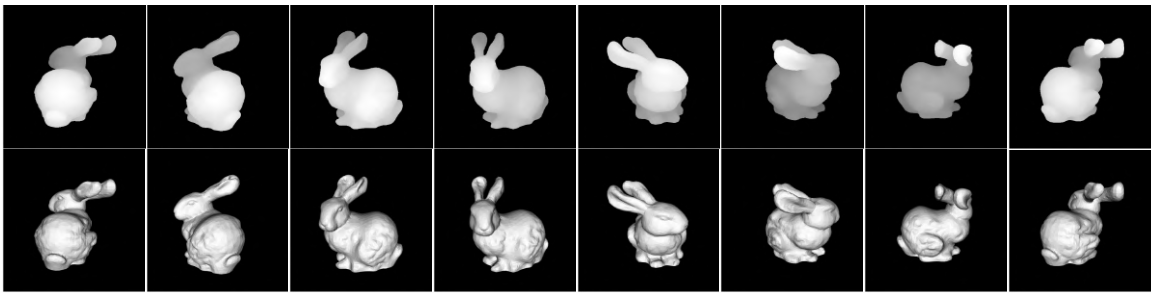
3.2.1 Point Clouds

Point clouds are an example of *raw* 3D data. Formally, a point cloud $\mathcal{P}$ is a set of unordered points $\{p_i\}_{i=1}^{n}$ sampled from some connected smooth surface, representing the scanned object [20]. Each measurement (i. e. each point) might contain additional information like color or its normal. Point clouds are usually obtained by scanning real-world objects or environments. The *sampling density* should be sufficient to ensure that the geometry of the real-world object is preserved. In general, this representation preserves the original geometric information without any discretization but suffers from measurement errors, resulting in points sampled inaccurately near the surface.

Point clouds do not carry explicit information on *connectivity*. To exploit the principle of locality, *neighborhood* of points is examined. Neighborhood of a point $p_i \in \mathcal{P}$ is a set N of k points close to p_i with respect to some metric, such that $N \subset \mathcal{P}$. There are several neighborhoods used in point cloud processing and analysis – *k-nearest neighborhood*, *ball-neighborhood* and *Delaunay neighborhood*, among others. K-nearest neighborhood of $p \in \mathcal{P}$ is formed by the first k points of $\mathcal{P}$ sorted in ascending order by their distance to p. Ball-neighborhood contains all points of $\mathcal{P}$ that are of a smaller distance to p than



(a) Visualization of extrinsic camera parameters



(b) Multi-view representation as depth images (upper row) and rendered geometry (bottom row)

Figure 3.5: Multi-view representation of a triangular mesh. In this particular illustration, a triangular mesh of *Stanford Bunny* is rendered under 8 distinct viewpoints. Corresponding cameras are illustrated within scene in **(a)** as blue pyramids. Individual cameras follow a circular trajectory around mesh, with the focal point being the same as the mesh's centroid. **(b)** shows two multi-view representations capturing the same objects under the same camera viewpoint settings, obtained using perspective projection. While the first row represents a multi-view representation composed of depth maps, the second row contains renderings of the geometry. Illumination and shading parameters are not specified in the example, although the individual elements of the multi-view representation would look different with them set up differently.

some r. In both cases, it depends on the algorithm design whether p_i is included within its neighborhood or not. To obtain the Delaunay neighborhood, N is projected on a *tangent plane* at the processed point p_i and the projected points are *Delaunay-triangulated*. All points that share an edge with p_i within the triangulation form the Delaunay neighborhood of p_i [3]. To apply learning over point clouds, each point $p \in \mathcal{P}$ is attached with a feature vector $f_u \in \mathbb{R}^d$. Typical values stored in f_u are the position of the point in $\mathbb{R}^3$, color, normal point, or geometric values derived from them such as local curvature.

3.2.2 Graphs

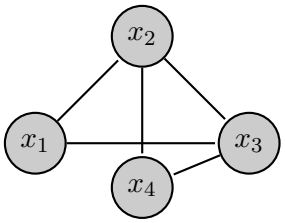
Graph is a fundamental geometric structure since graphs come in many applications either naturally (e.g. protein analysis) or are latent (social networks, recommendation systems, or physics). Graph data structure is often denoted as $\mathcal{G}(V, E)$, where V is a finite set of graph *vertices* or *nodes* and E is a finite set of *edges*. If two vertices x, $y \in V$ are adjacent in the graph, they are incident to an edge, and so there is a corresponding entry in E: $\{x, y\} \in E$. Each node (and edge, but let's consider nodes only) can be associated with a real-valued attribute vector. These feature vectors, $f_n \in \mathbb{R}^d$ usually form the input to a neural network. *Node neighborhood* is a fundamental aspect to describe the receptive field of (spatial) convolutions on a graph. The *i-th neighborhood* $N_i(x)$ of a node x is the set of nodes at most i steps (or hops) from the target node. There are multiple types of graphs based on their characteristics (directed vs. undirected, cyclic vs. acyclic, connected vs. disconnected, among many others). If interested in their specification or other terminology from graph theory, the reader is referred to the literature [17].

An *adjacency matrix* $\mathbf{A}$ of $\mathcal{G}$ is a binary n-by-n matrix, where $n = |V|$. Each non-zero element corresponds to an edge from x_i to x_j, such that $\forall x_i, x_j \in V, \mathbf{A}_{ij} = 1$. A *degree matrix* $\mathbf{D}$ is a diagonal matrix of $\mathbf{A}$ where each entry is the number of incident edges. The most prominent matrix for deep graph learning, the Laplacian graph, is $\mathbf{L} = \mathbf{D} - \mathbf{A}$. $\mathbf{L}$ is a symmetric, positive semi-definite matrix, thus an eigendecomposition of the matrix yields a full set of eigenvectors. Then, the graph information can be projected into the spectral domain, where the convolution of graphs is defined as multiplication [17]. See Figure 3.6 for examples of these matrices.

The representation of 3D shapes or 3D data in the form of a graph is not considered to be inherently natural. However, despite this, graphs are frequently utilized in practice due to their advantageous properties and the effectiveness of algorithms from graph theory. Other types of representations (like meshes and point clouds) are frequently transformed into graphs, to allow the use of these algorithms. This pattern also holds for deep learning methods.

3.2.3 Polygonal Meshes

Polygonal mesh representation, or *mesh*, for short, approximates surfaces of 3D shapes via a set of 2D polygons in 3D space and can be formally denoted as a tuple $\mathcal{M} = (V, E, F)$ defined by a set of vertices, $V \in \mathbb{R}^{N \times 3}$, a set of edges E that connect the vertices, and finally by a set of faces $F \in \mathbb{R}^3$ formed by a set of vertices – three in case of triangular meshes, which are connected by edges: $F = \{(u, v, q) \mid u, v, q \in V \wedge (u, v), (u, q), (q, v) \in E\}$. In some literature, the polygonal mesh definition is extended by an additional element. This element $\mathbf{A}$, the *adjacency matrix*, defines the connection between two vertices (for example $\mathbf{A}_{i,j}$ is set to 1 if vertex i is connected with vertex j, otherwise $\mathbf{A}_{i,j}$ is set to 0). Neighborhoods N on meshes are often defined in terms of k-ring neighborhoods, $k \in \mathbb{N}$. For example, a 1-



(a) Graph $\mathcal{G}$

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

(b) Adjacency matrix $\mathbf{A}$

$$\mathbf{D} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

(c) Degree matrix $\mathbf{D}$

$$\mathbf{L} = \begin{pmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & 2 \end{pmatrix}$$

(d) Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$

Figure 3.6: Representation of sample graph $\mathcal{G}$ via various matrices. $\mathcal{G}$ in **(a)** can be represented by adjacency matrix $\mathbf{A}$ **(b)**, degree matrix $\mathbf{D}$ **(c)** or by subtracting $\mathbf{A}$ from $\mathbf{D}$, obtaining graph Laplacian $\mathbf{L}$ **(d)**.

ring neighborhood N_i of the vertex $v \in V$ contains all the points connected to v by $e \in E$. Although the vertex neighborhood is the most commonly utilized type of neighborhood, its size cannot be universally predetermined, as it is dependent on the number of vertices that are connected. This property is analogous to that of point clouds and graphs; however, in the case of meshes, the neighborhoods for other mesh elements, such as edges or faces, can be defined. In situations where certain general assumptions are met, such as mesh *manifoldness* or *closeness*, the sizes of these neighborhoods become fixed, such as for edges. Figure 3.7 visualizes some of the neighborhood phenomena in meshes.
Likewise, it is also feasible to compose per-element feature vectors for any mesh elements. For example, for each $v \in V$, it is possible to construct $f_v \in \mathbb{R}^d$ with *extrinsic* and/or *intrinsic* features. Extrinsic features depend on the mesh transformation, i.e. change when the mesh is translated or rotated within the scene – for example, vertex position or normal. Intrinsic parameters are invariant under translations and rotations: various angle ratios, adjacent face areas, and such.

An essential concept, similar to the one from graph theory, is the *Laplacian matrix* $\mathbf{L}$, as it completely encodes the intrinsics of a mesh and allows its analysis in the spectral domain [59]. Given $n = |V|$, it is calculated as $\mathbf{L} = \mathbf{M}^{-1}\mathbf{C}$, where $\mathbf{M} \in \mathbb{R}^{n \times n}$ is the *diagonal matrix* whose i-th entry along the diagonal is twice the influence area of the vertex v_i, and $\mathbf{C}$ is the *sparse cotangent weighted matrix* defined as follows:

$$\mathbf{C}_{ij} = \begin{cases} -(cot\alpha_{ij} + cot\beta_{ij}) & i \neq j, v_j \in N_1(v_i), \\ \sum_{v_j \in N_1(v_i)}(cot\alpha_{ij} + cot\beta_{ij}) & i = j, \\ 0 & v_j \notin N_1(v_i), \end{cases} \tag{3.6}$$

where $N_1(v_i)$ is the 1-ring neighborhood of the vertex $v_i \in V$, and α_{ij} and β_{ij} are angles opposite to the edge formed by vertices $v_i \in V$ and $v_j \in V$ [13]. An analogous parallel can be drawn with graph processing: the eigendecomposition of $\mathbf{L}$ enables the transformation into the spectral domain, where many mesh characteristics can be examined, similar to graphs. Typical applications of mesh Laplacian are, in general, out of the scope of this

work, but important for a group of neural mesh learning approaches that operate in the spectral domain (discussed later in Section 4.2.4).

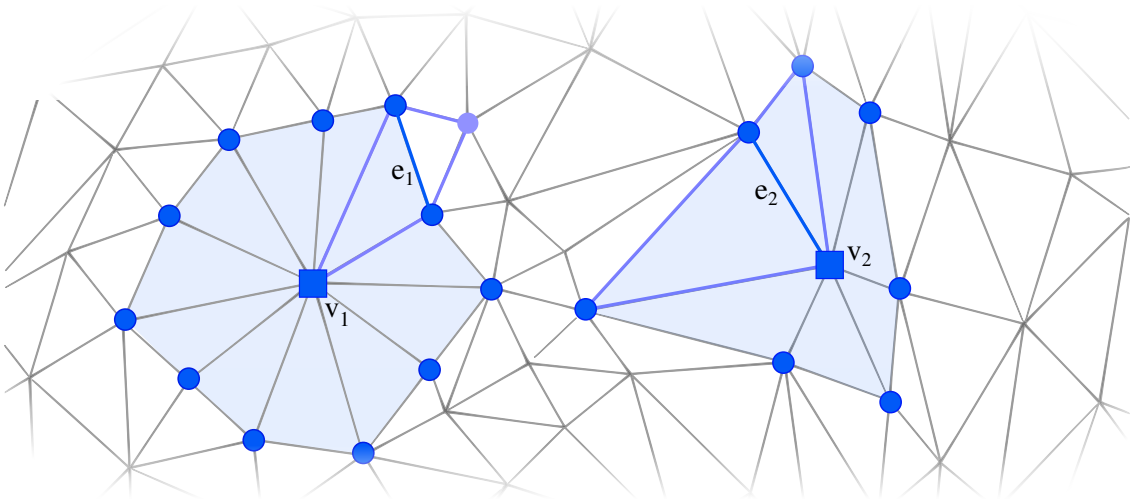


Figure 3.7: Various 1-ring vertex neighborhoods and fixed edge neighborhoods. Given two vertices from mesh $\mathcal{M}$, such that $v_1, v_2 \in V$, the sizes of $N_1(v_1)$ and $N_1(v_2)$ does not equal. However, if $\mathcal{M}$ is manifold and closed, edges $e_1, e_2 \in E$ are guaranteed to have fixed 1-neighborhood of 4 edges.

Chapter 4

Current Approaches towards Deep Learning Analysis of 3D Geometry

Deep learning methods have brought promising results in many 2D computer vision tasks. These representation-learning methods with multiple-level representations of various abstractions take advantage of natural signals: local connections, shared weights, pooling, and the use of numerous stacked layers. The convolutional layer detects local conjunctions of features from the previous layer, whereas the pooling layer merges semantically similar features into one [42]. However, this concept inspired by neuroscience is strongly based on the regular spatial structure of Euclidean data (see Section 3.1 and Section 3.2). Hence, extending this concept to irregular structures like 3D meshes is non-trivial yet of high potential, based on the success of the concept in regular domains.

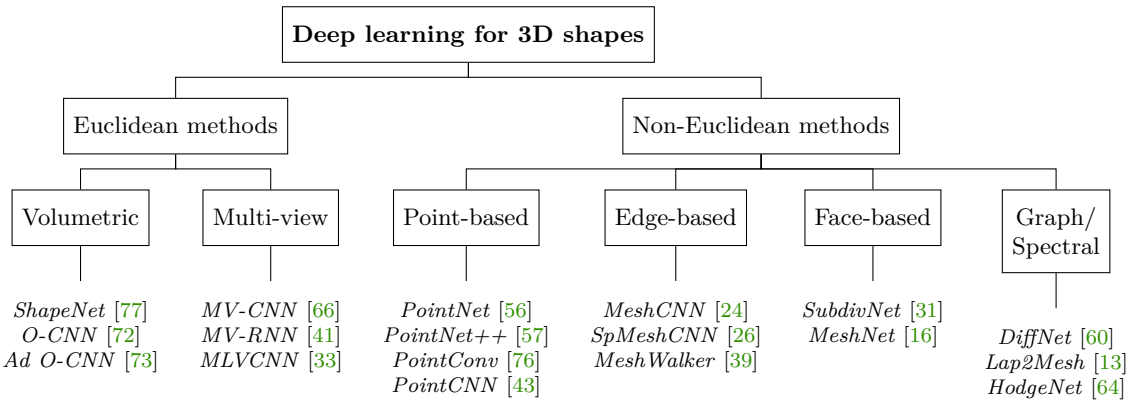


Figure 4.1: Euclidean/non-Euclidean taxonomy of deep learning 3D shape analysis. This taxonomy classifies works according to whether the core operations of neural networks are defined in the Euclidean or non-Euclidean domain. Please notice that it is not a comprehensive listing of all papers in the literature, but a selection of the most significant ones in each of the categories.

This chapter provides a comprehensive overview of the existing literature on deep learning methods for 3D geometry analysis. It starts with the introduction of general techniques known as *Euclidean methods*. These methods bypass the challenges posed by irregular mesh shapes and neighborhood ambiguity by converting the mesh geometry into Euclidean forms. Subsequently, the chapter discusses *Non-Euclidean methods*. To exploit the natural

potential of such data, these techniques redefine convolution and pooling operations over non-Euclidean representations. Note that this is one of many taxonomies that might be employed. Since this work was conceived in this way in the previous chapter as well, this division is followed for methods descriptions as well. Refer to Figure 4.1 for a visualized taxonomy and overview of significant works for each category. In the concluding section, the text discusses how these general approaches are applied to a specific task in medicine – segmentation of teeth in surface dental scans, which is the task addressed in this thesis.

4.1 Euclidean Methods

An effective approach to leveraging deep learning techniques for 3D shape analysis involves the conversion of input mesh into a Euclidean representation. This approach allows for direct utilization of the established concepts of convolutional neural networks commonly applied in regular domains. Within the literature, the following approaches have arisen: the conversion of meshes into volumetric grids or octrees, followed by 3D convolutional processing, and the efficient representation of meshes in a form of view-based 2D descriptors.

4.1.1 Volumetric-Based Methods

In volumetric-based methods, triangular mesh is modeled as a function sampled on voxel grid, for example by the rasterization process. Since the volumetric representation preserves a regular spatial structure, it is possible to use principles known from the analysis of volumetric data, such as CT or MRI scans. Due to the variety of tasks addressed in the analysis of medical CT and MRI images, research has yielded many methods built on top of 3D convolution and pooling [14]. The pioneering work in *mesh-to-grid* approaches was that of Wu *et al.*, and is called 3D ShapeNet [77]. In this study, the authors represent the 3D mesh using a binary tensor with a value of 1 indicating that the voxel lies inside the mesh, while a value of 0 indicates an empty space voxel. 3D convolution and pooling layers are then used to extract features from the voxelized mesh, as depicted in Figure 4.2. However, baseline ShapeNet is affected by two major shortcomings:

- Method is **limited to low-resolution grids**. In the original paper, a grid of size $30 \times 30 \times 30$ was used in experiments.
- Their approach is computationally intensive, since it performs a significant fraction of calculations on empty parts of the grid.

Additionally, due to the significant undersampling, intricate details of geometry are lost, making the method unsuitable for tasks that analyze geometry with intricate details, such as in medicine. Various works have sought to remedy these shortfalls. Of note, for example, are the works of Wang *et al.* [72, 73], who introduced O-CNN and its adaptive version – architectures that use an octree representation of volume. 3D CNN operations are performed on the sparse octants occupied by the boundary surfaces of 3D shapes only, making training and inference more memory and computation efficient. Convolution for a particular octant is computed from the feature vectors of neighboring octants, i.e. those in the same depth of the tree. The pooling operator resembles max-pooling from 2D images: the largest value in a given octet is selected. Both O-CNNs and Adaptive O-CNNs offer advantages in terms of memory efficiency and reduced computation times, enabling the analysis of larger grids, such as $256 \times 256 \times 256$, on powerful graphics cards in the order of seconds. However, the

experiments conducted in this study demonstrated that these methods may exhibit lower accuracy than non-Euclidean approaches, particularly in classification tasks.

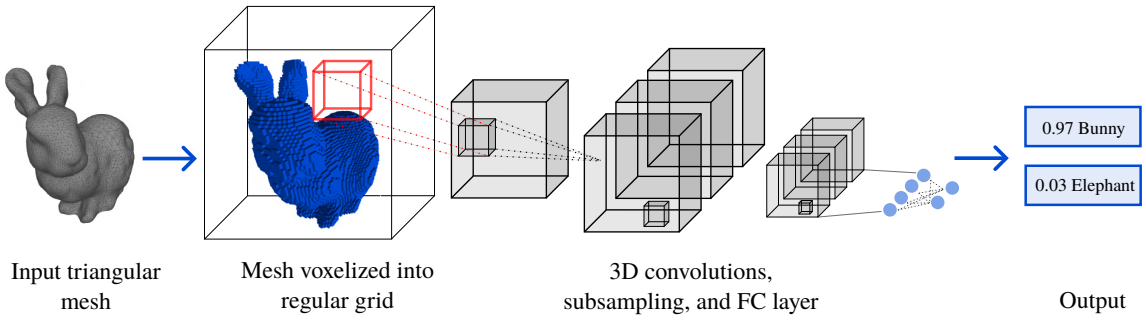


Figure 4.2: High-level overview of ShapeNet-based classifier. Input 3D mesh is at first voxelized into regular 3D grid. Then, features are extracted using 3D convolution and pooling. Fully connected layer is connected at the end, producing final output probability tensor.

4.1.2 Multi-View Approaches

As discussed in Section 3.1.3, in multi-view representation, 3D mesh is represented by a series of 2D renderings. Learning over such representation then aims to obtain a function that models each 2D view separately (i. e. extraction of view-based features using 2D CNNs) with the final joint optimization of all functions into a compact object descriptor [66]. This approach is often referred to as MV-CNNs (Multi-View Convolutional Neural Networks). Therefore, the basic principle can be thought of as a general feature extractor from 3D meshes and can be used to solve downstream tasks such as classification and segmentation. Furthermore, a significant advantage of this approach is the ability to utilize pre-trained models from the 2D domain. The authors in [41] extend this idea to address the task of semantic segmentation of human body parts. In this paper, they interpreted individual 2D views as a sequence and hence adopted a recurrent network, specifically *LSTM* [29]. The first module employed in this approach is the *CNN module*, responsible for generating a boundary map for each view, indicating the boundaries of the segmented regions. However, the authors discovered that the produced boundary maps exhibited issues related to poor localization and inconsistency. It was observed that treating the inputs individually did not guarantee viewpoint consistency, which is a crucial aspect in the segmentation process. Figure 4.3 outlines these modules. The second significant part of the framework — *the LSTM module* — exploits the power of LSTM loop connections, which allow the network to capture long-range dependencies by gates and memory structures, resulting in correlated multiple views and consistent boundary maps. The authors showed by experiments that combining multi-view CNNs and recurrent units into so-called *MV-RNNs* is meaningful, since this combination outperformed both the MV-CNN frameworks and conventional segmentation algorithms. This approach was consequently generalized and described in detail in [33].

This approach has, however, its shortcomings. The problem is the multi-view configuration, which cannot be determined generically for all domains and tasks. The number of viewpoints and the distribution of viewpoints within scene must always be tailor-made for each task. On top of that, without satisfying the assumption of consistently aligned input

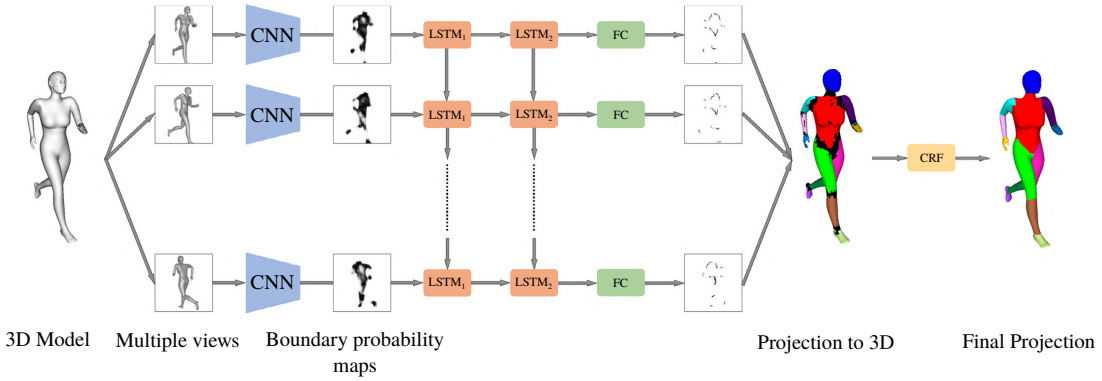


Figure 4.3: Overview of the multi-view RNN approach. Authors render input mesh as a sequence of ordered 2D geometry renders. Each of the rendered map is passed to a CNN (shared weights among viewpoints) to obtain a boundary probability map. This map is correlated by a two-layer LSTM followed by a fully connected layer. The consistent edge images from multiple views are transformed to 3D, followed by a region growing and CRF for boundary smoothing. Whole pipeline can be trained in an end-to-end manner. Outline adapted from [41].

mesh (for example upright oriented around a specific axis), one does not know beforehand which camera parameters would yield the most informative views. The latter is partially addressed in [51], so the network can learn on sequences of most informative views and subsequently infer the next view in a *next-best-view* fashion.

