ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Image Processing and Computer Vision

SKULL RECONSTRUCTION THROUGH SHAPE COMPLETION

CANDIDATE

Simone Gayed Said

SUPERVISOR

Prof. Luigi Di Stefano

CO-SUPERVISORS

Prof. Giuseppe Lisanti

Prof. Samuele Salti

Dott. Riccardo Spezialetti

Academic year 2020-2021

Session 2nd

Contents

List of Figures

List of Tables

Abstract

In this study, we present a shape completion approach to skull reconstruction. Our final goal is to reconstruct the complete mesh of a skull starting from its defective point cloud. Our approach is based on an existing deep neural network, opportunely modified, trained to reconstruct a complete 3D point cloud from an incomplete one. The complete point clouds are then processed through a multi-step pipeline in order to reconstruct the original skull surface. Moreover, we analyze and refine the Sant'Orsola skull dataset, designing functional pipelines for its processing. On the test set, the proposed approach is able to complete missing areas effectively, reaching high accuracy in terms of the predicted point locations and a good qualitative approximation of the complete skull.

1 Introduction

Cranioplasty is the surgical process where a skull defect, caused by a brain tumor surgery or by trauma, is repaired using a cranial implant, which must fit precisely against the borders of the skull defect as an alternative to the removed cranial bone. The creation of a cranial implant is a difficult task. It entails three steps: (1) acquiring 3D imaging data from CT or MRI of the skull with the defect, (2) translating the 3D imaging data into a 3D mesh model, and (3) designing an implant as a 3D mesh model for 3D printing. The last step is currently conducted by using costly commercial software and highly-trained professional users [54]. Fully automated, low-cost cranial completion could provide significant benefits and improvements to the current cranioplasty clinical workflow [32]. Previous research has resulted in open-source CAD tools for cranial implant design [14], however these methods are time-consuming and require human interaction. These methods usually make use of the geometric information contained in the shape of the skull. For example, one of the most common techniques is to find the skull's symmetry plane and mirror the defective region [13]. However, mirroring is not the best approach possible, considering that human heads are not perfectly symmetric.

Therefore, a fast and automatic design of cranial implants is highly desired. In this thesis we tackle the challenge of automated cranial reconstruction in a data-driven fashion, instead of relying explicitly on geometric shape priors of human skulls, to provide a meaningful starting point to help the maxillofacial surgeons in the design of cranial implants. Our goal is not to provide the final, perfect to millimetre implant but to develop a good qualitative approximation of the complete skull that will be used as a base to facilitate the cranioplasty clinical workflow. An effective estimate of the original, unaffected skull will

help the doctors to reduce the time and expenses needed to design the final implant. The cranial reconstruction task may be expressed as a volumetric shape learning problem, in which the shape of the implant is learned directly or indirectly from the shape of a damaged skull. On the one hand, the shape of the implant can be inferred directly from a defective skull. On the other hand, a complete skull can be generated by learning to fill the defective region on a defective skull. In this case, the implant is obtained indirectly by subtracting the completed and defective skulls. In this sense, cranial implant design can be formulated as a shape completion problem. For direct implant generation, the ground truth is the implant, which is the region removed from a complete skull. For skull shape completion, the ground truth is the original complete skull. The input of either formulation is the defective skull. Since our goal is to provide an effective qualitative approximation of the complete skull, in this study, we primarily elaborate on the second formulation, i.e., given a defective skull shape, we predict the shape of the complete skull.

Unlike existing skull completion methods, we directly operate on raw point clouds without any structural assumption (e.g. symmetry) or annotation (e.g. semantic class) about the underlying shape. Since point clouds or meshes are not in a regular format, most existing techniques voxelized the 3D data into occupancy grids or distance fields where convolutional networks can be applied. However, the cubically growing memory cost of 3D voxel grids limits the output resolution of these methods. Further, detailed geometry is often lost as an artifact of discretization that can obscure the natural invariances of the data. For this reason, we used point clouds as input representation for 3D geometry. Point clouds are unified and simple structures; this prevents the high memory cost and loss of geometric information caused by voxelization, avoid the combinatorial irregularities and complexities of meshes and allows our network to generate more fine-grained completions.

2 Related Work

As discussed in Chapter 1, cranial implant design can be formulated as a shape completion problem. 3D shape completion from partial point clouds is a fundamental problem in computer vision and computer graphics. This problem occurs when only a single view of an individual object is provided or large parts of the object are occluded as, e.g., in autonomous driving applications. Traditional geometry-based approaches, such as Poisson surface reconstruction [24] can only handle minor gaps in the collected 3D data. Unfortunately, these techniques frequently fail to reconstruct significant missing areas. Learning-based techniques are more suited for this purpose because of their capacity to learn powerful 3D shape priors from huge online 3D model datasets for repairing such missing regions. This section reviews the general shape completion algorithms used for various data modalities (points, meshes and voxel grids).

- **Voxel Grid Completion.** Traditional shape completion techniques [21, 11] work with volumetric data that has been voxelized from a point representation typically using a signed distance function. Voxel grid approaches have been widely used in recent experiments using convolutional neural networks (CNN) on volumetric images. Both works use an encoder-decoder network, which is limited to accepting voxel grids as input. Meshes can be retrieved from the final completed grids.

- **Point/Mesh Completion.** Recent advances in deep learning enable a CNN to learn from unstructured point clouds efficiently. An encoder-decoder can be used to perform shape completion directly on the raw point data [67, 74]. The widespread benchmark in litterature is based on ShapeNet [8], which is often used as a benchmark dataset for both

the voxel grid and point-based shape completion studies. To complete dense point clouds, Liu et al. [37] propose a two-step approach. They used an encoder-decoder network; the first step predicts a completed but coarse point cloud. In the second phase, a residual network generates a dense (high-fidelity) version of the completed point cloud, given as input a combination of the coarse output from the previous step and the partial point cloud. Early studies from Liepa [35] and Kraevoy et al. [27] perform shape completion directly on triangular meshes using classical geometry processing and mesh editing techniques. Shape completion on triangular meshes is significantly more complex than on binary voxel grids because the former data structure can carry far more information (e.g., texture and colour) about an object than the latter.

- **Medical Images.** Shape completion has also been applied to medical images. Prutsch et al. [53] used a GAN to complete 2D aortic dissection (AD) images (CT) in order to create healthy aorta images before dissection. Armanious et al. [2] developed a GAN network to complete arbitrary formed regions on 2D brain images. Gapon et al. [16] used a patch similarity matching approach to eliminate metal artifacts from 2D CT and MRI images. A multi-layer perceptron (MLP) was trained to search an image for the best matching patches to the missing region. Manjón et al. [39] employed a 3D U-Net to eliminate lesions on brain MRI images. While other studies all require an explicit definition of the region of interest (usually done manually) before completion, the trained network can complete the missing region without manual delineation of the lesions.

  It should be emphasized that because medical pictures are often grayscale, many medical shape completion applications necessitate the restoration of the shape and the voxel/pixel intensities of the missing region.

Existing 3D shape completion methods can then be divided into three main categories: geometry-based, alignment-based, and learning-based.

- **Geometry-based** techniques complete shapes without the use of external data by exploiting geometric hints from partial input. Surface reconstruction algorithms [5, 12, 76, 47, 63, 56], for example, generate smooth interpolations to fill holes in locally partial scans. Instead, symmetry-driven algorithms [42, 43, 51, 52, 61, 65] discover symmetry axes and regular recurring structures in the partial input to replicate pieces from observed regions to unobserved regions. These approaches assume that the geometry of the missing sections can be inferred directly from the seen regions. Unfortunately, this assumption does not hold on to most incomplete data from the real world.

- **Alignment-based** approaches complete shapes using template models from an extensive shape database to which the partial input is matched. Some of these methods [23, 25, 60, 64] retrieve object parts and then assemble them to obtain the complete shape, while others [20, 33, 46, 50, 58] retrieve the complete shape directly. Other studies [6, 15, 19, 30, 31, 55] alter the returned model to create shapes that are more consistent with the input. In some cases [9, 34, 45, 57, 73], geometric primitives such as planes and quadrics are used instead of a shape database. These approaches require time-consuming optimization during inference, making them unsuitable for online use. They are also susceptible to background noise.

- **Learning-based** techniques complete shapes using a parameterized model (typically a deep neural network) that directly maps partial input to a complete shape, allowing faster inference and generalization. This is where our method fits in. While most existing learning-based methods [21, 62, 71, 11, 59, 48] use voxels to represent shapes, which are

helpful for convolutional neural networks, our method employs point clouds, which preserve all geometric information about the shapes while being memory efficient.

Existing networks used in point cloud completion and reconstruction can be roughly categorized into MLP-based, graph-based, and convolution-based networks according to the network architecture employed.

- **MLP-based Networks**.Because of its simplicity and high representation ability, MLP is used in a number of works for point cloud processing [1, 36] and reconstruction [74, 38]. These approaches use multiple Multi-layer Perceptrons to model each point separately and then use a symmetric function to aggregate a global feature (e.g., Max Pooling). However, the geometric connections between 3D points are not fully taken into account.

- **Graph-based Networks** consider each point in a point cloud as a vertex of a graph.These kinds of networks generate directed graph's edges based on the neighbours of each point. Convolution is usually operated on spatial neighbours, and pooling is used to produce a new coarse graph by aggregating information from each point's neighbours. Graph-based networks consider local geometric structures in contrast to MLP-based techniques.

- **3D Convolution-based Networks**. 3D CNNs based on the volumetric representation of 3D point clouds were commonly used in early works [11, 21, 30]. Converting point clouds to 3D volumes creates a quantization effect which discards some data details [66] and is ineffective for expressing fine-grained information. Several works [40, 22, 29, 28] develop CNNs that operate on discrete 3D grids transformed from point clouds.

3 Dataset

3.1 Analysis

The maxillofacial unit of the Sant'Orsola polyclinic in Bologna made available for this study a dataset consisting of 415 meshes obtained from anonymized head CT scans in DICOM format. The meshes in the dataset were first moved to their Natural Head Position (NHP), a standardized and reproducible position of the head in an upright posture and then aligned, w.r.t a reference skull, according to 3 matching points, two on the frontozygomatic sutures and one on the basion. The initial alignment is finally refined using the Iterative closest point (ICP) algorithm [75].

For each skull in the dataset, a series of metadata have been manually annotated Tab 3.1. They describe the skulls in terms of their quality and extension. Most metadata can assume just two values, 0 or 1, to indicate the absence or presence of the corresponding skull's portion. The metadata for which this does not hold are Qual Score, Mirror, and the ones concerning the teeth, as shown in Figure 3.1.

Metadata	Description	Possible Values
Alv L	Mandible alveolar left part	0, 1
Alv R	Mandible alveolar right part	0, 1
Ang L	Left mandibular angle	0, 1
Ang R	Right mandibular angle	0, 1
Corp L	Left mandibular corpus	0, 1
Corp R	Right mandibular corpus	0, 1
Fron L	Frontal left bone	0, 1
Fron R	Frontal right bone	0, 1
Glen/Cond L	Left Mandibular glenoid/condyles	0, 1
Glen/Cond R	Right Mandibular glenoid/condyles	0, 1
Max Sin L	Maxillary left sinus	0, 1
Max Sin R	Maxillary right sinus	0, 1
Ment	Ment	0, 1
Mirror	Mirroring applicability	0, 1, 2
Occ	Occipital bone	0, 1
Orb L	Left orbit	0, 1
Orb R	Right orbit	0, 1
Par L	Parietal left bone	0, 1
Par R	Parietal right bone	0, 1
Qual Score	Comprehensive skull's quality score	1, 2, 3, 4, 5
Ramus L	Left mandibular ramus	0, 1
Ramus R	Right mandibular ramus	0, 1
Teeth Up L	Upper left dental arch	0, 1, 2
Teeth Up R	Upper right dental arch	0, 1, 2
Teeth Lw L	Lower left dental arch	0, 1, 2
Teeth Lw R	Lower right dental arch	0, 1, 2
Temp L	Temporal left bone	0, 1
Temp R	Temporal right bone	0, 1
Zyg L	Zygomatic left bone	0, 1
Zyg R	Zygomatic right bone	0, 1

Table 3.1: Dataset metadata and description.

