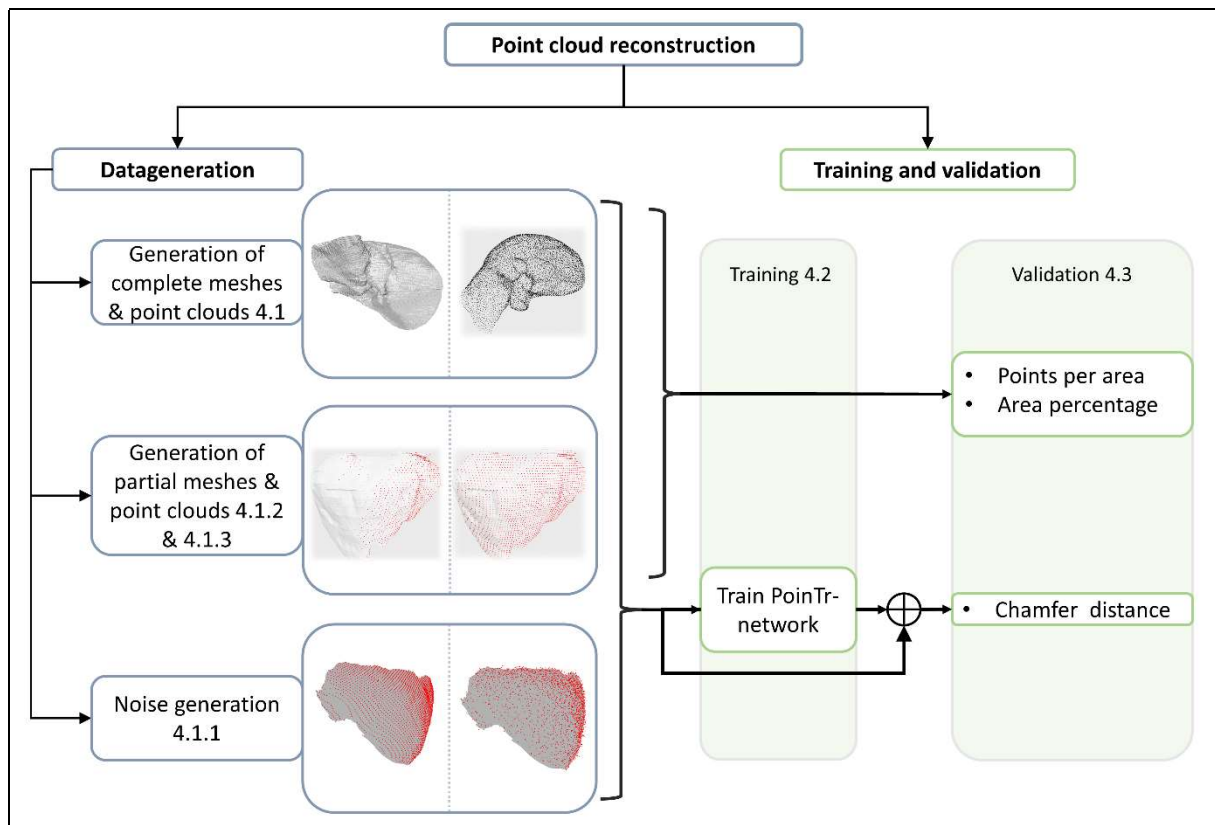


Bachelorarbeit zum Thema

DEEP LEARNING METHODS FOR PARTIAL TO WHOLE OBJECT REGISTRATION IN MIXED REALITY GUIDED LIVER SURGERY - PRELIMINARY EXPERIMENTS



Vorgelegt von
Guido Reinfurt
aus Markttheidenfeld

eingereicht: 12.04.2024
Betreuer: Prof. Dr. Stefanie Remmele



HOCHSCHULE LANDSHUT
HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN

Fakultät für Elektrotechnik und Wirtschaftsingenieurwesen

Erklärung zur Bachelorarbeit

(gemäß § 9 d, Abs. 3 APO)

Name, Vorname der/des Student(in)en: **Reinfurt, Guido**

Hochschule
Fakultät Elektrotechnik und Wirtschaftsingenieurwesen

Landshut

Hiermit erkläre ich, dass ich die Arbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benützt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

12.04.2024

(Datum)

.....

(Unterschrift der/des Student(in)en)



HOCHSCHULE LANDSHUT
HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN

Fakultät für Elektrotechnik und Wirtschaftsingenieurwesen

FREIGABEERKLÄRUNG DER/DES STUDENT(IN)EN

Reinfurt Guido

Hiermit erkläre ich, dass die vorliegende Bachelorarbeit in den Bestand der Hochschulbibliothek aufgenommen werden kann und

ohne Sperrfrist

oder nach einer Sperrfrist von

- 1 Jahr
- 2 Jahren
- 3 Jahren
- 5 Jahren oder länger

über die Hochschulbibliothek zugänglich gemacht werden darf.

12.04.2024

(Datum)

.....

(Unterschrift der/des Student(in)en)

Table of Contents

| | | |
|-------|---|----|
| 1 | Purpose..... | 6 |
| 2 | Motivation..... | 7 |
| 3 | A theoretical base for point cloud reconstruction..... | 12 |
| 3.1 | Point cloud theory..... | 12 |
| 3.2 | Marching cubes algorithm..... | 12 |
| 3.3 | Ray casting..... | 14 |
| 3.4 | Transformer theory..... | 15 |
| 3.4.1 | Transformer finetuning..... | 17 |
| 3.5 | PoinTr: Diverse point cloud completion with geometry-aware transformers..... | 17 |
| 4 | Method..... | 19 |
| 4.1 | Data generation..... | 20 |
| 4.1.1 | Noise generation..... | 22 |
| 4.1.2 | Training data generation..... | 23 |
| 4.1.3 | Test data generation..... | 25 |
| 4.2 | Training of PoinTr..... | 26 |
| 4.3 | Description of experiments..... | 28 |
| 4.3.1 | Experiment 1. Test data description..... | 28 |
| 4.3.2 | Experiment 2. Comparison of model trained on noisy data to model trained on clean data..... | 28 |
| 4.3.3 | Experiment 3. Comparison of different test data categories on Model A and Model B..... | 28 |
| 4.4 | Materials..... | 30 |
| 5 | Results..... | 31 |
| 5.1 | Test data description..... | 31 |
| 5.2 | Comparison of model trained on noisy data to model trained on clean data..... | 34 |
| 5.3 | Comparison of different test data categories on Model A and Model B..... | 35 |
| 5.4 | Qualitative Results..... | 36 |
| 6 | Discussion..... | 39 |
| 6.1 | Test data validation:..... | 39 |
| 6.1.1 | Analysis of Captured Area Percentage..... | 39 |
| 6.1.2 | Analysis of Points Per Area..... | 39 |
| 6.2 | Comparison of model trained on noisy data to model trained on clean data..... | 40 |
| 6.3 | Comparison of the influence of test data parameters on the reconstruction performance..... | 40 |
| 6.4 | Data generation..... | 42 |
| 6.4.1 | Mesh generation..... | 42 |
| 6.4.2 | Partial point cloud generation..... | 44 |

| | | |
|-----|-----------------------|----|
| 6.5 | Training process..... | 44 |
| 7 | Conclusion | 46 |
| 8 | Acknowledgment..... | 47 |
| 9 | References..... | 48 |
| 10 | List of figures | 53 |
| 11 | List of tables..... | 56 |

1 Purpose

This work primarily focuses on improving registration procedures within augmented reality (AR) applications, by analyzing the reliability of point cloud reconstruction as a preprocessing step. Therefore, a state-of-the-art, transformer base network is utilized. For the training process a comprehensive dataset specifically designed for training deep learning models is created. This dataset will be tailored to the task of reconstructing partial liver views for augmented reality guidance in medical applications. The validation process for this dataset will be two-fold:

- **Deep Learning Network Training:** The created dataset will be used to train a state-of-the-art deep learning network, specifically a transformer-based model, the “PoinTr”-network. This training will evaluate the dataset's effectiveness in facilitating accurate point cloud reconstruction.
- **Testing with Real-World Scenarios:** A separate testing dataset, meticulously designed to reflect various real-world conditions encountered in AR guidance, such as variations in point cloud density, occlusions, and sensor noise, will be used to assess the performance of the trained model. This evaluation will analyze the model's ability to generate accurate reconstructions under diverse and potentially challenging scenarios.

2 Motivation

The field of liver surgery has historically faced significant challenges. In the early 1950s, limitations in preoperative diagnosis led to low rates of successful liver resections, as evidenced by the first reported case of right hepatic lobe resection [1]. Between 1968-1977, due to the limited possibility of obtaining a preoperative diagnosis, in 4031 liver cases in Japan, only 4 % were treated by resection. 518 patients were subjected to exploratory laparotomy, with 3-year and 5-year survival probability as poor as 19.6 % and 11.8 %, respectively [2].

Clinical outcomes improved rapidly in the 1980s due to refinements in technology like intraoperative ultrasonography and vein embolization [3]. However, even with these advancements, challenges remain. Performing liver resections using ultrasound presents a significant challenge. Techniques like ultrasound require a high level of skill and experience to intellectually integrate the ultrasound images with the actual surgical environment. Because of these constraints, a significant proportion of patients undergo suboptimal resections, characterized by inadequate surgical margins [4]. To address these obstacles, image-guidance systems have been integrated into the surgical process [5]. While these systems offer some advantages by projecting the orientation of surgical instruments onto a digital representation of anatomy, there are some challenges. The operators utilizing image-guided therapy must alternate their attention between the surgical area and the live feed displayed on the monitor which displaying real-time data. The complexity arises from the demand of intellectually aligning various imaging modalities. Accurately identifying targets within the patient's body requires careful examination and interpretation of the visual information on the screen, increasing the overall complexity of the procedure and potentially leading to higher surgical risks [6].

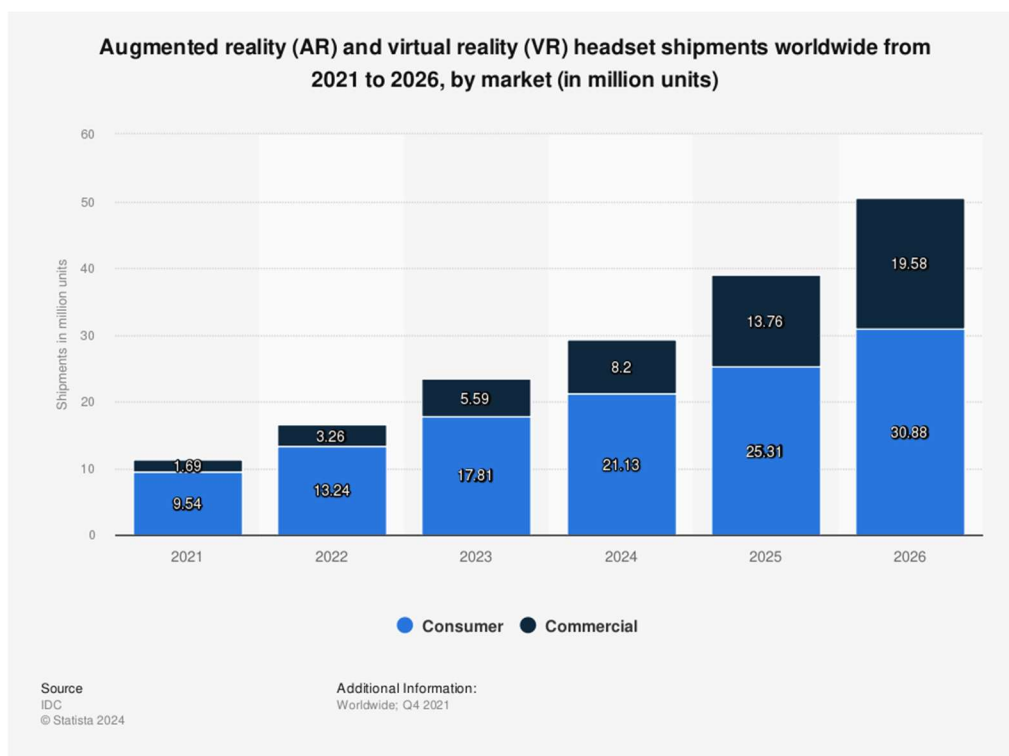


Figure 1. Illustrates the projected shipment forecast for augmented reality (AR) and virtual reality (VR) headsets globally, segmented by market (consumer and commercial) from 2021 to 2026 [7].

The limitations of image-guided therapy highlight the need for improved visualization and cognitive support for surgeons. This is where augmented reality (AR) emerges as a promising solution. The growing availability of advanced AR hardware, including Microsoft HoloLens, Oculus Quest, and Google Glass 2, has promoted their integration into healthcare settings (Figure 1). When augmented reality (AR) technology is integrated with image-guided interventions, it allows surgeons to directly visualize target organs and their surroundings within the actual surgical field. This means that hidden parts of organs inside the patient’s body become visible to the surgeon, enhancing their perception during the interventional procedure. Such advancements are exemplified by methods developed by Lonedì for distal locking screw procedures during intramedullary nail placement and Elmi-Terander for pedicle screw placement [8], [9]. By eliminating the need for numerous fluoroscopic images, these methods led to improvements in both surgical speed and accuracy.

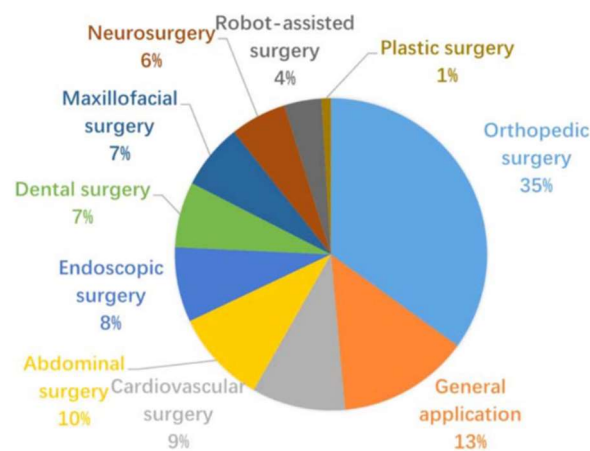


Figure 2. Illustrates the distribution of 103 research papers focusing on augmented reality (AR) applications in various surgical procedures [10].

Image registration is a critical step when using augmented reality technologies. It involves aligning feature points from patient imaging data, such as CT scans or MRIs, with corresponding points in the actual anatomy of patients. Other registration methods, such as surface registration, may also be employed. This ensures that the virtual information overlaid by AR matches the patient's real-world anatomy, with both sharing the same spatial coordinates [11]. Figure 2 presents the number of research publications across various surgical procedures in studies related to augmented reality surgical navigation. Notably, AR navigation is applied frequently in surgeries involving organs characterized by minimal motion and deformation, like orthopedic surgery, where tracking is often facilitated through invasive fiducial implantations [12]. However, the field of hepatic surgery poses additional challenges. The liver is a soft, non-rigid organ, that is frequently obscured by surrounding structures. These include the ribs, intestines, and other organs, significantly limiting the surgeon's direct view during the critical image registration process (Figure 3). Despite this, efforts have been made to apply mixed reality techniques to liver surgery.