4.2 Non-Euclidean Geometry Analysis

Methods that belong to this category attempt to preserve the native representation of non-Euclidean 3D representations, so they redefine necessary operators over geometric structures. Individual methods then differ in what simplex is considered as the basic element: vertex, edge, or triangle (possibly an arbitrary polygon), analogous to pixels or voxels in Euclidean domains. Alternatively, some methods use principles of deep learning on graphs (GDL), typically by analyzing meshes in the spectral domain. What these methods have in common is that they operate directly on the irregular structure of non-Euclidean representations. This has the effect of coming up with a **redefinition of convolution** and **pooling**, the basic building blocks of convolutional neural networks, and exploiting various geometric features such as mesh curvature. Please note that the distinction into these classes might be blurry, as some methods may fall into several categories.

4.2.1 Point-Based Methods

Point-based methods typically sample points from input shape either by neglecting the mesh adjacency and thus taking the vertices as the input point cloud or by sampling points on faces. As point clouds avoid the combinatorial irregularities and complexities of meshes, they are typically easier to learn from. This is one of the reasons why deep learning on point sets has gained momentum more quickly compared to learning over surfaces. The pioneering work was that of Stanford University researchers called *PointNet* [56], a universal framework

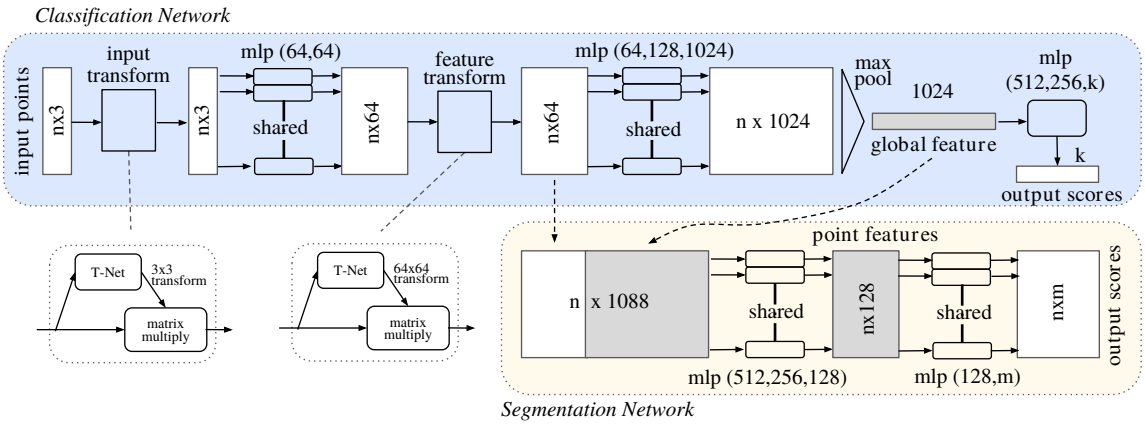


Figure 4.4: PointNet architecture for classification and segmentation. Encoder (blue part) consists of series of applied learnable transformations and feature extractions using MLPs. The global feature is calculated using the permutation-invariant max-pool symmetry function. For the per-point scores (yellow part), the global feature is transformed to the desired dimensionality by using shared MLP units. Figure adapted from [56].

for learning on a set of unordered points that, in addition, well respects the permutation invariance of points in the input. In the basic setting, PointNet operates over vectors of point's (x, y, z) coordinates, possibly extendable by other features like normal vectors. The framework is composed of several components (see Figure 4.4):

- **Input and feature transform.** Small trainable neural network (named *T-Net*) is learned to generate an $N \times N$ transformation matrix, where N is the dimensionality of the points in a given layer. The idea is that there might be a better coordinate system (translated/rotated point cloud) in which it is easier to extract the features. This matrix is applied to the input, preserving its shape.
- **Learnable MLP units.** Each point forms an input to a shared multi-layer perceptron, increasing the dimensionality.
- **Permutation-invariant maxpool.** Maxpool is implemented as a symmetry function to aggregate the information form each input point. As it takes the maximum, no matter what order the points are in, the maximum will be the same.

Authors argue that such approach is invariant under transformations, respects the interaction among points (neighboring points form a meaningful subset), and respects the nature of the unordered set of points, while using simple concepts like MLPs.

As PointNet uses single maxpooling operation only to aggregate the whole point set, it lacks the ability to capture local context at different scales. Hence, the work has been extended by hierarchical clustering of points where larger and larger regions are abstracted progressively along the hierarchy, similar to deep networks operating over images. This approach is called PointNet++ [57]. To introduce a hierarchical learning structure over point sets, authors introduce *set abstraction* levels where, at each level, a set of points is processed and abstracted to produce a new set with a reduced number of point samples. Each set is composed of three key layers:

1. **Sampling layer.** It selects a set of points from input samples at a given set abstraction, representing the centroids of local regions using *iterative farthest point sampling*. Note the difference with CNNs that scan the vector space agnostic of data distribution; point sets typically come with non-uniform density in different areas, so this strategy generates data-dependent receptive fields.

2. **Grouping layer.** This layer constructs new sets of local regions by examining the neighborhoods of the centroids sampled. In images, the local region of a pixel consists of pixels within certain Manhattan distance (defined by kernel size in CNNs). In point sets, one must specify a metric distance and some range query to define a local region of a point (see Section 3.2.1). Authors in this work employ the ball query as it guarantees a fixed region scale.

3. **PointNet layer.** Small PointNet network is used for local pattern learning in grouped local regions.

Figure 4.5 illustrates the PointNet++ architecture. Combination of hierarchical structure and thoughtful aggregation of multi-scale information according to local point densities enabled the framework to achieve state-of-the-art performance on classification and segmentation benchmarks of 3D point clouds. It continues to serve as an important work even as other compelling approaches have emerged [43, 76].

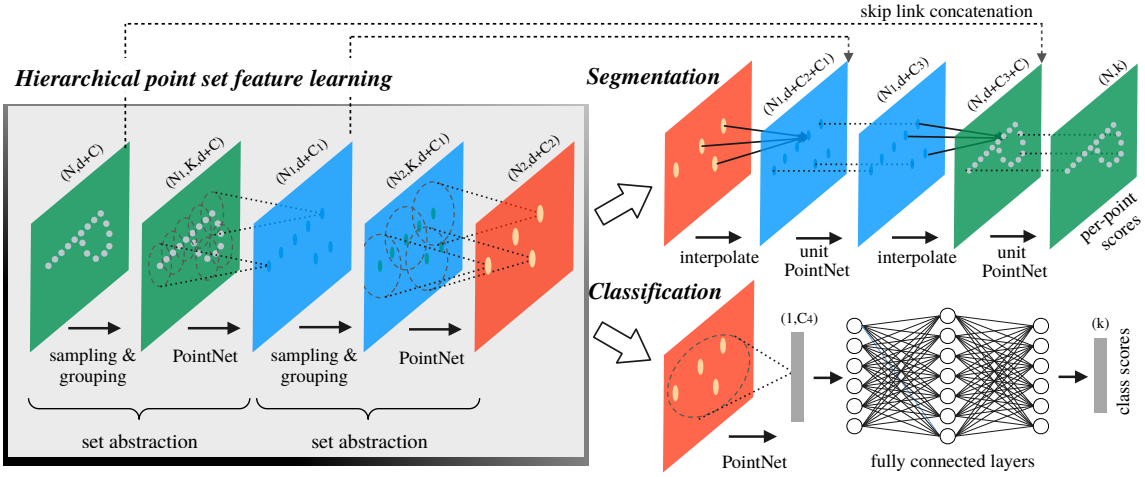


Figure 4.5: Illustration of PointNet++ hierarchical feature learning architecture. This example shows high-level PointNet++ application for 2D point set segmentation and classification. Figure adapted from [57].

4.2.2 Edge-Based Methods

Research in this category of methods gained momentum after the publication of the work by Hanocka *et al.* [24], presented in 2019 at *SIGGRAPH* conference. The authors introduced a general architecture operating directly over triangular meshes called *MeshCNN*, allowing to solve domain-specific downstream tasks like mesh classification or segmentation. As their approach is considered edge-based, convolution, pooling, and unpooling layers are applied on the mesh edges. Such a formulation comes with certain assumptions about the input mesh. In order to define a clear neighborhood of each edge, each 3D shape must be

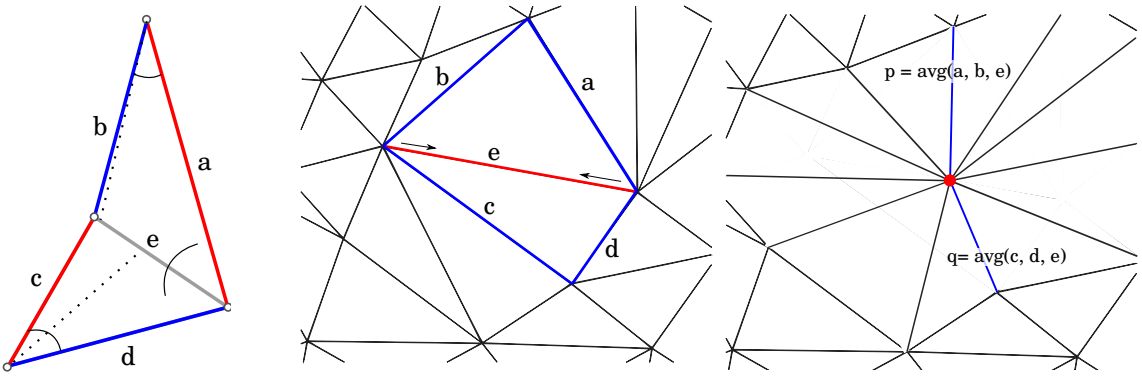
represented as a manifold triangular mesh with possibly boundary edges. This guarantees that each edge is incident to two triangles at most and thus adjacent to two or four other edges. Taking this into account, the authors defined the order-invariant *mesh convolution* for an edge e and its four adjacent edges as follows:

$$e_r = e \cdot k_0 + \sum_{j=1}^{4} k_j \cdot e^j, \tag{4.1}$$

where e^j is the feature vector of j^{th} convolutional neighbor of target edge e, and k_0 and k_j are corresponding kernels. To achieve the invariance to the ordering of the input data, a set of simple symmetric pairs is applied, resulting in the following definition of the receptive field of edge e:

$$(e^1, e^2, e^3, e^4) = (|a - c|, a + c, |b - d|, b + d), \tag{4.2}$$

where a, b, c, and d are the 1-ring neighbors of e. Following the convolution operation, a new feature tensor is generated, where the new number of features is equal to the number of kernels in the same way as in images.



(a) Edge features and mesh convolution

(b) Mesh pooling

Figure 4.6: Fundamental building blocks of edge-based methods. (a) shows the 1-ring edge neighborhood of edge e. Features of each edge are composed of intrinsic geometry properties: dihedral angle (the one between two incident triangles), inner angles and two edge-length ratios (between edge and the perpendicular dotted line). **(b)** demonstrates the mesh pooling accomplished by edge collapse. The red edge collapses into a point and the four blue edges merge into two edges. In a one edge collapse step, five edges are converted into two and a new vertex is created. Both illustrations are adapted from [24].

Another key feature of MeshCNN is the *mesh (un)pooling* operation that is capable of spatial adaptation to eliminate less informative features. For the downsampling process, the authors adapted the mesh simplification technique called *edge collapse*. Unlike conventional edge collapse presented by Hoppe [30], which removes edges that introduce minimal geometric distortion, mesh pooling chooses collapse candidates in a task-specific manner. The collapsed edges are those whose features contribute the least, so the network learns the shape parts of importance with respect to its objective. The operation transforms five edges into two, averaging the corresponding edge features. The mesh unpooling records the

mesh topology before pooling the mesh and then reinstates the same mesh connectivity. See Figure 4.6 for an illustration of the basic MeshCNN building blocks.

The authors chose intrinsic geometric features as input features. They form a five-dimensional vector for every edge: the *dihedral angle*, two *inner angles* and two *edge-length ratios* for each face, where the edge ratio is between the length of the edge and the perpendicular line for each adjacent face. When selecting intrinsic geometric features, the framework becomes invariant to translation, rotation, and uniform scale of the input shape.

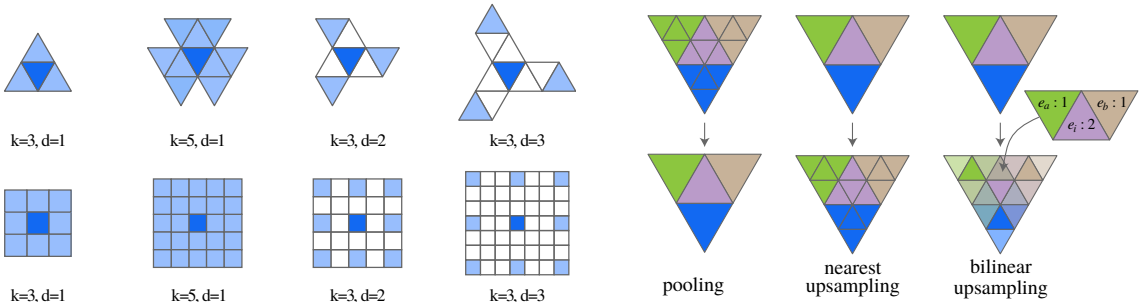
Overall, their work yields an interesting new approach to extracting features from meshes. However, there are some limitations. First, the evaluation was performed insufficiently in terms of the chosen metrics, especially to experiment with the segmentation network. Second, the chosen method is limited by the resolution of the input mesh – the work set an upper limit of 750 edges for classification tasks and 2 250 edges for segmentation tasks. From the perspective of applying their framework to medical data, where fine meshes capturing subtle anatomical properties need to be analyzed, the task formulated in this way is inapplicable. The latter was addressed in the work of Hansen *et al.* [26]. With these limitations in mind, they presented *SparseMeshCNN*. When the mesh unpooling is performed in the decoder, the situation is not that straightforward as in data over a regular grid (images or voxel structures). It is necessary to store information about collapsed edges during downscaling, so when expanding, the proper edges are unpooled. To diminish the memory requirements, the information about edge collapses is stored in a sparse matrix rather than a dense one. Their framework can process meshes with more than 60 000 edges with the same memory requirements. Other interesting works have been built on the basic idea of MeshCNN [25, 39].

4.2.3 Face-Based Methods

Face-based methods focus on efficient information sharing between neighboring faces. One of the first works in which the face is considered the primary data element is the one of Xu *et al.* [78]. For the task of semantic shape segmentation, the authors proposed a rotationally invariant k-ring neighbor convolution guided by face curvature. Several other methods have advanced the idea of k-ring convolutions for various tasks [16, 28]. Such methods, however, typically lack regular and uniform downsampling scheme to establish a fine-to-coarse mesh hierarchy – analogous to a 2D image pyramid.

The authors in [31] proposed so-called *SubdivNet* which aims to achieve the closest analogy to hierarchical processing over regular data. To aggregate local features into large-scale features at different levels, the authors employ *subdivision surfaces* – a smooth surfaces produced by refining coarse mesh. There are many mesh subdivision algorithms (for example *Catmull-Clark subdivision* [6] or *Loop subdivision* [46]), while in the work presenting SubdivNet, the Loop subdivision was used due to its Loop property. As a result, the Loop subdivision scheme always gives a *1-to-4 face mapping* between coarse and fine mesh, leading to a uniform fine to coarse hierarchy when applied several times. The 4-to-1 face mapping establishes a step-by-step injection of faces from input mesh to the finest one, allowing local to global feature aggregation.

The framework expects a closed 2-manifold triangular mesh on the input, where each face is exactly surrounded by 3 other faces. This *3-regular property* is analogous to the lattice connectivity of pixels in 2D images and allowed the authors to introduce a general mesh convolutional operation that allows variable kernel size, stride, and dilatation, as well as pooling and unpooling. See Figure 4.7 for the visualization of various convolution kernel



(a) Face convolution design pattern **(b)** Face pooling and unpooling

Figure 4.7: Fundamental building blocks of SubdivNet. (a) shows face convolution kernels with different kernel size k and dilation d (upper row) and corresponding 2D convolution kernels (bottom row). **(b)** depicts pooling and two types of upsampling: nearest and bilinear based on face distance, to provide smoother interpolation. Both illustrations are adapted from [31].

patterns and pooling/upsampling techniques. In the experiments carried out, each face was represented by a 13-dimensional vector composed of shape and pose descriptors (face area, interior triangle angles, inner products of face normal with vertex normals, face center, and face normal). The method outperformed other point- and edge-based approaches in both classification and segmentation tasks by a large margin. However, the proposed method expects remeshing of the input mesh, so the whole feature extraction is performed on a topologically different mesh. In addition, the method is still limited by memory requirements, though less memory-sensitive compared to, for example, MeshCNNs.

4.2.4 Methods Based on Graph Representation and Spectral Analysis

One of the important strands of literature in this category is formed by methods, where learning is performed on locally encoded points. There, the kernel functions mimic those from the image domain. In other words, learning is carried out on *parameterized* shapes. Several different approaches appear in the literature, which differ in the parameterization method used. For example, the authors in [50, 55, 69] parameterize *geodesic fields* (e.g. *tangent planes*) to apply surface convolution. Alternatively, the *global parametrization* is used by authors in [21, 49]. Due to parametrization, the methods are insensitive to different tessellation and meshing.

The other group of methods relies on the basic principles of learning over graph structures. Note that this is a widely studied field, so only the most relevant methods for mesh analysis are covered in the text. The graph learning methodology is typically divided into two classes: *spatial* and *spectral* methods. Convolution is defined spatially on the graph by analogy to convolution on a regular domain, whereas in spectral methods, graph signals are transformed from the vertex domain to the spectral domain. Although many important concepts can be found in spatial methods [18, 22, 34], spectral methods are used mainly in the context of 3D shape analysis, so they will be addressed. The graph can be translated to the spectral domain using graph Laplacian $\mathbf{L}$ (see Section 3.2.2). To do so, the eigendecomposition of $\mathbf{L}$ is performed by calculating the eigenvectors and the corresponding eigenvalues:

$$\mathbf{L} = \mathbf{U}\hat{\mathbf{\Lambda}}\mathbf{U}^T, \tag{4.3}$$

where $\mathbf{U} = [\mathbf{u}_0^T, \ldots, \mathbf{u}_{n-1}^T]$ are the Fourier bases of n eigenvectors and $\hat{\mathbf{\Lambda}} = \text{diag}([\lambda_0, \ldots, \lambda_{n-1}])$ the diagonal matrix of the eigenvalues [11]. Then, one of the spectral convolution approaches define operation of convolution over graph signals as follows:

$$f * \mathbf{g} = \mathbf{U}((\mathbf{U}^T f) \odot (\mathbf{U}^T g)), \tag{4.4}$$

where $\odot$ is element-wise multiplication [5]. Although more computationally optimal approaches emerged later [23], the main takeaway message is the principle: from graph signal, the Laplacian is calculated, and thanks to its properties, eigenvectors produced by eigendecomposition are used to define convolution in spectral domain, which becomes simple multiplication.

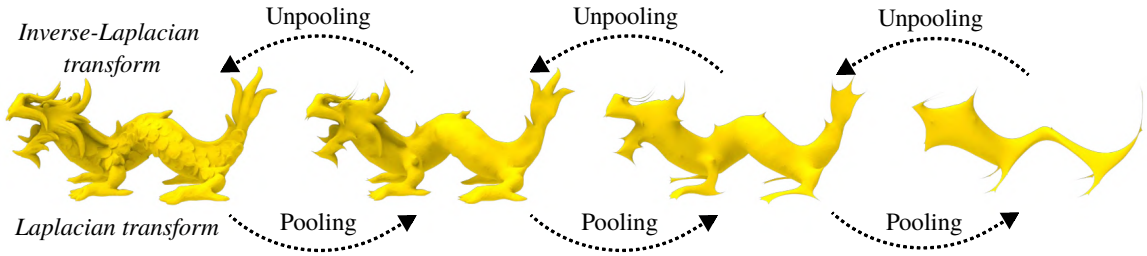


Figure 4.8: Laplacian pooling and unpooling, various number of low-frequency eigenvectors selection for building the spectral basis. From left to the right, the less smallest eigenvectors used, the less high-frequency features are preserved in the spectral surface reconstruction. The basic building blocks for hierarchical understanding: Laplacian pooling and unpooling are based upon this idea. Figure adapted from [13].

Similarly to signals like images and graphs, 3D shape information with varying frequency can be encoded in spectral domain as a set of eigenvectors and corresponding eigenvalues. In the rest of this section, $\mathbf{L}$ refers to mesh Laplacian (recall the definition in Section 3.2.3). $\mathbf{L}$ completely encodes the intrinsic geometry. The eigendecomposition of the Laplacian matrix $\mathbf{L}$ enables the transformation between the spatial and spectral domains, similar to the graphs presented above. Upon conducting eigendecomposition, it is possible to choose k eigenvectors that correspond to the k smallest eigenvalues. This selection $\mathbf{\Phi}$ is in fact a low-pass filter, as the lower the number of eigenvectors of smallest eigenvalues is selected, the less high-frequency geometry is preserved when the spectral surface reconstruction is performed. Based on this idea, Dong *et al.* recently presented *Laplacian2Mesh* [13]. The authors construct an input feature matrix $\mathbf{G} \in \mathbb{R}^{n \times 39}$ in the spatial domain, where n is the cardinality of vertex set V of input mesh $\mathcal{M}$. The authors employ various intrinsic and extrinsic features to define this synthesized descriptor, such as vertex positions, normals, 1-dimensional Gaussian curvature, Heat Kernel Signature [68], and low-frequency eigenvectors corresponding to the 20 lowest frequencies. This feature matrix, upon projected to the spectral domain, becomes:

$$\tilde{\mathbf{G}} = \mathbf{\Phi}_k^T \mathbf{G}, \tag{4.5}$$

and enables to encode meshes of arbitrary topology, complexity, and geometry, into a $k \times 39$ matrix $\tilde{\mathbf{G}}$. Then, $\tilde{\mathbf{G}}$ serves as the input to the neural network. The key element of Laplacian2Mesh framework is the usage of various spectral bases $\mathbf{\Phi}_k^T$ to simulate the hierarchical

processing of regular data. To do so, authors select three different resolutions of $\tilde{\mathbf{G}}$. The resolution depends on the number of low frequency eigenvectors used to construct corresponding $\mathbf{\Phi}$ that generates $\tilde{\mathbf{G}}$. The three spectral matrices are then processed by regular convolution blocks. Using similar idea for *Laplacian pooling* and *unpooling*, authors built architectures for different tasks, such as classification and segmentation. For better understanding, see Figure 4.8, which demonstrates how these concepts look after spectral surface reconstruction.