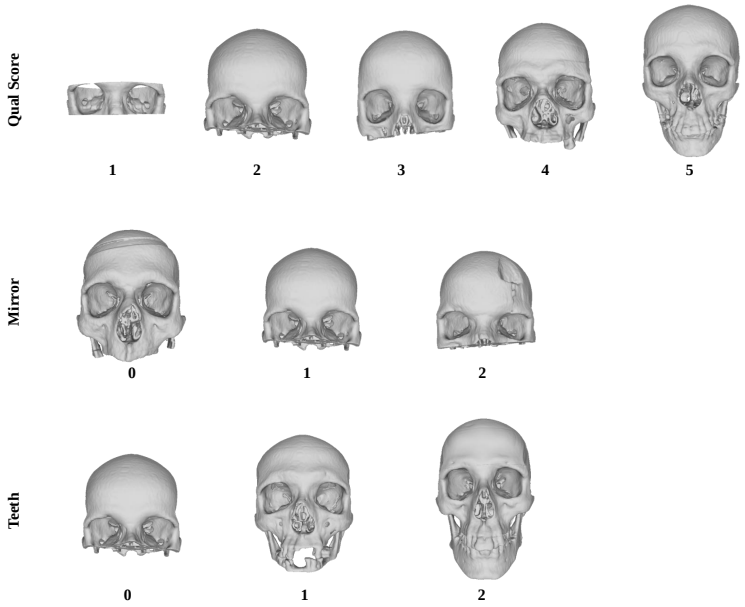


Figure 3.1: Examples per different Qual Score, Mirror and Teeth values.

The **Qual Score** metadata represents the comprehensive quality score of the skull. A domain expert gave this score based on his/her experience. It can assume a value between 1 and 5 (included); the higher the value, the better the overall quality.

The **Mirror** metadata refers to the possibility to apply a mirror operation to augment the available data. It can assume a value between 0 and 2 (included):

- $0 =$ *midline defects, mirroring not applicable*
- $1 =$ *mirror applicable*
- $2 =$ *mirroring can be used to correct single-sided defects*

The **Teeth Up R, Teeth Up L, Teeth Lw R, Teeth Lw L** metadata can assume 3 different values:

- $0 =$ *edentulous or not present in the CT cut*

- 1 = *partial edentulous or partially cut teeth in CT*
- 2 = *reasonably complete dental arch (continues up to at least the first molar)*

Considering that the dataset was acquired from many different, mostly segmental, data sources (e.g. CT scan of eumorphic patients, CT of composite or conservatively reduced fractures, CT for the evaluation of a head injury), we discarded the skulls that present a severe deformity or damage (e.g. meshes annotated with qual score 1 or mirror value other than 1). The resulting dataset contains 386 skulls in total.

By analyzing the dataset, both the original one and its cleaned version, we have found that most of the samples have between 250,000 and 400,000 vertices (points), most of them have quality score 3 and mirror value 1. Therefore, they are not fully segmented skulls, but there is the possibility of applying a mirror operation to augment the available data. As expected, there is a clear correlation between the quality score and the number of vertices, as shown in Figure 3.2.

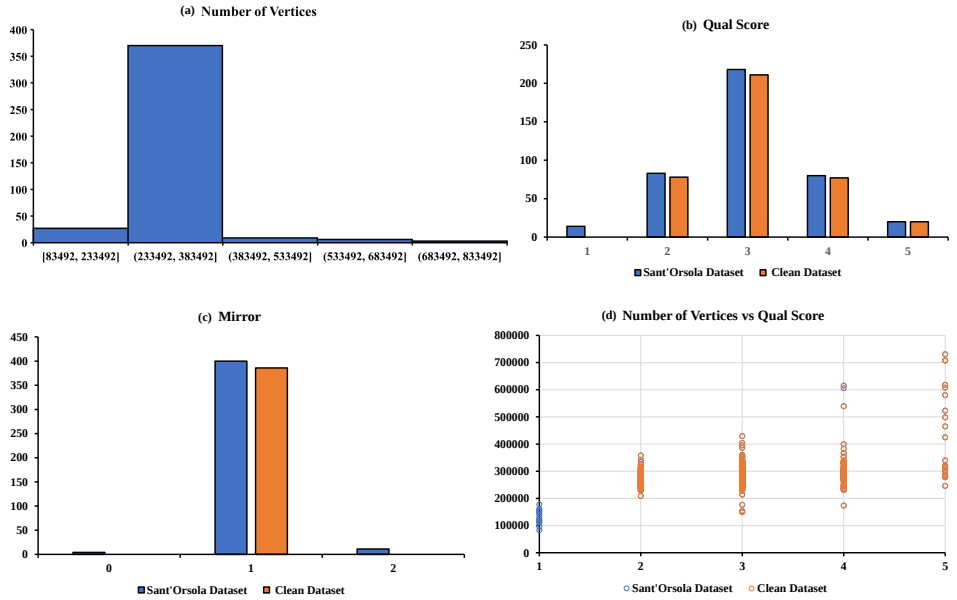


Figure 3.2: Dataset Analysis (a) Histogram of the number of vertices in the Sant'Orsola dataset, (b) Number of samples per different Qual Score value, (c) Number of samples per different Mirror value, (d) Correlation between the number of vertices and the quality score.

3.2 Processing

As anticipated in the previous chapters, we operate directly on raw point clouds without structural assumptions (e.g. symmetry) or annotation (e.g. semantic class) about the underlying form. Therefore, we have to convert the meshes in the dataset into point clouds. In 3D computer graphics and solid modelling, a polygon mesh is a geometric data structure that allows the representation of surface subdivisions by a set of polygons. Meshes are particularly used in computer graphics, to represent surfaces, or in modeling, to discretize a continuous or implicit surface. A mesh is a collection of vertices, edges, and faces that define a polyhedral object's shape, where:

- **Vertex:** A position (usually in 3D space) and other information such as colour, normal vector and texture coordinates.
- **Edge:** A connection between two vertices.

- **Face:** A closed set of edges, in which a triangle face has three edges.

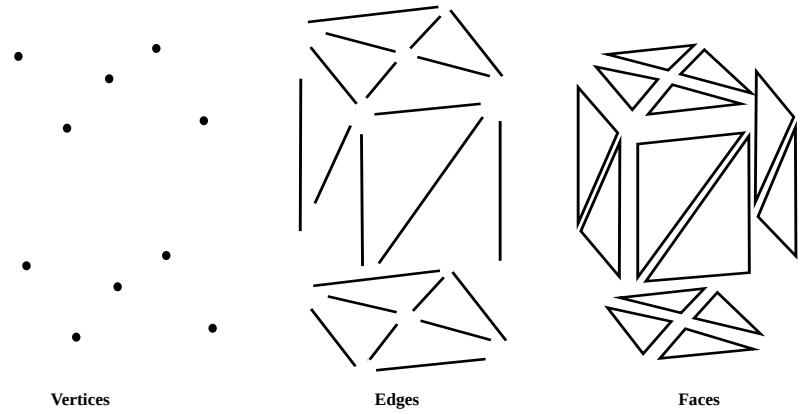


Figure 3.3: Mesh elements.

Thus, the simplest way to convert a mesh into a point cloud is to take its vertices and consider them as points. This approach, in most cases, is enough, but not in our situation. The skull meshes have many vertices hidden inside the model (Figure 3.4), which describe the internal skull bones structures: not very useful for our purposes. Most inner bones structures are typically not reconstructed faithfully since their structural features are much more important than the aesthetic ones. Moreover, use the internal points is deleterious from the point of view of point cloud processing. Unfortunately, there is no straightforward method to remove them.

Figure 3.4: Due to the internal points, there can be distinguished two main volumes. The innermost is made of points that can be removed to simplify the point clouds and are not necessary for this study's scope. (a) Frontal view, (b) Left Lateral view, (c) Parietal view, (d) Basilar view.

To eliminate the internal vertices, we have defined a specific pipeline, illustrated in Figure 3.5, that can be summarized as follows:

1. Take many snapshots of the 3D model from different angles.
2. Extract the depth map from each snapshot taken.
3. Convert the depth information in point clouds.
4. Merge all the point clouds so obtained.
5. Simplify the final point cloud.

The first step requires taking several snapshots of the skull mesh from different points of view, Figure 3.5 (a). To be sure to capture every portion of the skulls, even the most hidden ones, we have taken 8 snaps by moving the camera along the horizontal axis, 45 degrees at a time, a parietal and a basilar view of the model. From each of these snapshots, the depth map has been extracted, Figure 3.5 (b), and converted in a point cloud by simply deprojecting the depth image into 3D points, considering the camera rotation and translation to keep all the point clouds in the same reference system, Figure 3.5 (c). The point clouds for every camera position are merged (concatenated) to obtain a

complete point cloud of the skull without internal points, which is finally simplified by using a Poisson Disk Sampling (PDS) algorithm [7], Figure 3.5 (d). PDS yields smoother complete point clouds than uniform sampling, making them better to represent the underlying object models.

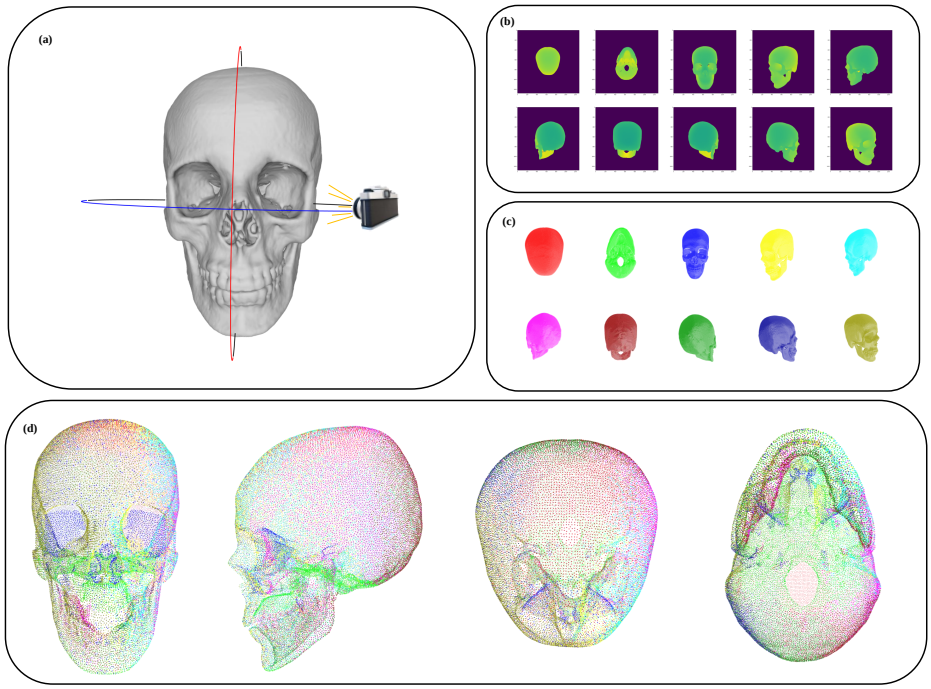


Figure 3.5: Internal points removal pipeline (a) Snapshots of the 3D model from different angles, (b) Depth maps of the snapshots, (c) Point clouds obtained from depth information, (d) Final result.

After the cleaning and processing steps, the resulting dataset is finally split into train (272), validation (29) and test set (73). In splitting the dataset, we made sure to keep the same proportion of samples w.r.t their quality score, Figure 3.6. This is a good practice in every deep learning task because separate the data into train, validation and test splits prevent overfitting. Overfitting happens when the model learns an overly specific function that performs well on training data but does not generalize to unseen data. Dataset splitting is also helpful to accurately evaluate the model performances.