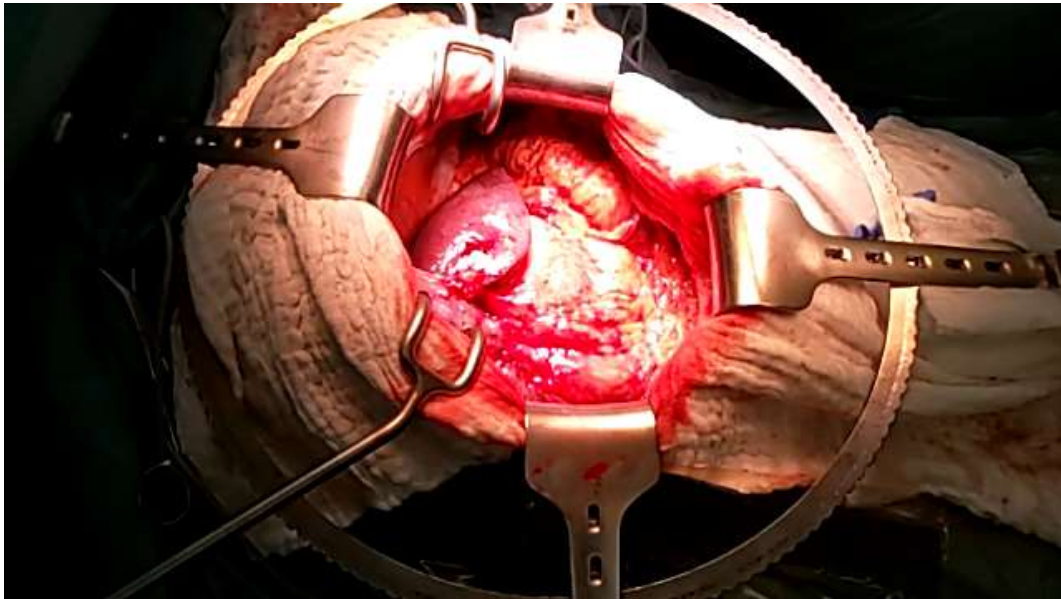


Figure 3. Illustrates a surgeon's view of a partially exposed liver during open liver surgery. Image: Acquisition from the AliAS data study, Lakumed Hospital Landshut-Achdorf. The study was approved by the Ethics Committee of the Technical University of Munich. BfArM study register number: DRKS00032826.

While Golse et al. explored using depth cameras (RGB-D) to access spatial information during open liver surgery, these methods can struggle with significant occlusions caused by surrounding organs. To address these limitations and achieve robust registration in such scenarios, alternative methods that handle occlusions effectively are necessary [4].

Prior to the rise of deep learning, traditional optimization-based methods offered solutions for registration tasks, such as the iterative closest point (ICP) and simultaneous localization and mapping (SLAM) algorithms [13], [14], [15]. Even though these algorithms utilized registration of objects in occluded point clouds, their high computational cost limits their application primarily to sparse point clouds [16]. The unique capabilities of deep learning algorithms to learn specific information from a dataset, reduce hardware and software costs and enhance registration performance [17]. However, deep learning methods face challenges in accurate registration of heavily occluded objects, especially when dealing with deformable structures [10].

Therefore, a second approach focuses on point cloud preprocessing techniques. These techniques play a crucial role in this process, as they aim to reconstruct a complete and accurate representation of the scene or object from incomplete or noisy data to enhance the performance of registration methods [18].

Reconstructing a point cloud poses a significant challenge due to the inherently disordered and unstructured nature of point clouds, which runs counter to the structured information needed for point cloud completion. Therefore, understanding the structural features of the point cloud is essential for achieving a more accurate and comprehensive reconstruction. Early efforts in applying deep learning for point cloud reconstruction borrowed methods used for image reconstruction tasks [19], [20], [21]. However, adapting these 2D techniques to 3D data significantly increased processing requirements due to the additional dimension [22].

PointNet and PointNet++ achieved a breakthrough in the field of point cloud processing by enabling the direct computation of point clouds without the need for transforming them into voxel grids [23], [24]. This method was further applied in other completion techniques, that utilized an encoder-decoder architecture [25], [26].

The current progress in deep learning has significantly improved point cloud completion capabilities. As illustrated in Figure 4, existing point cloud completion network architectures can be broadly categorized into point-based, view-based, convolution-based, graph-based, transformer-based, and generative model-based approaches.

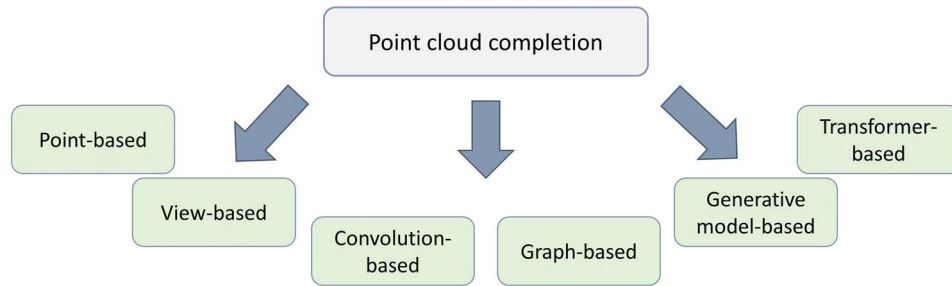


Figure 4. Illustrates different point cloud completion techniques.

Point-based methods treat each point independently, employing multi-layer perceptron’s (MLPs) to model them individually. To capture global features, a symmetric function like max-pooling is often utilized. An early example derived from PointNet was PU-Net, but due to the structural loss brought by multi-layer-perceptron, it is mainly used for upsampling of sparse point clouds [27]. To address the limitations of point-based methods, attention assisted mechanisms have been used to adaptively weight relevant information. Additionally folding methods that reconstruct point clouds by deforming a 2D grid into a 3D surface were implemented [28], [29]. While point-based methods work around the permutations issue, the independent treatment of points misses geometric relations between points and their neighbors [22]. One limitation of most point-based methods is their high computational cost. This stems from working directly with points, which can become computationally expensive, especially for complex objects with many data points.

View-based methods leverage image information to improve the reconstruction process. Zhang et al. proposed a view-guided network that enhances partial point cloud reconstruction [30]. However, this model produced unsatisfactory completion results on real-world data, suggesting a potential limitation to its generalizability. This issue might be addressed by using different datasets for training.

Convolution-based methods typically voxelize the point cloud data, converting it into a volume representation. This approach suffers from high storage requirements, because the grid also contains empty spaces in areas where the object is not present [22]. Additionally, voxelization leads to an irreversible loss of geometrical information. To address these challenges, Xie et al. proposed a method that utilizes differentiable Gridding and Gridding Reverse layers [31]. These layers allow for converting a point cloud directly, without the need for voxelization, thus preserving valuable structural information. While the method enables convolutional networks to work on point clouds, the computational demand grows cubically with the resolution.

Graph-based methods treat every point like vertices of graphs. The proposed method of Wang et al. introduced edge convolution [32]. This method aggregates the edge features associated with edges from each connected vertex. The convolution is applied on a set of k-nearest neighbors and the graphs are dynamically updated, leading to nonlocal information diffusion throughout the point cloud. This method has been further improved by utilizing attention mechanism [33].

Generative-based models like generative adversarial networks (GANs) or variational autoencoders (VAEs) face additional challenges in extending the initial 2D problem to 3D objects. To address these limitations, researchers have explored advanced approaches like multi-view GANs, which can leverage information from multiple viewpoints to create a more comprehensive 3D representation. Additionally, integrating generative models with techniques like reinforced learning and graph convolution has shown promise in tackling complex 3D tasks [34], [35], [36].

Transformer architecture was first proposed for natural language processing [37]. Since then, transformer architectures have shown remarkable success in various fields, including point cloud completion. One such application is the PoinTr-network by Yu et al., which formulates point cloud completion as a set-to-set translation problem [38], [39], [40]. While the transformer showed promising results in the reconstruction of point clouds, the size of the model limits its deployment on devices compared to other methods [22]. Most networks leverage a combination of the techniques shown in Figure 4 with each method offering unique strengths and limitations.

However, deep learning methods require large amounts of annotated datasets. Even though several large 3D-Object datasets like KITTI, ShapeNet, PartNet are publicly available, this is not the case for medical imaging data [41], [42], [43]. The abundance of annotated liver data for registration tasks poses a significant challenge. The DEPOLL dataset, designed to evaluate registration accuracy in liver surgery using augmented reality, serves as an example [44]. Its limitation to a single pig liver with only 13 deformations underscores the critical need for more extensive, human-specific data for registration algorithms. The aim of this thesis is twofold:

1. Generating a dataset for liver point cloud reconstruction in open liver surgery applications.
2. Assessing the suitability of the generated dataset for training a deep learning model that can reconstruct partial liver views during open liver surgery procedures.

3 A theoretical base for point cloud reconstruction

3.1 Point cloud theory

The use of 3D models is becoming increasingly prevalent in daily life. Applications such as autonomous driving and virtual reality rely on advanced processing and analysis of collected 3D data. Three-dimensional objects can be represented in various formats, including voxels, point clouds and polygon meshes, as depicted in Figure 5.

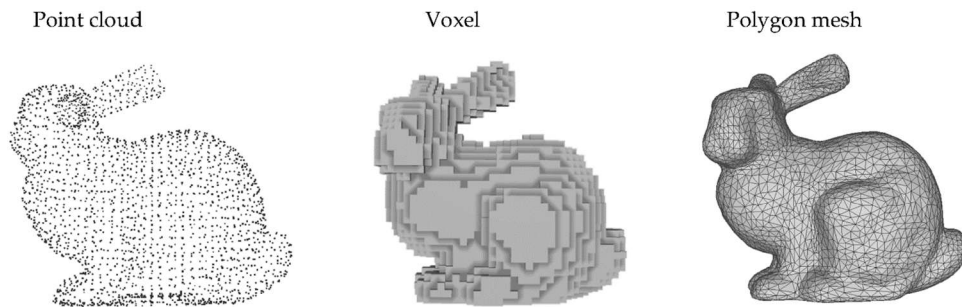


Figure 5. Illustrates different representations of a rabbit as point cloud, voxel grid, and polygon mesh [45].

The rise of reality capture techniques has propelled point clouds to become a crucial data source, ranking as the third most important format after vector maps and imagery [46]. Hence, algorithms must be utilized for the conversion between data types. While CT scans are often stored as voxels, particular attention is given to the conversion into triangular mesh formats [47]. Various algorithms, including convolutional neural networks, can process voxel data to create triangular meshes by encoding the voxel data and decoding a spherical mesh and features into a triangular mesh [48]. However, this work specifically emphasizes the use of marching cubes.

3.2 Marching cubes algorithm

Marching cubes is one of the most popular algorithms for surface reconstruction [49]. Since its development in 1980, many variations have been developed, like marching cubes based on edge growth [50]. The original algorithm mainly works in two steps.

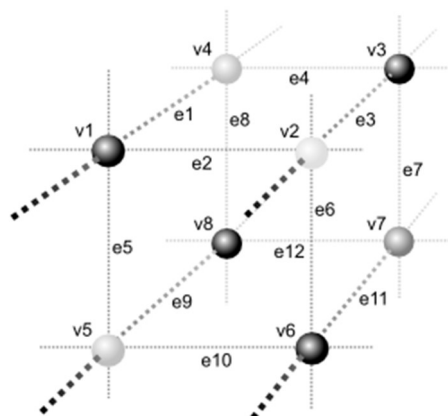


Figure 6. Illustrates a logical cube, a key element in the marching cubes algorithm for 3D surface reconstruction. The algorithm analyzes a grid of data points (voxels) at the cube's vertices. Each voxel in the grid can be assigned a value based on a specific property, such as density or intensity [51].

Logical cubes, used in image processing, are constructed from eight pixels drawn from two neighboring image slices (Figure 6). The marching cubes algorithm analyzes each cube to determine how a surface intersects it, then proceeds to the next cube in the data. Within each cube, vertices above the surface are marked as "inside," and those below are marked as "outside". This helps identify where the surface intersects the cube's edges.

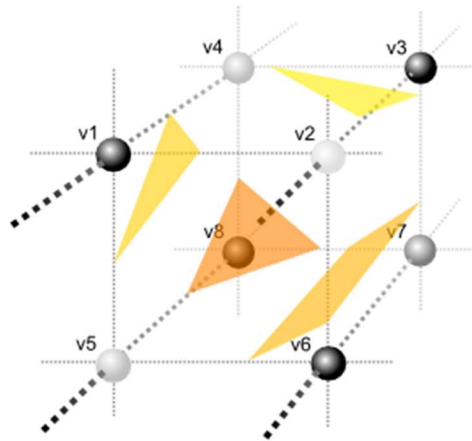


Figure 7. Illustrates example surfaces reconstructed from a logical cube. The figure illustrates how different configurations of vertex values within a cube can result in various triangulated surfaces [52].

This approach allows for determining the surface's topology within each cube, with the exact intersection location determined later. Considering all possible combinations of vertices inside and outside would result in 256 different cases, but by making use of symmetries, this can be reduced to 14 patterns that are usually stored in a look-up table. In the second step, unit normals for each triangle vertex are calculated (Figure 7). This involves estimating the gradient vector at the cube vertices and then linearly interpolating it at the point of intersection.

3.3 Ray casting

Since the HoloLens2 utilizes a Time-of-Flight sensor, initial point clouds obtained directly from these devices tend to be sparse and incomplete due to various factors such as occlusions, reflections, transparency and limitations in device resolution and viewing angles [46]. Data used for training and testing should reflect these properties. One method to create photorealistic pictures of virtual objects is ray casting.

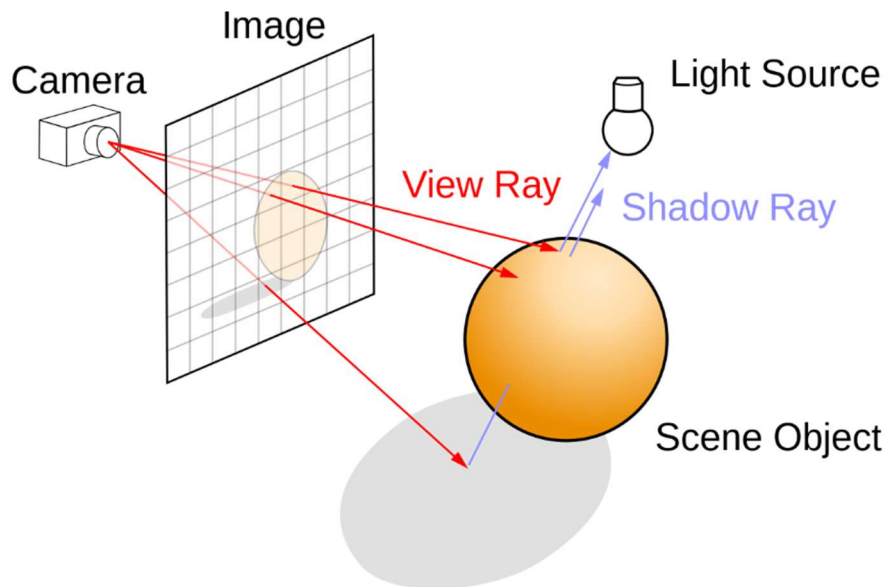


Figure 8. Shows the concept of ray casting in computer graphics. It shows a camera, a light source, a scene object and an image with the projected object [53].

This technique was first described in 1968 and simulates how light interacts with objects in a scene to create a photorealistic image [54]. Since then, ray casting found its way into many applications like the visualization in CAD solid modeling [55], [56]. In computer graphics, ray casting is a technique used to render images by simulating how light travels from a virtual camera to objects in the scene. The system then checks for the closest object each ray intersects. As illustrated in Figure 8, based on the properties of the object at the intersection point, the color and lighting of that pixel are determined. This approach allows for efficient rendering, making it a valuable tool for various applications.

However, it's important to consider limitations of ray casting. One constraint is aliasing, which arises due to the sampling of the scene at discrete points. This effect can be amplified with sparse data, as there is less information to reconstruct smooth surfaces. Filtering can help mitigate aliasing problems [57]. Ray casting simulations also typically neglect complex light interactions like reflections, refractions, and soft shadows. The realistic simulation of reflections can be made possible by calculating secondary beams, but this can significantly increase the computational effort [58].