Laplacian2Mesh is, of course, just one of several graph-inspired spectral domain methods, but provides an insight into neural mesh analysis in the spectral domain. As an example, *HodgeNet* [64] also operates in the spectral domain but is based on the mapping via the calculation of the *Hodge star operator* [12]. As spectral methods like [60, 64] are conceptually more difficult, interested readers are referred to the original papers.

4.3 State-of-the-Art in Mesh Teeth Segmentation

Prior to the advent of deep learning techniques, several non-learning attempts has launched to address the challenging dental mesh segmentation. These traditional geometry-based methods utilize hand-crafted features, such as curvatures and surface contour lines. These features are used to design decision rules for conventional segmentation algorithms such as thresholding [38], watershed-based region growing [44], active contours [35], and such [74, 80]. Apart from the fact that these methods often require thresholds to be manually regulated or key points to be sequentially annotated in order to fine-tune the resulting segmentation, these methods lack the robustness required to represent intricate tooth shapes and complex anatomical abnormalities, which make the segmentation process error-prone.

Several works tackle this problem using deep learning methods, although their frequency in literature is relatively limited, and, moreover, all approaches are based on a very similar design principle. As already discussed, among the 3D representations, the 3D point cloud became popular in many narrow tasks, mainly due to its flexibility and memory efficiency. This is also true for the task of automatic mesh teeth segmentation, where, the goal of the automatic framework is to perform per-face-classification of teeth regions and gingiva within input mesh. Although pioneering works such as PointNet [56] and PointNet++ [57] are capable of learning characteristics on different contextual scales, their performance is limited in the task of segmenting teeth. From that reason, several frameworks have been built on this idea that extend it to be applicable to the intricate character of dental scans. For example, the authors in [67] employ a coupled segmentation and dense correspondence network stacked on existing graph convolution network architectures. Similarly, by analyzing point cloud representation, Cui *et al.* [10] propose a method for distance-aware tooth centroid voting scheme and confidence-aware cascade segmentation module. Alternatively, the method in [75] is defined as a two-stage framework, where, first, the teeth regions are segmented using *iMeshSegNet* network [45], and landmarks are subsequently regressed within the proposed ROIs (regions of interest, that is, teeth regions).

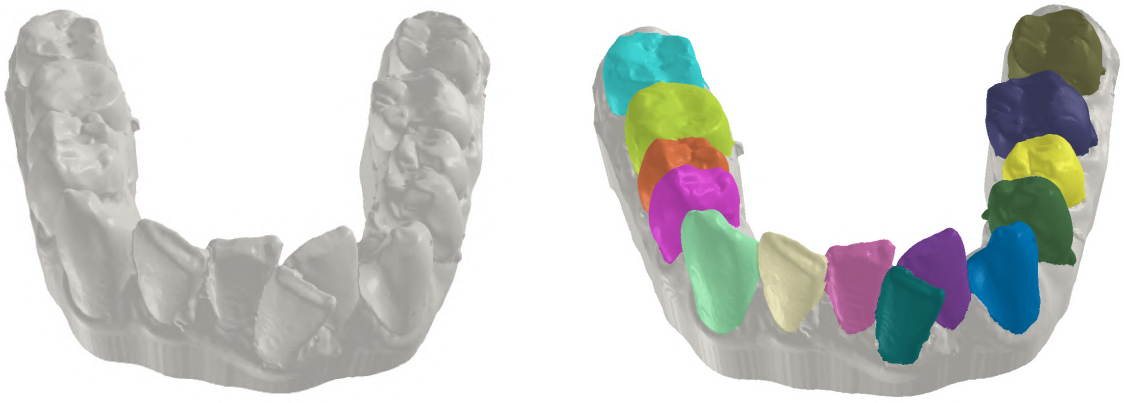
Although the results of the CNN-based approaches are promising and far beyond the results of conventional methods, these automatic frameworks are often evaluated on simple anatomical cases, and it is not clear how they would generalize on commonly occurring complicated orthodontic cases and hence how helpful they would be in industry. In addition, the methods often combine the task of detecting landmarks and segmenting teeth (performed in varying order). However, from a practical point of view, it is a desirable practice to separate these two tasks, as correction of the positions of detected orthodontic

landmarks is less time-consuming than correcting the segmented tooth regions. Finally, current state-of-the-art methods in the mesh teeth segmentation task do not dispose of any public benchmark dataset. This, combined with often poorly designed metrics, yields limited credibility in method evaluations and comparisons, so it is non-trivial to determine which of the current methods is superior.

Chapter 5

Proposed Solutions for Automatic Teeth Segmentation in Surface Scans

To address the problem of automatic teeth segmentation in surface meshes, in this thesis, three supervised solutions are proposed. The main contribution of this work is the multi-view segmentation approach. This is a novel approach to tackle this task, as to my knowledge, there is no solution in the literature that experiments with this Euclidean approach on this particular task. For fair comparisons, two non-Euclidean geometric approaches are further proposed: one is based on the PointNet [56] and PointNet++ [57] architectures (i.e., they operate on point clouds) and one is based on SparseMeshCNN [26], i.e., directly analyses the mesh representation. The first part of this chapter formally defines the problem to be solved. Subsequently, the dataset is introduced as a key element of supervised methods. After establishing an insight into the dataset, the proposed methods are presented in detail.



(a) Input triangular mesh **(b)** Segmented regions by proposed method

Figure 5.1: Example of an input surface dental scan (a) and generated output (b) of methods dealing with automatic teeth segmentation. Goal of this framework is to segment the input mesh into separate regions that represent individual teeth and gingiva.

5.1 Problem Definition

Given the 3D shape represented as a triangular mesh $\mathcal{M} = (V, E, F)$; see the full definition in Section 3.2.3. Unless stated otherwise, there are no assumptions about $\mathcal{M}$ with regard to manifoldness, closeness, resolution, orientation, or size. It is expected to be a geometric object representing a single human dentition – either a maxillary or a mandibular dental arch. The problem solved in this work is then formally denoted as

$$\mathcal{F} \subseteq F \times \mathcal{C}, \tag{5.1}$$

where $\mathcal{C}$ is a set of classes. The function $\mathcal{F}$ can be perceived as a mapping from a set of mesh faces to a set of classes, which is in fact how the downstream task of segmentation or per-element classification is formulated. In this work, individual classes have following semantics:

$$\mathcal{C} = \{L1,\ L2,\ L3,\ \ldots,\ R1,\ R2,\ R3,\ \ldots,\ G\}, \tag{5.2}$$

such that $|\mathcal{C}| = 17$. They represent sixteen classes for individual teeth regions encoded as L(eft) or R(ight) quadrant followed by a number starting from central incisor to 3rd molar (1 to 8), and one class G that represents gingiva. In the context of this work, no distinction is made between the upper and lower jaw, i.e. both are treated equally (the lower jaw is aligned to match the upper jaw). See Figure 5.2 for a visual demonstration of the target regions and nomenclature. Note that the region colors and nomenclature hold throughout the work. Mapping $\mathcal{F}$ is *injective*, since each face from $\mathcal{M}$ is assigned a specific class. However, it is not *surjective*, since in majority of cases the scanned dental arch is incomplete, i.e. teeth are missing, therefore some classes may not be associated with any face. Example of an input mesh $\mathcal{M}$ and visualized mapping $\mathcal{F}$ is presented in the teaser image, Figure 5.1. Each face is assigned a class (injectivity property), but some classes like L8 and R8 are not present, as corresponding teeth are missing (which means that surjectivity is not satisfied). Faces of each class should always form a single connected component.

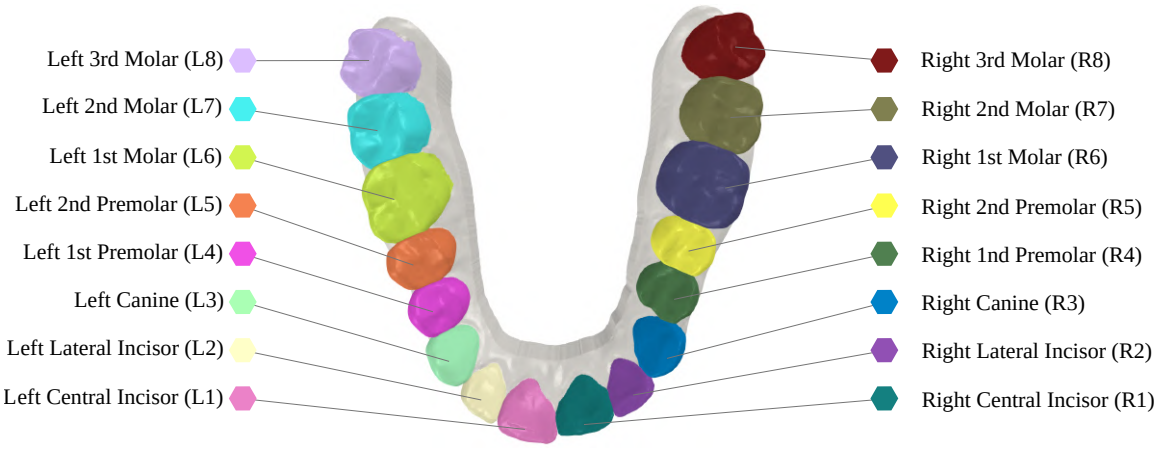


Figure 5.2: Visual notation and naming convention of segmented classes employed throughout the thesis. This work aims to segment 17 different classes, out of which sixteen correspond to individual teeth in human jaw. Mesh faces that are colored in gray represent gingiva class G.

5.2 Dataset of Surface Dental Scans of Orthodontic Patients

The dataset consists of **142 triangular meshes**, with an nearly ideally balanced distribution of the upper and lower jaws (76 represent the human maxilla and the remaining 66 represent mandibles). All data samples are **real orthodontic cases** scanned by various intra-oral scanners and by indirect scanners of wax impressions. The cases were anonymized so it was not possible to undertake a thorough analysis in regards to the ethnic, gender, or age; however, data come from multiple clinical sites across several continents. The majority of the data are from patients scanned prior to the orthodontic procedure. This brings a necessary and, moreover, natural complexity in terms of malocclusion and teeth shifts, providing a rich source of data for training and evaluation. In addition, there are scans with dental braces and wires in the dataset – that reflects the orthodontic nature of data, and it is desirable that the method is robust towards such cases. Finally, some cases contain geometry with blurred geometric signals, for example, at the tooth-gingiva boundary or inbetween two teeth.

Table 5.1: Exploratory data analysis. Table contains essential characteristics about the geometry of the samples in the dataset, which is important to consider when designing neural mesh analysis approaches. This exploratory analysis also contains characteristics important for non-Euclidean methods, such as mesh closeness, self-intersections, and manifoldness. †Physical dimensions were measured from oriented bounding box.

		Train Set	**Test Set**
Vertex count	min	42 837	49 374
	mean, std	74 783 ± 40 524	71 686 ± 14 551
	max	531 584	102 749
Face count	min	85 670	98 744
	mean, std	149 560 ± 81 051	143 356 ± 29 101
	max	1 063 188	205 494
Edge count	min	257 010	296 232
	mean, std	448 698 ± 243 154	430 107 ± 87 305
	max	3 189 564	616 482
Physical dimensions $(W \times H \times D)^{\dagger}$ [mm]	min	$34.60 \times 52.11 \times 12.74$	$26.33 \times 50.18 \times 13.70$
	mean	$49.24 \times 63.33 \times 17.83$	$46.53 \times 61.65 \times 17.62$
	max	$64.79 \times 81.06 \times 25.04$	$52.68 \times 68.91 \times 20.02$
Percentage of closed meshes [%]		100	100
Meshes with self-intersections [%]		0	0
Non-manifold cases [%]		0	0

All provided scans contain geometry information on vertex positions and topology, with an average face count of approximately 150 000, and do not carry any information on colors or textures. Regarding mesh topology and quality, meshes with self-intersections, non-manifold edges, or flipped normals are not present in the dataset. Table 5.1 provides in-depth insight into the fundamental geometric properties of the dataset. The geometry across the data is of varying complexity, most noticeable on the occlusal surfaces of the molars, where a number of samples contain intricately detailed cups and grooves, but others

have a contrasting absence of detail, with the occlusal surface comprising a rather smooth geometry (for a visual demonstration, zoom in on the molars in Figure 5.3).

Table 5.2: Rate of presence of teeth region classes within the dataset. Values represent the percentage of cases in which the (annotated) tooth is present. The values for categories represent the percentage of cases in which at least one of the teeth of that category is present. Note that the 3rd molar class (circled in red) is significantly underrepresented.

	Incisors		**Canines**	**Premolars**			**Molars**	
	L1, R1	L2, R2	L3, R3	L4, R4	L5, R5	L6, R6	L7, R7	L8, R8
by category [%]	96.41		94.65	96.41			91.66	
by tooth [%]	96.41	96.16	94.65	96.02	93.98	90.83	83.51	11.52

Individual tooth regions (segmentation masks) for each data sample were accurately labeled by a clinician in each case and subsequently reviewed and refined (quality-controlled) to ensure high quality of the labels. The shapes of the regions (except for the gingival region) follow typical tooth anatomy and vary primarily across tooth categories (incisors, canines, premolars, and molars). Minor nuances in shapes caused by anatomical variability are evident. The greatest variation in the segmentation regions occurs across tooth categories. Thus, Table 5.2 shows the data distribution of teeth classes and their categories. Except for the 3rd molar region, all classes are fairly uniformly represented. Note that each region is always formed by a single connected component.

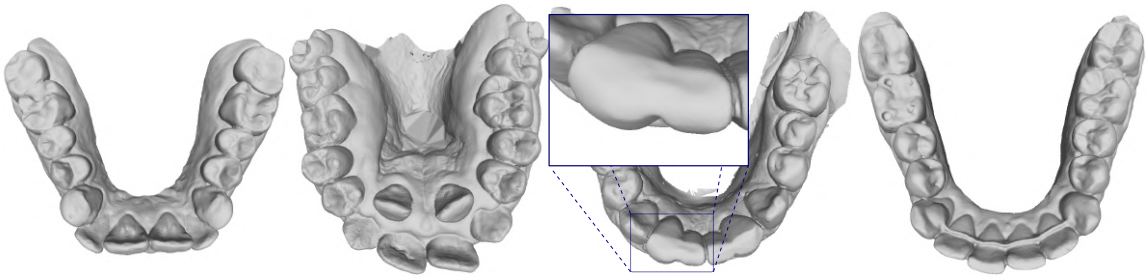


Figure 5.3: Example cases from the test set. From left: case with minor anomalies (sample from *basic test set*), case with severe transposition of lateral incisors, case with smooth incisal surfaces, and case with wire on lingual surfaces.

In order to ensure the best coverage of data diversity, 30 cases were thoroughly selected to form a test set. The process of rigorous selection of the test cases was assisted and discussed with an experienced dental planning software tester from TESCAN 3DIM, s.r.o. This holdout set was then divided into two smaller sets: *basic test set* with 10 cases with minimal or no dental irregularities and *complex test set* of 20 complex cases with various positional anomalies, scanned dental braces and wires and/or smooth surface parts without clear geometric features. See Figure 5.3 for sample dental scans of both the normal and abnormal subsets. Generally speaking, the tooth segmentation method must be able to segment teeth with as little error as possible, not only in simple cases but also especially in complicated orthodontic cases, in order to be usable in clinical practice. Splitting the test set can indicate whether the method is able to generalize well or whether there is a significant difference between the accuracy in the simple and complicated test set.

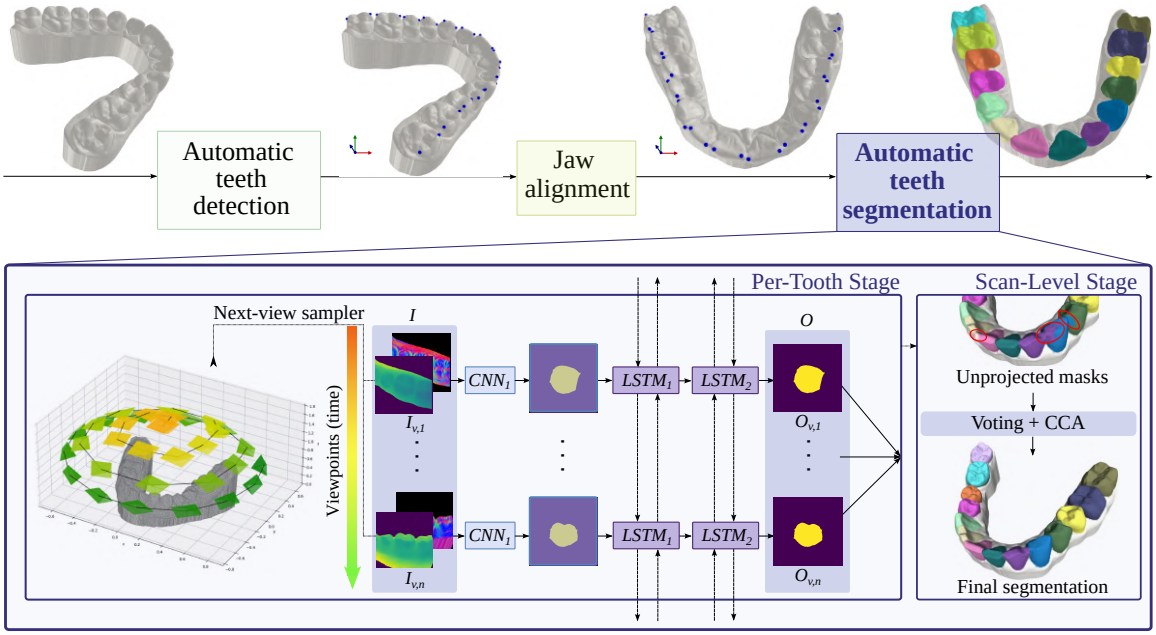


Figure 5.4: Schematic overview of the multi-view framework. Orthodontic landmarks are automatically detected using my previously published method [37]. Subsequently, teeth regions are individually segmented in aligned mesh by applying multi-view approach with inter-view context (*Per-Tooth Stage*). Teeth regions are post-processed in 3D by region voting and connected component analysis (*Scan-Level Stage*).

5.3 Recurrent Multi-View Segmentation Approach

A schematic view of the proposed multi-view deep learning framework is given in Figure 5.4. The whole pipeline consists of two automatic steps, with an intermediate conventional computation of alignment matrix:

1. **Automatic teeth detection.** This automatic module detects two occlusal/incisal landmarks on each tooth. Running tooth detection first has two advantages. In case of failure of the automatic method, from a practical standpoint, it is less laborious and much faster for the clinician to adjust the positions of the points than to adjust the boundaries of the 3D masks on the mesh. Second, preserving information about the location of landmarks makes it easier to solve other tasks by allowing for local processing. Since this part is not the focus of this work, the technical details are not further elaborated. The automatic module used in this work was designed and experimented within my Bachelor's thesis [36]. It is based on a multi-view representation, and it detects two anatomical landmarks on the occlusal/incisal surface of each tooth. If interested in the details of this step, refer to that thesis or to the published paper [37], which will provide a brief insight into how the method works. Note that any other teeth detection module could be employed.

2. **Jaw alignment.** To further facilitate the segmentation task, the positions of the detected points are used to align the model to a common coordinate system. Using the input points and employing the ICP algorithm [2], this step generates a transformation

matrix, that is applied to the analyzed geometry and is always upright oriented along the z-axis.

3. **Automatic teeth segmentation.** This step is the main goal of this thesis and will be the focus of the rest of this chapter.

To model the function $\mathcal{F}$, step 3. consists of two stages: (1) *per-tooth stage*, which is applied on each detected tooth separately – computations for each tooth can run in parallel, and (2) *scan-level stage*, which merges the per-tooth predictions and applies post-processing methods.

Per-Tooth Stage

Since the work relies on prior automatic detection and subsequent alignment to a common coordinate system, it is possible to perform feature extraction and mask generation **locally**, **separately for each tooth**. Thus, the 2D neural network has a higher yield of useful intricate geometric information at the input compared to a situation where a rendering of the entire 3D shape would form the input (considering the same amount of trainable parameters and input map size). Intuitively, when the third molar segmentation mask is generated in the left quadrant, the shape of the jaw in the right quadrant is irrelevant. This makes the method robust to positional teeth anomalies and, moreover, it suppresses errors that would be introduced by underrepresented tooth classes. The following sequence of steps is performed on each tooth individually:

1. **Generating input maps using next-view sampler.** Thanks to prior occlusion alignment, it is possible to obtain suitable initial camera extrinsic. The camera is positioned in a way that allows capturing the entire occlusal surface of the tooth, together with the nearby neighborhood, to preserve the contextual information of the adjacent teeth. A neighborhood of $\pm 12\,\text{mm}$ is captured with additional $3\,\text{mm}$ for molars. The following views are obtained by moving the camera in a spherically spiral manner with constant angular increase. The camera is always pointing at the center of given tooth (position is derived from landmarks). The multi-view configuration (viewpoint number and step size) is subject to experiment.

2. **Extraction of 2D segmentation masks.** From each camera viewpoint v, the following 2D input maps of size $H \times W$ are generated:

$$\boldsymbol{D_v} \in [0,\, 1]^{H \times W}, \text{ and} \tag{5.3}$$

$$\boldsymbol{N_v} \in [-1,\, 1]^{3 \times H \times W}, \tag{5.4}$$

such that $\boldsymbol{D_v}$ is the *depth map* where 0 indicates intersection at infinity, i.e. no intersection with the analyzed mesh $\mathcal{M}$, and the remaining values are normalized lengths of rays cast orthogonally in the direction of camera's direction (look-at) vector until first hit with $\mathcal{M}$ (distance between camera's z coordinate and z-coordinate of hit primitive). $\boldsymbol{N_v}$ is the *normal map* obtained in a similar fashion, but the tuple in each pixel holds information about the normal vector of first hit primitive (normal vectors in screen-space). Figure 5.5 visualizes the aforementioned map renderings. The depth map brings the 3D context into 2D processing, while the normal map yields information about curvature. These maps are then stacked to form a single input for the network:

$$\boldsymbol{I_v} := \{\boldsymbol{D_v}, \boldsymbol{N_v}\} \in \mathbb{R}^{4\times H\times W}. \tag{5.5}$$

Defined in this way, the method is also robust against various lightning, shading, reflectance models, etc. Since the network is based on a multi-view approach, there is a set of such inputs $I = \{\boldsymbol{I_{v,1}}, \ldots, \boldsymbol{I_{v,n}}\}$, where $\boldsymbol{I_{v,i}}$ is the map from the i-th $(1 \leq i \leq n)$ position of camera, and n the total number of viewpoints. Each element of I then serves as an input for the segmentation network denoted as CNN_1. This network is shared across all viewpoints and adapts *U-Net shape* [58], with minor modifications (see Section 6.3 for details). U-Net is chosen because it is a well-known fully-connected network architecture, achieving huge success in biomedical image segmentation.