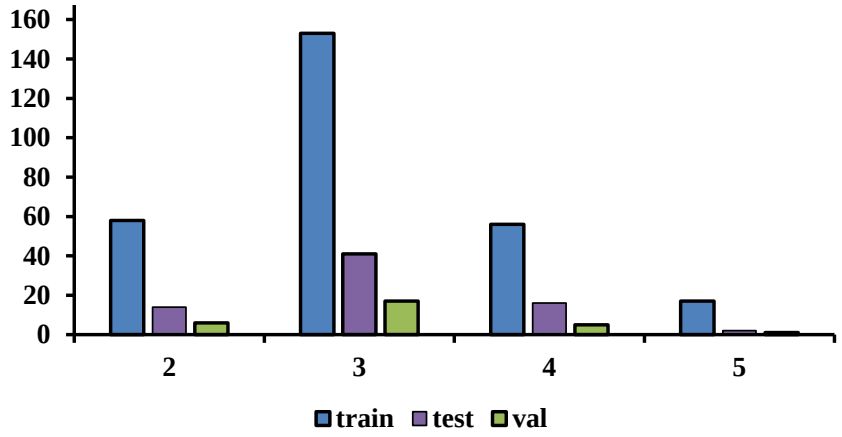


Figure 3.6: Quality score distribution for each dataset split.

3.3 Defect Injection

We had to create the defects artificially since the dataset we were provided consisted only of uninjured skulls. The artificial surgical defects were created by removing skull portions from the complete point clouds.

Figure 3.7 shows how defects are created. A point is picked randomly as the "centre" of the hole. For skulls of quality score 4, we take a random point on the face portion of the skull, Figure 3.7 (b). Instead, for skulls of quality score 5, a random point on the maxilla and mandible part is taken, Figure 3.7 (a). This is done because those areas are the ones of more interest for maxillofacial surgeons. The chosen point is then used as the centre of a rectangular parallelepiped with a square base of random, between 3cm and 10cm (included), height and width, Figure 3.7 (c). The range of dimensions of the defects was chosen under the advice of domain experts since it reflects the size of most real-case situations. In the end, every point that falls into the parallelepiped is removed from the complete skull point cloud, leaving a hole.

We created for each point cloud of quality score 2 or 3 only one partial cloud; instead, for the samples of quality score 4 or 5, we created as many partial clouds as to match the number of the sum of the available samples of quality

score 2 or 3 in the same dataset split. In this manner, we have more or less the same number of partial point clouds for each skull category. Finally, it is worth noticing that the partial clouds for the training set are created on the fly, so they are always different. For the validation and test set, they are fixed.

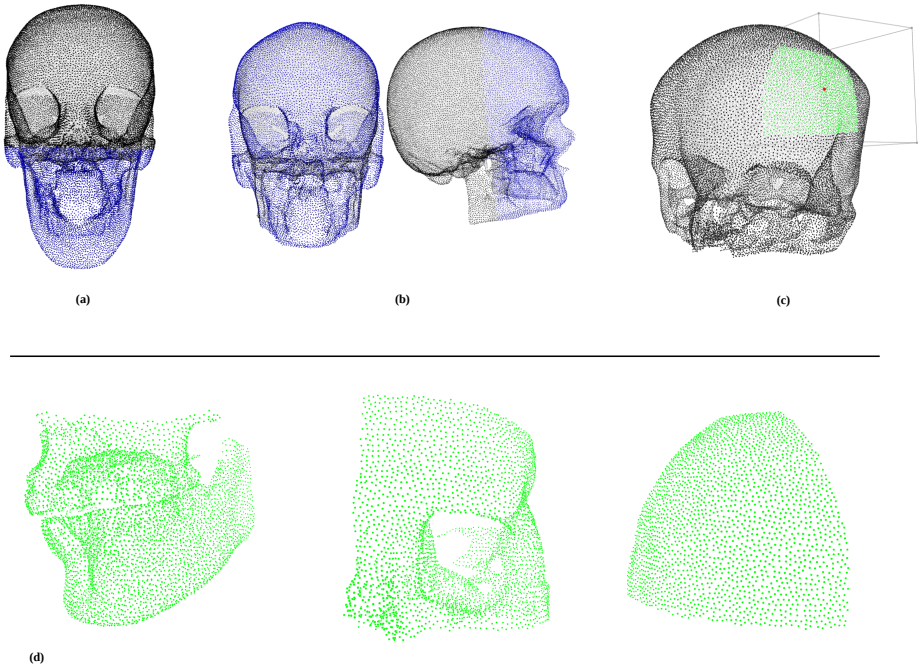


Figure 3.7: Defect Injection (a) Area from which to take a random point for $qualscore = 5$ point clouds, (b) Area from which to take a random point for $qualscore = 4$ point clouds, (c) Defect creation: in red, the random point used as the centre of the parallelepiped with a square base. In green, the selected points that will be removed to create the defect, (d) Defect examples.

Our datasets' defective skulls are generated artificially from complete skulls. Actual craniotomy surgery defects are generally more complicated and irregular than synthetic defects. However, we expect that deep learning networks trained on fake flaws will be able to generalize to genuine surgical defects, which will require the network to be resilient in terms of the defects' form, location, and size.

3.4 Normalization

In the end, each point cloud $P = \{p_i\}_{i=1}^{n}$, where n is the number of points in P, is normalized to guarantee that the coordinates of each of its points $p_i = (x_i, y_i, z_i)$ satisfy $-1 < x_i, y_i, z_i < 1$. In datasets like ShapeNet [8], the point clouds are normalized by subtracting their centroid to centre them in (0,0,0) and dividing by a scalar value:

$$m = \max_{p_i \in P}(\sqrt{\sum(x_i^2, y_i^2, z_i^2)}) \tag{3.1}$$

In our case, this cannot be done. The problem is that we have both complete skulls ($qualscore = 5$) and segmented skulls in our dataset, for example, the ones without mandibles. Therefore, depending on the ”completeness” of the skull and the dimension of the defect, m could vary a lot and consequently changing the skull scale, Figure 3.8 (b). This is undesirable since the point of normalization is to standardise the inputs to the network.

To overcome this issue, we have computed the maximum m of the point clouds in the training set ($m = 155$) and use this value as a fixed scale factor. In this way, the partial point clouds are normalized by subtracting their centroid and then diving by $m = 155$, Figure 3.8 (c). The ground truth complete point clouds are normalized using the centroid computed on the partial point cloud to keep the alignment between them. Note that we cannot simply normalize the point clouds before creating the defects because in a real case scenario, we may not have the original complete skull.

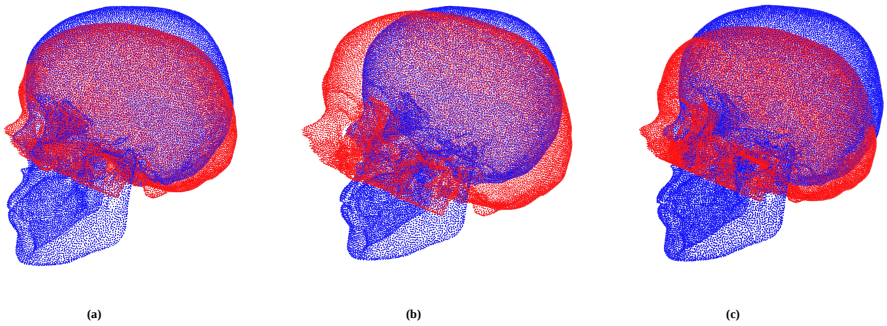


Figure 3.8: (a) Pre-normalization alignment, (b) Alignment after ShapeNet like normalization, (c) Alignment after using a fixed scale factor m for normalization.

From Figure 3.8 (a) and Figure 3.8 (c), it can be seen that normalization has introduced a degree of misalignment between the point clouds. We have found that this misalignment is not a problem; instead, it is necessary to avoid what we have called the ”phantom mandible issue”. This issue refers to the reconstruction errors due to the inability of the network to distinguish between skulls with defects and not fully segmented skulls. For instance, between a complete skull with a defect on the mandible and a skull that does not include that bone portion. In those cases, the network did not reconstruct the mandible from this the name ”phantom mandible issue”. As we will see in the next chapter, our model reconstructs the entire skull, not only the missing portion. The network learns, without any external indication, what to reconstruct and what not. For example, in the case of skulls without the mandible ($qualscore <= 5$), it learns that in those cases, the mandible should not be reconstructed because it is not a defect, but it is just the skull scan that does not include it. Without the misalignment introduced by normalization, it is harder for the network to catch this hidden assumption.

4 Skull Reconstruction

This chapter presents the pipeline designed to generate starting from the point cloud of the defective skull the mesh of the complete skull. The skull reconstruction pipeline can be summarized as follows:

1. **Completion:** recover the complete 3D skull point cloud from an incomplete one.
2. **Merging:** merge the partial point cloud with the reconstructed points, i.e. the points needed to fill the defect.
3. **Normal Estimation:** estimate the normals for the reconstructed points.
4. **Surface Reconstruction:** use a surface reconstruction technique to obtain the completed skull's mesh.

4.1 GRNet

The neural network used is based on the Gridding Residual Network (GRNet) [70], a very successful network able to recover a complete point cloud from an incomplete one in a coarse-to-fine fashion by introducing 3D grids as intermediate representations to regularize unordered point clouds. We chose to base our model on this architecture because it is able to recover better details of objects than other point cloud completion methods, its explicit architecture makes it easy to modify and improve it, and it is manageable from the computational point of view. The main drawback of GRNet is that it moves the input points, leading to a noisier reconstruction. Another missing of this network is that it is not able to recover points normals. We have modified the GRNet architecture in order to be able to work with point clouds with many more

points and to achieve better results. GRNet comprises five parts: Gridding, 3D Convolutional Neural Network, Gridding Reverse, Cubic Feature Sampling, Multi-layer Perceptron and Gridding Loss, as shown in Figure 4.1.

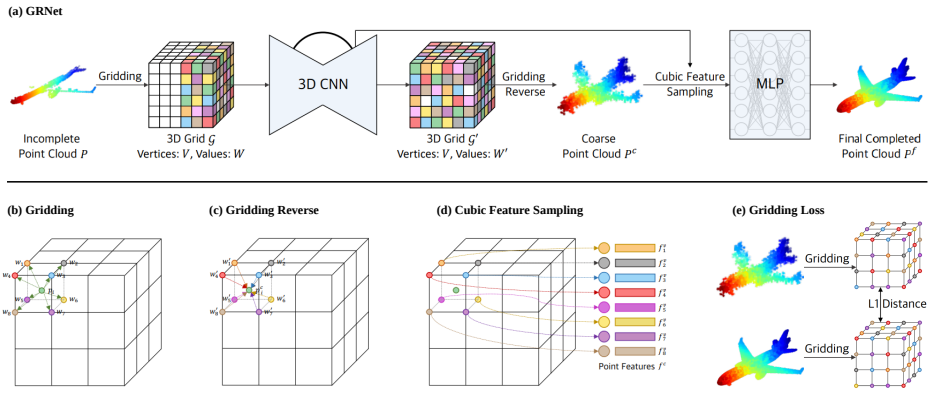


Figure 4.1: GRNet overview taken from the original paper [70] (a) GRNet, (b) Gridding, (c) Gridding Reverse, (d) Cubic Feature Sampling, and (e) Gridding Loss.

4.1.1 Gridding

In chapters 1 and 2 we have seen that most of the existing methods voxelize the point clouds into 3D voxels where 3D convolutions can be applied. However, detailed geometry is often lost as an artefact of discretization that can obscure the natural invariances of the data. Recent methods use MLP for point cloud reconstruction and aggregate information using a symmetric function (e.g., Max Pooling). The problem with MLP-based methods is that the geometric relationships among 3D points are not fully considered.