3.4 Transformer theory

Introduced by Vaswani et al. in 2017, the transformer, revolutionized the field of natural language processing (NLP)[37]. Its core concept is the self-attention mechanism, which allows a model to attend to relevant parts of its own input sequence, rather than relying solely on sequential processing, like RNNs. Since then, the transformer architecture itself has been adopted in many different fields, including computer vision and point cloud completion. This section should give brief introduction to the functionality of the original concept that was proposed in the paper, "Attention is all you need".

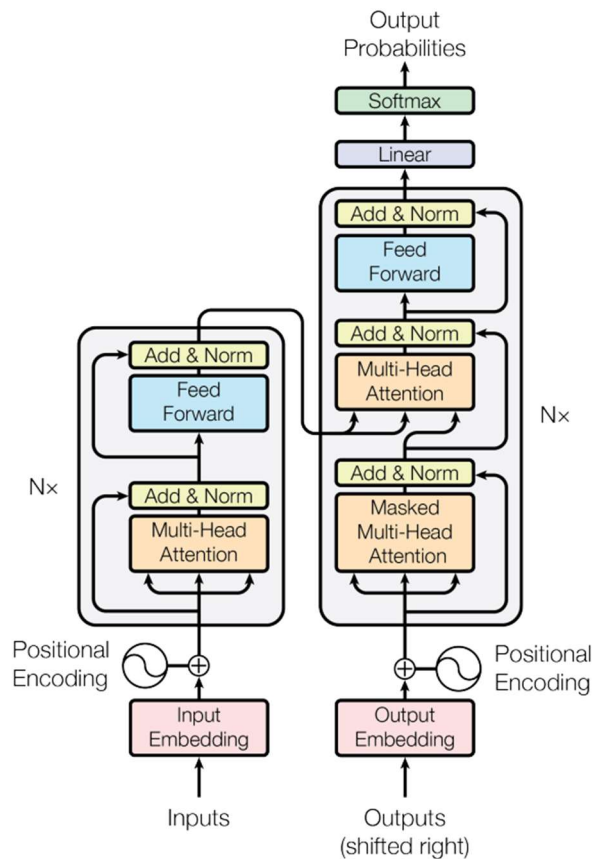


Figure 9. Illustrates the transformer architecture like proposed in "Attention is all you need" utilizing cross a and self-attention [37].

Figure 9 illustrates the proposed model architecture, featuring an encoder on the left and a decoder on the right. These components are connected through a cross-attention layer, where the encoder's output values are passed as queries and keys to a multi-head attention layer within the decoder. The paper proposes an encoder-decoder architecture with six identical layers, each referred to as a "head" by the authors. This architecture allows each head of the transformer to access every value of the input sequence with a reduced dimension, as depicted in Figure 10.

The encoder begins with positional encoding, a crucial step for transformers, as they do not inherently possess a sense of sequence order. To address this, positional encoding involves adding a fixed value to each position of the input sequence. The input is passed through a multi-head-attention layer (Figure 10) that calculates the self-attention for every input value.

The **decoder** also utilizes positional encoding, followed by a self-attention layer. Additionally, for every sequence, cross attention is utilized. The output of the encoder, which consists of the entire input sequence is passed as queries and keys. For every output of the decoder, cross-attention is used, for which the output-value of the encoder is reused as keys and queries to adapt the decoder output to the input values. Subsequently, self-attention in the decoder takes all already generated outputs into account, while the cross-attention takes the user prompt from the encoder into account.

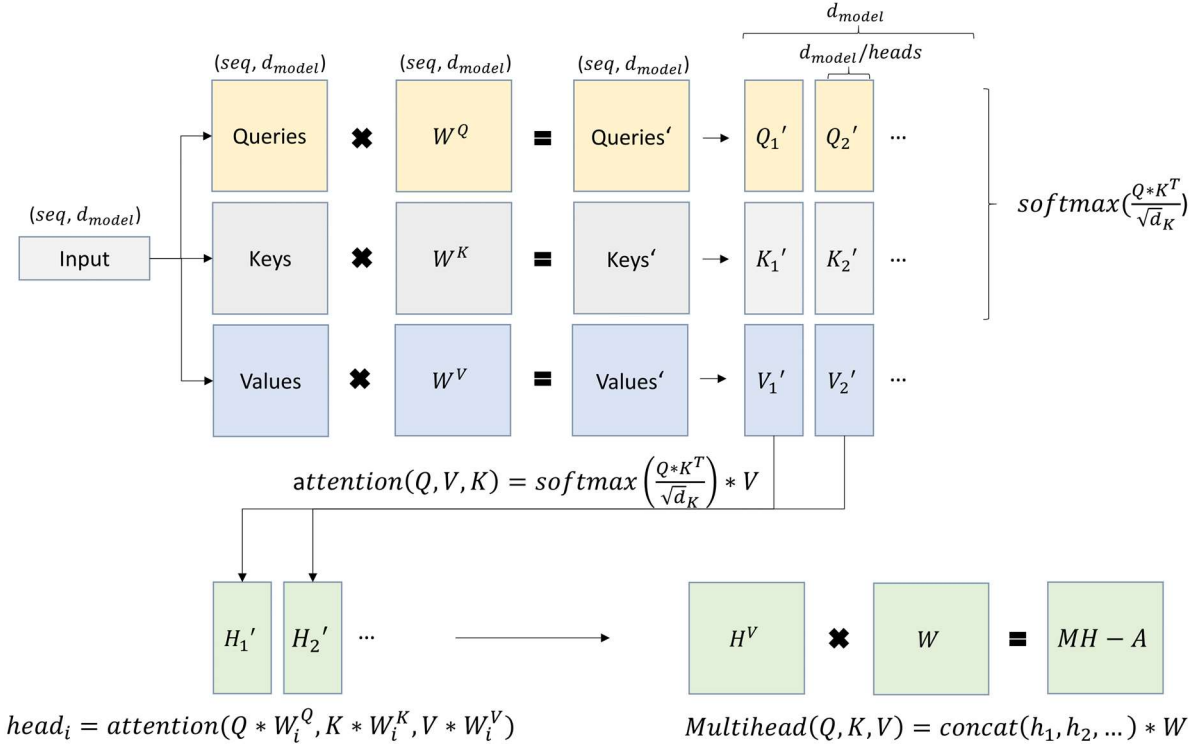


Figure 10. Illustrates a multi-head attention layer, a core component in transformer models used for various natural language processing (NLP) tasks such as machine translation and text summarization. The multi-head attention mechanism allows the model to focus on the most relevant parts of an input sequence for each word or token, enabling it to capture long-range dependencies within the sequence.

3.4.1 Transformer finetuning

Fine-tuning is a powerful technique used to leverage pre-trained deep-learning networks for new tasks. Research in convolutional neural networks (CNNs) suggests that lower layers capture general representations, while higher layers learn more task-specific features [59]. Similar observations were made in natural language processing with the BERT model. Clark et al. demonstrated that the bottom layers of BERT attend broadly, while the top layers focus on capturing linguistic syntax [60]. BERT, introduced by Devlin et al., is a pre-trained transformer model trained on a massive unlabeled dataset. It is often fine-tuned on labeled datasets for supervised learning tasks, significantly reducing the computational cost compared to training from scratch. Studies have shown that fine-tuning only a small portion of the final layers can achieve high performance, with some studies indicating that fine-tuning as few as a quarter of the final layers can still achieve 90 % of the original quality [61].

Several fine-tuning approaches exist for transformers. **Selective methods**, arguably one of the earliest examples, involve fine-tuning only a few top layers of the network [62]. **Additive methods** augment the pre-trained model with additional parameters or layers, while only optimizing the newly added parameters during training [63]. Although these models introduce additional parameters to the network, significant improvements in training time and memory efficiency are made by reducing the size of the gradients. By saving memory through frozen model parameters, much larger networks can be trained. **Reparameterization-based methods** utilize low-rank representations to minimize the number of trainable parameters. Aghajanyan et al. demonstrated that the size of the subspace that needs adaptation is smaller for larger models [64].

3.5 PoinTr: Diverse point cloud completion with geometry-aware transformers

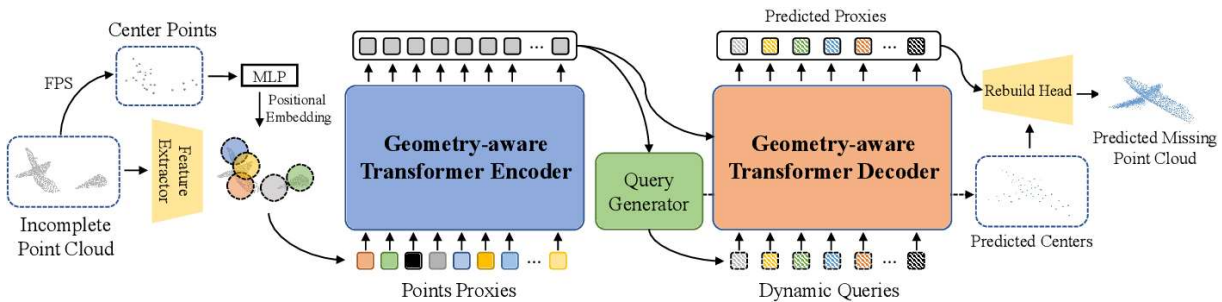


Figure 11. PoinTr pipeline: A partial point cloud is downsampled to extract local features around center points. Positional encoding is added before feeding these features to a transformer encoder-decoder for point proxy prediction. Finally, a two-stage refinement process with an MLP and FoldingNet completes the point cloud [38].

Inspired by the huge success of transformers in language translation, Yu et al. adapted the point cloud completion task as a transformer-based set-to-set translation [38]. As illustrated in Figure 11, the network architecture is split into five main parts. Overall, an **encoder-decoder** architecture is utilized to convert the completion into a translation problem. **Point proxies** that represent the local point cloud as a set of feature vectors are extracted by first localizing a fixed number of center points through furthest point sampling. With the use of dynamic graph CNN, features of the center points are extracted. A point proxy combines features around a center point with its location information. This combined data is then passed to the encoder. A **geometry-aware transformer (encoder)**, that

facilitates a kNN-attention layer, captures the geometric relation in the point cloud. These feature vectors are passed to a **query generator** that dynamically creates queries by summarizing the encoder output with a linear projection to a higher dimension, followed by a max pooling operation to reshape the features as coordinates. The features are then concatenated with the coordinates and passed through a multilayer perceptron (MLP) to produce query embeddings for the encoder. A **geometry-aware transformer (decoder)** utilizes self- and cross-attention to predict point proxies of the missing part while taking into account local features in addition to global information and relationship between queries and outputs of the encoder. Lastly, a **multiscale point cloud generation** is used to convert the generated point proxies into the missing point cloud. To recover the detailed shapes around the center points of the predicted proxies, Folding Net is fed with the features of each point proxy [29].

During **training**, PoinTr receives a partial point cloud as input. Unlike transformers for natural language processing (NLP), which can incorporate previous predictions, PoinTr utilizes a refinement process. This process involves iteratively predicting new points based on the current partial cloud. These predicted points are evaluated based on their coherence with existing points and their contribution to the overall reconstruction compared to the ground truth. While some points might be rejected, they still contribute to the loss function for weight updates. The chamfer distance is adapted as the loss function to calculate the loss separately for the predicted centers and the completed dense point cloud. The metric measures the similarity between two sets of points by calculating the sum of the squared distances between the closest points in each set.

4 Method

This work investigates the potential of incorporating a PoinTr network into the pre-processing pipeline for partial liver point clouds. The goal is to improve the performance of downstream registration tasks. The influence of various parameters, including noise, point cloud density, and viewpoint, is analyzed. Therefore, this task can be separated into two main stages:

Data generation: This stage focuses on creating a training dataset suitable for reconstructing liver point clouds from partial data, similar to scenarios encountered in AR-guided open liver surgery applications. A key aspect of this process involves incorporating variations in noise levels.

Network training and evaluation: The generated data is then used to train and validate a point cloud completion model. The model's performance is evaluated under varying parameters that could affect completion accuracy.

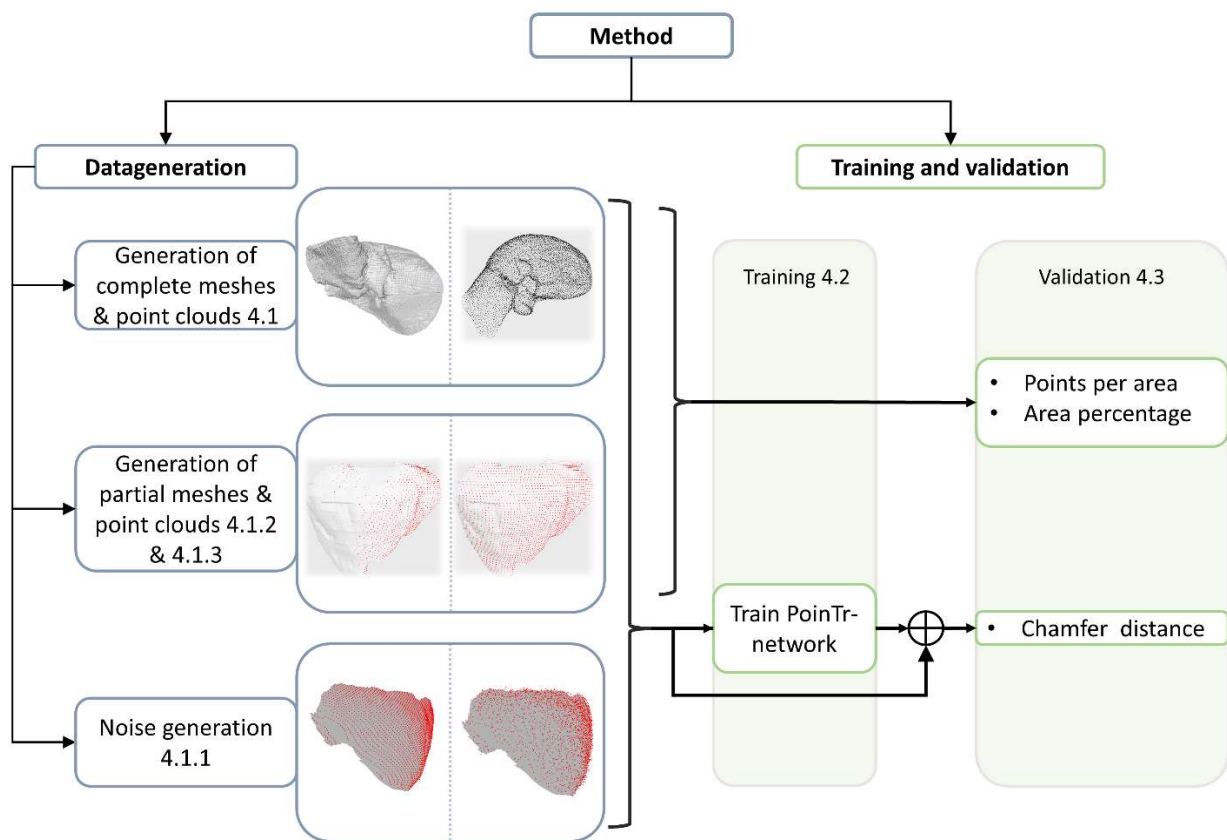


Figure 12. Illustrates the methodological framework employed in this work. The left side (blue) depicts the data generation process, where high-fidelity 3D object models (meshes) and extracted point clouds are created. Subsequently, partial meshes and point clouds with added noise were generated to simulate real-world scenarios. The right side (green boxes) showcases the training and validation process for a PoinTr network and the generated data.

A comprehensive literature review identified the PoinTr network as a suitable choice for point cloud completion tasks due to its transformer-based architecture [38]. Two PoinTr models are trained with different noise levels and their performance is evaluated. Additionally, the test dataset is analyzed to understand the impact of viewing angle, noise level and point cloud density.