Since the configuration of next-view sampler is fixed, the set I can be perceived as a **sequence** of n samples. As the outputs of CNN_1 lack the inter-viewpoint context, further tuning by *ConvLSTM* [62] is employed, to ensure **correlation of views and consistency of the boundaries of the segmentation regions**. As the input sequence I can be generated prior to the 2D segmentation mask prediction, it is possible to employ *Bi-Directional ConvLSTM* [7], allowing to capture the context of the sequence in both directions. So, the FCN components extract intra-view context, whereas the next RNN components can concentrate on the inter-view context.

The weights of the FCN and RNN components can be trained *end-to-end*, and together form the 2D segmentation mask extraction function. The output of this function for each tooth is then a sequence of 2D binary segmentation masks $O = \{\boldsymbol{O_{v,1}}, \ldots, \boldsymbol{O_{v,n}}\}$, where each $\boldsymbol{O_{v,i}} \in \{0,1\}^{H\times W}$ for each i in $(1 \leq i \leq n)$.

3. **Unprojection of the predictions to $\mathbb{R}^3$.** In this step, the sequence of output maps O is transformed into a set of faces $F_{\mathcal{C}} \subseteq F$ representing the segmented tooth. For each output map $O_{v,i}$, the extrinsic of camera is set to the same as when the corresponding input has been generated. Then, the rays are cast orthogonally only through those pixels of $O_{v,i}$, whose value is 1. For each face $f \in F$ hit by any of the rays, the value of $F_{\mathcal{C}}$ is set to $F_{\mathcal{C}} \cup \{f\}$.

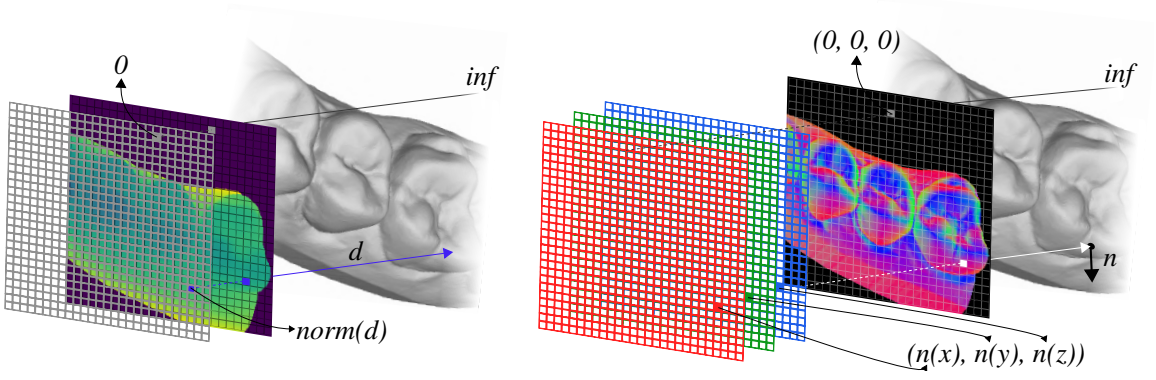


Figure 5.5: Rendering procedures of 2D maps for multi-view network inference. Pixel values in depth maps (left) correspond to the normalized value from camera's z-position to mesh intersection. RGB values of normal map (right) correspond to x, y, z values of normal vector of hit triangle, respectively.

Scan-Level Stage

Once the previous sequence of steps is applied to each tooth, it is necessary to merge all the regions on the target mesh $\mathcal{M}$. There are two major issues that might arise from the independent generation of the per-tooth region: (1) some faces $f \in F$ might be assigned to multiple classes from $\mathcal{C}$, and (2) the unprojected regions do not form a single connected component, although it is a desirable behavior based on the definition of the problem. The former is solved by *region voting*, whereas the latter is solved by *connected component analysis*. Both situations are illustrated in a simplified case in Figure 5.6.

1. **Region voting.** To tackle the cases when a face $f \in F$ is assigned to multiple classes from $\mathcal{C}$, let's first define *multi-view certainty* U as

$$U \subseteq F \times \mathcal{C} \times P, \tag{5.6}$$

such that $P = \{p \mid p \in [0, 1]\}$. Intuitively, each tuple $(F_i, \mathcal{C}_i, p_i) \in U$ defines certainty p_i of face F_i being a member of class $\mathcal{C}_i$, and is obtained as the ratio of the number of views from which the face was hit by a cast ray, and the number of views from which the face was initially visible. For a clearer demonstration, U can be perceived as a mapping

$$U \colon F \times \mathcal{C} \mapsto [0, 1]. \tag{5.7}$$

This mapping then assigns per-face class in the following fashion:

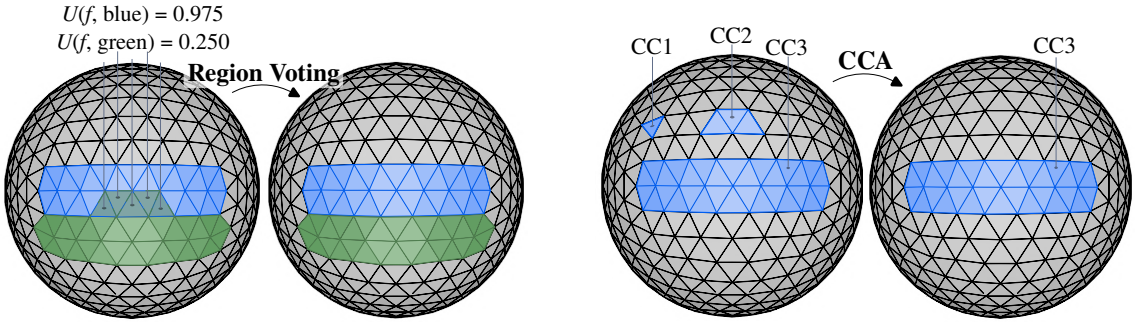
$$\mathcal{F} \leftarrow \begin{cases} \arg\max\limits_{c \in \mathcal{C}} U(f, c) & \text{if } \max\limits_{c \in \mathcal{C}} U(f, c) \neq 0, \\ G & \text{if } \max\limits_{c \in \mathcal{C}} U(f, c) = 0. \end{cases} \tag{5.8}$$

This means that each face is assigned to the class with highest multi-view certainty. If the certainty is 0 for each teeth classes, the face is associated with the class that represents the gingiva.

2. **Connected component analysis.** As previously stated, the problem is formulated in a manner in which each tooth in the dental scan corresponds to a single connected region. Therefore, it is necessary to remove all outlying faces. Face is considered outlying, when it is not part of the largest segmented region of given class. The CCA approach from graph theory is used to accomplish the removal of outlying faces. In case there is a class which is represented by more than one component, only the component with the highest face count is preserved. Faces of the remaining components are associated with classes respecting the multi-view certainty.

5.4 Proposed Non-Euclidean Frameworks

As an alternative to proposed recurrent multi-view framework, two other non-Euclidean approaches are proposed. They will primarily serve as a fair comparison of the previously introduced method to other approaches (so that they are trained and evaluated on the same dataset). In these two non-Euclidean frameworks, one considers vertices as the main element, and the other one considers edges as the main element. Both methodologies share



(a) Region voting **(b)** Connected component analysis

Figure 5.6: Scan-level stage operations. In **(a)**, five faces are assigned to two classes (blue and green classes overlap there). When the multi-view certainty is examined, given faces are assigned to the blue class, as it's of higher value. In **(b)**, only CC3 region is retained, as it is the component with highest number of primitives. Note that both cases may occur commonly in the task where the regions are classes of teeth: masks overlap on the adjacent teeth, isolated triangles arise for instance from backprojection step.

a common design principle in which they require prior teeth detection (see beginning of Section 5.3). This prerequisite is critical in their practical use, as it facilitates the analysis of mesh cuts comprising only a limited vicinity of the segmented tooth. Such an approach ensures that the analysis preserves intricate geometric details, which would be otherwise lost by severe undersampling of the analyzed point cloud or heavy mesh decimation. Note that undersampling/decimation would become necessary when analyzing the entire jaw simultaneously, given the memory limitations imposed by the non-Euclidean neural network architectures.

5.4.1 Point-Based Segmentation Approach

In a similar fashion as in the multi-view framework, feature extraction and region proposal is applied in **per-tooth manner**:

1. **Generating input point cloud of interest.** Given tooth center point $c \in \mathbb{R}^3$ (derived from landmarks), the target mesh $\mathcal{M}$ is first cut by six planes parallel to coordinate planes, such that the point-plane distance d of individual planes and point c define the amount of contextual information that will be provided to the neural network. The value of d is set to 12 mm (15 mm in case of molars) so that the geometry of the adjacent teeth is preserved in the cut mesh. The resulting mesh is then a tuple $\mathcal{M}_{cut} = (V_{cut}, E_{cut}, F_{cut})$, where $V_{cut} \subseteq V$, $E_{cut} \subseteq E$, and $F_{cut} \subseteq F$. Then, the *mesh to point cloud* procedure follows. As the modeled function $\mathcal{F}$ assigns classes to faces, it is necessary to preserve a correspondence between the points in the analyzed point cloud and the target faces. So, the input point cloud $\mathcal{P}_I = \{p_x^i, p_y^i, p_z^i, n_x^i, n_y^i, n_z^i\}_{i=1}^n$ yields the position of the point in $\mathbb{R}^3$ and its normal vector obtained as follows. For each face $f \in F_{cut}$, a point is sampled using barycentric coordinates u, v, w:

$$(p_x, p_y, p_z) = uf_1 + vf_2 + wf_3, \tag{5.9}$$

where f_1, f_2, and f_3 are the vertices of triangle f, and values of u, v, and w are randomly generated so the point lies within the face f ($0 \leq u, v, w \leq 1$ and also

$u + v + w = 1$). Random placement brings small perturbations, especially beneficial in training phase, which is advantageous as it reduces network's sensitivity to noise. The normal vector (n_x, n_y, n_z) is simply the normal of the corresponding face f. Note that if only one point is sampled from each face, the cardinality n of $\mathcal{P}_I$ equals to the number of faces of $\mathcal{M}_{cut}$, but multiple points can be sampled from one face, increasing the resolution of the point cloud. The whole procedure is illustrated in Figure 5.7.

2. **Extraction of point segmentation mask.** Point cloud $\mathcal{P}_I$ then directly forms an input to a neural network, which is designed as PointNet++ [57] (experiments were also performed with PointNet [56]). The result of inference is a set

$$O = \{o \mid o \in \{0, 1\} \wedge |O| = n\}, \tag{5.10}$$

where zero indicates that the corresponding point does not contribute to the tooth region, and 1 indicates that it does.

3. **Mapping point predictions to target mesh.** Each predicted value of O corresponds to a point $p \in \mathcal{P}_I$, and that point corresponds to a face $f \in F_{cut}$, which is also included in the set F of the target mesh $\mathcal{M}$. Using these correspondences, the prediction is transitively mapped to the target mesh.

The **scan-level stage** is then limited by the fact that there is no possibility for region voting, as the certainty of prediction can not be directly derived as in multi-view approach. It would be possible if multiple points were sampled for each triangle, however, the resulting point cloud would be prohibitively large for processing on average GPUs. Consequently, the scan-level stage resorts solely to connected component analysis, which satisfies the requirement of producing single region segmentation masks.

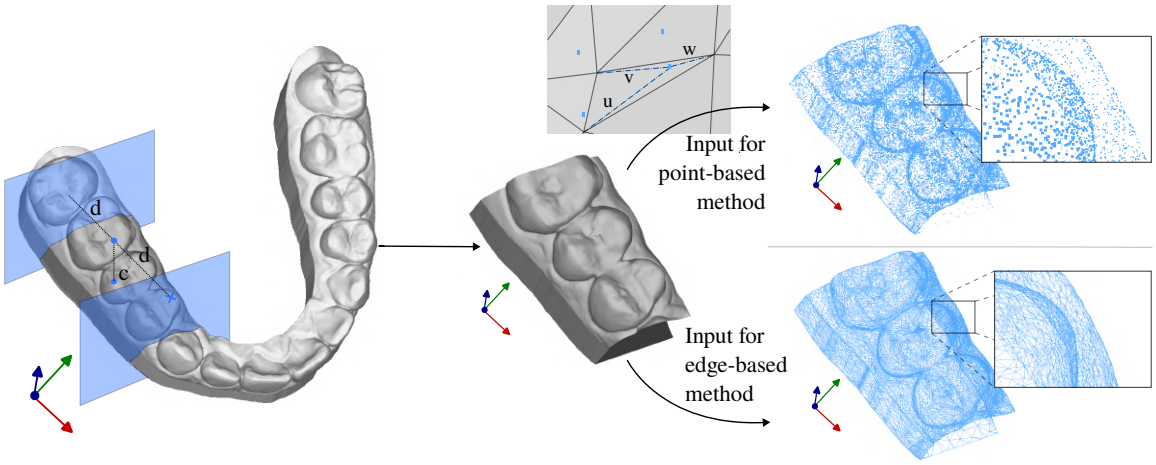


Figure 5.7: Input preparation procedures for proposed non-Euclidean frameworks. For each tooth within input scan, a mesh cut is first obtained by cutting the mesh by 6 planes, with distance d from tooth center c (for simplicity, only two planes are shown). This cut is either directly used as an input for edge-based method, or it is transformed into point cloud. In the latter, applied sampling technique randomly picks a point from each face by barycentric coordinates.

5.4.2 Edge-Based Segmentation Approach

Following the **per-tooth computation** principle, the edge-based solution also takes a tooth cut $\mathcal{M}_{cut}$ as an input, which is defined the same way as the point-based method. For each edge $e \in E_{cut}$, a 5-dimensional feature vector f_e is calculated and is composed of the dihedral angle, two inner angles, and two edge-length ratios for each face, such that the edge ratio is calculated from the length of the edge and the perpendicular line for each adjacent face. The use of these intrinsic features is inspired by the approach presented in [24]. Segmentation network based on the SparseMeshCNN architecture is employed [26], and it directly takes the 3D shape $\mathcal{M}_{cut}$ as input in the form of an edge-feature tensor. Suppose n to be $|E_{cut}|$ in this setup, the output is a set O with cardinality of n, containing class predictions of the same format as in Equation 5.10. Note that these predictions are **per-edge**. These predictions are effectively propagated to each $f \in F$ by examining the predicted class labels of its incident edges in the same fashion as presented in [24]. Subsequently, after mapping the regions to $\mathcal{M}$ for each tooth, the analysis of connected components is calculated at the scan level.

Chapter 6

Implementation and Training Details

This chapter outlines the details of the implementation of proposed methods. It provides a detailed overview of the equipment used, together with comprehensive information on the training and evaluation parameters. The objective is to adhere to the principles of reproducible research, ensuring that the readers have the necessary information and resources to accurately reproduce the results presented in this study.

6.1 Toolset

The entire segmentation framework is implemented using `Python`[1] due to its preeminent position within the domain of deep learning research. For building, training, and evaluating neural network models, open-source framework `PyTorch`[2] was used. Since `PyTorch` does not provide its users with a platform for monitoring training processes, evaluation runs and other visualizations, `WandB`[3] platform was employed for this particular purpose. For CPU computations, the `NumPy`[4] was utilized, as it provides vectorized versions of all the necessary computations required for this work. For all mesh and point cloud processing and visualization, `Trimesh`[5] package was used. It provides an abstraction for analyzed 3D shapes and point clouds as well as a viewer for programmable visualization of 3D geometry. This library does not provide implementation for some mesh processing operations, so other 3D shape analysis libraries were used as well, such as `Open3D`[6] for the ray casting method and `PyMeshLab`[7] for various mesh augmentation techniques.

6.2 Training and Evaluation Infrastructure

In order to facilitate the training and experimentation of the proposed approaches, it was necessary to devise a thoughtful infrastructure. This was particularly crucial given the utilization of diverse representations of 3D shapes, a range of networks that process various

[1] https://www.python.org/
[2] https://pytorch.org/
[3] https://wandb.ai/site
[4] https://numpy.org/
[5] https://trimsh.org/index.html
[6] https://github.com/isl-org/Open3D
[7] https://pymeshlab.readthedocs.io/en/latest/index.html

input types, and others. It also allowed for seamless experimentation with different subsets of the dataset, augmentations, and other variables. The following sections provide brief descriptions of crucial components of the infrastructure, their interconnections, and how the other libraries are used. It also clarifies a portion of the code that was adapted from existing implementations (several classes for point-based and edge-based approaches). Note that unless explicitly specified, the code is an original implementation.

Data Handling

Datasets are organized in `JSON`[8] format. The reason for this is twofold: in addition to geometry and segmentation masks information, each data sample contains annotated landmark positions, alignment matrix, jaw type, and others. These metadata need to be properly organized to allow for their seamless processing. Second, this brings a simple way to merge multiple datasets, calculate dataset statistics, versioning, and querying for specific records. Each data sample (i.e. each record in `JSON` dataset) is represented as an instance of `GenericMesh` class. This abstraction contains reference to a `TriMesh` object, i.e. the geometry of given 3D shape, and all other annotations and metadata in the form of class variables. Within the behavior, methods like jaw alignment or visualizations are defined. Instances of `GenericMesh` also contain a reference to a list of `ToothMeshCut` instances. These represent data samples of individual teeth, since the proposed methods operate locally on each tooth. When instantiating, it is possible to specify the width of the cut – how far are the planes that cut the geometry of parent `GenericMesh` from given tooth center. More important is the behavior of the `ToothMeshCut` class, which contains methods for converting local mesh to multi-view representation, point cloud representation, and a representation suitable for the MeshCNN-based method.[9] These methods can be used either to pre-generate the dataset of input features on disk or for on-the-fly sampling during training and evaluation procedures. Prior to any dataset analysis (before any conversion of `JSON` records to a list of `GenericMesh` instances), the processing goes through `JsonInterface` layer. This class contains *endpoint-like* methods for filtering only cases with specific characteristics. This is handy for experimenting with particular subsets of the dataset. It is possible to filter for example by jaw types or only cases with a specific tooth present. Individual fully parameterizable augmentation techniques are implemented within classes `Augmentator2D` and `Augmentator3D`. Finally, datasets for individual approaches are wrapped by classes `DatasetMultiView`, `DatasetMultiViewLSTM`, `DatasetPointNet`, and `DatasetMeshCNN`. These inherit from `PyTorch`'s `Dataset`[10] class. Combined with `DataLoader`[11], the filtered `JSON` records in the form of `ToothMeshCut` instances are possible to automatically sample in batches, process in parallel, and others, which simplifies the training procedure. The values of the monitored metrics and loss functions are stored within `WandB` run specified by the corresponding experiment configuration and unique id. The platform is also used to store trained weights and traced models throughout training.

[8] https://www.json.org/json-en.html
[9] Input feature generation adapted from https://github.com/s183983/Sparse_MeshCNN_w_Attention.
[10] https://pytorch.org/docs/stable/data.html#torch.utils.data.Dataset
[11] https://pytorch.org/docs/stable/data.html#torch.utils.data.DataLoader

Training and Evaluation Pipeline

The hyperparameters of each training run are specified by command-line arguments for individual training scripts (each of the proposed approaches runs different script – for example `train_multi_view.py` and `train_point_based.py`). Furthermore, each run is presented by a unique identifier that is used for associating experiments, trained models, and evaluation results. The specified hyperparameters and unique identifier are also used to create a training run on the `WandB` logging platform. The training procedure is carried out in the `Trainer` class. In its constructor, objects of appropriate classes are initialized, according to the specifications: one of the aforementioned `Dataset` classes for handling data, network architecture to be trained – `UNet2D`, `UNetLSTM`, `PointNetSeg`[12], `PointNet2Seg`[13], or `SparseMeshCNNSeg`[14], loss functions, optimizers, and other parts needed for training. To evaluate the performance of the framework based on a specific training configuration, it is essential to define a training run. The provided script automatically loads the corresponding weights and executes the evaluation process. The evaluation metrics obtained during this process are stored in a `JSON` format, facilitating easy visualization and analysis of the quantitative results. This enables comparisons between different runs and approaches, aiding in the analysis and interpretation of the performance.

6.3 Architectures of Employed Models

A deep neural network architecture was designed for each of the proposed approaches. In the following subsections, the individual architectures are introduced, with the primary focus on the main method of this work – the multi-view approach with recurrent blocks.

6.3.1 Recurrent Multi-View Segmentation Model

As already discussed in Section 5.3, the task of automatic tooth region segmentation in 2D is approached as a sequence processing. First, the fully-connected CNN component constructs a feature map f_{FCN} for each of the viewpoints. It aims to extract visually object-relevant information (e.g. intricate tooth surface geometry), while the object-irrelevant information is discarded. Then, the recurrent component concentrates on the inter-view context by processing f_{FCN} maps sequentially, adjusting the displacements of produced masks and thus bringing consistency on the boundaries. Please refer to Figure 6.1 to see details of the designed multi-view segmentation model. The FCN part is formed by a U-shaped encoder-decoder, inspired by the U-Net architecture [58]. Common characteristics with U-Net are employed skip connections, which enable the flow of information between the contracting (encoder) and expanding (decoder) stages at multiple resolutions. These connections help in preserving and combining low-level and high-level features during the upsampling process, allowing to access both local and global information of analyzed maps. To reduce overfitting by regularization, additional batch normalization layers were added in each stage of the encoder, between the convolutional and ReLU layers. Upsampling is accomplished by transposed convolutional layers. The architecture consists of four stages and a bottleneck. With hidden features dimensionality set to $32, 64, 128$, and 256, the network is of sufficient capacity given the task, size and character of the analyzed data. It

[12] Model architecture adapted from https://github.com/yanx27/Pointnet_Pointnet2_pytorch.
[13] See footnote 12.
[14] Model architecture adapted from https://github.com/s183983/Sparse_MeshCNN_w_Attention.

is trained to extract features from input maps of size $256 \times 256 \times 4$, which correspond to stacked depth and normal map, and producing a feature map f_{FCN} with dimensionality of $256 \times 256 \times 1$. Additionally, *deep supervision* [71] is employed in the fully connected part of the network. To take the advantage of deep supervision, the total loss value $\mathcal{L}_{total}$ is defined as follows:

$$\mathcal{L}_{total} = \sum_{i}^{S-1} \mathcal{L}_i + \mathcal{L}_{final}, \tag{6.1}$$

where $\mathcal{L}_i$ is the loss value computed at the decoder stage i and $\mathcal{L}_{final}$ corresponds to the loss value computed from the output layer. Deep supervision provides shorter paths for gradients to flow during backpropagation and amplifies hierarchical feature learning.