The core idea behind GRNet is the introduction of 3D grids as intermediate representations to regularize point clouds. Given an irregular and unordered point cloud $P = \{p_i\}_{i=1}^{n}$, where $p_i \in \mathbb{R}^3$ and n is the number of points in P, they convert it into a regular 3D grid $\mathcal{G} = \langle V, W \rangle$ and at the same time preserving the point cloud's spatial arrangements. $V = \{v_i\}_{i=1}^{N^3}$, $W = \{w_i\}_{i=1}^{N^3}$, $v_i \in \{(-\frac{N}{2}, -\frac{N}{2}, -\frac{N}{2}), ..., (\frac{N}{2} - 1, \frac{N}{2} - 1, \frac{N}{2} - 1)\}$, $w_i \in \mathbb{R}$ and N is the resolution of the 3D grid $\mathcal{G}$. A cell is defined as a cube, as shown in Figure

4.1 (b). The set of neighboring points $\mathcal{N}(v_i)$ for each vertex $v_i = (x_i^v, y_i^v, z_i^v)$ of the 3D grid is defined as the set of points that lie in the adjacent 8 cells of the considered vertex. Therefore, a point $p = (x, y, z)$, $p \in P$, if satisfies $x_i^v - 1 < x < x_i^v + 1$, $y_i^v - 1 < y < y_i^v + 1$, $z_i^v - 1 < z < z_i^v + 1$ then it belongs to $\mathcal{N}(v_i)$.

Moreover, the differentiable *Gridding layer*, as shown in Figure 4.1 (b) computes the corresponding value w_i of a vertex v_i as:

$$w_i = \sum_{p \in \mathcal{N}(v_i)} \frac{w(v_i, p)}{|\mathcal{N}(v_i)|} \tag{4.1}$$

where $|\mathcal{N}(v_i)|$ is the number of neighboring points of v_i so if $|\mathcal{N}(v_i)| = 0$ then also $w_i = 0$. In standard voxelization, instead, w_i at the vertex v_i is computed as:

$$w_i = \begin{cases} 0 & \forall p \notin \mathcal{N}(v_i) \\ 1 & \exists p \in \mathcal{N}(v_i) \end{cases} \tag{4.2}$$

This voxelization process, however, creates a quantization effect, which discards some object details. Furthermore, because voxelization is not differentiable, it cannot be used for point clouds reconstruction.

Finally, the interpolation function $w(v_i, p)$ is defined as:

$$w(v_i, p) = (1 - |x_i^v - x|)(1 - |y_i^v - y|)(1 - |z_i^v - z|) \tag{4.3}$$

4.1.2 3D Convolutional Neural Network

The purpose of the 3D Convolutional Neural Network (3D CNN) with skip connections is to fill in the missing parts in the incomplete point cloud. It is based on the concept of a three-dimensional encoder-decoder with U-net connections [68, 69]. The 3D CNN can be formulated as follows, given W as

input:

$$W' = 3DCNN(W) \tag{4.4}$$

where $W' = \{w_i'\}_{i=1}^{N^3}$ and $w_i' \in \mathbb{R}$.

4.1.3 Gridding Reverse

Gridding Reverse is the component used in GRNet to generate the coarse point cloud, as illustrated in Figure 4.1 (c). It takes the 3D grid $\mathcal{G}' = \langle V, W' \rangle$ and returns a sparse point cloud $P^c = \{p_i^c\}_{i=1}^m$, where m is the number of points in the coarse point cloud and $p_i^c \in \mathbb{R}^3$. It generates one point coordinate p_i^c for each grid cell by a weighted combination of its eight vertices coordinates and corresponding values:

$$p_i^c = \frac{\sum_{\theta \in \Theta^i} w_\theta^i v_\theta}{\sum_{\theta \in \Theta^i} w_\theta^i} \tag{4.5}$$

where $\Theta^i = \{\theta_j^i\}_{j=1}^8$ is the index set of vertices of the i−th 3D grid cell.

4.1.4 Cubic Feature Sampling

To overcome the inability of MLP-based methods to take into account the geometric relationships among 3D points, in GRNet *Cubic Feature Sampling*, Figure 4.1 (d), is used to combine features for the coarse point cloud P^c. This is useful for the following MLP layers to retrieve more fine-grained details. More precisely, given the feature map of 3D CNN $\mathcal{F} = \{f_1^v, f_2^v, ..., f_{t^3}^v\}$, where $f_i^v \in \mathbb{R}^c$ and t^3 is the size of the feature map, for a coarse point cloud P^c its features are computed as:

$$f_i^c = [f_{\theta_1^i}^v, f_{\theta_2^i}^v, ..., f_{\theta_8^i}^v] \tag{4.6}$$

so by contatenating the features of eight vertices of the i-th 3D gird cell where p_c^i lies in. In the end, a chosen number of points is randomly sampled from the coarse point cloud to reduce redundancy and generate a fixed number of points.

4.1.5 Multi-layer Perceptron

The Multi-layer Perceptron (MLP) takes the coarse point cloud P^c and its features F^c as input and returns $P^f = \{p_i^f\}_{i=1}^k$, where $p_i^f \in \mathbb{R}^3$ and k is the number of points in the final completed point cloud, as:

$$Pf = MLP(F^c) + Tile(P^c, r) \tag{4.7}$$

Tile creates a new tensor of size $rm \times 3$ by replicating P^c r times. In GRNet, r is set to 8. The MLP is used to recover the details from the coarse point cloud by learning residual offsets between the coarse and final completed point cloud points.

4.1.6 Gridding Loss

The last notable proposal by the authors of GRNet is the *Gridding Loss* which computes the L1 distance between the generated points and ground truth by representing them in regular 3D grids exploiting the aforementioned *Gridding layer*. Hence, let $\mathcal{G}_{pred} =< V^{pred}, W^{pred} >$ and $\mathcal{G}_{gt} =< V^{gt}, W^{gt} >$ be the 3D grids obtained by Gridding the predicted and ground truth point clouds, respectively, the *Gridding Loss* can be defined as:

$$\mathcal{L}_{Gridding}(W^{pred}, W^{gt}) = \frac{1}{N_G^3} \sum \|W^{pred} - W^{gt}\| \tag{4.8}$$

where N_G is the resolution of the two 3D grids.

4.2 Proposed Model

4.2.1 Architecture

Figure 4.2 illustrates the architecture used in our study. The number of layers and their connections match the one proposed in GRNet, whereas their dimension has been changed in order to be able to take in input incomplete point clouds with $35,000$ points and return a complete reconstructed point cloud of $40,000$ points. The original GRNet proposal could accept partial point clouds of only 2048 points and produce complete point clouds of $16,384$ points. Please note that this does not mean that the partial clouds must have precisely $35,000$ points, but if it $N < 35,000$, where N is the number of points, zero padding is added. Instead, if $N > 35,000$, $35,000$ points are randomly sampled.

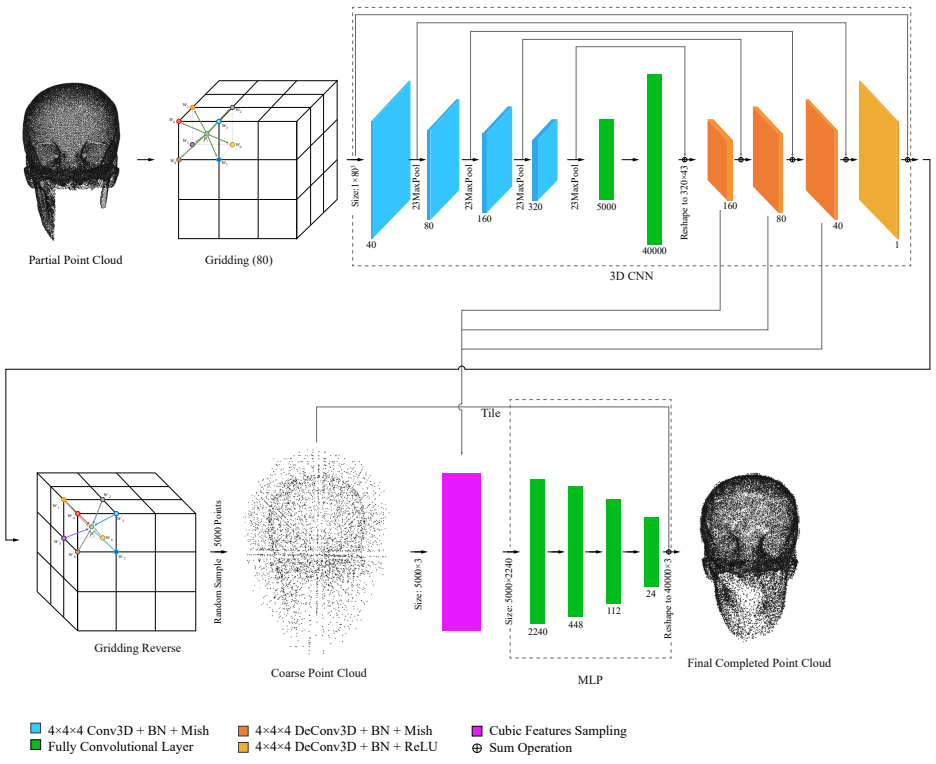


Figure 4.2: Neural Network architecture.

We have designed the network in this way because we have found that $40,000$ is a good enough number of points both to capture every skull detail and, at the same time, to maintain the number of network parameters accountable by the computational resources available to us.

As shown in Figure 4.2, the 3D CNN encoder consists of four 3D convolutional layers, each having a bank of 4^3 filters with padding of 2, followed by batch normalization and a max-pooling layer with a kernel size of 2^3. Convolutional layers have 40, 80, 160 and 320 output channels, respectively. Finally, two fully connected layers with sizes of 5000 and $40,000$ follow the encoder. The decoder is made up of four transposed convolutional layers, each having a bank of 4^3 filters with padding of 2 and stride of 1, followed by a batch normalizing layer.

Every convolution, except for the last one (followed by a ReLU activation), is followed by the MISH activation function [41], as we found that it improved our training. MISH provides a smoother energy landscape, resulting in less peaked training losses than those obtained with ReLU.

Cubic Feature Sampling collects point features from feature maps produced by 3D CNN's first three transposed convolutional layers. 5000 points from the coarse point cloud P^c are randomly picked to minimize redundancy and create a predetermined number of points. As a result, it generates a feature map with a size of 5000×2240.

The MLP consists of four fully connected layers with dimensions of 2240, 448, 112, and 24, respectively. The output of MLP is reshaped to $40,000 \times 3$, which corresponds to the offsets of the coordinates of $40,000$ points.

4.2.2 Loss Functions

Our model is trained end-to-end, and the training loss consists of two parts: $\mathcal{L}_{sparse}$ and $\mathcal{L}_{dense}$. Where $\mathcal{L}_{sparse}$ is the loss computed on the coarse point

clouds P^c, and $\mathcal{L}_{dense}$ is the loss computed on the final reconstructed point clouds P^f. Considering the training efficiency, we choose the symmetric Chamfer Distance (CD) as loss for both $\mathcal{L}_{sparse}$ and $\mathcal{L}_{dense}$:

$$\mathcal{L}_{CD}(P,Q) = \frac{1}{|P|}\sum_{x\in P}\min_{y\in Q}||x-y||^2 + \frac{1}{|Q|}\sum_{y\in Q}\min_{x\in P}||x-y||^2 \tag{4.9}$$

where x and y denote points that belong to two point clouds **P** and **Q**, respectively. Consequently, the joint loss function can be formulated as:

$$\mathcal{L} = \mathcal{L}_{sparse} + \mathcal{L}_{dense} \tag{4.10}$$

where:

$$\mathcal{L}_{sparse} = \mathcal{L}_{CD}(P^c, P^{gt}) \tag{4.11}$$

and

$$\mathcal{L}_{dense} = \mathcal{L}_{CD}(P^f, P^{gt}) \tag{4.12}$$

During the fine-tuning step, we have used the *Gridding Loss* (Equation 4.8), instead of the Chamfer Distance as loss for the loss on coarse clouds, $\mathcal{L}_{sparse} = \mathcal{L}_{Gridding}(P^c, P^{gt})$.

4.3 Reconstructed Points

Working directly on the complete reconstructed point cloud to reconstruct the skull surface is not a good idea. GRNet reconstruct the entire point cloud, so even the points belonging to the partial input point clouds are moved and subject to noise. Moreover, we already have the normals of those points. For these reasons, we decided to find and then extract from the completed reconstructed point clouds only the reconstructed points, i.e., the points needed to fill the defects, and merge them with the partial point clouds to obtain better

complete skull point clouds.