To this end, the methodology covers five key steps. This initial step focuses on constructing a pipeline specifically designed to generate suitable data for training the point cloud completion model (4.1). A crucial aspect of this pipeline involves incorporating variations in noise levels (4.1.1). The second step

centers around constructing the training dataset, that will be used to train the PoinTr network (4.1.2.). Third, a test dataset is constructed that mirrors real-world applications using Hololens2 time-of-flight (ToF) sensor (4.1.3). Fourth, a description of the specific modifications made to the PoinTr network to tailor it for this particular application (4.2). The final step involves providing detailed descriptions of the experiments conducted. To enhance readability and ensure clarity, a traceability matrix is included (Table 4). This matrix links the research questions to the corresponding experiments, allowing for easier comprehension of how each experiment contributes to addressing the overall research objectives (4.3).

4.1 Data generation

This section details the process of generating a dataset suitable for training a point cloud completion model, for reconstructing partial liver point clouds. As mentioned earlier, publicly available 3D data of human organs is scarce.

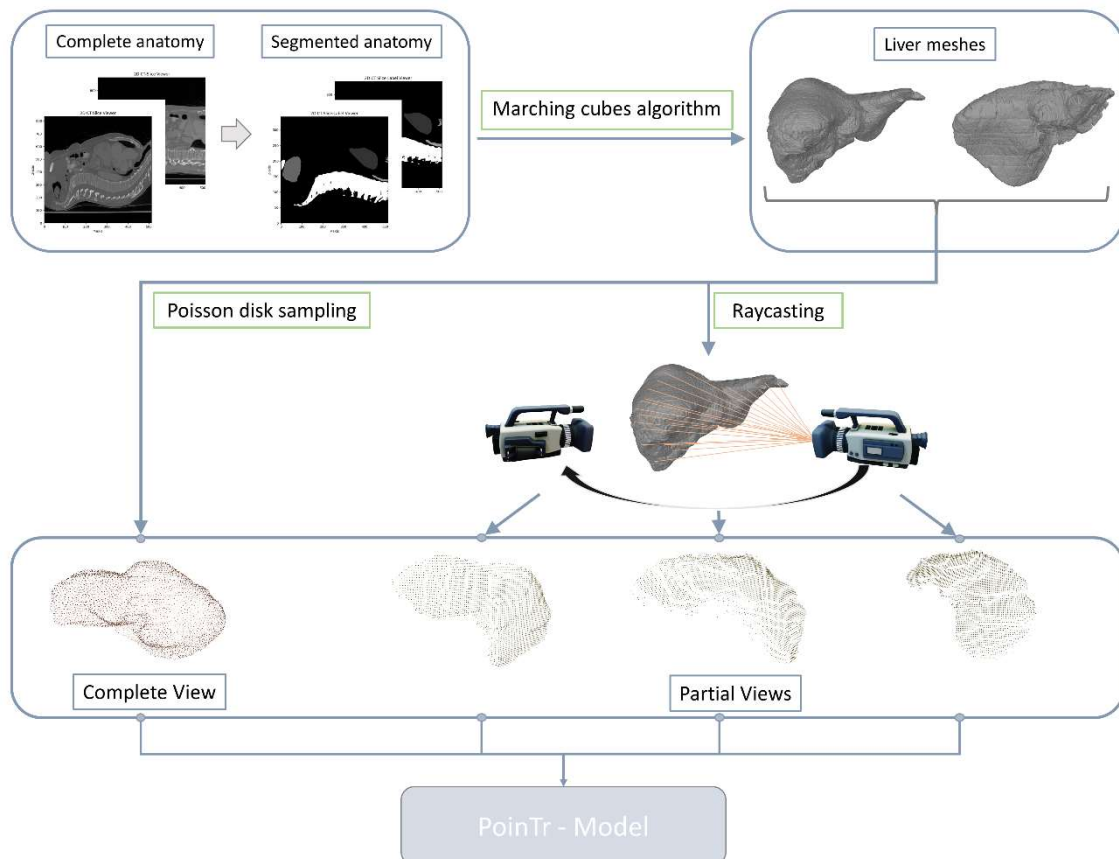


Figure 13. Illustrates the data generation process. Top left: Annotated CT data. Top right: Generation of meshes through marching cubes algorithm. Bottom left: Generation of complete point cloud through poisson disk sampling. Bottom right: Generation of partial views through raycasting.

While there is no publicly available data specifically for liver point cloud completion, there are public datasets of anonymized patient CT data, such as CT-ORG and liver tumor segmentation (LiTS) datasets [65], [66]. LiTS offers a valuable advantage: it provides comprehensive segmentation labels for all datasets, including segmentations for background tissue, liver and cancer. The LiTS dataset consists of

Guido Reinfurt 20

131 annotated CT datasets. This data is stored in a unified NIFTI format and includes pre- and post-therapy abdominal CT scans.

It is important to note that the CT scans within the LiTS dataset are acquired using different CT scanners, resulting in varying image resolutions (ranging from 0.56 mm to 1.00 mm) and slice thicknesses (between 0.45 mm and 6 mm). As shown in Figure 13, a selection process is applied to the voxels within the CT data. Only voxels that carry specific labels are included in this process. In this case, voxel with label “liver” or “tumor” are selected. These selected voxels, along with their corresponding voxel size information, are then fed into the marching cubes algorithm to create a watertight mesh representing the liver and potential tumors.

The mesh generated from the segmented LiTS data serves as the foundation for creating both complete and partial point clouds. For the generation of complete point clouds, the Poisson-disc sampling is utilized to generate uniformly distributed point clouds with 7500 points. This value is empirically tested to reflect the amount of data points generated by the Hololense2 depth sensor when scanning a full liver phantom.

Several techniques exist for creating partial point clouds of objects. One approach, as proposed by Pfeiffer et al., involves cropping a subvolume around the object's bounding box from the original point cloud [67]. This work employs the ray casting technique to simulate real-world views captured by a Hololens2 sensor. To enable fast computation, a virtual ray casting scene must be created, where the mesh is transformed into a tensor format. Single rays can be calculated by providing a start point and a direction as a 6-dimensional vector. A helper function is used that creates a pinhole camera in the virtual space. The parameters of the function are listed in Table 1.

Table 1. Details of virtual pinhole camera parameters for ray casting scene

| Parameter | Values |
|-----------------------------------|--|
| Angel of view (fov_deg) | 120° |
| Camera view of direction (center) | (x_1, y_1, z_1) |
| Camera position (eye) | $(x_2, y_2, z_2) \cdot cameradistance + (x_1, y_1, z_1)$ |
| Vector in upward direction (up) | [0, 1, 0] |
| Rays in x-direction (width_px) | 1024 |
| Ray in y-direction (height_px) | 1024 |

While the parameters "Angle of view" and “Rays in x-direction” and “Rays in y-direction” are directly taken from the Hololens2 ToF sensor datasheet, other virtual pinhole camera parameters require further determination [68]. The vector specifying the up-position of the camera is empirically evaluated within the liver mesh and set as a unit vector pointing in the positive y-direction.

The “camera view of direction (center)” parameter, denoted in Equation 1, determines a 3D vector specifying the direction in which the virtual camera is pointing. This vector is calculated based on the center of mass of the mesh, which is computed by finding the mean value of all vertex coordinates within the mesh.

$$(x_1, y_1, z_1) = \frac{1}{n} \sum_{i=1}^n (x_{vertices, i}, y_{vertices, i}, z_{vertices, i})$$

Equation 1

$$(x_1, y_1, z_1) = \textit{centerpoint}$$

$$(x_2, y_2, z_2) = \textit{camera position}$$

The “camera position (eye)” consists of the camera position as a unit vector (x_2, y_2, z_2) , that is multiplied by the camera distance. Then the center of the mesh (x_1, y_1, z_1) is added to compensate for the translation of the liver.

4.1.1 Noise generation

Real-world sensors, including the time-of-flight (ToF) sensor in the Hololens2, are subject to limitations such as noise during data acquisition. The Hololens2 datasheet specifies a depth uncertainty of 0.2 % of the camera distance [68]. To incorporate this, Gaussian noise is added to the length of each ray during the virtual scene generation. This noise is introduced in the direction of the ray itself. The standard deviation of the added Gaussian noise is set to the specified depth uncertainty from the Hololens2 datasheet, as illustrated in Figure 14.

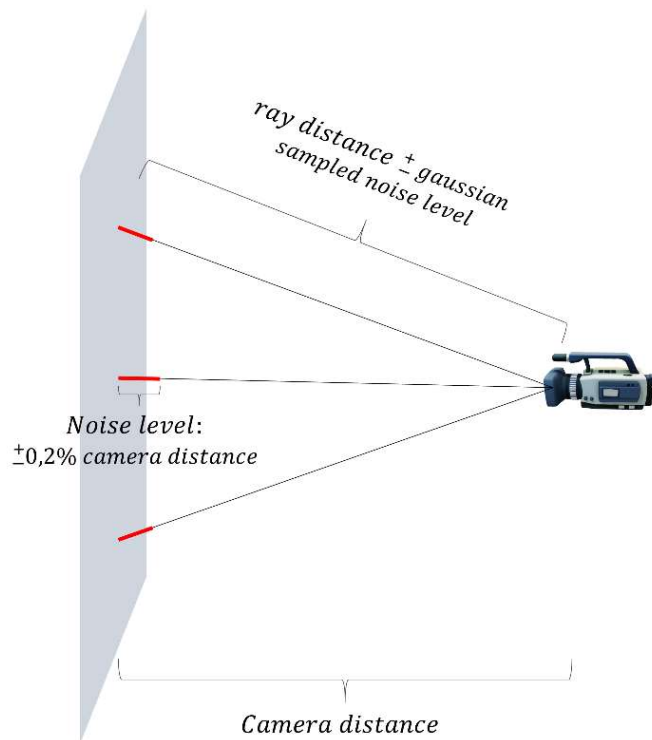


Figure 14. Illustrates the noise generation process. For each point where a ray intersects an object, a Gaussian-sampled value with a standard deviation of 0.2 %, of the camera distance along the ray direction is added to the intersection point.

To create a comprehensive training and testing dataset, two approaches were employed for calculating the camera position. These are further shown for the training dataset in 4.1.2 and for the testing dataset in 4.1.3. For both the training and testing data, two versions are generated with and without noise added to the partial point clouds as illustrated in Figure 15.

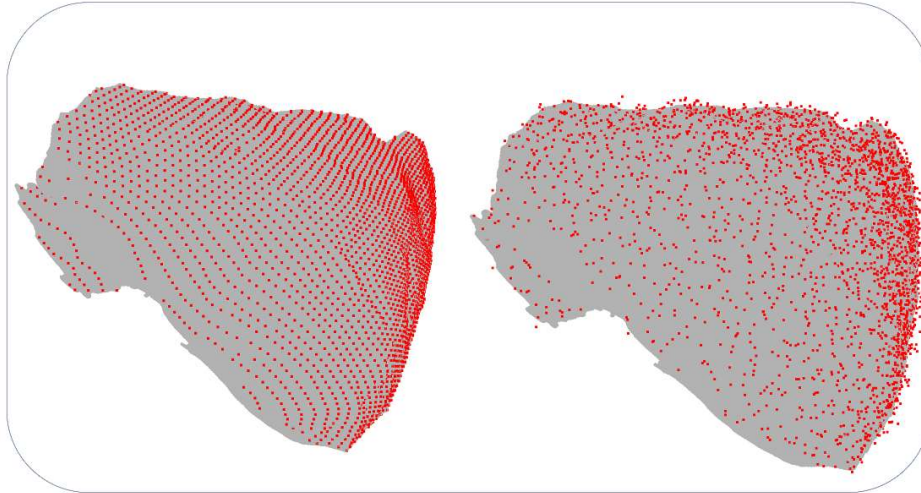


Figure 15. Ray casting results without noise (left) and with noise added (right).

4.1.2 Training data generation

The training process aims to equip the network with a deep understanding of the liver's 3D structure. As illustrated in Figure 17, golden angle sampling typically used to distribute an arbitrary number of points with maximum uniformity on a circle, is leveraged here to generate 250 different views of the liver. This technique ensures that viewpoints encompass the entire 3D structure effectively. However, a direct adaptation to 3D would not work. To extend this into a three-dimensional world, in each step, the golden angle is multiplied by an index variable “ i ” to create an azimuth angle (β). Simultaneously, another angle iterates uniformly across π , representing the polar distance angle. This angle controls the distance from a reference point. The combination of azimuth and pole distance angle is then calculated into a unit vector representing the camera angle.

For each liver, a suitable camera distance must be calculated to ensure a full view of the liver. To achieve this, the maximum distance between every node in the mesh is calculated and the maximum value selected. This value represents the maximum point distance within the mesh and is used to determine the appropriate camera distance as shown in Figure 16.

To investigate the impact of real-world sensor noise on the network's performance, the dataset was generated in two versions: with and without noise added to the partial point cloud generation. Training the network on both noisy and clean versions of the partial point clouds allows for the assessment of how exposure to realistic variations affects the reconstruction process. This has the potential to improve the network's overall robustness.

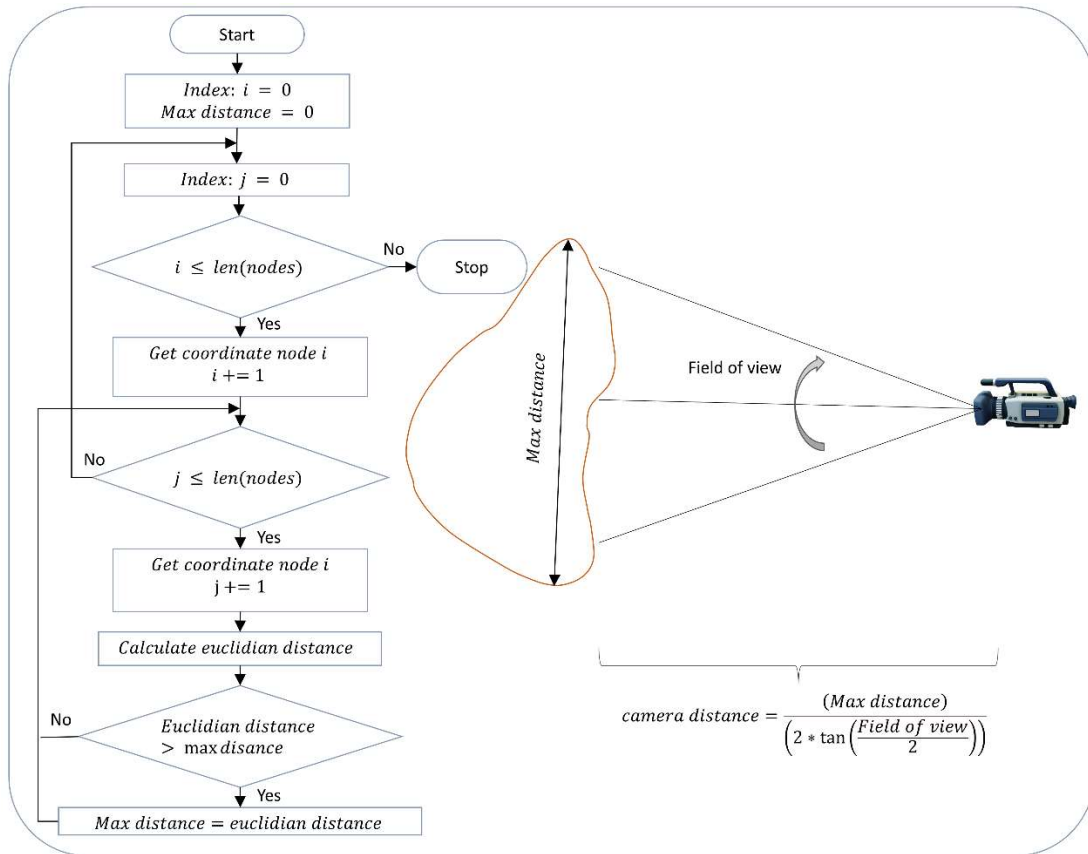


Figure 16. Calculating camera distance for full liver coverage. Left side: Calculating of maximum mesh distance, Right side: Camera distance calculation based on maximum mesh distance.