To capture the dependencies between features obtained from individual viewpoints, they are fed into a recurrent unit. The recurrent unit is composed of two layers of bi-directional convolutional LSTM (ConvLSTM) [7], where input, forget, and output gates of LSTM cells are computed through convolutional operations instead of fully connected layers. The feature size of each of the LSTM units is 32. In the training process, the loss is finally computed as the sum of the deep supervision losses and a final loss value, and optimized through the final FCN layer and the recurrent unit.

6.3.2 Design of Point-Based and Edge-Based Architectures

The architectures of the two chosen non-Euclidean approaches are conceptually the same as the U-Net-like architecture employed in the multi-view approach. Obviously, they differ in the operators used in each layer, as presented in Section 4.2, as well as in the input format. In their principle, the architectures follow an encoder-decoder structure with skip-link concatenations. Thus, it is sufficient to define the dimensionality of the features at each level, together with the corresponding point cloud/mesh resolutions. Refer to Table 6.1 to see the values of the architecture parameters of the best-performing designs. Additionally, for the edge-based method, attention modules were added to two bottom layers.

6.4 Mesh Pre-Processing and Augmentation Techniques

Prior to any processing and analysis, all shapes are uniformly scaled to fit into a unit cube. In addition. All augmentation techniques that are common for all approaches are directly applied on the 3D shape. As the edge-based approach is similarity-invariant, applying random mesh rotation, translation, or isotropic scaling would not generate new informative data samples. For that reason, anisotropic scaling of vertex locations is applied. Values are sampled from a normal distribution with $\mu = 1$ and $\sigma = 0.1$. Random vertex displacements are also employed, where 30% of vertices are displaced by a random value from range $\langle 0.0\,\text{mm}, 0.08\,\text{mm}\rangle$. This produces new features mainly for point-based approach. To improve the robustness to various geometry tessellations, operations like random edge flips and edge collapses are applied on the shapes.

Most of the augmentation techniques focus on the geometric aspect of the input shape. To further increase the robustness and generalization ability of multi-view solutions, one can benefit from the augmentation techniques used in the image domain. When rendering the maps, the focal point of the camera is perturbed by ± 0.2 mm and the viewing direction is altered by an angle up to $\pm 8^\circ$ during sequence generation. 2D maps are augmented by random scaling factor of range $[0.90, 1.10]$ and rotation of range $[-30, 30]$ degrees.

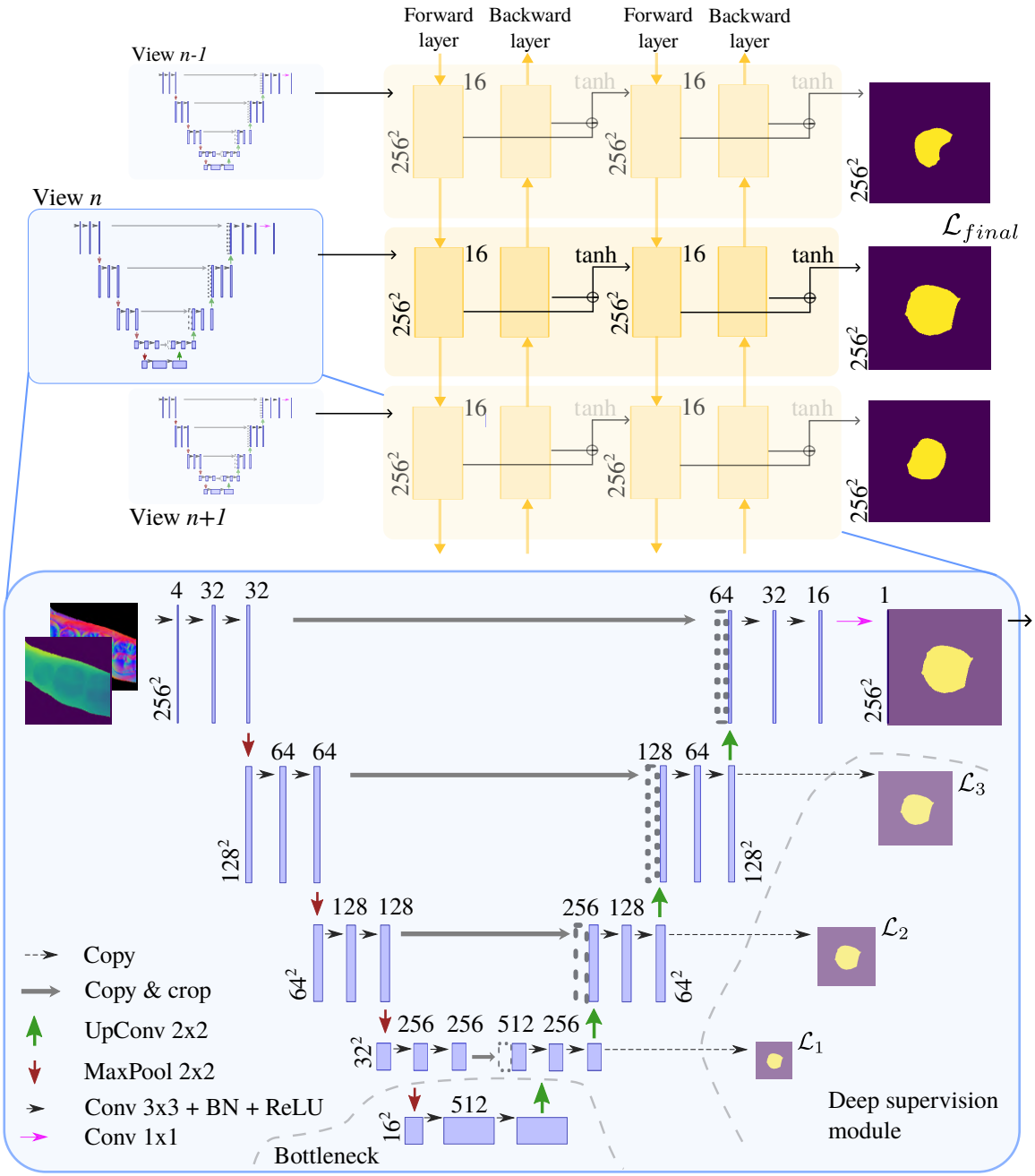


Figure 6.1: Detailed scheme of proposed multi-view recurrent segmentation network. Blue part of the scheme represents the fully-connected CNN part. Deep supervision module is present only during training. Yellow part represents the recurrent module with bi-directional ConvLSTM units.

6.5 Specification of Training Configurations

To simplify the reproducibility of this work, this section presents the configurations of the training parameters for the best-performing setup of each of the proposed approach (multi-view, point-based, and edge-based). All important parameter values are summarized in Table 6.1. The augmentation specifications are excluded from the table, as they are already

presented in Section 6.4. The most sensitive to changes in configuration is the edge-based approach, where even slight changes in hidden features dimensionality drastically decrease method's performance.

Table 6.1: Summary of training configurations for proposed approaches. Table shows hyperparameters for the best-performing configurations. *Extrinsic geometric features in point-based methods are formed by a vector of point position and normal vector. †Intrinsic geometric features in edge-based method are composed of dihedral angle (the one between two incident triangles), inner angles, and two edge-length ratios.

Hyperparameter	**multi-view solution**	**point-based**	**edge-based**
architecture design	U-Net + ConvLSTM	PointNet++	SparseMeshCNN
iteration count	400 000	200 000	200 000
initial learning rate	0.001	0.001	0.001
learning rate scheduler	cosine annealing	cosine annealing	cosine annealing
lr scheduler patience (# iterations)	25 000	10 000	10 000
optimizer	AdamW	AdamW	AdamW
betas for optimizer	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)
weight decay	0.001	0.01	0.01
loss function	2D Dice	3D Point Dice	3D Edge Dice
expected input resolution	256 × 256 px	16 384 points	65 536 edges
type of input features	depth + normal map	extrinsic*	intrinsic†
batch size	2	8	4
hidden features dimensionality	$[32, 64, 128, 256]$	$[32, 64, 128, 256, 512]$	$[16, 32, 64, 128, 256]$
resolutions (in encoder)	$[256^2, 128^2, 64^2, 32^2]$	$[4\,086, 1\,024, 256, 64, 16]$	$[20k, 16k, 8k, 4k, 2k]$
2D deep supervision	✓	-	-
self-attention edge blocks	-	-	✓

Chapter 7

Conducted Experiments and Achieved Results

The main objective of the experiments is to investigate whether the proposed multi-view method handles tooth segmentation in simple cases, while being able to generalize on complex cases (refer to Section 5.2 to see the description of individual datasets). Only in such a case the proposed approach is applicable to clinical practice. Moreover, the aim is to show how individual design decisions affect the quality of the outputs. This may be relevant for future research since it will become apparent which parts of the framework contribute to the overall performance and what kind of errors they can suppress. All of this forms the *first axis of experimentation*, focusing on the primary method of this thesis. In Section 7.2, experiments with this method are presented in turn, primarily in the form of an ablation study.

Subsequently, in Section 7.3, an *evaluation along the second axis* is presented. In this group of experiments, the best configuration of the main multi-view method is selected, which is then compared to several variants of the proposed non-Euclidean approaches (point-based and edge-based) and one conventional segmentation algorithm based on the idea of Graph-Cut. The goal is to determine which of the approaches is best suited for the task of segmentation of teeth. Furthermore, it is also desirable to determine what may be causing the underperformance of some of the approaches. These experiments also include comparisons with the state-of-the-art tooth segmentation methods in the literature. However, since none of these methods provides a public implementation and it is also unclear what the nature of their dataset is, this comparison is not entirely fair. However, in general, it can provide a preliminary assessment of the comparative performance of the methodology proposed in this study with respect to current state-of-the-art approaches.

For the experiments conducted in this work, NVIDIA GeForce RTX 3060 with 12GB of memory was utilized to assess the performance.

7.1 Evaluation Metrics

For meaningful and fair quantitative evaluation, two complementary metrics are employed: one *overlap* and one *boundary* metric. Each of them reveals distinct types of errors in the generated segmentation masks. Together, they provide a more comprehensive assessment of segmentation performance [1]. To give an example, if only a boundary metric was employed, it might not be sufficient to identify the issue of over-segmentation, as it does not provide

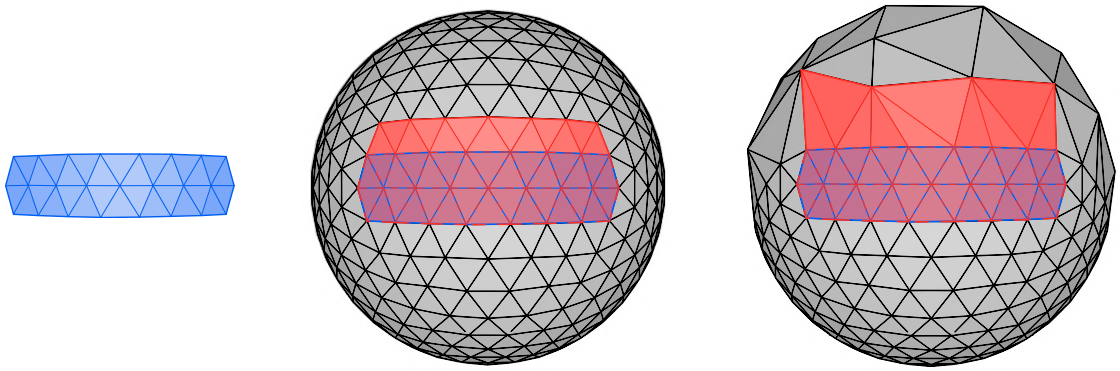
information about the extent to which the predicted regions are correctly overlapping with the ground truth. Alternatively, a good result on the overlap metric would not indicate a poor boundary localization in the form of narrow but long false positive or false negative blobs. The section follows with the formulation of particular metrics employed. Please note that the performance is **always evaluated on the mapped target masks** in $\mathbb{R}^3$ in original physical scan dimensions.

Overlap Metric – Weighted Intersection over Union

In contrast to the IoU computed, for example, over images, where each pixel is usually of the same size, this metric takes into account the fact that various triangle sizes contribute differently to the final value. The *W-IoU* is calculated as follows:

$$W\text{-}IoU = \frac{\sum\limits_{f \in F_\cap} A_f}{\sum\limits_{f \in F_\cup} A_f}, \tag{7.1}$$

where A_f is area of face f, $\hat{F}$ and F are faces of ground truth and prediction masks, $F_\cap = \hat{F} \cap F$, and $F_\cup = \hat{F} \cup F$. *W-IoU* detects errors such as under/over-segmentation, or shift errors. Note that the use of an overlap metric that does not weight values by the polygon area is considered inaccurate and incorrect. Refer to Figure 7.1 which illustrates the superior performance indication compared to the unweighted version.



(a) GT Mask

(b) $IoU = 0.69$
$W\text{-}IoU = 0.69$

(c) $IoU = 0.75$
$W\text{-}IoU = 0.48$

Figure 7.1: Impact of weighting factor on IoU metric. Red mask is the prediction, that overlaps the GT mask, but also produces false positive prediction of a triangle stripe above. Attempting to predict the mask depicted in **(a)**, the weighting has no effect in **(b)**, since all triangles have a uniform area equal to 1. However, **(c)** clearly shows the importance of weighting. Area-wise, there is larger error in the prediction as compared to **(b)**. However, the metric without employed weighting in fact reports a smaller error, since the area is composed of less triangles.

Boundary Metric – Symmetric Hausdorff Distance at 95 Percentile

Let $\hat{V}$ be a point cloud of ground truth boundary vertices (vertices forming the edges of the mask boundary), and V be a point cloud of prediction boundary vertices. Let d be the

Euclidean distance in $\mathbb{R}^3$. Symmetric Hausdorff distance, d_H, is calculated as

$$d_H(\hat{V}, V) = \max(d_{\hat{V}V}, d_{V\hat{V}}) = \max\{\max_{x\in\hat{V}} \min_{y\in V} d(x,y), \max_{y\in V} \min_{x\in\hat{V}} d(x,y)\}. \tag{7.2}$$

d_{H95} is its 95th percentile. This metric detects artifacts in segmented masks such as missing details or narrow protruding parts. The value corresponds to real-world dimensions in *millimeters* (*mm*).

7.2 Evaluation Axis 1: the Recurrent Multi-View Approach

This section presents in-depth experiments with the proposed recurrent multi-view method. The ablation study, the effect of multi-view configuration, and the exploitation of multi-view certainty to highlight possible errors in the detected mask are presented in turn.

7.2.1 Ablation Study of Key Framework Components

Quantitative results of the ablation study are summarized in Table 7.1. Further elaboration on the presented results follows.

Usefulness of Different Types of Information in Input Maps

The results regarding the input format demonstrate two things. First, the depth information is crucial for robust representation of intricate tooth shapes, whereas normal maps alone represent the teeth less informatively. Second, stacking normal maps with depth maps brings into learning a promising inductive bias for improving the results at the tooth-gingiva boundary. Curvature information in normal maps enables the network to generate smoother and more consistent segmentation results, particularly in complex and irregular boundary regions. So, for all the remaining experiments with multi-view solution, the combination of these two maps is always considered.

Table 7.1: Measured results for the ablation study. *Ds*: depth maps. *Ns* normal maps. *Deep Sup*: deep supervision. *LSTMs*: bi-directional ConvLSTM units. *Voting*: region voting. *CCA*: connected component analysis. Values in the table present means calculated on all teeth in cases of particular test sets.

Method Component						Basic cases		Complex cases	
Ds	*Ns*	*Deep Sup*	*LSTMs*	*Voting*	*CCA*	*W-IoU* ↑	d_{H95} ↓	*W-IoU* ↑	d_{H95} ↓
✓						0.8799	1.3742	0.8613	1.4060
	✓					0.8515	1.6715	0.8492	1.8221
✓	✓					0.9112	1.1621	0.8898	1.3199
✓	✓	✓				0.9154	1.1500	0.8815	1.2986
✓	✓	✓	✓			0.9222	1.0372	0.9071	1.2435
✓	✓	✓	✓	✓		0.9718	0.4311	0.9503	0.5400
✓	✓	✓	✓	✓	✓	**0.9781**	**0.3611**	**0.9553**	**0.4033**

Deep Supervision

Next, the contribution of training under deep supervision was analyzed. When applied to the training process, one can observe increased training stability and faster convergence. These two aspects of training are particularly important when training on medical data (or on any limited datasets in terms of size), decreasing the risk of overfitting. Furthermore, there is a visible marginal improvement in performance metrics. These findings support the notion that the multi-view approach yields good results also because it draws on the well-studied field of 2D CNNs, from which concepts like deep supervision can be easily applied for the analysis of geometric data. Please refer to Figure 7.2 for the illustration of deep supervision module on the stability of the training.

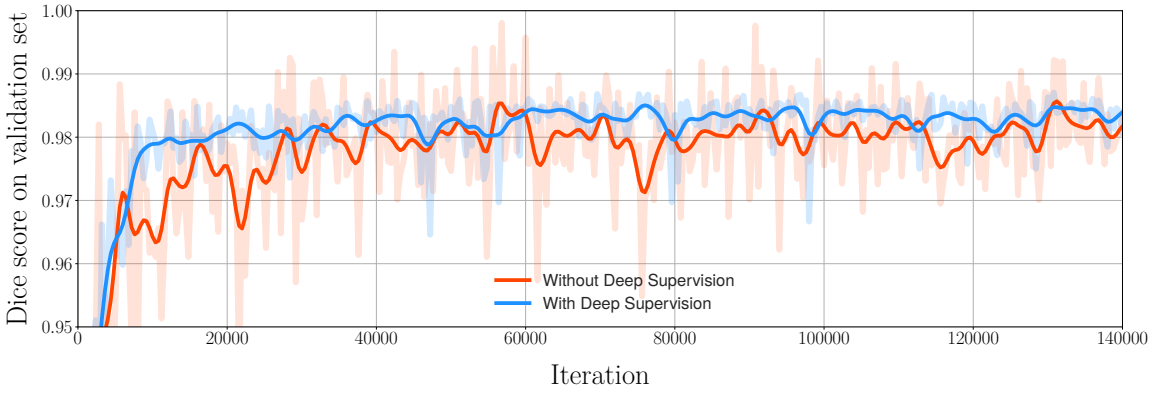
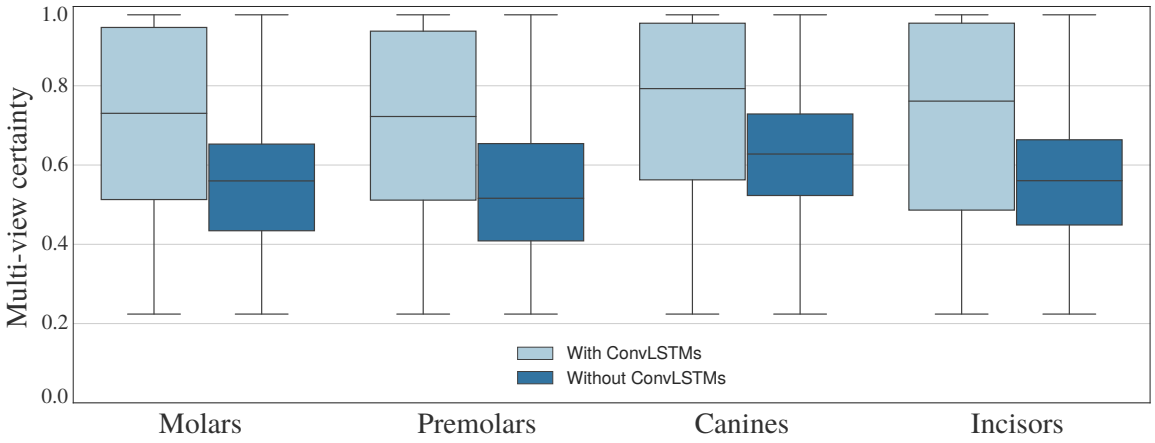


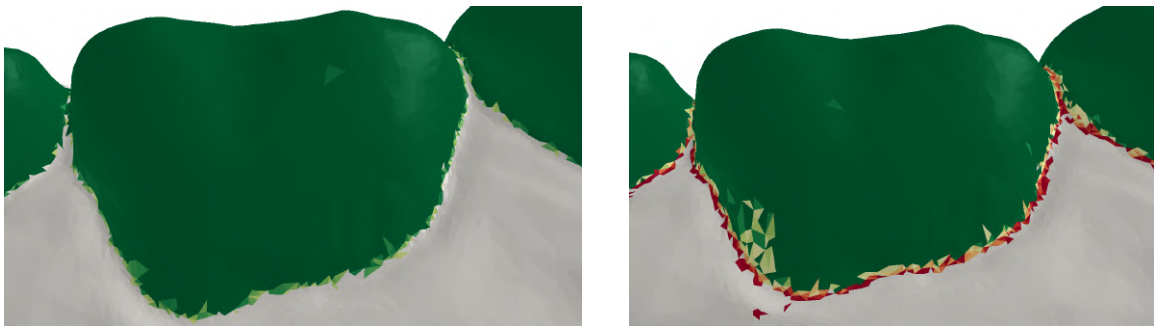
Figure 7.2: Impact of deep supervision on training stability and convergence. With employed deep supervision during the training process, both the values of the loss as well as of measured metrics are more stable, oscillate less, and converge faster. The semi-transparent plots represent the original Dice values on validation set, while the bold lines are their versions smoothed by 1D Gaussian with sigma of 2.

Recurrent Bi-Directional ConvLSTM Blocks

Without imposing an inter-view context, the generated segmentation masks have inconsistent region boundaries. This results in region boundaries appearing unnatural in some cases, for example, spiky and jagged artifacts can be seen. Recurrent units bring substantial increase in performance metrics, which support the hypothesis of the importance of inter-view coherence. The presence of a wider 3D context mainly results in generation of more consistent and coherent tooth region boundaries. To demonstrate the influence of recurrent blocks, *multi-view certainty* (previously described in Section 5.3) is used. Indeed, the inconsistency in region boundaries should be reflected in decreased multi-view certainty on triangles close to the boundary, as they are less often hit from all viewpoints. The graph and visualization in Figure 7.3 shows that this is true for cases from all teeth categories. Without the recurrent units, even though the triangles were assigned to a given tooth class, the certainty near the boundaries was lower by approximately 20%. In the medical community, certainty of predictions fosters trust and acceptance of the technology, and overall might help in clinical decision making (a more detailed attempt will also be introduced in Section 7.2.3). In this particular case, it mainly serves as a tool for model selection and comparison.



(a) Quantitative multi-view certainty comparison at mask boundaries



(b) Visual comparison with certainty on teeth masks encoded into green-to-red color map

Figure 7.3: Comparison of multi-view certainties at region boundaries with and without bi-directional ConvLSTM units. Values in **(a)** are measured on 5-ring neighbourhood of segmented region boundary and are trimmed by ones and values less than 0.2. In **(b)**, certainty is encoded into heatmap, where green indicates high segmentation certainty. Left visualization shows the certainty encoding for a setup with incorporated recurrent units and the one on the right without. Measured with viewpoint number set to 49.