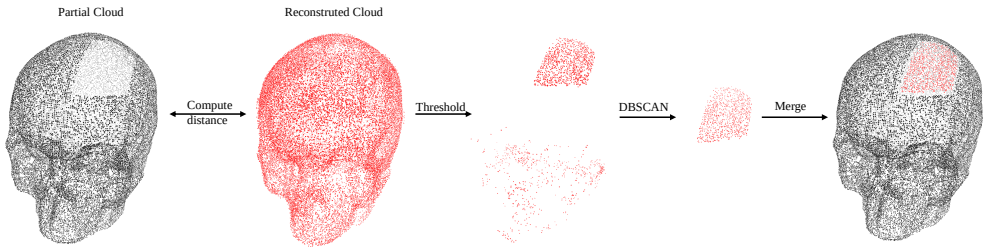


Figure 4.3: Reconstructed points selection pipeline.

Obtained the final reconstructed complete point clouds, we can find the points relative to the defect by simply computing the distance between the partial point clouds and the reconstructed ones. Hence by computing for each point in the partial cloud the distance to the closest point in the reconstructed cloud. Let $P^{partial}$ and $P^f = \{p_i^f\}_{i=1}^k$, be respectively a partial point cloud and its final completed reconstruction, the set of reconstructed points, so the ones added to fill the defect, is given by:

$$P^R = \left\{ p_i^f \,|\, \min(d(P^{partial}, p_i^f)) < \varepsilon \right\} \tag{4.13}$$

where d is the Euclidean distance.

Due to the noise present in the reconstructed point cloud, not only the points needed to fill the defect will be selected. Therefore, we then apply the DBSCAN algorithm to select only the largest cluster of points. Finally, we merge the partial point cloud with the found points. The pipeline is summarized in Figure 4.3.

4.4 Normal Estimation

Many surface reconstruction algorithms require the use of normals to generate a mesh starting from a point cloud. Our network does not estimate the point normals, so we need to compute them afterwards. The normals of the points added to the partial cloud to fill the defect are estimated by finding adjacent points and determining the principal axis of the adjacent points using covariance analysis. The two key parameters are the search radius and the maximum number of the nearest neighbour. As normal candidates, the covariance analysis algorithm generates two opposing directions. Both can be valid without knowing the global structure of the geometry. This is referred to as the normal orientation problem. For this reason, we then used a further orientation function to orient the normals with respect to consistent tangent planes.

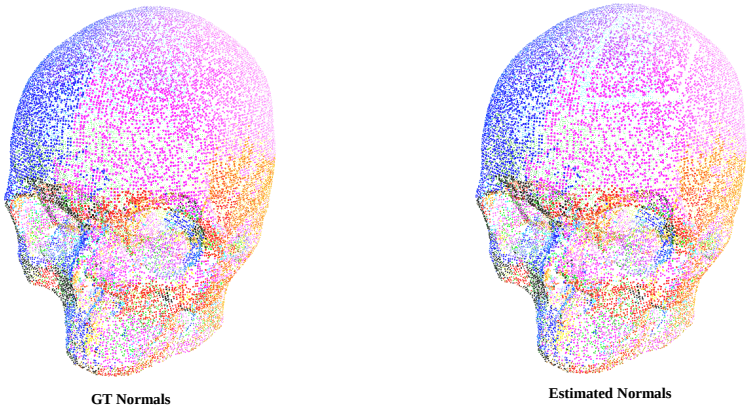


Figure 4.4: Example of normal estimation result.

For complex 3D point clouds, plot normals as vectors is not very informative. It is tough to see where the little arrows point. Therefore, we mapped normal vectors to the RGB cube. The function takes a unit vector (a point on the sphere), finds where it intersects the RGB cube, and uses that colour as the vector representation [3]. In Figure 4.4 it is reported an example of this kind of visualization and a normal estimation result.

4.5 Surface Reconstruction

Surface reconstruction from point clouds is a core topic in geometry processing [4]. It is an ill-posed problem: an infinite number of surfaces approximate a single point cloud, and a point cloud does not define a surface in itself. Thus, the user must define additional assumptions and constraints, and reconstruction can be achieved in many different ways.

Using the reconstruction algorithms on outlier-ridden point clouds produce overly distorted output. Therefore we strongly filter these outliers before performing reconstruction.

We chose Poisson reconstruction [24] as a reconstruction algorithm. It computes an implicit function whose gradient matches the gradient of the input normal vector field. Inside and outside the inferred shape, this indicator function has opposing signs (hence the need for closed shapes). This approach necessitates the use of normals and yields smooth, closed surfaces. If the surface is intended to interpolate the input points, it is not suitable. On the contrary, it works well when the goal is to approximate a noisy point cloud with a smooth surface, like in our case.

Finally, we performed an explicit remeshing of the triangular mesh obtained Poisson reconstruction algorithm by repeatedly applying edge flip, collapse, relax and refine to improve aspect ratio and topological regularity.

5 Experiments

5.1 Implementation Details

Our network is implemented using PyTorch [49], an open-source machine learning framework with automatic differentiation and eager execution, and CUDA. The leading third-party libraries on which the project is based are Open3D [77] and PyMeshLab [44]. Open3D is an open-source library that supports the rapid development of software that deals with 3D data; instead, PyMeshLab is a Python library that interfaces to MeshLab [10], the popular open-source application for editing and processing large 3D triangle meshes.

All the training and validation processes were executed on Google Colaboratory [17], a platform that gives the possibility to exploit some computational resources for free. In particular, Colab allows you to select a GPU runtime that boosts the training time of neural models and most of the time we were assigned an NVIDIA Tesla T4 GPU with 16GB of RAM. We use the Adam optimiser [26] with learning rate 10^{-4}, $\beta 1 = 0.9$, $\beta 2 = 0.999$ and batch size 4. We trained the network for 250 epochs, with the learning rates decayed by 0.5 after 100 epochs. We also fine-tuned the network using as training loss the *Gridding Loss* for other 50 epochs still using Adam as optimizer and 10^{-4} as initial learning rate.

5.2 Evaluation Metrics

One of the challenges in point cloud completion is the comparison with the ground truth. As evaluation metrics, we have chosen these existing similarity metrics:

Chamfer Distance. For two point clouds P and Q, CD measures the mean distance between each point in one point cloud to its nearest neighbour in the other point cloud, Equation 4.9.

Earth Mover Distance. Compared to the Chamfer Distance (CD), the Earth Mover's Distance (EMD) is more reliable to distinguish the visual quality of the point clouds [37]. EMD is only defined when P and Q have the same size:

$$EMD(P,Q) = \min_{\phi:P\to Q} \frac{1}{|P|} \sum_{x\in P} ||x - \phi(x)||_2 \tag{5.1}$$

By solving the linear assignment problem, EMD forces the output to have the same density distribution as the ground truth and is thus more discriminative to the local details and the density distribution.

F-Score. F1 score is defined as the harmonic average of precision (the accuracy) and recall (the completeness), where F1 reaches its best value at 1 (perfect accuracy and completeness) and worst at 0. In other terms:

$$\textit{F-Score}(\tau) = \frac{2P(\tau)R(\tau)}{P(\tau)+R(\tau)} \tag{5.2}$$

where $P(\tau)$ and $R(\tau)$ denote the precision and recall for a distance threshold τ, respectively. Let $\mathcal{G} = \{(x_i, y_i, z_i)\}_{i=1}^{n_\mathcal{G}}$ be the ground truth and $\mathcal{R} = \{(x_i, y_i, z_i)\}_{i=1}^{n_\mathcal{R}}$ be a reconstructed point set being evaluated, where $n_\mathcal{G}$ and $n_\mathcal{R}$ are the numbers of points of $\mathcal{G}$ and $\mathcal{R}$, respectively:

$$P(\tau) = \frac{1}{n_\mathcal{R}} \sum_{r\in\mathcal{R}} \Big[\min_{g\in\mathcal{G}} ||g - r|| < \tau\Big] \tag{5.3}$$

$$R(\tau) = \frac{1}{n_\mathcal{G}} \sum_{g\in\mathcal{G}} \Big[\min_{r\in\mathcal{R}} ||g - r|| < \tau\Big] \tag{5.4}$$

Cosine Distance. We have used the cosine distance to measure the similarity between vertices normals. The cosine distance is simply $D_c = 1 - S_c$, where

S_c is the cosine similarity between the vectors. However, it is essential to note that this is not a proper distance metric as it does not have the Schwarz inequality, and it violates the coincidence axiom. The cosine distance is a significant metric because, as said in section 4.5, the normals orientation is fundamental when dealing with surface reconstruction algorithms such as Screened Poisson.

Finally, for the sake of clarity, in the following sections, we will refer to each of these metrics in two different ways depending on if they were computed on all the 40,000 points of the reconstructed point clouds or if they were computed only on the reconstructed points. Hence, the ones predicted to fill the defect. In the former case, the ground truth is the complete skull, and to the metric name will be set the subscript *C* in the other case, the ground truth will be the removed portion of the skull, and *R* will be set as subscript to the metric acronym. Thus, for example, the Chamfer Distance will assume the acronym CD_C if computed on the entire set of 40,000 points returned by the model; instead, CD_R if it refers to the value computed just on the reconstructed points.

5.3 Baselines

We compare our method against two baselines:

1. **AtlasNet** [18], which generates points by sampling from the parametric surface element. AtlasNet naively duplicate primitive-decoder pairs to comply with different shapes.

2. **FoldingNet** [72], which proposes folding-based decoder to directly generate points. FoldingNet adopts a PointNet-like encoder to produce latent representations. Like the PointNet encoder, FoldingNet decoder is a point-wise network shared among points. Given a predefined 2D primitive (e.g., 2D square patch or a sphere), the FoldingNet decoder

takes a 2D coordinate from the primitive and latent codeword as input, then maps the 2D point to a 3D coordinate. The set of 3D points mapped by the FoldingNet decoder constitute the reconstructed point cloud.

For a fair comparison, we train all the models using the Chamfer Distance loss with their released codes. All models utilize the same training set.

5.4 Quantitative Evaluation

To achieve a fair comparison, we train all methods on our dataset using the same training strategy. For all evaluated methods in Table 5.1 are reported the Chamfer Distance, the F-score and the Earth Mover Distance. Our network clearly outperforms all the other models. Moreover, we have computed the evaluation metrics only for the reconstructed points, so the ones added to fill the defect, Table 5.2. Finally, in Tables 5.3 and 5.4, we have reported the results obtained by our network (still computed only on the reconstructed points) for different defect positions and defect dimensions, respectively.

Method	$CD_C \downarrow$	$EMD_C \downarrow$	F-score$_C$(3mm)$\uparrow$
Ours	**2.3672**	**4.5473**	**0.9696**
AtlasNet	11.667	12.5332	0.4995
FoldingNet	7.462	25.1551	0.4732

Table 5.1: Point completion results compared using Chamfer Distance(CD), Earth Mover Distance(EMD) and F-Score(3mm) computed on 40,000 points. The best results are highlighted in bold.

Method	$CD_R\downarrow$	$EMD_R\downarrow$	F-score$_R$(3mm)$\uparrow$	Cosine Distance$_R\downarrow$
Ours	**9.3813**	**2.9719**	**0.7532**	**0.3124**
AtlasNet	62.5015	25.1945	0.1456	0.9222
FoldingNet	61.5883	24.0986	0.1356	0.8984

Table 5.2: Point completion results compared using Chamfer Distance(CD), Earth Mover Distance(EMD), F-Score(3mm) and Cosine Distance computed on the new data. The best results are highlighted in bold.

Defect Position	$CD_R\downarrow$	$EMD_R\downarrow$	F-score$_R$(3mm)$\uparrow$	Cosine Distance$_R\downarrow$
Cranium	9.5883	2.9825	**0.8219**	**0.2545**
Face/Mandible	**8.9706**	**2.8121**	0.6169	0.4271

Table 5.3: Point completion results for defect positions.