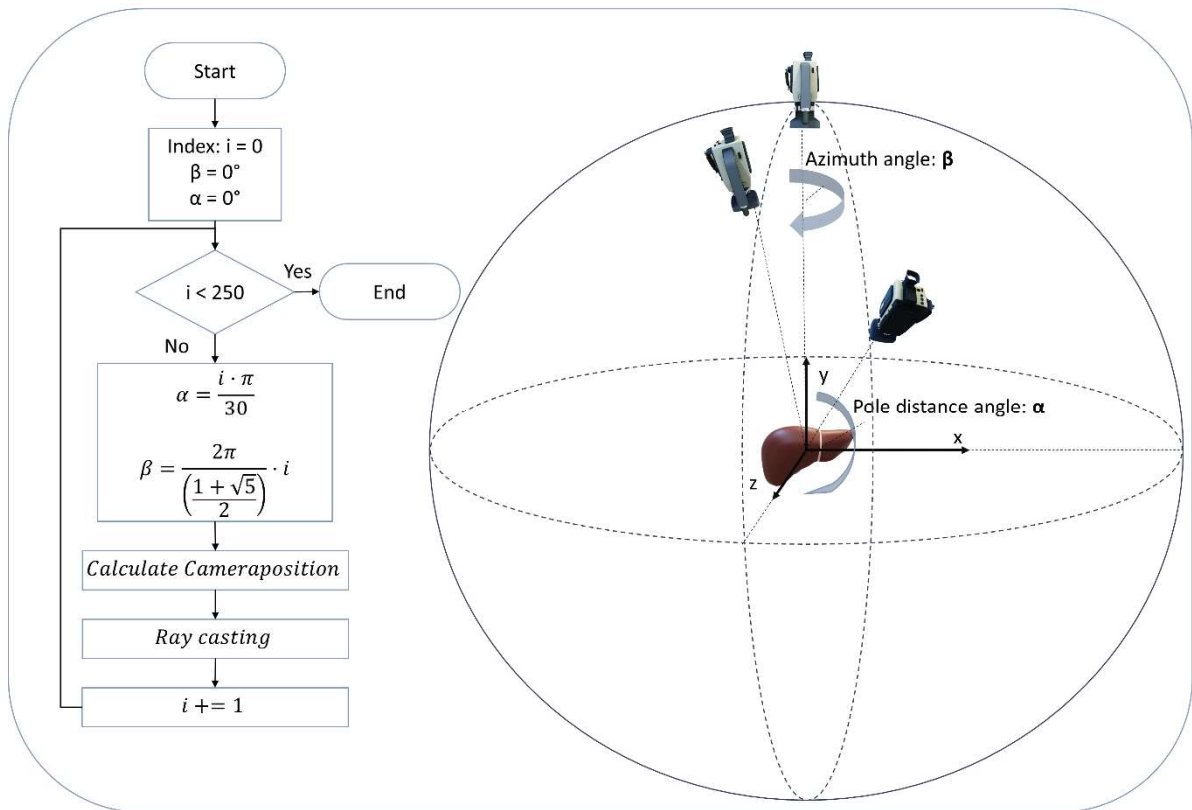


Figure 17. Training data generation. The flowchart on the left illustrates how camera positions are generated in each step by varying two angles and utilizing golden angle sampling. The angles used for this are azimuth and pole distance angle as depicted on the right.

4.1.3 Test data generation

The testing dataset is designed to simulate data captured by HoloLens 2 during open liver surgery, particularly focusing on procedures like right or left hepatectomies. Unlike the training data, which included all possible views, the test data focuses on three key influencing factors specifically relevant to this surgical context as shown in Figure 18:

- Camera distance
- Camera view point
- Noise level

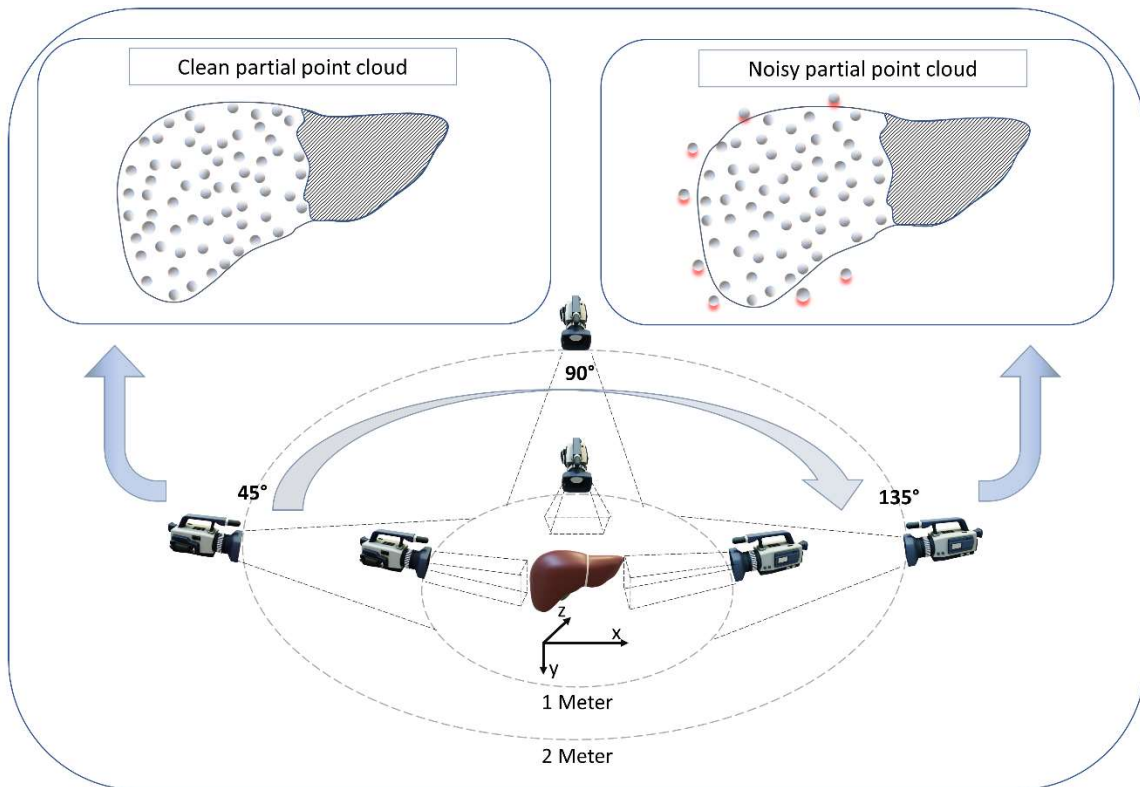


Figure 18. Test data generation. The figure shows three different views to reflect different surgeon views in open liver surgery.

To assess the network's performance under realistic surgical conditions, the testing dataset incorporates limitations similar to those encountered during surgery. The camera distance is set to 1 m and 2 m to effectively limit the number of points available in the partial point cloud. This simulates situations where the surgeon might be working at a distance, leading to fewer points captured by the depth sensor. For testing the dependencies on viewpoints, the virtual camera is rotated by 45°, 90° and 135° around the z-Axis as shown in Figure 19. Also, noise is introduced to the test data according to the method in 4.1.1.



Figure 19. Partial point clouds generated through raycasting. For a better understanding of the process the complete mesh is also displayed.

4.2 Training of PoinTr

The network is trained on the generated training datasets including 103 complete liver point clouds with associated partial point clouds. In this case, an 80/20 split was used, allocating 80 % of the data for training and 20 % for evaluation. Originally, PoinTr was trained on big datasets like ShapeNet55 where 55 different classes with about 51,300 unique 3D-models are included [42]. The generated dataset only consists of 131 3D-objects with the associated 250 partial views for each liver. Due to the limited size and specific focus of the dataset compared to ShapeNet55, directly training PoinTr from scratch might be inefficient. Consequently, the approach chosen here involves fine-tuning (3.4.1) a pre-trained PoinTr model on this focused dataset. The authors of PoinTr offer several pre-trained models on different datasets (ShapeNet55, ShapeNet34, PCN, and KITTI [26], [41], [42]). The model trained on ShapeNet55 achieved the best performance, with a Chamfer distance of $1.09e-3$ compared to the others (ShapeNet34: $2.05e-3$, PCN-dataset: $7.26e-3$). Lower Chamfer distance values (in meter) indicate better reconstruction accuracy. Given its superior performance, the ShapeNet55-trained model was chosen as the baseline for the fine-tuning process. Therefore, the pretrained model was fine-tuned on the created training data. Two models were generated, trained by the original and the noisy data for 250 epochs, while the best checkpoints, evaluated on the validation data during the training, is utilized for further testing.

While PoinTr was designed for complete point clouds like those in ShapeNet55, it included a function to generate partial views by randomly selecting points from the ground truth data. However, this approach would not be suitable for training on the customized dataset. As shown in Figure 19, a new function specifically tailored for the dataset is implemented. This function loads a corresponding partial view from the training data for each ground truth point cloud. The parameters used for training the network are listed in Table 2.

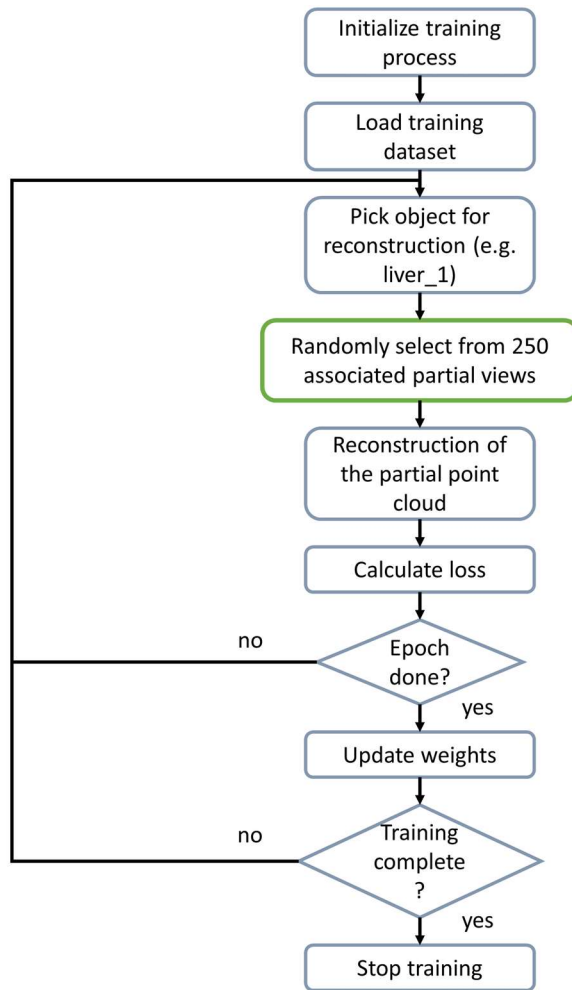


Figure 19. Flow chart of PoinTr training process. This flowchart illustrates the PoinTr training process. The blue boxes represent the standard training pipeline, while the green box highlight adaptations made specifically for training with ray-casted data.

Table 2. Hyperparameters of the PoinTr network used in training process and reconstruction.

| Optimizer | | Scheduler | | BNM-Scheduler | | Model | |
|----------------|--------|----------------------|----------|---------------|--------|------------------------|--------|
| Type: | Adam | Type: | LambdaLR | Type: | Lambda | Name: | PoinTr |
| Learning rate: | 0.0005 | Decay step: | 21 | Decay step: | 21 | Number of predictions: | 5625 |
| weight decay: | 0.0005 | Learning rate decay: | 0.76 | BN-decay: | 0.5 | Number of queries: | 96 |
| | | Lowest decay: | 0.02 | BN-momentum: | 0.9 | Knn-Layers: | 1 |
| | | | | Lowest decay: | 0.01 | Trans-dimensions: | 384 |
| | | | | | | Batch size: | 96 |

4.3 Description of experiments

Three experiments are carried out to answer the research question. The initial experiment (Test data description: 4.3.1) evaluates the quality of the test data itself. The second experiment (Comparison of model trained on noisy data to model trained on clean data: 4.3.2) focuses on evaluating the robustness of the reconstruction performance. In the third experiment both trained models are evaluated on the entire test dataset to determine which one yields superior results. The final stage (Comparison of different test data categories on model trained on noise-free and model trained on noisy data: 4.3.3) employs a Design of Experiments (DoE) approach to compare the influence factors camera distance, field of view and noise on the reconstruction performance.

4.3.1 Experiment 1. Test data description

This stage is about creating simulated depth sensor data using raycasting methods like shown in 4.1.3. Two metrics were used to evaluate the quality of the ray casted data:

- Percentage Area coverage: This metric quantifies the portion of the object's surface area captured in the partial mesh compared to the complete mesh. A higher percentage indicates a more comprehensive representation of the object, potentially providing more global information for the neural network during reconstruction.
- Points per visible Area: This metric measures the density of the point cloud within the visible region of the object. A higher value indicates a denser point cloud, potentially providing more local information for the neural network during reconstruction.

Both values combined should provide a comprehensive overview of the information included in the partial point clouds for each category.

4.3.2 Experiment 2. Comparison of model trained on noisy data to model trained on clean data

Two separate neural network models were trained:

- Model A: Trained on noise-free ray casted data.
- Model B: Trained on ray casted data with simulated noise.

Both models are trained using data created according to 4.1.2. This initial evaluation focuses on comparing the performance of Model A and Model B on the entire test dataset generated according to section 4.1.3. Reconstruction quality is assessed using the chamfer distance metric. A paired t-test is used to statistically compare the chamfer distance values obtained from both models at each category across the test dataset.

4.3.3 Experiment 3. Comparison of different test data categories on Model A and Model B

Both models are subjected to further testing using a design of experiment (DoE) approach. The DoE involves systematically varying factors such as:

- Viewing angles
- Noise levels
- Camera distances

By systematically varying these factors within the DoE framework, as shown in Table 3, the main effects of each factor on the reconstruction quality can effectively be calculated. The chamfer distance metric is applied to quantify the reconstruction accuracy and completeness under different DoE conditions.

Table 3. Full factorial experiment design table. This table presents a full factorial design experiment, outlining all combinations of tested parameters to comprehensively analyze their potential influence on the reconstruction process.

| Test | Angle | Distance [m] | Noise |
|------|-------|--------------|-------|
| 1 | 45° | 1 | + |
| 2 | 45° | 1 | - |
| 3 | 90° | 1 | + |
| 4 | 90° | 1 | - |
| 5 | 135° | 1 | + |
| 6 | 135° | 1 | - |
| 7 | 45° | 2 | + |
| 8 | 45° | 2 | - |
| 9 | 90° | 2 | + |
| 10 | 90° | 2 | - |
| 11 | 135° | 2 | + |
| 12 | 135° | 2 | - |

Table 4. Traceability of the research question to related experiments and results.

| Traceability matrix | | | |
|---|--|----------|------------------------|
| Subject of experiment: | Experiment description: | Results: | N-samples (test data): |
| Test data with categories: <ul style="list-style-type: none"> • Camera distance • Noise-level • Viewpoints | Validation of test data on metrics: Area percentage and points per visible area. | 5.1 | 126 |
| Two PoinTr-models trained on clean and noisy data. | Validation of PoinTr performance on test data using chamfer distance. | 5.2 | 126 |
| Test data with categories: <ul style="list-style-type: none"> • Camera distance • Noise-level • Viewpoints | Validation of the parameter influence on the reconstruction results using a design of experiment approach. | 5.3 | 126 |

4.4 Materials

The used deep learning network PoinTr, which is described under 3.5, is publicly available [69]. The source code has been modified to fit the specific needs for using a dataset generated through raycasting. For both training and testing of the PoinTr framework and the created dataset, the used materials are listed in Table 5.