Region Voting

The inclusion of the region voting procedure resulted in the largest improvement in the segmentation performance of all components. For example, in terms of the overlap metric, the increase was around 5% on both test sets. This is due to the fact that, in the successive backprojection of each tooth, it was simply sufficient to induce inaccuracy from one view that superimposed on the previous tooth. However, this can be easily corrected thanks to the computed multi-view certainty. The significant impact of this module on qualitative results can be seen in the randomly selected test cases in Figure 7.4. There, it is evident that it suppresses significant errors on surfaces close to areas of contacts of neighboring teeth, also resulting in notable performance improvement in terms of both overlap and boundary metrics.

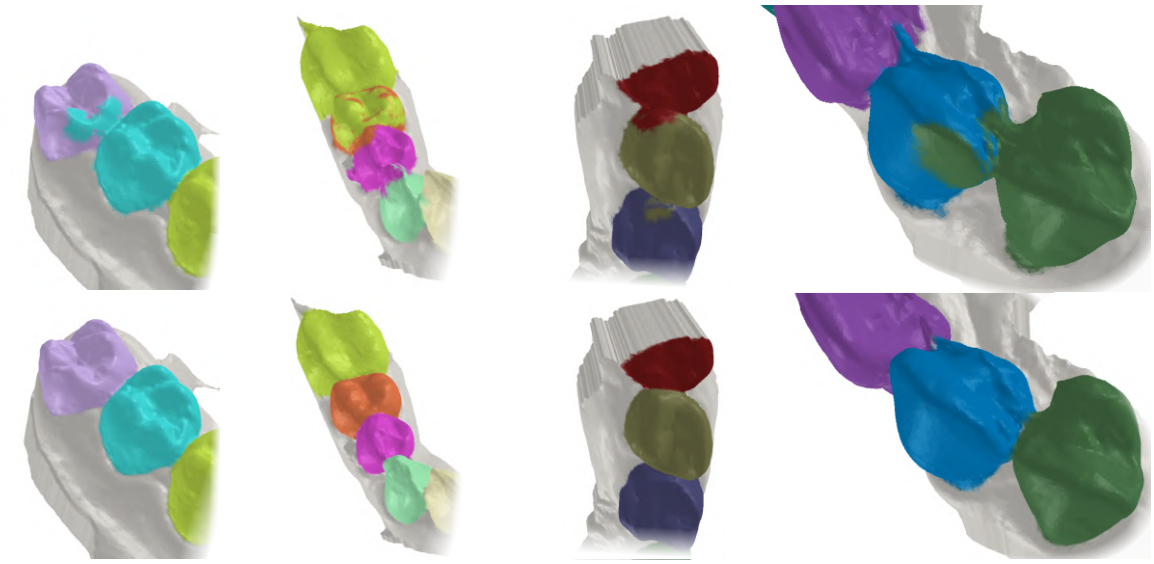


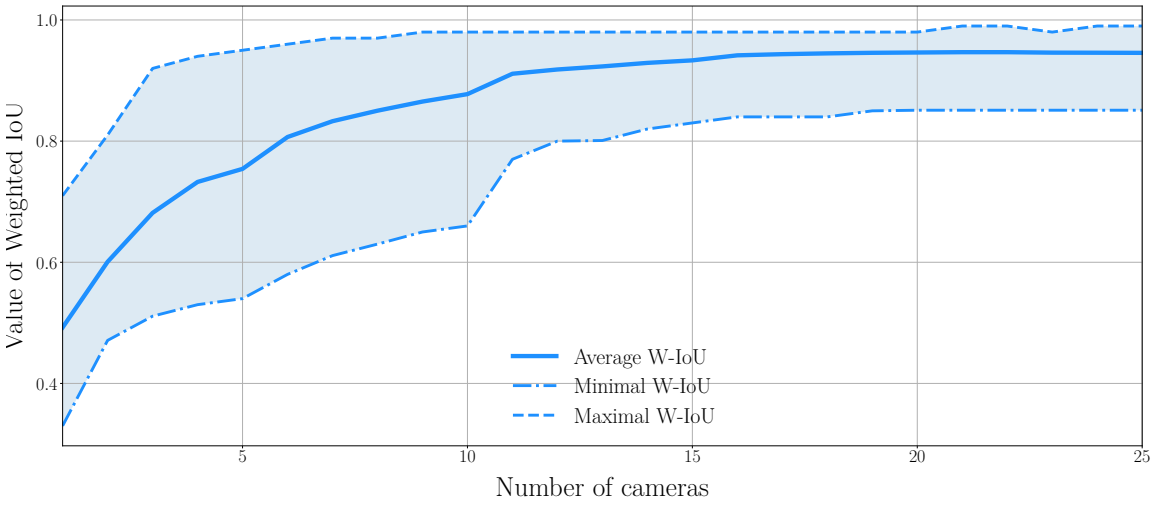
Figure 7.4: Impact of region voting in post-processing. Top row: segmentation masks before region voting. Note the artifacts of superimposed neighboring class masks. Bottom row: masks after region voting with suppressed neighbor class artifacts.

Connected Component Analysis

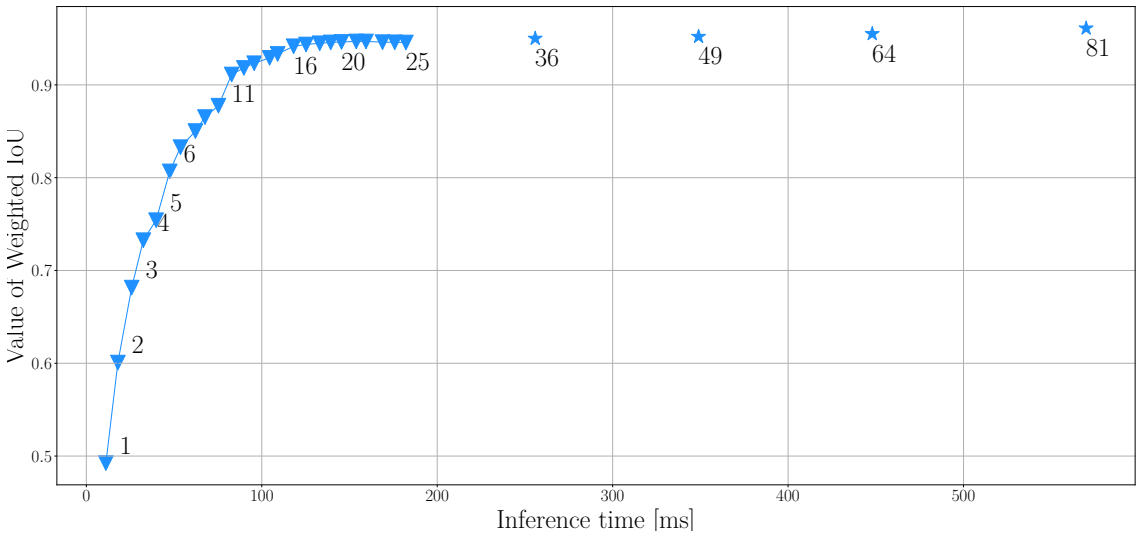
Connected component analysis is the last component of the pipeline. First, it fulfills the assumption that every region is composed of a single connected component. Second, often miniature isolated regions form the smaller components that need to be removed. Their manual correction by the clinician would be a challenging task due to the difficult localization of elements such as single false positive triangles. These isolated islands were present in approximately 93% of the test cases, with a connected compound count of 11 ± 8 across both test sets altogether. Isolated false positive islands form less than 1% of the whole detected mask (47 ± 69 triangles). The impact of CCA on performance is minor in terms of overlap metrics, as the areas of the isolated islands are typically very small but notable for the boundary metric, which is another indication of why it makes sense to combine metrics from these two families of metrics.

7.2.2 Multi-View Configuration

One of the biggest concerns in adopting a multi-view approach is how to properly distribute the camera positions and what the optimal number of these cameras is. In the experiments, the method is evaluated with different numbers of cameras that are distributed proportionally in a helix, as described in Section 5.3. The quantitative results in Table 7.2 provide compelling evidence that with increasing viewpoint number, performance increases, although it begins to converge at around 16 viewpoints. Therefore, the final viewpoint number should be set as a trade-off between performance and inference time. From the analysis of Figure 7.5, again, the steady convergence of performance is evident. Its increment starts to be negligible and unfavorable from the perspective of the segmentation quality to the computational time ratio. The final number of cameras is set to 49. The reason is that from the aspect of the multi-view certainty and its subsequent use in post-processing, it is advisable to set this number slightly higher, as this increases the *certainty depth*. With



(a) Impact of viewpoint number to the performance



(b) Trade-off curve between inference time and overlap metric

Figure 7.5: Trade-offs between viewpoint number and overall performance. Graph in **(a)** reveals a clear upward trend in performance for smaller number of viewpoints, with an incipient convergence from around 16 viewpoints. Graph in **(b)** illustrates the relationship between overlap metric and incipient single sequence – **single tooth**. Triangles connected by line correspond to values from upper graph, stars are additional values to emphasize the convergence. Integer values in graph correspond to viewpoint number. Inference time was measured as an average of 200 runs and GPU warm-up was excluded from measurements. Note that for clear visualization only overlap metric is shown, but a very similar trend can be observed when analyzing the boundary metric.

higher certainty depth, it is possible to express the certainty more precisely and bring more reliable post-processing, which is of great importance for high-quality segmentation results.

Table 7.2: Quantitative results for various multi-view settings. The angular increase during sampling is always set to cover the entire upper hemisphere with a spiral. Provided values are the averages calculated from all samples within given test set.

Test suite	Viewpoint number & metric							
	$N = 9$		$N = 16$		$N = 25$		$N = 49$	
	W-IoU ↑	d_{H95} ↓	*W-IoU* ↑	d_{H95} ↓	*W-IoU* ↑	d_{H95} ↓	*W-IoU* ↑	d_{H95} ↓
Basic cases	0.8415	3.782	0.9442	0.880	0.9547	0.509	**0.9781**	**0.361**
Complex cases	0.8303	3.911	0.9172	2.062	0.9402	0.655	**0.9553**	**0.403**

7.2.3 Multi-View Certainty Analysis

The multi-view certainty provides a meaningful asset to help with the final model selection. It is possible to examine whether this apparatus can be exploited to locate potential error regions in segmented masks. This is based on the hypothesis that parts of the masks with less certainty are more likely to introduce errors. If the hypothesis was true, these regions could be highlighted to the clinician. The software could then guide clinicians' attention towards areas of potential errors for further investigation and corrective actions.

This particular preliminary experiment was empirically evaluated on meshes from a complex test set. They were per-face colored by multi-view certainty encoded into green-to-red color map, indicating high to low segmentation certainty. The results of some of the cases are visualized in Figure 7.6. In the experiment, the focus was directed to parts of the model with complex geometric features, such as braces, wires, blurred transitions between teeth or crooked teeth. In fact, these are the most likely to show reduced certainty.

The analysis has shown that, in most cases, the decreased certainty coincides with complex geometric phenomena, as mentioned above. The exception is crooked teeth. However, these findings do not correlate as well with the lower quality of segmented results. Usually, although the certainty is lower in geometrically complex areas, the segmentation quality itself is overall very good. The outstanding case was found in the data sample that contains extremely blurred tooth-gingiva and tooth-tooth boundaries. There, the alarmingly low certainty correlates with the low segmentation quality. In conclusion, further investigation will be needed to get the best separation of false positive and true positive error region alarms in order to make the segmentation process more efficient.

In summary, the final number of cameras is set to 49. The reason is that from the aspect of the multi-view certainty and its subsequent use in post-processing, it is advisable to set this number slightly higher, as this increases the *certainty depth*. With higher certainty depth, it is possible to express the certainty more precisely and bring more reliable post-processing, which is of great importance for high-quality segmentation results.

7.3 Evaluation Axis 2: Comparative Analysis

This section presents the results of experiments, where the best-performing multi-view method was compared to other approaches. First, the method was compared with custom proposed non-Euclidean frameworks (point-based and edge-based solutions, see Section 5.4.1 and Section 5.4.2, respectively), which were trained and evaluated on the same dataset as the multi-view solution. Additionally, the method was compared with a conventional mesh segmentation algorithm that is based on Graph-Cut. Its implementation was

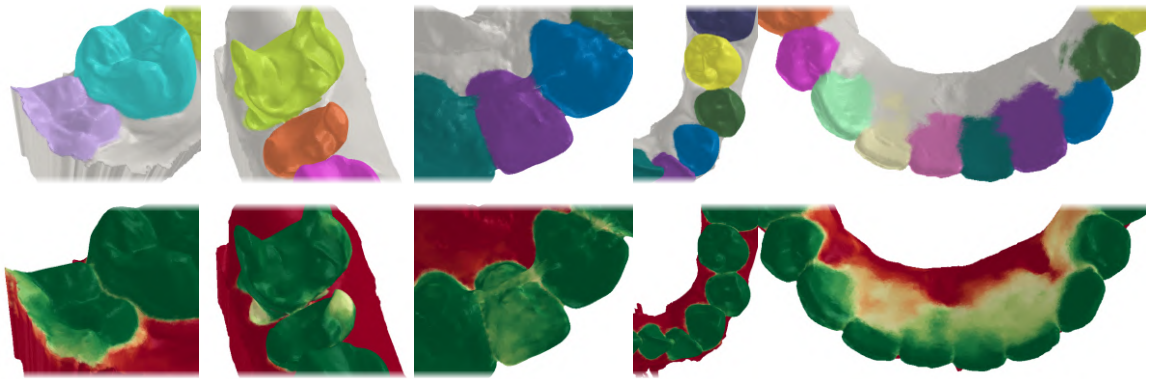


Figure 7.6: Certainty analysis in anatomically/geometrically complex models. Top row: detected segmentation masks. Bottom row: corresponding certainty maps. The certainty is of lower values at geometry representing incompletely scanned molar, fractured cusp, visible tooth root, or wires. On a shape with extremely smooth geometry, the certainty is very low, which correlates with low segmentation accuracy.

provided by TESCAN 3DIM, s.r.o., and I am not the author of the method. All these approaches will be referred to as *in-house approaches*. The objective of these experiments is to investigate the limitations of these methods in various aspects (segmentation performance and inference time). This comparison is the most fair that can be provided in this work, since none of the state-of-the-art teeth segmentation solutions provides an open dataset or trained models. For completeness, the remaining part of this section presents a rough comparison with the performance derived from state-of-the-art publications [10, 45, 67, 75], although all were trained and evaluated on different datasets, so the comparison may not be entirely fair.

7.3.1 A Comparative Study of In-House Approaches

Three main experiments are presented with in-house methods: general quantitative and qualitative comparison, inference times, and tessellation robustness. The best-performing multi-view solution is compared with the provided Graph-Cut solution, with four point-based versions, varying in employed architecture and input features, and with two versions of edge-based methods, which vary in the architectural design. Note again that all supervised solutions are trained on the same dataset and all solutions are evaluated on the same test sets.

Qualitative and Quantitative Comparison

Table 7.3 showcases the quantitative comparison between our method and other in-house approaches. These experimental findings reveal several important pivotal results:

- The multi-view solution surpasses the performance of all other approaches. Although the performance increase is not that significant, this also holds true for the Graph-Cut method, which is considered to be tuned as best as possible and is an industry solution in a commercial software. This shows that when thoughtfully designed, even simple 2D Euclidean approaches are capable of effective handling of intricate patterns, variations, and nuances present in real-world 3D geometric medical data.

- Each segmentation method shows a drop in performance on cases from complex test set. More importantly, the greatest generalizability with regard to data complexness is evident in edge-based methods. Compared to multi-view solution, these methods report a smaller performance drop in overlap metric and only a slightly larger drop in boundary metric. Apparently, this non-Euclidean approach is capable of meaningful feature extraction that enables it to uncover intricate patterns in the geometry.
- The inclusion of curvature information (normal vectors) in point-based methods brings a significant improvement in both suggested metrics for the PointNet version only. However, for the approach based on PointNet++, the improvement is negligible for basic cases, and it even decreases the performance on complex cases. As PointNet operates on each point individually, normal vectors provide additional geometric information about each point, enabling the model to capture finer details and better understand the local surface characteristics. In the PointNet++ version, it might cause redundancy within the hierarchical feature learning, which indicates that for this particular task and architecture, such geometric information is irrelevant.

In addition to quantitative metrics, other valuable insights into the performance of the methods were obtained by visually examining the segmentations. Qualitative analysis revealed several notable findings regarding **typical error patterns of individual approaches**. The Graph-Cut solution is based on spreading the segmentation mask from landmarks on cusps until some significant change in curvature is found. That should represent the tooth-gingiva boundary. However, in cases where such a boundary is blurred, the central pit of the occlusal surface is deep, or dental appliances are present, the method fails. Outputs for PointNet-based approach without normal vector information are characteristic by over-segmentation. As there is no curvature information, the method learns blobs around teeth that are not bounded by the tooth-gingiva transition. On the other hand, the remaining point-based methods produce masks with higher false negative triangle count (under-segmentation). During the qualitative evaluation of edge-based methods, it was observed that in some cases, the generated masks exhibit incompleteness, where the mask is not fully connected, and small unconnected blobs or areas within the mask are present. Finally, the Euclidean multi-view solution introduces errors in teeth areas that are occluded during rendering, such as in narrow interdental spaces. For illustration, see example cases in Figure 7.7 and Figure 7.11.

Segmentation Performance versus Inference Time

Another important aspect that should be evaluated for all methods is the ratio between inference time and segmentation quality. Trade-offs are visualized in Figure 7.8. The observed trends show that the fastest methods are those that process raw point clouds. For a case when PointNet architecture is employed and the feature vector contains normals, the trade-off seems even more favorable than for multi-view solutions. The slowest methods are those based on SparseMeshCNN, which is also reflected in training time. To give a rough indication, it takes on average 10 times longer to converge to this state, in which the performances are comparably good with other methodologies.

Tessellation Robustness

Finally, robustness experiments toward various geometry tessellations are introduced. Naturally occurring small changes in topology and geometry may be observed in the acquired

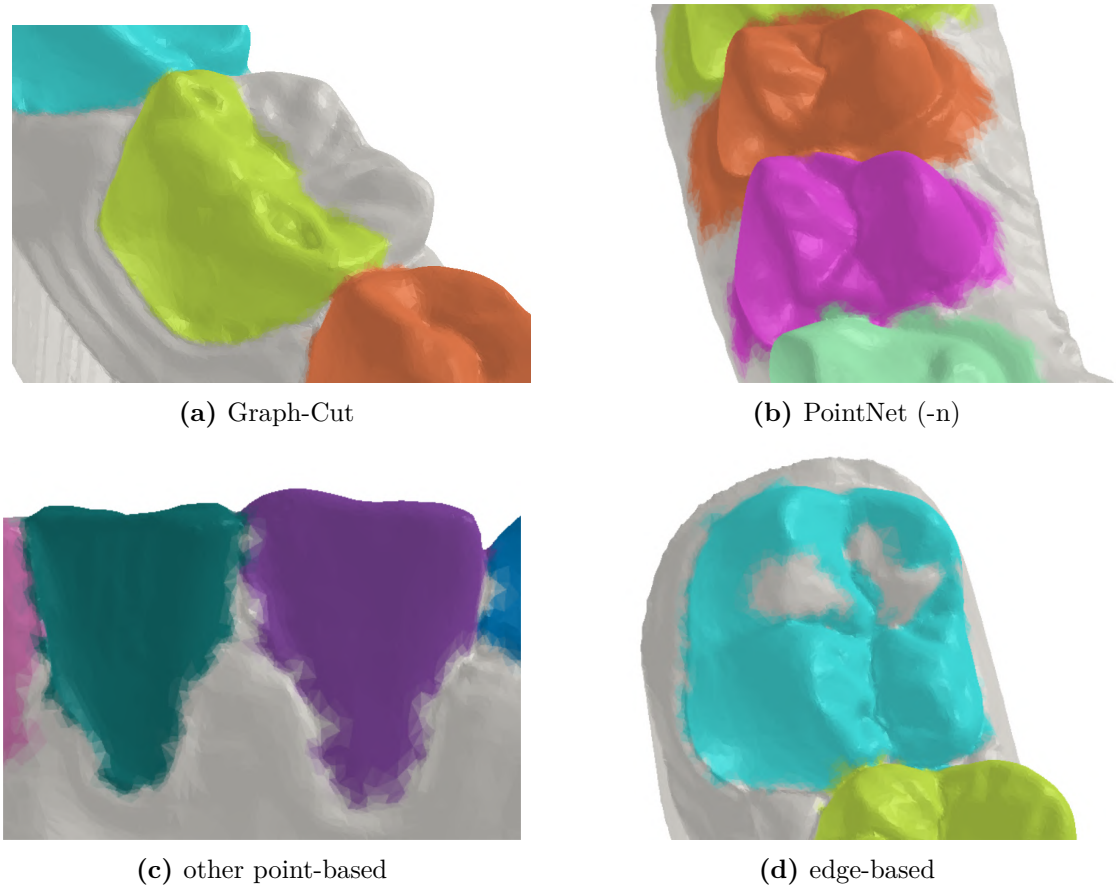
Table 7.3: Segmentation performance of in-house methods evaluated on both test sets. Notation in brackets: presence (+) or absence (-) of normal maps in the input point cloud (n) or edge self-attention blocks in two bottom stages of SparseMeshCNN (a).

Segmentation method	Basic cases		Complex cases		Perf. drop
	W-IoU ↑	d_{H95} ↓	*W-IoU* ↑	d_{H95} ↓	both ↓
Graph-Cut	0.9622	0.413	0.9318	0.625	3.0%; 0.21 mm
PointNet (-n)	0.8020	1.615	0.7654	2.121	3.6%; 0.50 mm
PointNet (+n)	0.9516	0.515	0.9279	1.198	2.3%; 0.68 mm
PointNet++ (-n)	0.9167	0.758	0.8883	1.302	2.8%; 0.54 mm
PointNet++ (+n)	0.9182	0.741	0.8696	1.517	4.8%; 0.77 mm
SparseMeshCNN (-a)	0.9284	0.646	0.9196	0.701	0.8%; 0.05 mm
SparseMeshCNN (+a)	0.9444	0.598	0.9401	0.653	**0.4%**; 0.05 mm
MV-RNN	**0.9781**	**0.361**	**0.9553**	**0.403**	2.2%; **0.04 mm**

data, as they can be influenced by the scanning properties, which may vary across different clinics. To verify the robustness to geometric and topological changes, the test sets were modified with subsequent methods:

- **Random vertex displacements.** Vertices were displaced by a value randomly sampled from range $\langle 0.0\,\text{mm}, 0.08\,\text{mm}\rangle$. It is expected that the methods will demonstrate good robustness towards these changes as a similar augmentation is applied during training.
- **Planar flipping optimization.** *Edge flip* is a common operation in the process of increasing the quality of the mesh. This operation is often performed to increase the local triangle quality. Again, a similar procedure is applied during training data augmentation.
- **Explicit remeshing.** This method, by repeatedly applying *edge flip*, *edge collapse*, *relax* and *refine*, improves triangle quality and topological regularity.
- **Laplacian smooth**. This operation averages each vertex position with weighted positions of neighbor vertices. The original version that does not preserve the surface of a triangular mesh was applied in three iterations [65].