Defect Dimension	$CD_R\downarrow$	$EMD_R\downarrow$	F-score$_R$(3mm)$\uparrow$	Cosine Distance$_R\downarrow$
[3,5[cm	9.4049	3.1174	0.6981	**0.2785**
[5,7[cm	11.2443	3.249	0.7591	0.3102
[7,10] cm	**7.7831**	**2.6875**	**0.7662**	0.3254

Table 5.4: Point completion results for defect sizes.

Please note that the metrics reported in these tables are computed on the de-normalized samples to improve the interpretability of the results. It is also worth noticing how the network seems to generalise pretty well w.r.t the defect dimension, whereas it seems more susceptible to the defect position.

5.5 Qualitative Evaluation

The qualitative comparison results are shown in Figure 5.1. The proposed network, based on GRNet, can generate better complete shapes with fine details than the other methods. It can effectively reconstruct complete shapes by learning structural and context relations from the incomplete point cloud, including geometrical symmetries, regular arrangements, and surface smoothness. Unfortunately, FoldingNet and AtlasNet fail to represent point clouds with complex topologies even if the networks are scaled up.

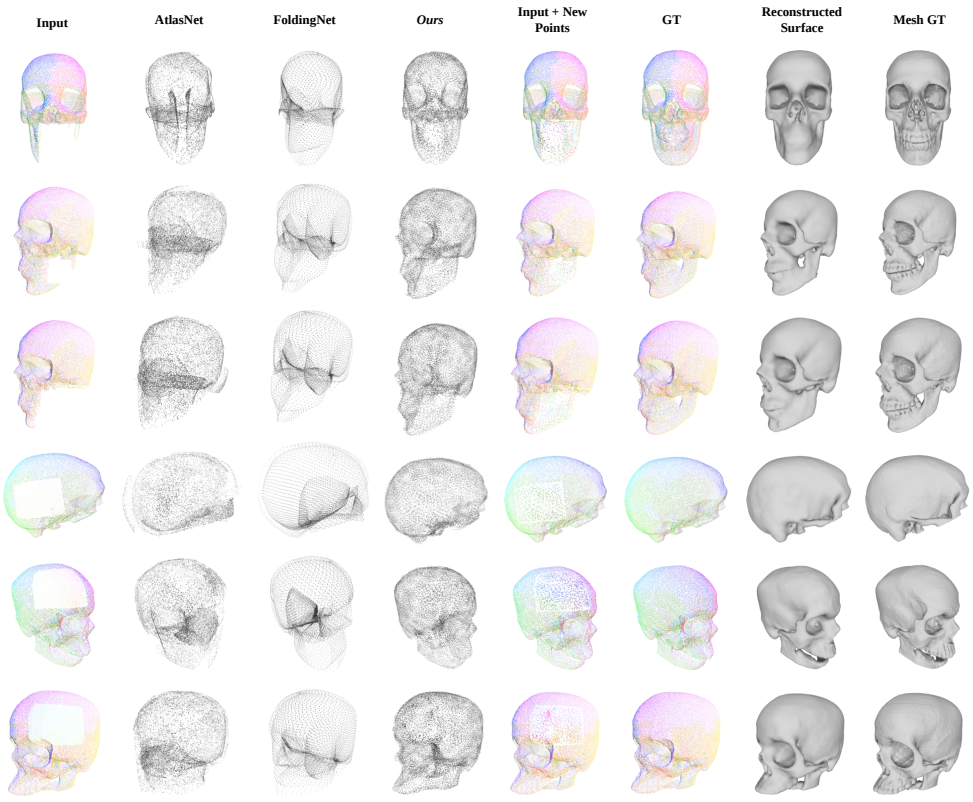
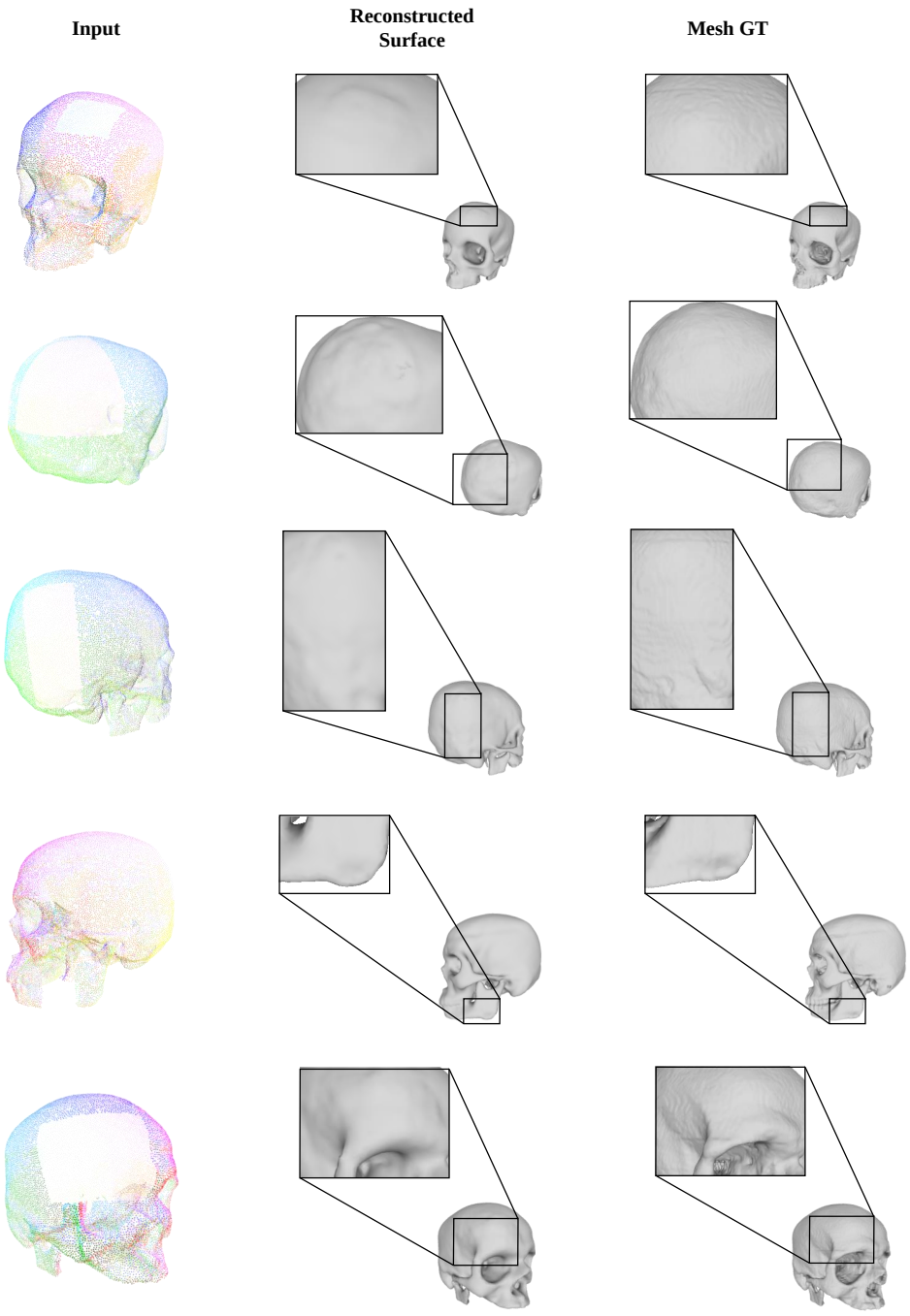


Figure 5.1: Qualitative comparison and results.

5.5.1 Additional Results

In this section, we provide more visual results highlighting the areas where the defect was present.


Input
Reconstructed Surface
Mesh GT

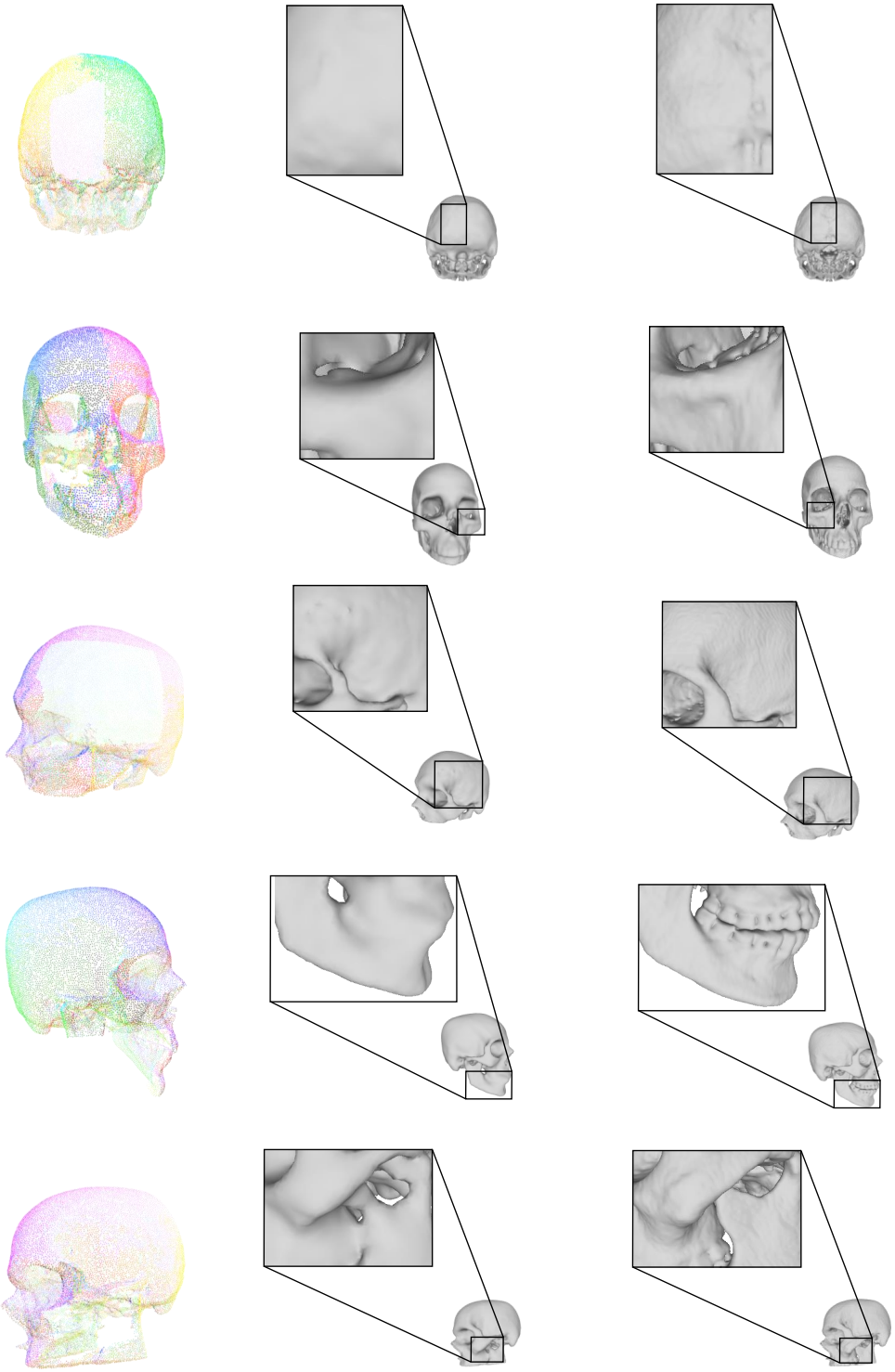


Figure 5.2: Qualitative results with a focus on the defective region.

5.5.2 Error Analysis

We can cluster the errors made by our network in three main categories.

The first one deals with the errors due to a poor normals estimation. As mentioned in the section 4.5, we use the Poisson surface reconstruction algorithm. This method to work well requires that the normals of the points are correctly estimated and oriented. In some cases, this is not the case, and consequently, the results are badly affected by this.