Table 5. Overview of materials used for both training and test of the PoinTr framework.

| | Materials: |
|--------------------------------------|---|
| Computer used for training | CPU: Intel Core i5-13600KF 3.5GHz RAM: 32 GB GPU: NVIDIA GeForce RTX 4070 OS: Windows 10 Enterprise Version 22H |
| Programming language | Python 3.9.18 |
| IDE within anaconda framework | PyCharm Professional 2023.2.5 |
| Key packages used in the code | tensorboard 2.15.1 tensorflow 2.15.0 torch 2.1.2 + cu118 torchaudio 2.1.2 +cu118 torchvision 0.16.2+cu118 open3d 0.17.0 timm 0.9.12 |
| Software tool for imaging | Meshlab 2023.12 |

5 Results

5.1 Test data description

The first experiment explores the influence of two key parameters on partial point cloud generation: percentage of visible area (distribution visualized in Figure 20) and points per visible area (plotted in Figure 21), where a distribution of these values for all categories of the test dataset is plotted. Table 6 provides detailed mean and standard deviation values for both parameters across all categories. Figure 22, showcases a partial point cloud generated from the three characteristic viewpoints, to illustrate the impact of varying visible areas and point densities on the captured information.

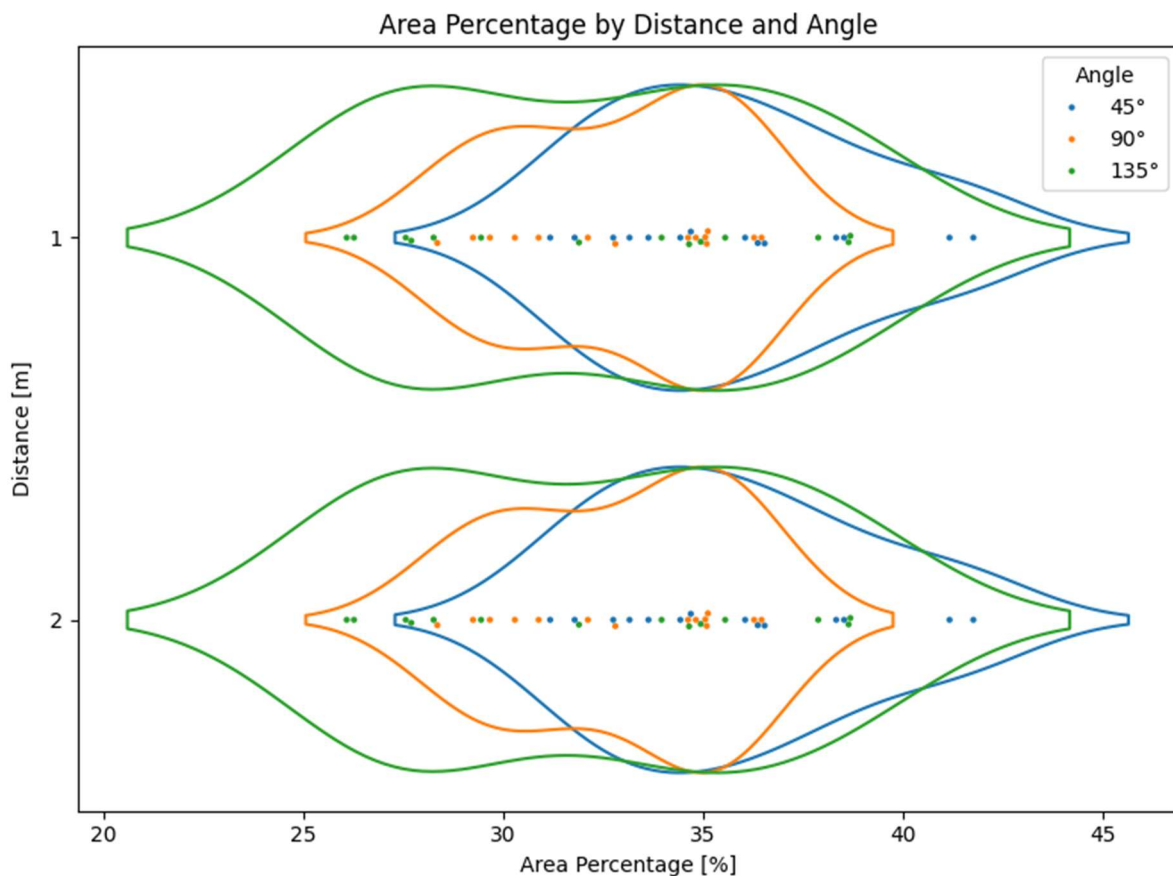


Figure 20. Illustrates area percentage by distance and angle. This figure illustrates the percentage of visible area with respect to the complete object. The distribution of these values is shown for the two camera distances 1 m (at the top) and 2 m (at the bottom) for the camera positions at 45° (blue), 90° (orange) and 135° (green).

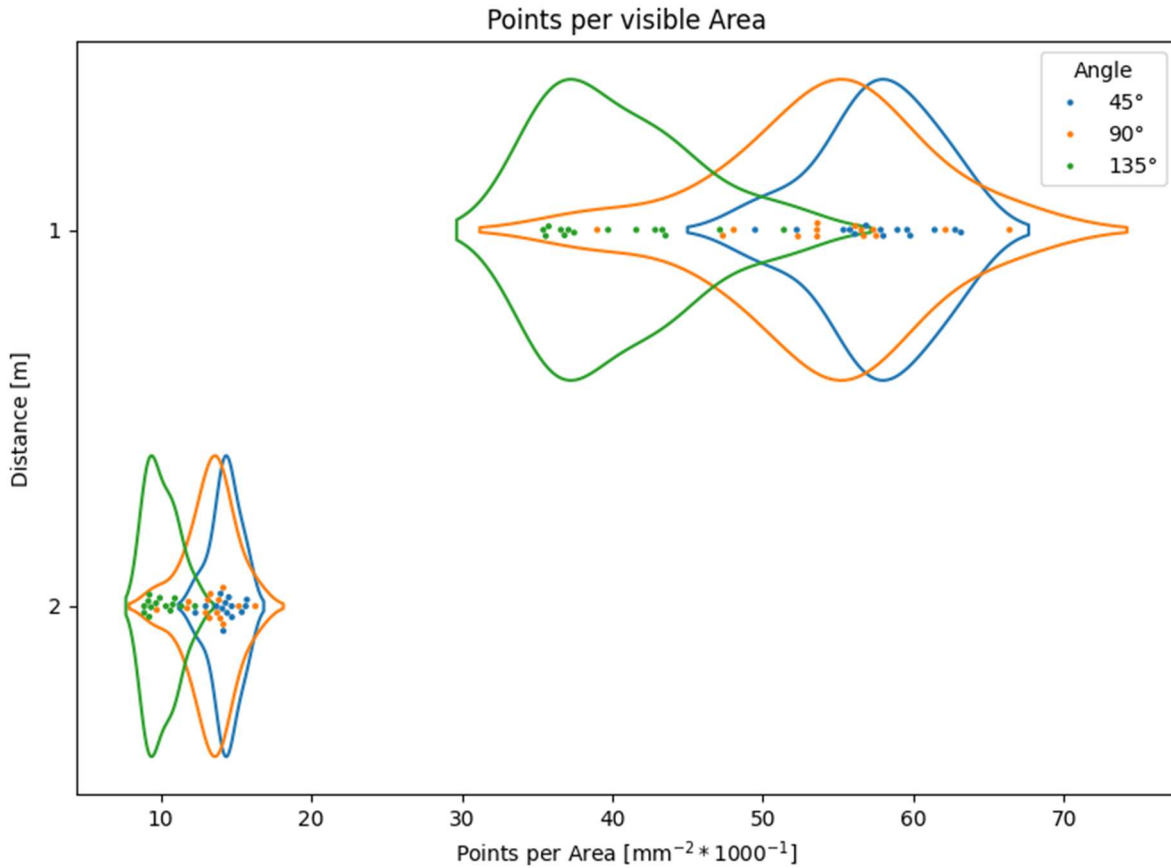


Figure 21. Illustrates the number of points in each partial point cloud relative to the visible area of the associated partial mesh. The distribution of these values is shown for the two camera-distances: 1 m (at the top) and 2 m (at the bottom) for the camera positions at 45° (blue), 90° (orange), and 135° (green).

Table 6. Lists mean and standard deviation for area in percentage and points per area of all test data categories.

| Category | N-samples | Points per Area [points/(mm ² x1000)] | | Area Percentage [%] | |
|----------|-----------|---|----------|------------------------|----------|
| | | Mean | Standard | Mean | Standard |
| 1m, 45° | 14 | 57.6619 | 3.8075 | 35.7364 | 3.2852 |
| 1m, 90° | 14 | 54.3066 | 6.6102 | 32.9139 | 2.7858 |
| 1m, 135° | 14 | 40.3023 | 4.8770 | 32.2509 | 4.6462 |
| 2m, 45° | 14 | 14.3290 | 0.9471 | 35.7364 | 3.2852 |
| 2m, 90° | 14 | 13.3816 | 1.5716 | 32.9139 | 2.7858 |
| 2m, 135° | 14 | 10.0569 | 1.0314 | 32.2509 | 4.6462 |

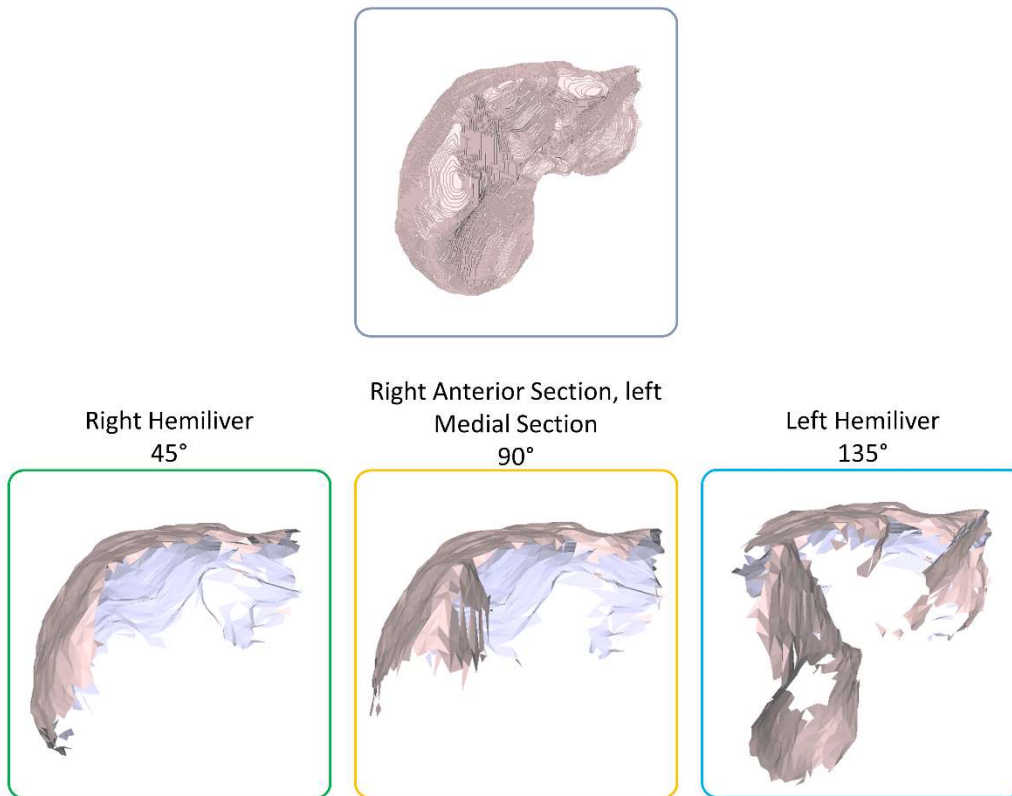


Figure 22. Illustrates the complete mesh (top) and the corresponding partial views generated from different viewpoints (45°, 90°, and 135°, from left to right).

Results “Test data validation” description:

Experiment “Test data validation” analyzed point cloud density across different categories. Figure 20 shows minimal variations in the mean area percentage across viewpoints. While the deviations from the mean increase slightly from 90° and 45° to 135°, the overall changes remain minor. Notably, the figure suggests no significant difference in area percentage based on camera distance.

Figure 21 illustrates how points per area vary across categories for each viewpoint. Interestingly, the value consistently increases from the 135° viewpoint to the 45° viewpoint for both 1 m and 2 m camera distances. Additionally, the figure suggests that partial point clouds generated from a 2 m camera distance have significantly fewer points per area, compared to those generated from a 1 m distance.

Table 6 summarizes the results from these graphs and adds for each metric and category the standard deviation. The table reveals small standard deviations for area percentage, ranging from +/- 2.8 % for the 90° viewpoint to 4.6 % for the 135° viewpoint. The points per area variations tend to be the lowest at the 135° viewpoint and highest at the 45° viewpoint.

5.2 Comparison of model trained on noisy data to model trained on clean data

Following the exploration of partial point cloud properties, this section investigates the performance of the model trained on clean and noisy data. Figure 23 plots the model's performance for different camera distances and noise levels within the test data. By comparing the clean and noisy data plots within each figure, the impact of training data noise on the model's reconstruction accuracy at different camera distances and noise levels is evaluated. For the plot, a table with the results of the paired t-test is provided.

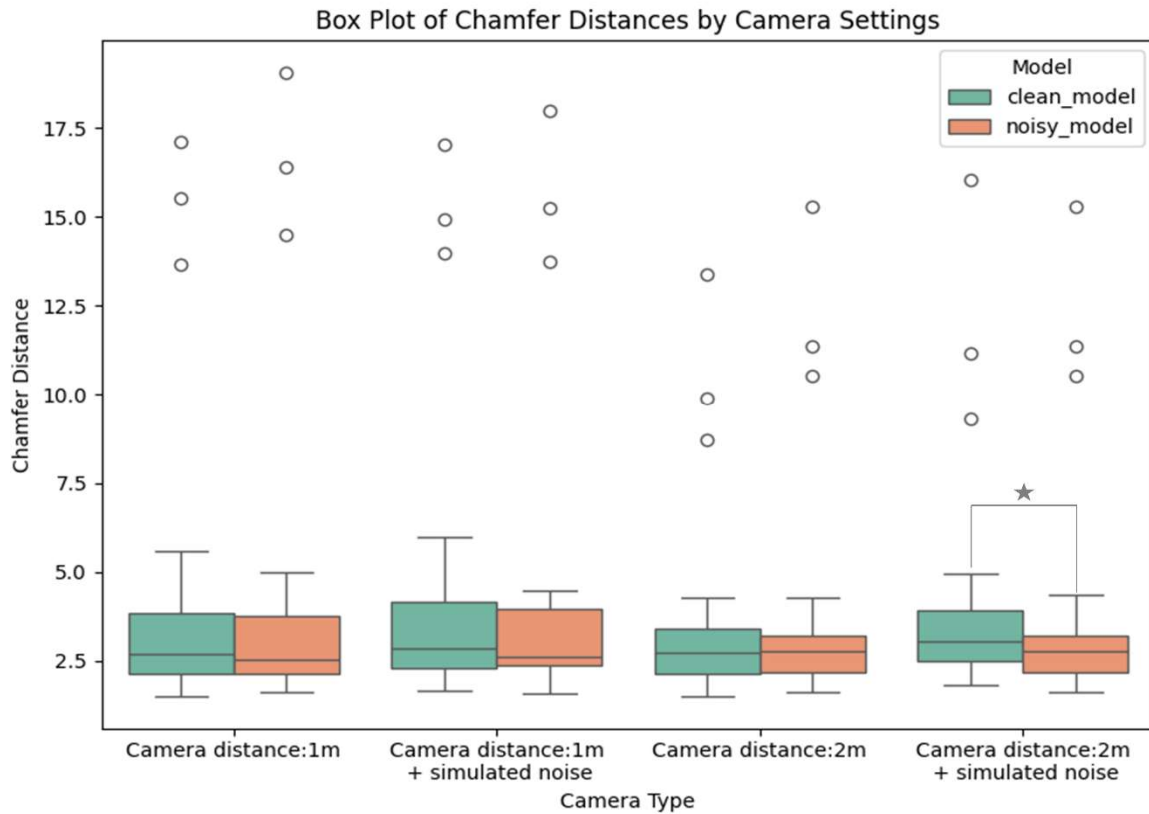


Figure 23. Shows the model's performance (reconstruction error) using boxplots. The turquoise boxes represent the clean model, while the orange boxes represent the model trained on noisy data. The x-axis categories represent camera distance (1 m and 2 m) with and without additional noise in the test data. Stars indicate statistically significant differences ($\alpha = 0.05$) between clean and noisy data performance for each category.