These operations still preserve the semantic characteristic of the represented object. See Figure 7.9 for an illustration of the mesh modifications mentioned above. The results of the analysis for key method configurations are presented in Table 7.4. Overall, all methods are robust toward perturbations in vertex positions and changes in topology. The highest drop in performance is observed when a random vertex displacement is applied, as this method introduces the biggest spatial geometry changes. This drop is noticeable especially in results of Graph-Cut and multi-view approaches. In the former, this is caused by evident changes in curvature, for which the algorithm would have to be additionally fine-tuned. In the latter, the performance drop is caused by false positive and false negative predictions at mask boundaries, where the generated outputs are still smooth. Another interesting observation is that methods which improve the quality and regularity of the mesh slightly increase the performance of edge-based methods. Despite these, no other significant trend was observed.



(a) Graph-Cut **(b)** PointNet (-n)

(c) other point-based **(d)** edge-based

Figure 7.7: Typical error patterns occasionally present in results of alternative in-house approaches. Graph-Cut **(a)** method incorrectly assesses the mask boundary during its propagation based on geometric properties. **(b)** PointNet without normals generates oversegmented masks due to lack of curvature information. **(c)** other point-based approaches occasionally produce undersegmented masks with jagged boundaries. Finally, edge-based approach **(d)** intermittently generates masks with holes. For typical error pattern of multi-view approach, see Figure 7.11.

7.3.2 Comparison with State-of-the-Art in Teeth Segmentation Methods

Comparison of the results achieved with other studies is limited and therefore very rough. This is because the works do not present a public test dataset and evaluate the outputs on different metrics. Moreover, the authors do not always precisely specify the dataset employed, so it is often challenging to determine how the methods would perform in complex real-world cases. Nevertheless, overlap metric results have been extracted from papers discussing the same problem, as not every paper provides an evaluation on some boundary metric. These are summarized in Table 7.5. For the most proximate comparison, the table presents values of the best-performing method (MV-RNN) on two additional metrics: the unweighted IoU and the Dice score, as these are the ones frequently found in the papers. The results suggest that the proposed 2D solution is comparable to and even outperforms some of the previous works. However, because the nature of the data is not clear, no specific conclusion can be drawn from this experiment. Rather, it sheds light on the desirability of having a public benchmark dataset for the task of surface teeth segmentation.

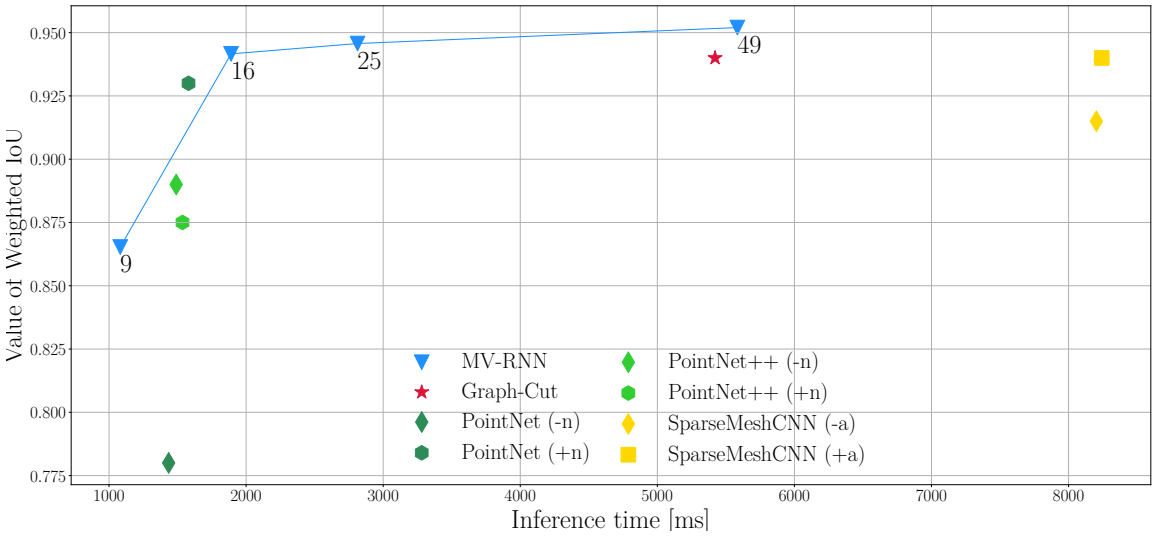


Figure 7.8: Average overlap metric results for combined test sets and inference times for segmentation in a dental scan representing complete dentition (16 teeth). Inference times of supervised methods were measured in Python as an average of 200 runs and GPU warm-ups were excluded. Graph-Cut computation was measured from C++ implementation within commercial dental planning software. Note that input generations from meshes are excluded from computations. Numbers at blue triangles represent the viewpoint number used in given multi-view setup. Average W-IoUs of both tests sets are plotted within the graph.

Table 7.4: Robustness of methods towards various geometry tessellations. Abbreviations of topology adjustment methods: RVD: Random Vertex Displacements, PFO: Planar Flipping Optimization, ER: Explicit Remeshing. Lap. Smooth stands for 3 iterations of Laplacian Smoothing. For simplicity, the reported results omit the boundary metric, but a very similar trend was also observed on this metric. Notation in brackets: $(+n)$ presence of normal maps in the input point cloud and edge self-attention blocks in two bottom stages of SparseMeshCNN $(+a)$.

Segmentation method	**Topology Adjustment & Performance**				
	Reference	RVD	PFO	ER	Lap. Smooth
	W-IoU ↑	*W-IoU* ↑	*W-IoU* ↑	*W-IoU* ↑	*W-IoU* ↑
Graph-Cut	0.947	0.911	0.940	0.943	0.952
PointNet $(+n)$	0.939	0.923	0.921	0.919	0.938
PointNet++ $(+n)$	0.893	0.849	0.899	0.892	0.885
SparseMeshCNN $(+a)$	0.942	0.931	0.949	0.944	0.958
MV-RNN	0.966	0.931	0.963	0.965	0.963

7.4 Summary of Results

This section summarizes the findings and contributions made. Collectively, the results presented on both evaluation axes appear consistent with the good performance of the proposed Euclidean surface teeth segmentation framework.

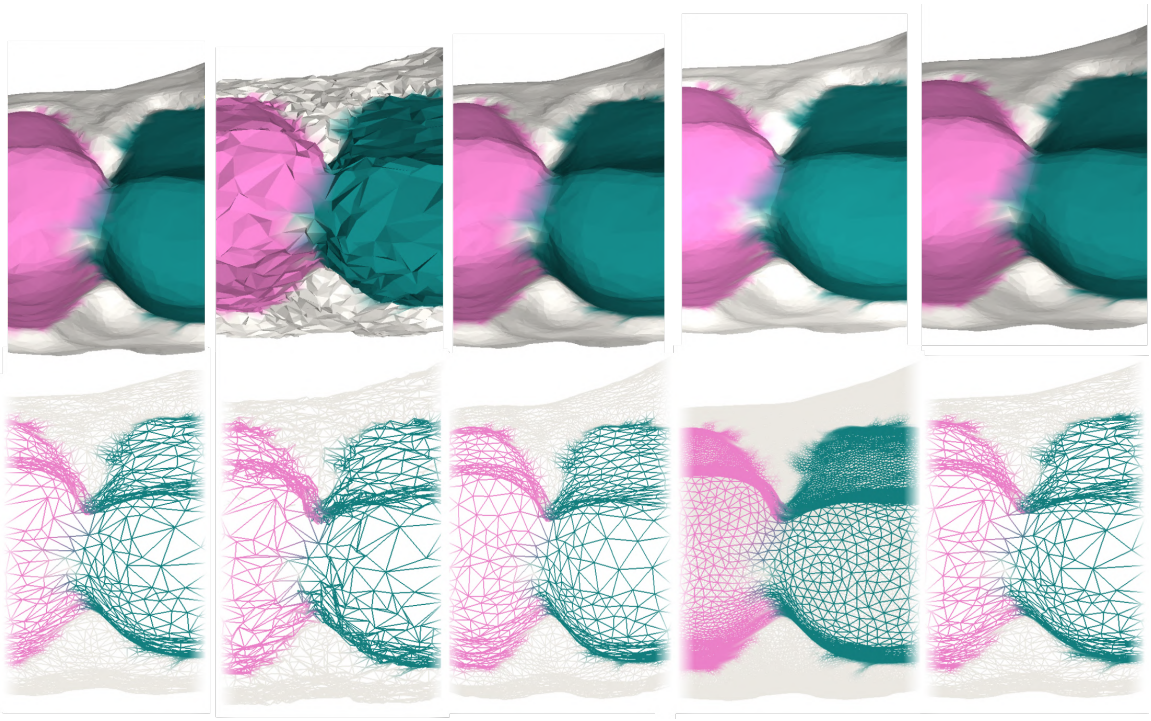


Figure 7.9: Example cases for tessellation robustness experiment on central incisors of randomly selected case. From left to right: reference mesh, random vertex displacements, planar flipping optimization, explicit remeshing, Laplacian smooth (two iterations).

Table 7.5: Performance comparison of proposed multi-view method and state-of-the-art approaches. Proposed method outperforms other methods when complex cases are excluded from the evaluation. When evaluated on combined test set, it still outperforms two out of four methods proposed in the literature.

Segmentation method	**Metric**	
	Unweighted *IoU* ↑	*Dice Score* ↑
Hao *et al.* [27]	0.9392	–
Cui *et al.* [10]	–	0.9690
Sun *et al.* [67]	–	0.9612
Wu *et al.* [75]	–	0.9530
MV-RNN (this work), basic set	0.9531	0.9880
MV-RNN (this work), combined	0.9358	0.9668

Overall results. The proposed recurrent multi-view approach reliably segments teeth with an average *W-IoU* score of 0.966 and Hausdorff distance at 95 percentile of 0.382 mm on combined test sets. Extensive results carried out together with a thoughtful division of the test set into two subsets of various complexity show that this method performs well not only on simple anatomical and geometric cases, but also on complex shapes with braces, wires, croocked teeth, or smooth transitions between teeth/gingiva. This generalization is of high importance for the practical application of the method, as orthodontic cases are

usually of this nature. Since the method is defined as a binary segmentation that operates locally on each tooth, it performs equally well on all teeth, so irregularities in the position of the teeth and the underrepresentation of the 3rd molars are not issues. There are also no significant differences in performance on maxillae and mandibles (for example the difference in the overlap metric is less than 1%). Refer to Figure 7.10 for additional qualitative results.

Ablation study. The ablation study conducted found evidence that each component of the method yields increasingly good results. The multi-view configuration was examined in the context of segmentation performance, inference time, and the importance for multi-view certainty analysis. The results also demonstrated that the multi-view certainty analysis can be considered when selecting the final model. If further examined, it appears that it might provide a promising way for highlighting potential errors in segmented regions, and so it could serve to draw the doctor's attention to such error areas.

Comparative analysis. The method was compared directly with a conventional method based on Graph-Cut and two proposed non-Euclidean supervised approaches. In general, the Euclidean method obtained the most robust results. The Graph-Cut solution fails if the shape is not perfectly concave, so it is error-prone to any unusual geometry, such as wires or deep buccal grooves. Custom trained non-Euclidean approaches are of higher complexity, but also provide interesting results. Point-based methods introduce better trade-off between performance and inference times, whereas edge-based methods generate masks with consistent boundaries. Overall, each methodology introduces a specific error pattern in the segmentation (Figure 7.7), which is true also for the multi-view method (Figure 7.11).

The proposed method was also indirectly compared with other state-of-the-art methods. The results indicate that the performance of this work is similar to or even better than the results presented in the works. It must be noted that they were trained and evaluated on different datasets.

Academic and industrial contribution. As part of an extensive evaluation, a large number of aspects of the Euclidean approach to geometric data segmentation were investigated and proven to be meaningful. Altogether with a comparison with conventional and non-Euclidean algorithms and the eventual release of the test sets, the results are valuable for researchers in designing and/or evaluating similar methods.

The preliminary results were consulted with experts in the development of commercial dental planning software. As these results on the limited test dataset indicate a higher accuracy than the currently used method, once tested on a larger dataset, the proposed method will be further fine-tuned and subsequently deployed in production.

7.5 Limitations and Future Work

Although the results demonstrate strong evidence for direct clinical application, they present some **limitations**, such as **when the analyzed dental scan contains narrow interdental spaces** or, in general, when there are **occluded geometry parts that are not visible** given multi-view configuration. In 29 out of 30 test cases, dental scans contain connected teeth without interdental details, which implies good performance of the 2D method. The limitation becomes clear in the one case with separated teeth, in which the interdental space is occluded from each camera viewpoint. In that particular case, primitives that form the mesial and distal tooth surfaces are incorrectly assigned to the gingiva class. See Figure 7.11 for visualization. **Another limitation** of this method stems from the **presence of unclear geometric transitions between adjacent teeth or between teeth and**

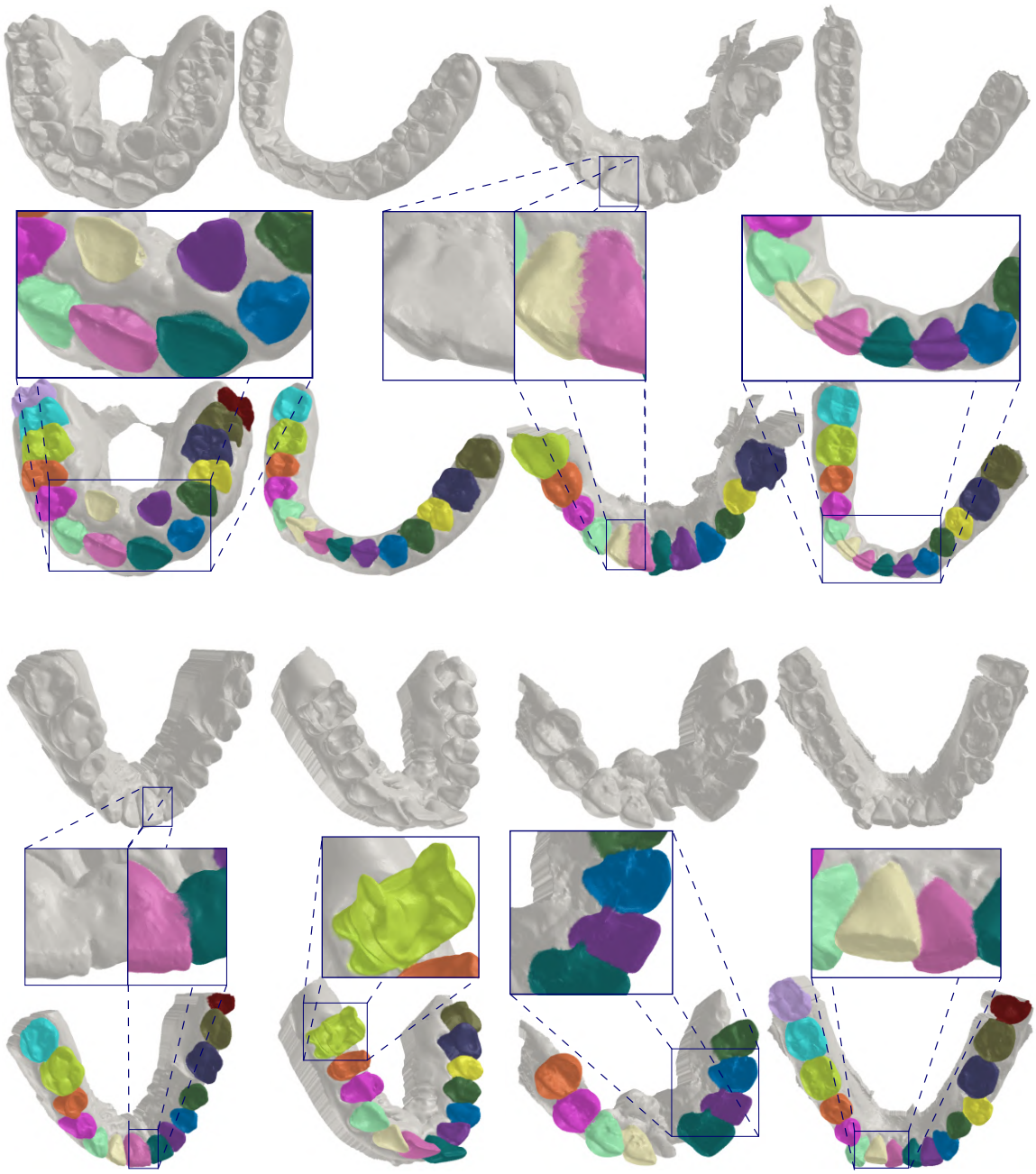


Figure 7.10: Qualitative results of multi-view approach on randomly selected test cases. The proposed method reliably segments teeth in geometrically and anatomically complex shapes.

the gingiva. In cases where the transition is smooth but still visually distinguishable, the method performs well with the necessary minor corrective actions. Such cases can be observed in Figure 7.10. However, there is one extreme case in the test set where it is difficult to determine the tooth regions correctly even for an expert in the field (empirically verified by letting the case be segmented by an experienced dental planning software tester). In this case, the method performs poorly (see the output in Figure 7.6), which indicates that the curvature information is essential in feature extraction. Finally, it should be noted that

the size of the test set is not sufficiently large to reliably detect all the patterns of defects that can be encountered in data within the medical domain and with this characteristic.

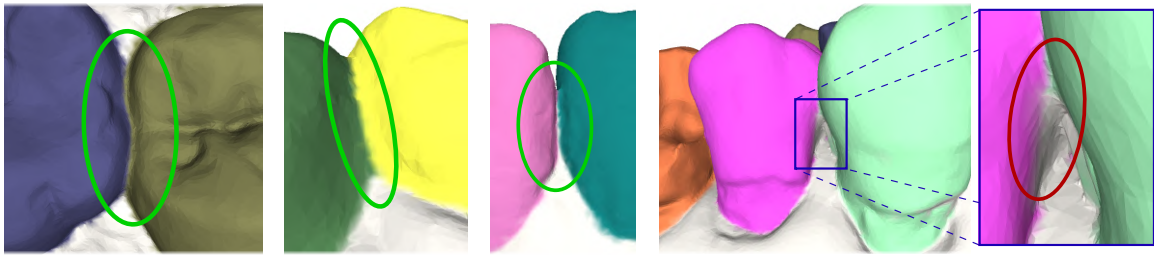


Figure 7.11: Detailed visualization of segmented region boundaries at interdental space vicinity. In vast majority of the cases, teeth are not completely separated (first three images) but rather fused together. When the interdental space is evident, the method does not correctly segment such space due to occlusions.

Interesting questions for future research that arise from the multi-view method are for example learnable estimation of the next-best camera position or the replacement of CNN blocks by vision transformers (ViTs) [70]. Further investigation could also focus on enhancing the method's robustness in handling instances characterized by narrow interdental spaces. One possible solution could be based on a combination of Euclidean and non-Euclidean approaches, which could lead to a more expressive encoding of the input shape by the combination of visual and geometric features in latent space [40]. In fact, this extension might be reasonable since it has been observed in the evaluation that methods from these categories produce different kinds of error. Lastly, it would be valuable for the research community to provide a public evaluation dataset for a fair comparison of methods.

Chapter 8

Conclusion

This work aimed to examine various approaches towards the analysis of 3D geometric medical data via deep learning. In the task of teeth mesh segmentation, three approaches were proposed and extensively evaluated.

The main Euclidean framework employs a multi-view approach. It bypasses the irregular nature of triangular meshes, and at the same time benefits from the well-studied field of 2D deep learning. The proposed method contains components like inter-viewpoint spatial correlation, deep supervision, and thoughtful post-processing steps. This approach was compared with other supervised methods based on PointNet [56], PointNet++ [57] and SparseMeshCNN [26] that are known as non-Euclidean since they directly tap into the geometric nature of the data.

The experimental part presents an ablation study of the main method. It provides strong evidence that the method is designed thoughtfully and that each of its components contributes to better performance. Evaluation of multi-view configuration and certainty follows. On a dataset of real-world orthodontic cases, the main method achieves an average weighted IoU score of 0.966 and Hausdorff distance at 95 percentile of 0.382 mm. The second axis of the experiments presents a comparative analysis. There, the main method was compared with the proposed point-based and edge-based methods, along with Graph-Cut algorithm. The main method yields superior results compared to any other in-house method. Lastly, in a rough comparison with current state-of-the-art methods, it achieves competitive results, but it is difficult to ascertain it since any of the work provides an open dataset.

Though the test set utilized in this work covers a wide range of possible anatomical or geometric anomalies (teeth dislocations, malocclusions, scanned appliances, or blurred boundaries between teeth), it is vital to further experiment with larger test set due to tremendous variance in data from the geometric and medical domains. Yet, the results that were achieved within the scope of this work are promising for the deployment of the method in dental planning software. This integration is approved by competent people. Furthermore, it is planned to publish a shortened version of this research and release the test set, allowing fair comparisons to be observed in future academic publications.

Bibliography

[1] Antonelli, M., Reinke, A., Bakas, S., Farahani, K., Kopp Schneider, A. et al. The medical segmentation decathlon. *Nature communications*. Nature Publishing Group UK London. 2022, vol. 13, no. 1, p. 4128. DOI: 10.1038/s41467-022-30695-9. Available at: https://doi.org/10.1038/s41467-022-30695-9.

[2] Arun, K. S., Huang, T. S. and Blostein, S. D. Least-Squares Fitting of Two 3-D Point Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1987, PAMI-9, no. 5, p. 698–700. DOI: 10.1109/TPAMI.1987.4767965. Available at: https://doi.org/10.1109/TPAMI.1987.4767965.

[3] Berg, M. de, Cheong, O., Kreveld, M. van and Overmars, M. *Computational Geometry: Algorithms and Applications*. Springer, 2008. ISBN 978-3-540-77973-5.

[4] Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A. and Vandergheynst, P. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*. 2017, vol. 34, no. 4, p. 18–42. DOI: 10.1109/MSP.2017.2693418. Available at: https://doi.org/10.1109/MSP.2017.2693418.

[5] Bruna, J., Zaremba, W., Szlam, A. and LeCun, Y. Spectral Networks and Locally Connected Networks on Graphs. In: Bengio, Y. and LeCun, Y., ed. *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. 2014. Available at: http://arxiv.org/abs/1312.6203.

[6] Catmull, E. and Clark, J. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*. 1978, vol. 10, no. 6, p. 350–355. DOI: 10.1016/0010-4485(78)90110-0. ISSN 0010-4485. Available at: https://doi.org/10.1016/0010-4485(78)90110-0.