The second major cause of poor results is the noise present in the point clouds returned by the network. The points are not equally distantiated one to the other, and they do not lay on the same plane. This lead to a non-smooth surface.

Finally, in the last category fall the mistakes due to a combination of factors from a poor normals estimation to noise and other possible sources of errors such as the failure of the pipeline to determine the reconstructed points or simply the failure of the network to reconstruct particular shapes or details due to little training data for certain defects.

In Figure 5.3, for each category of error are reported two visual examples.

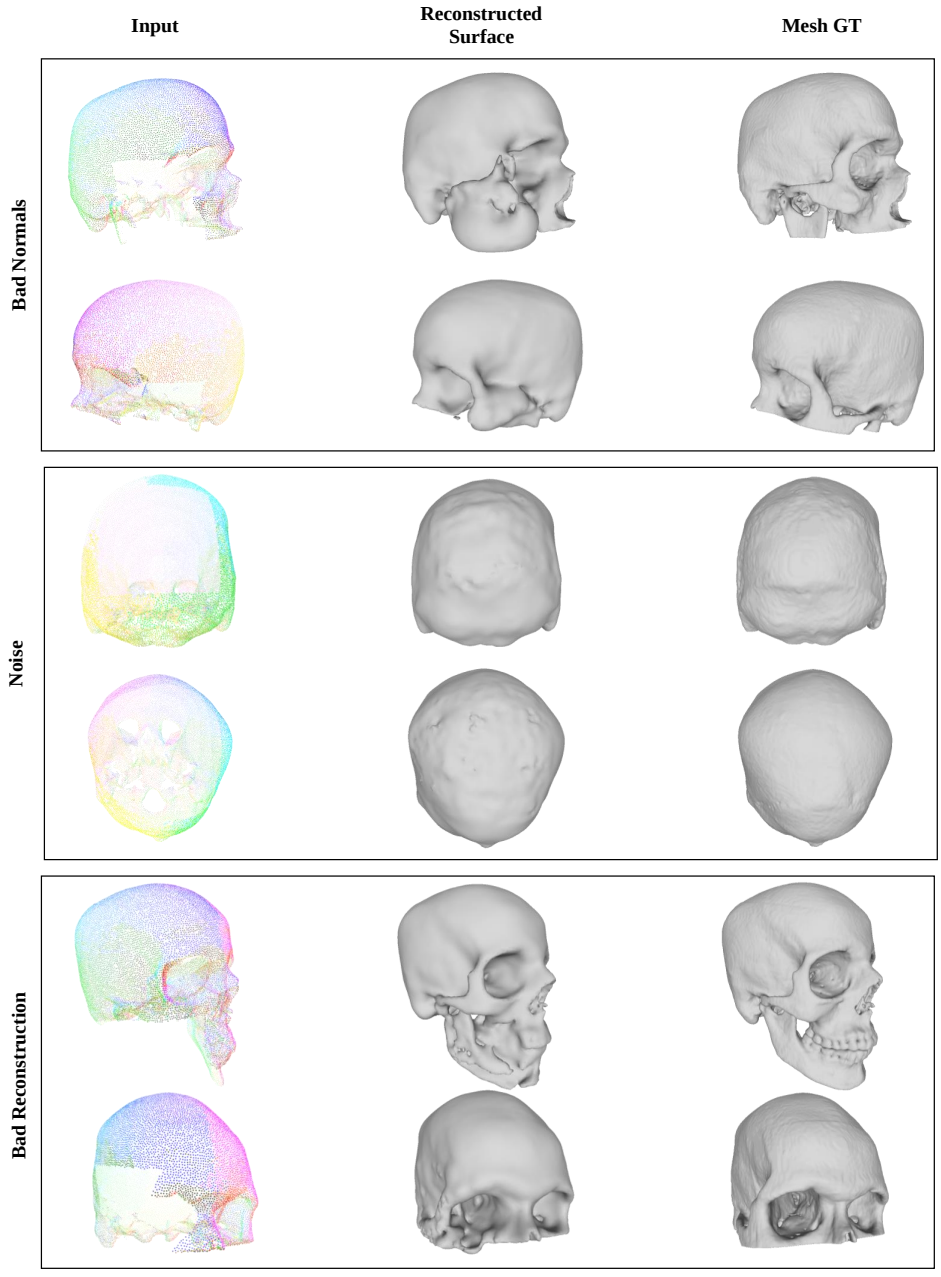


Figure 5.3: Results per different error categories.

6 Conclusion

In this thesis, we proposed a shape completion approach to skull reconstruction. Unlike existing skull completion methods, we directly operate on raw point clouds without any structural assumption or annotation about the underlying shape. Point clouds are unified and simple structures; this prevents the high memory cost and loss of geometric information caused by voxelization, avoid the combinatorial irregularities and complexities of meshes and allows our network to generate more fine-grained completions.

The model used is based on GRNet opportunely modified and improved to deal with the available dataset. In addition, we contribute to analyze and refine the Sant'Orsola skull dataset, designing functional pipelines for its processing. The proposed approach is able to complete missing areas effectively, reaching high accuracy in terms of the predicted point locations and a good qualitative approximation of the complete skull that will be used as starting point to facilitate the cranioplasty clinical workflow.

Future works could use this study as a baseline and improve it by investigating the possibility of filling in new data only where needed while keeping original samples. Another possible future development may deal with normals prediction. These extensions will simplify the pipeline to obtain the final reconstructed skull surface starting from the model output leading to better quantitative and qualitative results.

Bibliography

[1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning representations and generative models for 3d point clouds, 2018. arXiv: 1707.02392 [cs.CV]. 2.

[2] K. Armanious, V. Kumar, S. Abdulatif, T. Hepp, S. Gatidis, and B. Yang. Ipa-medgan: inpainting of arbitrary regions in medical imaging. *2020 IEEE International Conference on Image Processing (ICIP)*, October 2020. DOI: 10.1109/icip40778.2020.9191207. URL: http://dx.doi.org/10.1109/ICIP40778.2020.9191207. 2.

[3] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer. Nesti-net: normal estimation for unstructured 3d point clouds using convolutional neural networks. *arXiv preprint arXiv:1812.00709*, 2018. 4.5.

[4] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf, and C. T. Silva. A survey of surface reconstruction from point clouds. *Comput. Graph. Forum*, 36(1):301–329, January 2017. ISSN: 0167-7055. DOI: 10.1111/cgf.12802. URL: https://doi.org/10.1111/cgf.12802. 4.5.

[5] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, J. A. Levine, A. Sharf, and C. T. Silva. State of the Art in Surface Reconstruction from Point Clouds. In S. Lefebvre and M. Spagnuolo, editors, *Eurographics 2014 - State of the Art Reports*. The Eurographics Association, 2014. DOI: 10.2312/egst.20141040. 2.

[6] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 187–194, USA. ACM Press/Addison-Wesley Publishing Co., 1999. ISBN: 0201485605.

DOI: 10.1145/311535.311556. URL: https://doi.org/10.1145/311535.311556. 2.

[7] J. Bowers, R. Wang, L.-Y. Wei, and D. Maletz. Parallel poisson disk sampling with spectrum analysis on surfaces. *ACM Trans. Graph.*, 29(6), December 2010. ISSN: 0730-0301. DOI: 10.1145/1882261.1866188. URL: https://doi.org/10.1145/1882261.1866188. 3.2.

[8] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 2, 3.4.

[9] A.-L. Chauve, P. Labatut, and J.-P. Pons. Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1261–1268, 2010. DOI: 10.1109/CVPR.2010.5539824. 2.

[10] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In V. Scarano, R. D. Chiara, and U. Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008. ISBN: 978-3-905673-68-5. DOI: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136. 5.1.

[11] A. Dai, C. R. Qi, and M. Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis, 2017. arXiv: 1612.00101 [cs.CV]. 2.

[12] J. Davis, S. Marschner, M. Garr, and M. Levoy. Filling holes in complex surfaces using volumetric diffusion. In *Proceedings. First International Symposium on 3D Data Processing Visualization and Transmission*, pages 428–441, 2002. DOI: 10.1109/TDPVT.2002.1024098. 2.

[13] L. Di Angelo, P. Di Stefano, L. Governi, A. Marzola, and Y. Volpe. A robust and automatic method for the best symmetry plane detection of craniofacial skeletons. *Symmetry*, 11(2), 2019. ISSN: 2073-8994. DOI: 10.3390/sym11020245. URL: https://www.mdpi.com/2073-8994/11/2/245. 1.

[14] J. Egger, M. Gall, A. Tax, M. Ücal, U. Zefferer, X. Li, G. von Campe, U. Schäfer, D. Schmalstieg, and X. Chen. Interactive reconstructions of cranial 3d implants under mevislab as an alternative to commercial planning software. *PLOS ONE*, 12(3):1–20, March 2017. DOI: 10.1371/journal.pone.0172694. URL: https://doi.org/10.1371/journal.pone.0172694. 1.

[15] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010. DOI: 10.1109/TPAMI.2009.167. 2.

[16] N. Gapon, V. Voronin, R. Sizyakin, D. Bakaev, and A. Skorikova. Medical image inpainting using multi-scale patches and neural networks concepts. *IOP Conference Series: Materials Science and Engineering*, 680:012040, December 2019. DOI: 10.1088/1757-899X/680/1/012040. 2.

[17] Google. Colaboratory. URL: https://colab.research.google.com/. 5.1.

[18] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. Atlasnet: a papier-mâché approach to learning 3d surface generation, 2018. arXiv: 1802.05384 [cs.CV]. 5.3.

[19] S. Gupta, P. Arbelaez, R. Girshick, and J. Malik. Aligning 3d models to rgb-d images of cluttered scenes. In pages 4731–4740, June 2015. DOI: 10.1109/CVPR.2015.7299105. 2.

[20] F. Han and S. Zhu. Bottom-up/top-down image parsing by attribute graph grammar. In volume 31, 1778–1785 Vol. 2, November 2005. ISBN: 0-7695-2334-X. DOI: 10.1109/ICCV.2005.50. 2.

[21] X. Han, Z. Li, H. Huang, E. Kalogerakis, and Y. Yu. High-resolution shape completion using deep neural networks for global structure and local geometry inference, 2017. arXiv: 1709.07599 [cs.CV]. 2.

[22] B.-S. Hua, M.-K. Tran, and S.-K. Yeung. Pointwise convolutional neural networks, 2018. arXiv: 1712.05245 [cs.CV]. 2.

[23] E. Kalogerakis, S. Chaudhuri, D. Koller, and V. Koltun. A probabilistic model for component-based shape synthesis. *ACM Trans. Graph.*, 31(4), July 2012. ISSN: 0730-0301. DOI: 10.1145/2185520.2185551. URL: https://doi.org/10.1145/2185520.2185551. 2.

[24] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, pages 61–70, Cagliari, Sardinia, Italy. Eurographics Association, 2006. ISBN: 3905673363. 4.5.

[25] V. G. Kim, W. Li, N. J. Mitra, S. Chaudhuri, S. DiVerdi, and T. Funkhouser. Learning part-based templates from large collections of 3D shapes. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 32(4), July 2013. 2.

[26] D. P. Kingma and J. Ba. Adam: a method for stochastic optimization, 2017. arXiv: 1412.6980 [cs.LG]. 5.1.

[27] V. Kraevoy and A. Sheffer. Template-Based Mesh Completion. In M. Desbrun and H. Pottmann, editors, *Eurographics Symposium on Geometry Processing 2005*. The Eurographics Association, 2005. ISBN: 3-905673-24-X. DOI: 10.2312/SGP/SGP05/013-022. 2.

[28] S. Lan, R. Yu, G. Yu, and L. S. Davis. Modeling local geometric structure of 3d point clouds using geo-cnn, 2018. arXiv: 1811.07782 [cs.CV]. 2.

[29] H. Lei, N. Akhtar, and A. Mian. Octree guided cnn with spherical kernels for 3d point clouds, 2019. arXiv: 1903.00343 [cs.CV]. 2.