Table 7. Paired t-test results: Clean vs. Noisy Model Performance (n, t, p-value; $p < 0.05$ indicates statistical significance)

| Paired t-test (model clean to model noisy $\alpha = 5\%$) | | | | |
|--|------------|--------------------|------------|--------------------|
| | Camera 1 m | Camera 1 m + noise | Camera 2 m | Camera 2 m + noise |
| N-Samples | 42 | 42 | 42 | 42 |
| T-Value | 0.0810 | -0.9533 | 1.3781 | -3.4000 |
| P-Value | 0.9358 | 0.3459 | 0.1756 | 0.0015 |

Results “Comparison of model trained on noisy data to model trained on clean data” description:

The paired t-tests revealed a statistically significant difference (p -value < 0.05) between the models' performance for the category distance at 2 m and added noise to the partial point cloud data.

5.3 Comparison of different test data categories on Model A and Model B

Building upon the previous analysis of viewpoint-specific performance and noise sensitivity, this section delves deeper into the main effects of individual categories on the model's reconstruction quality. In this work, the design of experiments framework is leveraged to systematically analyze how variations within each category (viewpoint, camera distance, noise level) affect the reconstruction performance for both models. The main effects of Model A are presented in Figure 24, while the main effects of Model B are shown in Figure 25.

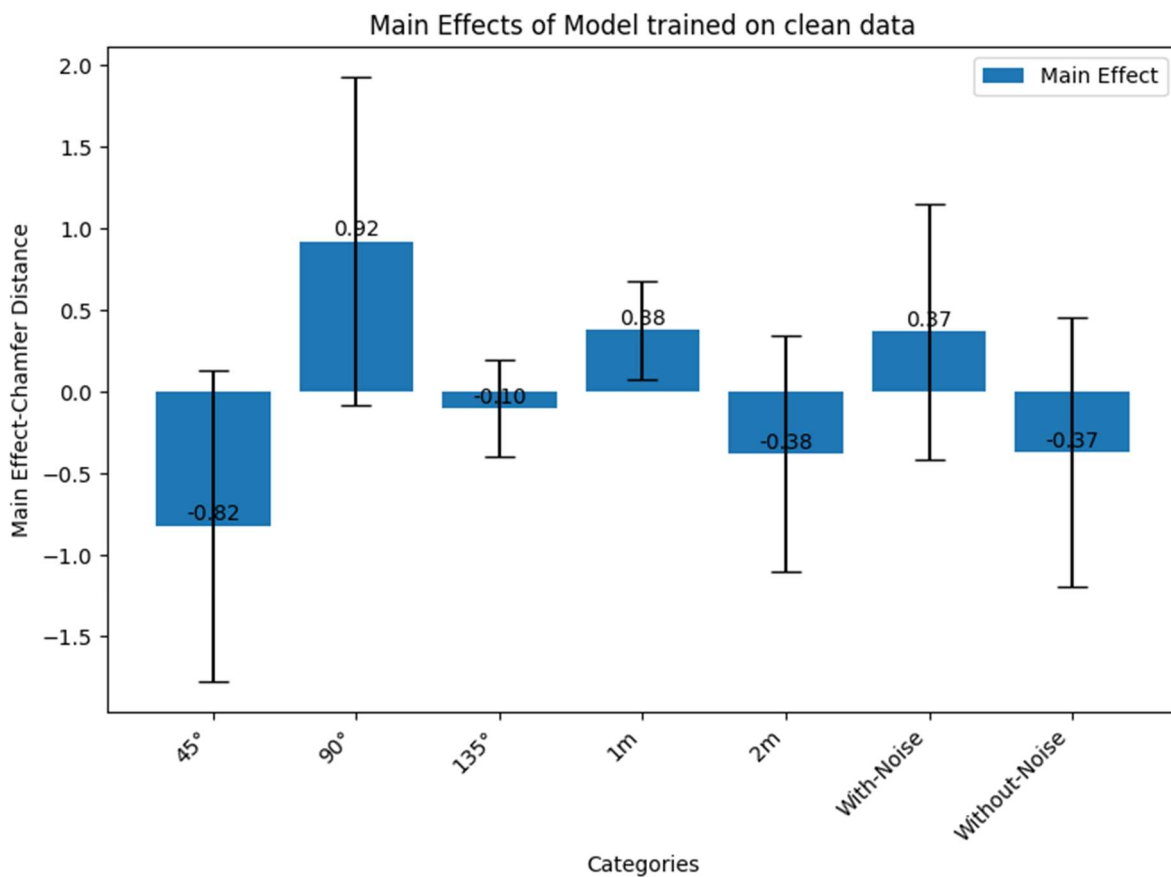


Figure 24. Illustrates the main effects of the clean model. This plot depicts the main effects of individual categories, on the reconstruction performance of the model trained on clean data, as identified through a design of experiments analysis. Each category (from left to right: view point at 45°, 90°, 135°; camera distance at 1 m, 2 m; noise-level with noise, without noise) represented on the x-axis is investigated independently to determine its impact on the model's output. The vertical axis represents the main effect, visualized through a change of the chamfer distance, which signifies the average change in reconstruction error (lower is better).

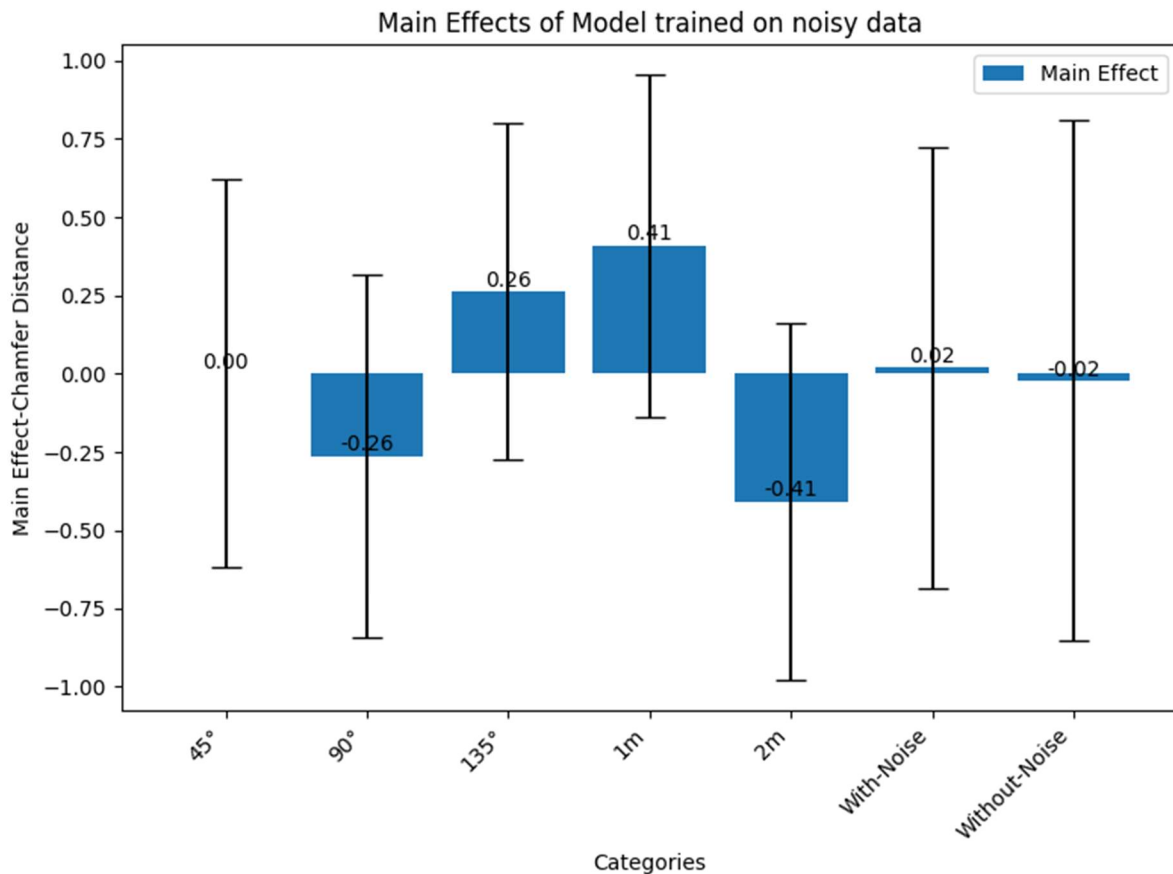


Figure 25. Illustrates the main effects of the noisy model. This plot depicts the main effects of individual categories, on the reconstruction performance of the model trained on clean data, as identified through a design of experiments analysis. Each category (from left to right: view point at 45°, 90°, 135°; camera distance at 1 m, 2 m; noise-level with noise, without noise) represented on the x-axis is investigated independently to determine its impact on the model's output. The vertical axis represents the main effect, visualized through a change of the chamfer distance, which signifies the average change in reconstruction error (lower is better).

Results “Comparison of different test data categories on Model A and Model B” description:

The model performance over the viewpoints varied depending on the viewing angle and on the model used for the reconstruction. Both models showed a decrease in reconstruction performance at a camera distance of 1 m compared to 2 m. This effect was more pronounced for the model trained on noisy data. While noise in the test dataset shows a decrease in performance over the model trained on clean data, the model trained on noisy data exhibits minimal performance degradation due to the noise addition.

5.4 Qualitative Results

Following the analysis of reconstruction errors through metrics like chamfer distance, this section delves deeper by showcasing the visual quality of reconstructions. Here, exemplars from the test data category of camera distance 1 m and camera position 45° are presented. One example representing the best reconstruction (Table 8) and another representing the worst reconstruction (Table 9) within this category. By visually comparing these two reconstructions alongside their corresponding point clouds and ground truth data, a qualitative understanding of how chamfer distance translates into reconstruction fidelity is gained.

Table 8. This table visualizes the best reconstruction (lowest chamfer distance) for the test data category with camera distance 1 m and camera position 45°. The left side shows the ground truth data: the point cloud representation at the top and the mesh representation at the bottom. The right side displays the reconstruction results in two different views: the reconstructed point cloud visualized in blue points and a partially mesh that represents the partial point cloud that the network receives as input for reconstructing the complete object.


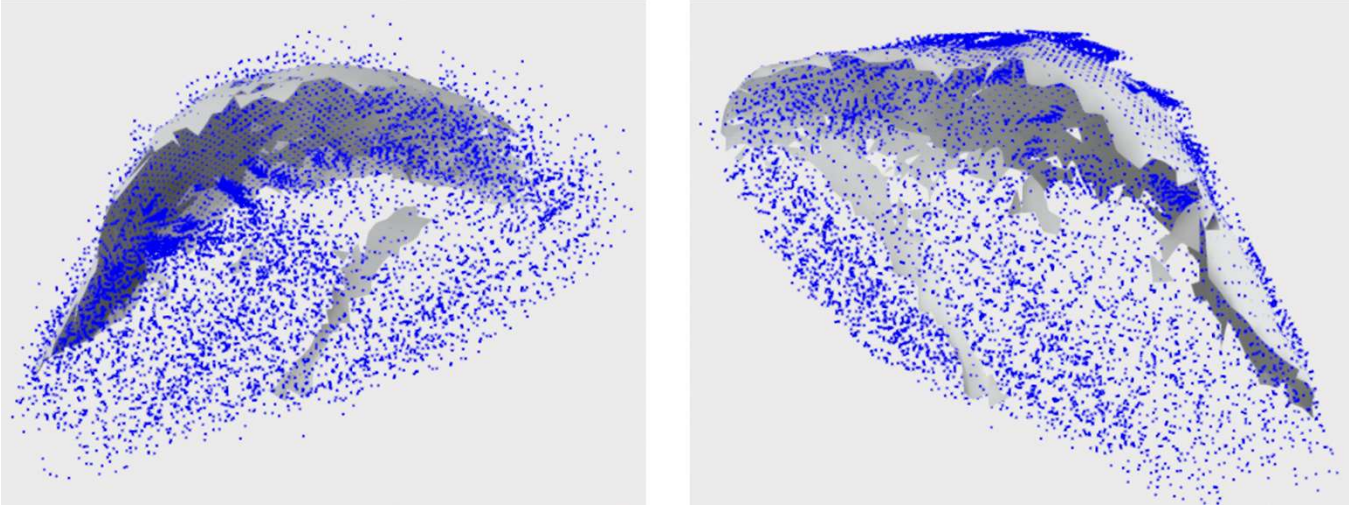
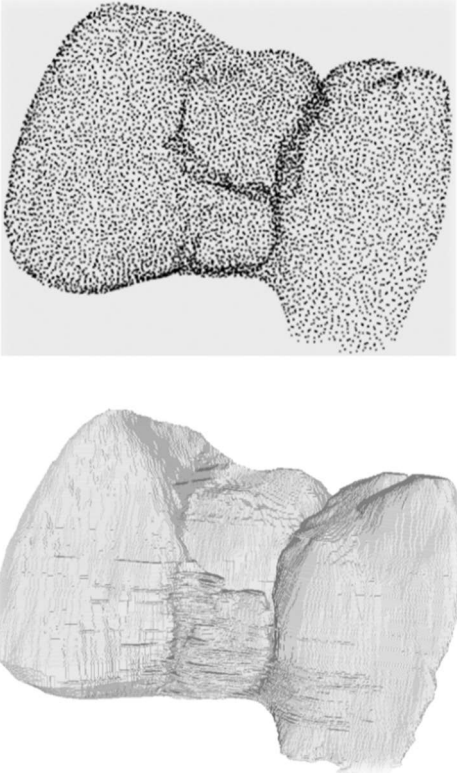

| Camera 1 m at 45° - Best reconstruction | |
|---|--|
| Ground Truth: | Reconstruction: (Chamfer Distance: 1.7074244) - Reconstructed points in blue, partial mesh in grey. |
|  <p>The ground truth visualization consists of two parts. The top part is a point cloud of a hand, showing a dense collection of grey points. The bottom part is a corresponding mesh representation of the same hand, showing a smooth, continuous surface with visible lines and shading.</p> |  <p>The reconstruction visualization consists of two parts. The left part shows a point cloud where the reconstructed points are blue and the partial mesh is grey. The right part shows a similar view from a different angle, highlighting the blue points and the grey mesh structure.</p> |

Table 9. This table visualizes the worst reconstruction (highest chamfer distance) for the test data category with camera distance 1 m and camera position 45°. The left side shows the ground truth data: the point cloud representation at the top and the mesh representation at the bottom. The right side displays the reconstruction results in two different views: the reconstructed point cloud visualized in blue points and a partially mesh that represents the partial point cloud that the network receives as input for reconstructing the complete object.

| Camera 1 m at 45° - Worst reconstruction | |
|--|---|
| Ground Truth: | Reconstruction: (Chamfer Distance: 16.06927) - Reconstructed points in blue, partial mesh in grey |
|  <p>The ground truth is shown in two parts: a point cloud at the top and a mesh at the bottom. The point cloud consists of a dense collection of black points forming a 3D object with a complex, irregular shape. The mesh below it is a grey wireframe representation of the same object, showing the underlying triangular structure.</p> |  <p>The reconstruction results are shown in two views. The left view shows a 3D scene with a grey mesh of the object and a point cloud of blue points. The blue points are concentrated on the front-facing surface, while the rest of the object is represented by the grey mesh. The right view shows a similar scene from a different angle, highlighting the blue point cloud and the grey mesh.</p> |

6 Discussion

This chapter builds upon the results presented earlier, delving deeper into their significance and implications. The discussion focuses on the performance of the deep learning reconstruction network on the categorized test dataset. Key observations regarding reconstruction accuracy and the impact of different categories are explored. Additionally, the limitations of the current approach will be addressed, and potential avenues for future research to further improve the network's capabilities will be suggested.