[7] Chen, J., Yang, L., Zhang, Y., Alber, M. S. and Chen, D. Z. Combining Fully Convolutional and Recurrent Neural Networks for 3D Biomedical Image Segmentation. In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 2016, p. 3036–3044. ISBN 9781510838819. Available at: https://dl.acm.org/doi/10.5555/3157382.3157438.

[8] Cohen, T. and Welling, M. Group Equivariant Convolutional Networks. In: Balcan, M. and Weinberger, K. Q., ed. *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. JMLR.org, 2016, vol. 48, p. 2990–2999. JMLR Workshop and

Conference Proceedings. Available at: http://proceedings.mlr.press/v48/cohenc16.html.

[9] Crane, K., Goes, F. de, Desbrun, M. and Schröder, P. Digital Geometry Processing with Discrete Exterior Calculus. In: *ACM SIGGRAPH 2013 courses*. New York, NY, USA: ACM, 2013. SIGGRAPH '13. DOI: 10.1145/2504435.2504442. Available at: https://doi.org/10.1145/2504435.2504442.

[10] Cui, Z., Li, C., Chen, N., Wei, G., Chen, R. et al. TSegNet: An efficient and accurate tooth segmentation network on 3D dental model. *Medical Image Anal.* 2021, vol. 69, p. 101949. DOI: 10.1016/j.media.2020.101949. Available at: https://doi.org/10.1016/j.media.2020.101949.

[11] Defferrard, M., Bresson, X. and Vandergheynst, P. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Curran Associates Inc., 2016, p. 3837–3845. ISBN 9781510838819. Available at: https://dl.acm.org/doi/10.5555/3157382.3157527.

[12] Dillen, F. J. and Verstraelen, L. C. *Handbook of Differential Geometry, Volume 1*. Elsevier, 2005. ISBN 9780080461205.

[13] Dong, Q., Wang, Z., Li, M., Gao, J., Chen, S. et al. Laplacian2Mesh: Laplacian-Based Mesh Understanding. *IEEE Transactions on Visualization and Computer Graphics*. Institute of Electrical and Electronics Engineers (IEEE). 2023, p. 1–13. DOI: 10.1109/tvcg.2023.3259044. Available at: https://doi.org/10.1109%2Ftvcg.2023.3259044.

[14] Dou, Q., Chen, H., Jin, Y., Yu, L., Qin, J. et al. 3D deeply supervised network for automatic liver segmentation from CT volumes. In: Springer. *International conference on medical image computing and computer-assisted intervention*. 2016, p. 149–157. DOI: 10.1007/978-3-319-46723-8_18. ISBN 978-3-319-46722-1. Available at: https://doi.org/10.1007/978-3-319-46723-8_18.

[15] Elseberg, J., Magnenat, S., Siegwart, R. and Nuchter, A. Comparison on nearest-neigbour-search strategies and implementations for efficient shape registration. *Journal of Software Engineering for Robotics (JOSER)*. 2012, vol. 3, p. 2–12. DOI: 10.6092/JOSER_2012_03_01_P2. Available at: https://doi.org/10.6092/JOSER_2012_03_01_P2.

[16] Feng, Y., Feng, Y., You, H., Zhao, X. and Gao, Y. MeshNet: Mesh Neural Network for 3D Shape Representation. In: *The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, p. 8279–8286. DOI: 10.1609/aaai.v33i01.33018279. Available at: https://doi.org/10.1609/aaai.v33i01.33018279.

[17] Georgousis, S., Kenning, M. P. and Xie, X. Graph Deep Learning: State of the Art and Challenges. *IEEE Access*. 2021, vol. 9, p. 22106–22140. DOI:

10.1109/ACCESS.2021.3055280. Available at: https://doi.org/10.1109/ACCESS.2021.3055280.

[18] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O. and Dahl, G. E. Neural Message Passing for Quantum Chemistry. In: Precup, D. and Teh, Y. W., ed. *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. PMLR, 2017, vol. 70, p. 1263–1272. Proceedings of Machine Learning Research. Available at: https://dl.acm.org/doi/10.5555/3305381.3305512.

[19] Goodfellow, I. J., Bengio, Y. and Courville, A. C. *Deep Learning*. MIT Press, 2016. Adaptive computation and machine learning. ISBN 978-0-262-03561-3.

[20] Gross, M. and Pfister, H. *Point-based Graphics*. Morgan Kaufmann, 2007. Morgan Kaufmann series in computer graphics and geometric modeling. ISBN 978-0-12-370604-1.

[21] Haim, N., Segol, N., Ben-Hamu, H., Maron, H. and Lipman, Y. Surface Networks via General Covers. In: *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, p. 632–641. DOI: 10.1109/ICCV.2019.00072. Available at: https://doi.org/10.1109/ICCV.2019.00072.

[22] Hamilton, W. L., Ying, Z. and Leskovec, J. Inductive Representation Learning on Large Graphs. 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. 1024–1034. Available at: https://dl.acm.org/doi/10.5555/3294771.3294869.

[23] Hammond, D. K., Vandergheynst, P. and Gribonval, R. Wavelets on Graphs via Spectral Graph Theory. *CoRR*. 2009, abs/0912.3848. Available at: http://arxiv.org/abs/0912.3848.

[24] 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. DOI: 10.1145/3306346.3322959. Available at: https://doi.org/10.1145/3306346.3322959.

[25] Hanocka, R., Metzer, G., Giryes, R. and Cohen Or, D. Point2Mesh. *ACM Transactions on Graphics*. Association for Computing Machinery (ACM). 2020, vol. 39, no. 4. DOI: 10.1145/3386569.3392415. Available at: https://doi.org/10.1145%2F3386569.3392415.

[26] Hansen, B., Lowes, M., Ørkild, T., Dahl, A., Dahl, V. et al. SparseMeshCNN with Self-Attention for Segmentation of Large Meshes. *Proceedings of the Northern Lights Deep Learning Workshop*. april 2022, vol. 3. DOI: 10.7557/18.6281. Available at: https://doi.org/10.7557/18.6281.

[27] Hao, J., Liao, W., Zhang, Y., Peng, J., Zhao, Z. et al. Toward Clinically Applicable 3-Dimensional Tooth Segmentation via Deep Learning. *Journal of Dental Research*. 2022, vol. 101, no. 3, p. 304–311. DOI: 10.1177/00220345211040459. PMID: 34719980. Available at: https://doi.org/10.1177/00220345211040459.

[28] Hertz, A., Hanocka, R., Giryes, R. and Cohen Or, D. Deep geometric texture synthesis. *ACM Transactions on Graphics*. Association for Computing Machinery (ACM). aug 2020, vol. 39, no. 4. DOI: 10.1145/3386569.3392471. Available at: https://doi.org/10.1145%2F3386569.3392471.

[29] Hochreiter, S. and Schmidhuber, J. Long Short-Term Memory. *Neural Computation*. 1997, vol. 9, no. 8, p. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. Available at: https://doi.org/10.1162/neco.1997.9.8.1735.

[30] Hoppe, H. View-Dependent Refinement of Progressive Meshes. In: *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. USA: ACM Press/Addison-Wesley Publishing Co., 1997, p. 189–198. SIGGRAPH '97. DOI: 10.1145/258734.258843. ISBN 0897918967. Available at: https://doi.org/10.1145/258734.258843.

[31] Hu, S.-M., Liu, Z.-N., Guo, M.-H., Cai, J.-X., Huang, J. et al. Subdivision-based Mesh Convolution Networks. *ACM Transactions on Graphics*. Association for Computing Machinery (ACM). mar 2022, vol. 41, no. 3, p. 1–16. DOI: 10.1145/3506694. Available at: https://doi.org/10.1145%2F3506694.

[32] International, D. T. *SHINING 3D — AutoScan-DS-EX Pro* [online]. 2023 [cit. 2022-12-21]. Available at: https://www.dental-tribune.com/c/shining-3d/prod/shining-3d-autoscan-ds-ex-pro/.

[33] 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. DOI: 10.1609/aaai.v33i01.33018513. ISBN 978-1-57735-809-1. Available at: https://doi.org/10.1609/aaai.v33i01.33018513.

[34] Kipf, T. N. and Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *CoRR*. 2016, abs/1609.02907. Available at: http://arxiv.org/abs/1609.02907.

[35] Kronfeld, T., Brunner, D. and Brunnett, G. Snake-based segmentation of teeth from virtual dental casts. *Computer-Aided Design and Applications*. Taylor & Francis. 2010, vol. 7, no. 2, p. 221–233. DOI: 10.3722/cadaps.2010.221-233. Available at: https://doi.org/10.3722/cadaps.2010.221-233.

[36] Kubík, T. *Deep Neural Networks for Landmark Detection on 3D Models*. Brno, CZ, 2021. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Španěl, M. Available at: https://www.fit.vut.cz/study/thesis/23507/.

[37] Kubík, T. and Španěl, M. Robust Teeth Detection in 3D Dental Scans by Automated Multi-view Landmarking. In: INSTICC. *Proceedings of the 15th International Joint Conference on Biomedical Engineering Systems and Technologies - Volume 1: BIOIMAGING,*. SciTePress, 2022, p. 24–34. DOI: 10.5220/0010770700003123. ISBN 978-989-758-552-4. Available at: https://doi.org/10.5220/0010770700003123.

[38] Kumar, Y., Janardan, R., Larson, B. and Moon, J. Improved segmentation of teeth in dental models. *Computer-Aided Design and Applications*. Taylor & Francis. 2011, vol. 8, no. 2, p. 211–224. DOI: 10.3722/cadaps.2011.211-224. Available at: https://doi.org/10.3722/cadaps.2011.211-224.

[39] Lahav, A. and Tal, A. MeshWalker: Deep Mesh Understanding by Random Walks. *ACM Trans. Graph.* 2020, vol. 39, no. 6, p. 263:1–263:13. DOI: 10.1145/3414685.3417806. Available at: https://doi.org/10.1145/3414685.3417806.

[40] Lamb, N., Wiederhold, N., Lamb, B., Banerjee, S. and Banerjee, N. K. Using Learned Visual and Geometric Features to Retrieve Complete 3D Proxies for Broken Objects. In: *Proceedings of the 6th Annual ACM Symposium on Computational Fabrication*. New York, NY, USA: Association for Computing Machinery, 2021. SCF '21. DOI: 10.1145/3485114.3485118. ISBN 9781450390903. Available at: https://doi.org/10.1145/3485114.3485118.

[41] 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. DOI: 10.1016/j.cag.2017.05.011. ISSN 0097-8493. Shape Modeling International 2017. Available at: https://doi.org/10.1016/j.cag.2017.05.011.

[42] LeCun, Y., Bengio, Y. and Hinton, G. E. Deep learning. *Nature*. 2015, vol. 521, no. 7553, p. 436–444. DOI: 10.1038/nature14539. Available at: https://doi.org/10.1038/nature14539.

[43] Li, Y., Bu, R., Sun, M., Wu, W., Di, X. et al. PointCNN: Convolution On X-Transformed Points. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa Bianchi, N. et al., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018, vol. 31, p. 828–838. Available at: https://proceedings.neurips.cc/paper/2018/file/f5f8590cd58a54e94377e6ae2eded4d9-Paper.pdf.

[44] Li, Z., Ning, X. and Wang, Z. A fast segmentation method for STL teeth model. In: IEEE. *2007 IEEE/ICME International Conference on Complex Medical Engineering*. 2007, p. 163–166. DOI: 10.1109/ICCME.2007.4381713. ISBN 978-1-4244-1078-1. Available at: https://doi.org/10.1109/ICCME.2007.4381713.

[45] Lian, C., Wang, L., Wu, T.-H., Wang, F., Yap, P.-T. et al. Deep multi-scale mesh feature learning for automated labeling of raw dental surfaces from 3D intraoral scanners. *IEEE transactions on medical imaging*. IEEE. 2020, vol. 39, no. 7, p. 2440–2450. DOI: 10.1109/TMI.2020.2971730. Available at: https://doi.org/10.1109/TMI.2020.2971730.

[46] Loop, C. *Smooth Subdivision Surfaces Based on Triangles*. Dissertation. Available at: https://www.microsoft.com/en-us/research/publication/smooth-subdivision-surfaces-based-on-triangles/.

[47] Lorensen, W. E. and Cline, H. E. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In: *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: Association for Computing Machinery, 1987, p. 163–169. SIGGRAPH '87. DOI:

10.1145/37401.37422. ISBN 0897912276. Available at: https://doi.org/10.1145/37401.37422.

[48] Luebke, D., Reddy, M., Cohen, J. D., Varshney, A., Watson, B. et al. *Level of detail for 3D graphics*. Burlington, MA: Elsevier, 2002. The Morgan Kaufmann Series in Computer Graphics. ISBN 978-1-55860-838-2. Available at: https://cds.cern.ch/record/1990522.

[49] Maron, H., Galun, M., Aigerman, N., Trope, M., Dym, N. et al. Convolutional Neural Networks on Surfaces via Seamless Toric Covers. *ACM Trans. Graph.* New York, NY, USA: Association for Computing Machinery. jul 2017, vol. 36, no. 4. DOI: 10.1145/3072959.3073616. ISSN 0730-0301. Available at: https://doi.org/10.1145/3072959.3073616.

[50] Masci, J., Boscaini, D., Bronstein, M. M. and Vandergheynst, P. Geodesic Convolutional Neural Networks on Riemannian Manifolds. In: *2015 IEEE International Conference on Computer Vision Workshop, ICCV Workshops 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 2015, p. 832–840. DOI: 10.1109/ICCVW.2015.112. Available at: https://doi.org/10.1109/ICCVW.2015.112.

[51] Morrison, D., Corke, P. and Leitner, J. Multi-view picking: Next-best-view reaching for improved grasping in clutter. In: IEEE. *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*. 2019, p. 8762–8768. DOI: 10.1109/ICRA.2019.8793805. Available at: https://doi.org/10.1109/ICRA.2019.8793805.

[52] Mustra, M., Delac, K. and Grgic, M. Overview of the DICOM standard. In: IEEE. *2008 50th International Symposium ELMAR*. 2008, vol. 1, p. 39–44.

[53] Oleynikova, H., Millane, A., Taylor, Z., Galceran, E., Nieto, J. et al. Signed distance fields: A natural representation for both mapping and planning. In: University of Michigan. *RSS 2016 Workshop: Geometry and Beyond-Representations, Physics, and Scene Understanding for Robotics*. 2016. DOI: 20.500.11850/128029. Available at: https://doi.org/20.500.11850/128029.

[54] Orthodontics, R. F. *ITero 3D Digital Scanner* [online]. 2023 [cit. 2022-12-21]. Available at: https://www.riverfallsortho.com/services/itero-digital-impressions/.

[55] Poulenard, A. and Ovsjanikov, M. Multi-directional geodesic neural networks via equivariant convolution. *ACM Trans. Graph.* 2018, vol. 37, no. 6, p. 236. DOI: 10.1145/3272127.3275102. Available at: https://doi.org/10.1145/3272127.3275102.

[56] 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. DOI: 10.1109/CVPR.2017.16. Available at: https://doi.org/10.1109/CVPR.2017.16.

[57] 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.

[58] Ronneberger, O., Fischer, P. and Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab, N., Hornegger, J., III, W. M. W. and Frangi, A. F., ed. *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*. Springer, 2015, vol. 9351, p. 234–241. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-24574-4_28. Available at: https://doi.org/10.1007/978-3-319-24574-4_28.

[59] Rustamov, R. M. Laplace-Beltrami Eigenfunctions for Deformation Invariant Shape Representation. In: Belyaev, A. and Garland, M., ed. *Proceedings of the Fifth Eurographics Symposium on Geometry Processing, Barcelona, Spain, July 4-6, 2007*. The Eurographics Association, 2007, vol. 257, p. 225–233. ACM International Conference Proceeding Series. DOI: 10.2312/SGP/SGP07/225-233. ISBN 978-3-905673-46-3. Available at: https://doi.org/10.2312/SGP/SGP07/225-233.

[60] Sharp, N., Attaiki, S., Crane, K. and Ovsjanikov, M. DiffusionNet: Discretization Agnostic Learning on Surfaces. *ACM Trans. Graph.* 2022, vol. 41, no. 3, p. 27:1–27:16. DOI: 10.1145/3507905. Available at: https://doi.org/10.1145/3507905.

[61] Sheffer, A., Praun, E., Rose, K. et al. Mesh parameterization methods and their applications. *Foundations and Trends® in Computer Graphics and Vision*. Now Publishers, Inc. 2007, vol. 2, no. 2, p. 105–171. DOI: 10.1561/0600000011. Available at: https://doi.org/10.1561/0600000011.

[62] Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K. et al. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. 2015, vol. 28, p. 802–810. NIPS'15. Available at: http://arxiv.org/abs/1506.04214.

[63] Simon J. Littlewood, L. M. *An Introduction to Orthodontics*. 5th ed. Oxford University Press, 2019. ISBN 978-0-198-84726-7.

[64] Smirnov, D. and Solomon, J. HodgeNet: Learning Spectral Geometry on Triangle Meshes. *ACM Transactions on Graphics (TOG)*. ACM. August 2021, vol. 40, no. 4, p. 166:1–166:11. DOI: 10.1145/3450626.3459797. Available at: https://doi.org/10.1145/3450626.3459797.

[65] Sorkine, O. Laplacian Mesh Processing. In: Chrysanthou, Y. and Magnor, M., ed. *26th Annual Conference of the European Association for Computer Graphics, Eurographics 2005 - State of the Art Reports, Dublin, Ireland, August 29 - September 2, 2005*. The Eurographics Association, 2005, p. 53–70. DOI: 10.2312/egst.20051044. Available at: https://doi.org/10.2312/egst.20051044.

[66] 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. DOI: 10.1109/ICCV.2015.114. Available at: https://doi.org/10.1109/ICCV.2015.114.

[67] Sun, D., Pei, Y., Li, P., Song, G., Guo, Y. et al. Automatic tooth segmentation and dense correspondence of 3D dental model. In: Springer. *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2020, p. 703–712. DOI: 10.1007/978-3-030-59719-1_68. ISBN 978-3-030-59718-4. Available at: https://doi.org/10.1007/978-3-030-59719-1_68.

[68] Sun, J., Ovsjanikov, M. and Guibas, L. A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion. In: *Proceedings of the Symposium on Geometry Processing*. Goslar, DEU: Eurographics Association, 2009, p. 1383–1392. SGP '09. DOI: 10.1111/j.1467-8659.2009.01515.x. Available at: https://doi.org/10.1111/j.1467-8659.2009.01515.x.

[69] Tatarchenko, M., Park, J., Koltun, V. and Zhou, Q.-Y. Tangent Convolutions for Dense Prediction in 3D. arXiv. 2018. DOI: 10.48550/arXiv.1807.02443. Available at: https://arxiv.org/abs/1807.02443.

[70] Thisanke, H., Deshan, C., Chamith, K., Seneviratne, S., Vidanaarachchi, R. et al. Semantic Segmentation using Vision Transformers: A survey. In:. 2023. DOI: 10.48550/arXiv.2305.03273. Available at: https://doi.org/10.48550/arXiv.2305.03273.

[71] Wang, L., Lee, C.-Y., Tu, Z. and Lazebnik, S. Training Deeper Convolutional Networks with Deep Supervision. *CoRR*. may 2015. DOI: 10.48550/arXiv.1505.02496. Available at: https://doi.org/10.48550/arXiv.1505.02496.

[72] 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. DOI: 10.1145/3072959.3073608. Available at: https://doi.org/10.1145%2F3072959.3073608.

[73] Wang, P.-S., Sun, C.-Y., Liu, Y. and Tong, X. Adaptive O-CNN. *ACM Transactions on Graphics*. Association for Computing Machinery (ACM). dec 2018, vol. 37, no. 6, p. 1–11. DOI: 10.1145/3272127.3275050. Available at: https://doi.org/10.1145%2F3272127.3275050.

[74] Wu, K., Chen, L., Li, J. and Zhou, Y. Tooth segmentation on dental meshes using morphologic skeleton. *Computers & Graphics*. 2014, vol. 38, p. 199–211. DOI: 10.1016/j.cag.2013.10.028. ISSN 0097-8493. Available at: https://doi.org/10.1016/j.cag.2013.10.028.

[75] Wu, T.-H., Lian, C., Lee, S., Pastewait, M., Piers, C. et al. Two-Stage Mesh Deep Learning for Automated Tooth Segmentation and Landmark Localization on 3D Intraoral Scans. *IEEE Transactions on Medical Imaging*. IEEE. 2022, vol. 41, p. 3158–3166. DOI: 10.1109/tmi.2022.3180343. Available at: https://doi.org/10.1109%2Ftmi.2022.3180343.

[76] Wu, W., Qi, Z. and Fuxin, L. PointConv: Deep Convolutional Networks on 3D Point Clouds. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, p. 9613–9622. DOI: 10.1109/CVPR.2019.00985. Available at: https://doi.org/10.1109/CVPR.2019.00985.

[77] 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. DOI: 10.1109/CVPR.2015.7298801. Available at: https://doi.org/10.1109/CVPR.2015.7298801.

[78] Xu, H., Dong, M. and Zhong, Z. Directionally Convolutional Networks for 3D Shape Segmentation. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, 2017, p. 2717–2726. DOI: 10.1109/ICCV.2017.294. Available at: https://doi.org/10.1109/ICCV.2017.294.

[79] Yaqi, M. and Zhongke, L. Computer aided orthodontics treatment by virtual segmentation and adjustment. In: IEEE. *2010 International Conference on Image Analysis and Signal Processing*. 2010, p. 336–339. DOI: 10.1109/IASP.2010.5476100. Available at: https://doi.org/10.1109/IASP.2010.5476100.

[80] Zou, B.-j., Liu, S.-j., Liao, S.-h., Ding, X. and Liang, Y. Interactive tooth partition of dental mesh base on tooth-target harmonic field. *Computers in biology and medicine*. Elsevier. 2015, vol. 56, p. 132–144. DOI: 10.1016/j.compbiomed.2014.10.013. Available at: https://doi.org/10.1016/j.compbiomed.2014.10.013.

Appendix A

Contents of the Included Storage Media

- `data/test/` Folder with polygonal models for testing.[1]
- `data/train/` Folder with data for network training.[2]
- `trained-weights/` Folder with trained network weights.
- `src/` Folder with source files.
- `src-tech-report/` Folder with LaTeX source files.
- `LICENCE` Project licence.
- `poster.pdf` Poster summarizing this work.
- `README.md` Project description.
- `requirements.txt` List of required Python libraries.
- `tech-report.pdf` Technical report (this text).

[1] Note that the provided data are for demonstration purposes only and they are just a small portion of the whole dataset. The complete dataset of 3D scans provided by TESCAN 3DIM, s.r.o. as well as the generated input data in this work are not available for privacy reasons.

[2] See footnote 1.

Appendix B

Poster