[30] D. Li, T. Shao, H. Wu, and K. Zhou. Shape completion from a single rgbd image. *IEEE Transactions on Visualization and Computer Graphics*, 23(7):1809–1822, 2017. DOI: 10.1109/TVCG.2016.2553102. 2.

[31] G. Li, L. Liu, H. Zheng, and N. J. Mitra. Analysis, reconstruction and manipulation using arterial snakes. In *ACM Transactions on Graphics*, volume 29 of number 5, to appear, 2010. 2.

[32] J. Li, A. Pepe, C. Gsaxner, and J. Egger. An online platform for automatic skull defect restoration and cranial implant design, 2020. arXiv: 2006.00980 [physics.med-ph]. 1.

[33] Y. Li, A. Dai, L. Guibas, and M. Nießner. Database-assisted object retrieval for real-time 3d reconstruction. In *Computer Graphics Forum*, volume 34 of number 2. Wiley Online Library, 2015. 2.

[34] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N. J. Mitra. Globfit: consistently fitting primitives by discovering global relations. *ACM Trans. Graph.*, 30(4), July 2011. ISSN: 0730-0301. DOI: 10.1145/2010324.1964947. URL: https://doi.org/10.1145/2010324.1964947. 2.

[35] P. Liepa. Filling Holes in Meshes. In L. Kobbelt, P. Schroeder, and H. Hoppe, editors, *Eurographics Symposium on Geometry Processing*. The Eurographics Association, 2003. ISBN: 3-905673-06-1. DOI: 10.2312/SGP/SGP03/200-206. 2.

[36] H. Lin, Z. Xiao, Y. Tan, H. Chao, and S. Ding. Justlookup: one millisecond deep feature extraction for point clouds by lookup tables, 2019. arXiv: 1908.08996 [cs.CV]. 2.

[37] M. Liu, L. Sheng, S. Yang, J. Shao, and S.-M. Hu. Morphing and sampling network for dense point cloud completion, 2019. arXiv: 1912.00280 [cs.CV]. 2, 5.2.

[38] P. Mandikal and R. V. Babu. Dense 3d point cloud reconstruction using a deep pyramid network, 2019. arXiv: 1901.08906 [cs.CV]. 2.

[39] J. V. Manjón, J. E. Romero, R. Vivo-Hernando, G. Rubio, F. Aparici, M. de la Iglesia-Vaya, T. Tourdias, and P. Coupé. Blind MRI Brain Lesion Inpainting Using Deep Learning. In *International Workshop on Simulation and Synthesis in Medical Imaging*, pages 41–49, LIMA, Peru, October 2020. DOI: 10.1007/978-3-030-59520-3_5. URL: https://hal.archives-ouvertes.fr/hal-02966536. 2.

[40] J. Mao, X. Wang, and H. Li. Interpolated convolutional networks for 3d point cloud understanding, 2019. arXiv: 1908.04512 [cs.CV]. 2.

[41] D. Misra. Mish: a self regularized non-monotonic activation function, 2020. arXiv: 1908.08681 [cs.LG]. 4.2.1.

[42] N. J. Mitra, L. Guibas, and M. Pauly. Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics (SIGGRAPH)*, 25(3):560–568, 2006. 2.

[43] N. Mitra, M. Pauly, M. Wand, and D. Ceylan. Symmetry in 3d geometry: extraction and applications. *Computer Graphics Forum*, 32, September 2013. DOI: 10.1111/cgf.12010. 2.

[44] A. Muntoni and P. Cignoni. PyMeshLab, January 2021. DOI: 10.5281/zenodo.4438750. 5.1.

[45] L. Nan, A. Sharf, H. Zhang, D. Cohen-Or, and B. Chen. Smartboxes for interactive urban reconstruction. *ACM Trans. Graph.*, 29(4), July 2010. ISSN: 0730-0301. DOI: 10.1145/1778765.1778830. URL: https://doi.org/10.1145/1778765.1778830. 2.

[46] L. Nan, K. Xie, and A. Sharf. A search-classify approach for cluttered indoor scene understanding. *ACM Trans. Graph.*, 31(6), November 2012. ISSN: 0730-0301. DOI: 10.1145/2366145.2366156. URL: https://doi.org/10.1145/2366145.2366156. 2.

[47] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. Laplacian mesh optimization. In *Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia*, GRAPHITE '06, pages 381–389, Kuala Lumpur, Malaysia. Association for Computing Machinery, 2006. ISBN: 1595935649. DOI: 10.1145/1174429.1174494. URL: https://doi.org/10.1145/1174429.1174494. 2.

[48] D. T. Nguyen, B.-S. Hua, M.-K. Tran, Q.-H. Pham, and S.-K. Yeung. A field model for repairing 3d shapes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5676–5684, 2016. DOI: 10.1109/CVPR.2016.612. 2.

[49] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: an imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL: http://papers.neurips.cc/

paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. 5.1.

[50] M. Pauly, N. J. Mitra, J. Giesen, M. Gross, and L. Guibas. Example-based 3d scan completion. In *Symposium on Geometry Processing*, pages 23–32, 2005. 2.

[51] M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, and L. J. Guibas. Discovering structural regularity in 3d geometry. *ACM Trans. Graph.*, 27(3):1–11, August 2008. ISSN: 0730-0301. DOI: 10.1145/1360612.1360642. URL: https://doi.org/10.1145/1360612.1360642. 2.

[52] J. Podolak, P. Shilane, A. Golovinskiy, S. Rusinkiewicz, and T. Funkhouser. A planar-reflective symmetry transform for 3D shapes. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3), July 2006. 2.

[53] A. Prutsch, A. Pepe, and J. Egger. Design and development of a web-based tool for inpainting of dissected aortae in angiography images, 2020. arXiv: 2005.02760 [cs.CV]. 2.

[54] Renishaw. Digital evolution of cranial surgery, 2017. 1.

[55] J. Rock, T. Gupta, J. Thorsen, J. Gwak, D. Shin, and D. Hoiem. Completing 3d object shape from one depth image. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2484–2493, 2015. DOI: 10.1109/CVPR.2015.7298863. 2.

[56] K. Sarkar, K. Varanasi, and D. Stricker. Learning quadrangulated patches for 3d shape parameterization and completion, 2017. arXiv: 1709.06868 [cs.CV]. 2.

[57] R. Schnabel, P. Degener, and R. Klein. Completion and reconstruction with primitive shapes. *Computer Graphics Forum (Proc. of Eurographics)*, 28(2):503–512, March 2009. 2.

[58] T. Shao, W. Xu, K. Zhou, J. Wang, D. Li, and B. Guo. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Trans. Graph.*, 31(6), November 2012. ISSN: 0730-0301. DOI: 10.1145/2366145.2366155. URL: https://doi.org/10.1145/2366145.2366155. 2.

[59] A. Sharma, O. Grau, and M. Fritz. Vconv-dae: deep volumetric shape learning without object labels, 2016. arXiv: 1604.03755 [cs.CV]. 2.

[60] C.-H. Shen, H. Fu, K. Chen, and S.-M. Hu. Structure recovery by part assembly. *ACM Trans. Graph.*, 31(6), November 2012. ISSN: 0730-0301. DOI: 10.1145/2366145.2366199. URL: https://doi.org/10.1145/2366145.2366199. 2.

[61] I. Sipiran, R. Gregor, and T. Schreck. Approximate symmetry detection in partial 3d meshes. *Computer Graphics Forum*, 33(7):131–140, 2014. DOI: https://doi.org/10.1111/cgf.12481. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12481. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12481. 2.

[62] E. Smith and D. Meger. Improved adversarial systems for 3d object generation and reconstruction, 2017. arXiv: 1707.09557 [cs.CV]. 2.

[63] O. Sorkine and D. Cohen-Or. Least-squares meshes. In *Proceedings Shape Modeling Applications, 2004.* Pages 191–199, 2004. DOI: 10.1109/SMI.2004.1314506. 2.

[64] M. Sung, V. G. Kim, R. Angst, and L. Guibas. Data-driven structural priors for shape completion. *ACM Trans. Graph.*, 34(6), October 2015. ISSN: 0730-0301. DOI: 10.1145/2816795.2818094. URL: https://doi.org/10.1145/2816795.2818094. 2.

[65] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen. Shape completion enabled robotic grasping, 2017. arXiv: 1609.08546 [cs.RO]. 2.

[66] Z. Wang and F. Lu. Voxsegnet: volumetric cnns for semantic part segmentation of 3d shapes, 2018. arXiv: 1809.00226 [cs.CV]. 2.

[67] X. Wen, T. Li, Z. Han, and Y.-S. Liu. Point cloud completion by skip-attention network with hierarchical folding, 2020. arXiv: 2005.03871 [cs.CV]. 2.

[68] H. Xie, H. Yao, X. Sun, S. Zhou, and S. Zhang. Pix2vox: context-aware 3d reconstruction from single and multi-view images. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. DOI: 10.1109/iccv.2019.00278. URL: http://dx.doi.org/10.1109/ICCV.2019.00278. 4.1.2.

[69] H. Xie, H. Yao, S. Zhang, S. Zhou, and W. Sun. Pix2vox++: multi-scale context-aware 3d object reconstruction from single and multiple images. *International Journal of Computer Vision*, 128(12):2919–2935, July 2020. ISSN: 1573-1405. DOI: 10.1007/s11263-020-01347-6. URL: http://dx.doi.org/10.1007/s11263-020-01347-6. 4.1.2.

[70] H. Xie, H. Yao, S. Zhou, J. Mao, S. Zhang, and W. Sun. Grnet: gridding residual network for dense point cloud completion. In *ECCV*, 2020. 4.

[71] B. Yang, S. Rosa, A. Markham, N. Trigoni, and H. Wen. Dense 3d object reconstruction from a single depth view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(12):2820–2834, December 2019. ISSN: 1939-3539. DOI: 10.1109/tpami.2018.2868195. URL: http://dx.doi.org/10.1109/TPAMI.2018.2868195. 2.

[72] Y. Yang, C. Feng, Y. Shen, and D. Tian. Foldingnet: point cloud auto-encoder via deep grid deformation, 2018. arXiv: 1712.07262 [cs.CV]. 5.3.

[73] K. Yin, H. Huang, H. Zhang, M. Gong, D. Cohen-Or, and B. Chen. Morfit: interactive surface reconstruction from incomplete point clouds with curve-driven topology and geometry control. *ACM Trans. Graph.*,

33(6), November 2014. ISSN: 0730-0301. DOI: 10.1145/2661229.2661241. URL: https://doi.org/10.1145/2661229.2661241. 2.

[74] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert. Pcn: point completion network, 2019. arXiv: 1808.00671 [cs.CV]. 2.

[75] Z. Zhang. *Iterative closest point (icp)*. In *Computer Vision: A Reference Guide*. K. Ikeuchi, editor. 3.1. Springer US, Boston, MA, 2014, pages 433–434. ISBN: 978-0-387-31439-6. DOI: 10.1007/978-0-387-31439-6_179. URL: https://doi.org/10.1007/978-0-387-31439-6_179.

[76] W. Zhao, S. Gao, and H. Lin. A robust hole-filling algorithm for triangular mesh. In volume 23, pages 22–22, November 2007. ISBN: 978-1-4244-1579-3. DOI: 10.1109/CADCG.2007.4407836. 2.

[77] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 5.1.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my esteemed supervisor, Professor Luigi Di Stefano, for his unwavering support and belief in me.

I would also like to thank my co-supervisors, Professor Giuseppe Lisanti, Professor Samuele Salti and Dr Riccardo Spezialetti, for their insightful feedback that pushed me to sharpen my thinking and brought my work to a higher level.

Additionally, I would like to express gratitude to Dr Claudio Marchetti and Dr Mirko Beverini of the maxillofacial unit of the St.Orsola-Malpighi Polyclinic for their treasured support and domain knowledge which was really influential in shaping my experiment methods and critiquing my results. Moreover, this thesis work would not have been possible without the dataset they collected and made available to us.

Lastly, my appreciation also goes out to my family, particularly my brother Andrea, and friends for their encouragement and support throughout my studies.