6.1 Test data validation:

6.1.1 Analysis of Captured Area Percentage

The analysis of captured area percentage in Figure 20 reveals consistency, with an average variation of only $\pm 4\%$ across viewpoints (45° to 135°) and object distances (1 m, 2 m). This indicates that the data generation process captured a consistent portion of the ground truth object's surface area, averaging around 34%, regardless of these variations. Koo et al. showed that 30-50% of the liver surface is visible in laparoscopic surgery, therefore value of just 34% might not capture sufficient detail for high-fidelity reconstruction in open liver surgery applications [70].

6.1.2 Analysis of Points Per Area

As expected, Figure 21 shows that point cloud density, measured by points per area, increases for objects closer to the sensor (1 m) compared to farther distances (2 m), demonstrating a higher level of detail captured for nearby objects. Interestingly, points per area shows a slight decrease with changes in viewpoint (from 45° to 135°). This decrease can be attributed to the limited resolution of the sensor data. As the viewpoint changes, certain object surfaces become nearly parallel to the sensor rays, as shown in Figure 26. Surfaces that are nearly parallel experience fewer ray intersections which leads to a lower point cloud density in those regions compared to areas facing the sensor more directly. The figure also reveals a significant drop in points per area from 1 m to 2 m camera distance, which leads to a large area, from $15 - 35 \left[\frac{\text{points}}{\text{mm}^2 \cdot 1000} \right]$ not being covered by data points.

Preview for future optimizations:

The test data validation provides a controlled environment ideal for evaluating reconstruction performance, but it is highly idealized. The used method only considers self-occlusion of the object. In real-world applications, the visible portion of the target object would be significantly smaller, especially at viewpoints like 45° and 135° , as depicted in Figure 3. To generate a continuous point cloud density value, generating partial point clouds with multiple camera distances is recommended.

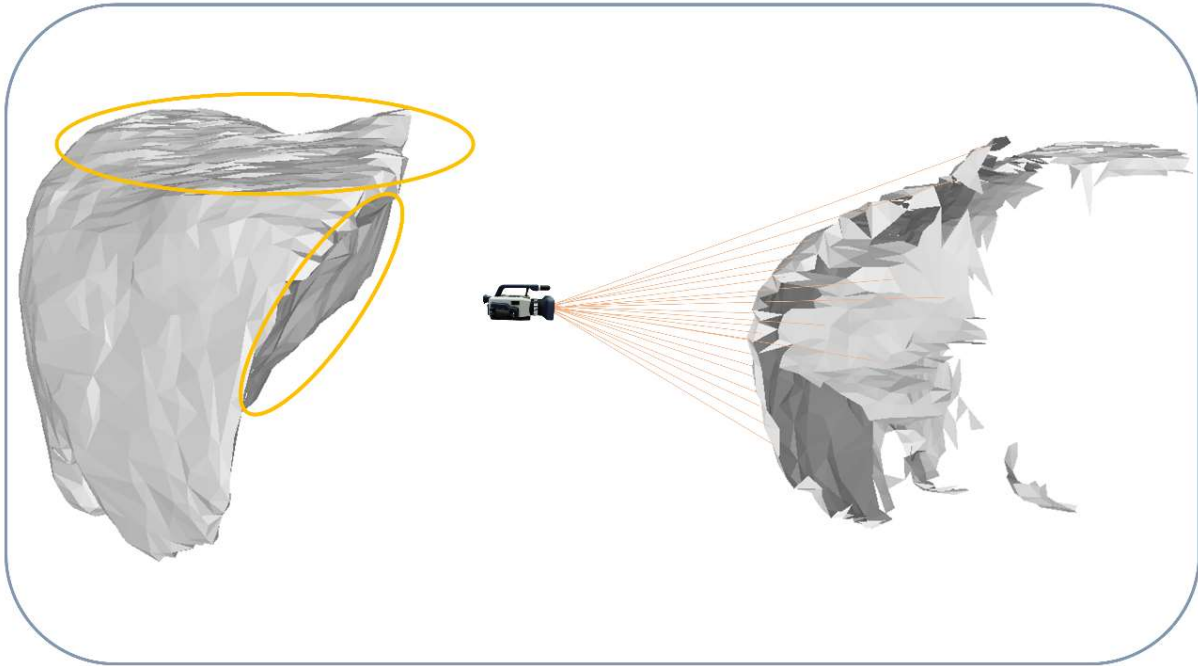


Figure 26. Illustrates the influence of object geometry on point cloud density captured by a virtual sensor (camera). The left side showcases a 135° viewpoint, highlighting surfaces in orange that are nearly parallel to the sensor rays in this view. The right side depicts the mesh and camera from a perpendicular viewpoint, emphasizing the near-parallel orientation of these marked surfaces relative to the sensor rays.

6.2 Comparison of model trained on noisy data to model trained on clean data

Figure 23 illustrates the difference in reconstruction performance between the model trained on clean and on noisy data. All medians from the model trained on noisy data, showed the same or slightly improved performance in reconstruction. Notably, one condition exhibited a statistically significant improvement according to a t-test, suggesting that training on noisy data can generate a more robust model that can outperform the clean data model under different conditions. While the limited number of data points used raises concerns about assuming normal distribution for the t-test, the overall trend suggests the potential benefit of incorporating noise in training.

Preview for future applications:

The beneficial behavior of artificial noise added to the training data is also observed by Neelakantan et al. and should be considered for future applications [71].

6.3 Comparison of the influence of test data parameters on the reconstruction performance

As illustrated in Figure 24 and Figure 25, the variation in performance across viewpoints and the two different models, suggest a potential interaction effect between training data noise and the geometric complexity of the visible surface. At the 45° view point, presenting the right liver lobe, the model trained on clean data shows superior performance compared to the 90° and 135° viewpoints. The model trained on noisy data exhibits medium performance at the 45° viewpoint.

Like demonstrated from Khajarian et al. both models might benefit from the presence of rich geometric features like those encountered at the 45° view point [72]. The best performance of the noisy model is observed at the 90° viewpoint. Through the training on noisy data the model might rely more on global features like the top and underside of the liver (Figure 22). These surfaces might offer the network more characteristic central points (global features) compared to the potentially richer geometric features (local features) encountered at viewpoints like 45°. The presence of these characteristic central points, even with potential noise in the training data, could be sufficient for the network to achieve superior performance to the clean data model for this specific view. Further investigation is needed to explore the specific reasons behind the model’s performance at 135°. This could involve analyzing the geometric features at this viewpoint in detail.

The usage of a model trained on noisy data proved beneficial, showing a more robust reconstruction performance for all viewpoints. Over all three camera positions the model showed less reliance to camera view point and only minor effects to noise added in the testing data. Additionally, the standard deviation for all parameters increased, suggesting a higher variability in performance across all parameters.

As displayed in Table 10, both the points per visible area and the points per partial point cloud in the training dataset correlate more closely with the test dataset at 1 m camera distance than 2 m. Counter to the expectations, both models exhibited lower performance when the partial point clouds were captured from a 1 m camera distance. This observation needs further investigation to understand the underlying mechanisms.

Table 10. Average points per area, number of points and average visible area for ground truth, training data, and testing data at 1 m and 2 m.

| | Ground Truth | Training data | Test data: 1m | Test data: 2m |
|---|--------------|---------------|---------------|---------------|
| Average points per area [$mm^{-2} \cdot 10^{-3}$] | 66.04 | 65.63 | 50.75 | 13.72 |
| Average number of points | 7500 | 2048 | 2102 | 524 |
| Average visible Area [mm^2] | 113552.32 | 32364.66 | 38191.88 | 38191.88 |

One explanation for this problem could arise through the used metric itself. In the original PoinTr network, a complete point cloud is randomly cropped to a fixed number of points. The partial point cloud is used for reconstruction while the complete point cloud is used as ground truth for calculating the loss. This workflow ensures that the points in the partial point cloud are at exactly the same position as in the ground truth, therefore with no contribution in the chamfer distance. For this work, ray casting was utilized to create partial point clouds, therefore no point in the partial point cloud is located at the same position as in the ground truth. In this case, the partial point cloud itself would contribute in the loss calculation. Additionally, Wu et al. showed a high dependency of chamfer distance on variations in point cloud density [73]. This complex relationship between point cloud density and average kNN-distance could lead to unexpected behavior, like higher chamfer distances in denser point clouds.

Preview for future optimizations:

For future work, exploring higher-order interaction effects using techniques like response surface methodology or plotting interaction terms would be beneficial. This could reveal more complex relationships between the parameters, providing a more comprehensive understanding of how they influence reconstruction accuracy. For evaluating the reconstruction performance, it should be considered to use metrics that are independent from variations in point cloud densities, like hyperbolic chamfer distance or density-aware chamfer distance [73], [74]. Additionally, incorporating a wider range of camera distances and potentially including dynamic viewpoints that simulate a surgeon's movements during an operation could provide valuable insights for real-world applications.

6.4 Data generation

The data generation process plays a fundamental role in achieving successful training for accurate reconstruction. While examining partial point clouds revealed that the generation process functions as intended in most cases, some challenges remain.

6.4.1 Mesh generation

The generation process leverages the publicly available LiTS dataset, which includes CT data with corresponding labels for background tissue, liver, and cancer. These labels are ideal for further processing. However, challenges arise due to the use of different CT units, resulting in CT images with varying slice thicknesses. For images with large slice thicknesses, the discrete segmentation process can lead to meshes with stepped surfaces, as illustrated in Figure 27. While Palomar et al. proposed mesh smoothing algorithms to address this problem, in this work, it was decided to not utilize smoothing operations, to avoid altering the ground truth data [75].

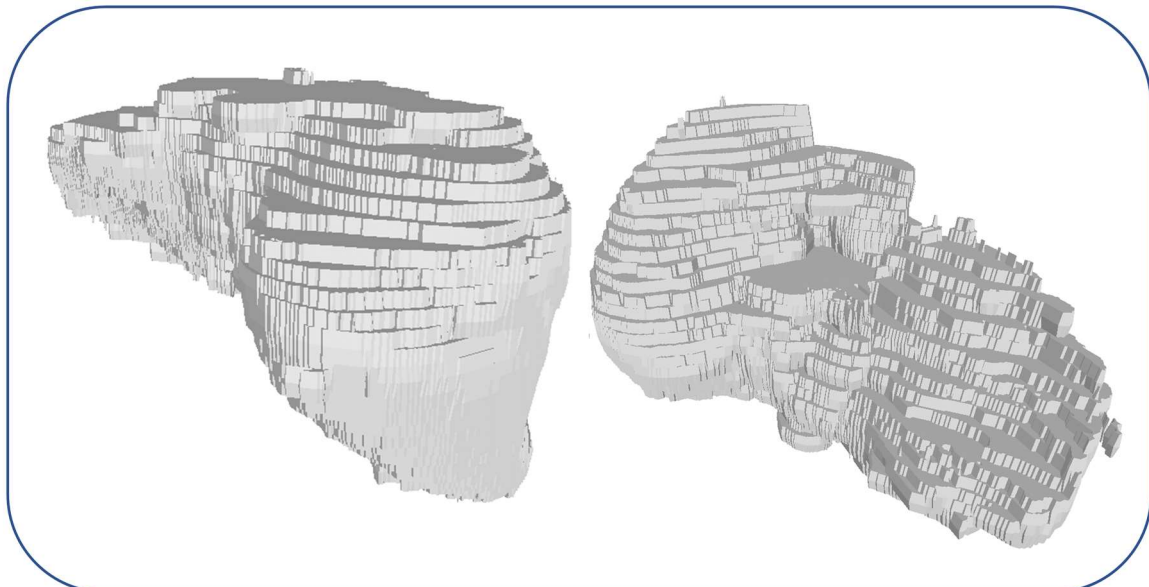


Figure 27. Illustrates a mesh generated from the LiTS dataset using CT scans with low voxel resolution. The low-resolution results in a mesh with a stepped surface, where the smooth anatomical features of the liver are not accurately captured.

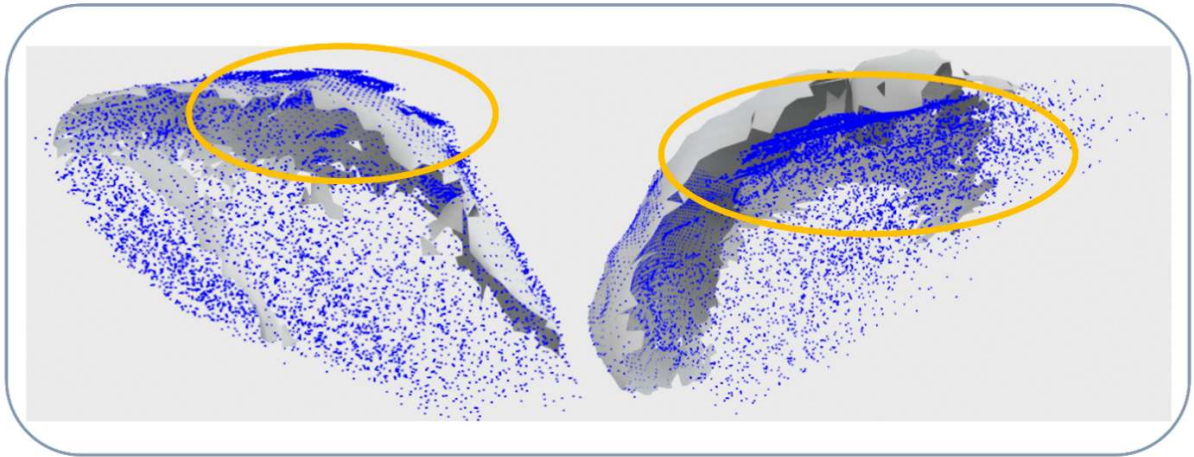


Figure 28. Illustrates planar surfaces with a stepped structure, highlighted in orange, extend over the visible mesh surface. These artifacts likely arise due to the presence of stepped surfaces within the training data.

As shown in Figure 28, an examination of the reconstructed point cloud overlaid on the partial mesh representing the input data reveals prominent stair-stepping artifacts. These artifacts, particularly evident on edges or boundaries, resemble the stepped surfaces observed in the training data with thick slices in Figure 27. This suggests the model has learned to reconstruct these features, potentially impacting the accuracy of reconstructions on unseen data with smoother surfaces.

The CT-data in the LITs-datasets includes three labels for liver, tumor and background tissue. For this work, all voxels with labels corresponding to liver and tumor were selected for further processing through the marching cubes algorithm to avoid holes in the mesh. However, since the dataset specializes in CT data with liver tumors, some CT images contain malignant tumors that extend over the liver surface, as visualized in Figure 29. In these cases, including the tumor labels, effectively changes the geometry of the organ. Due to the limited number of training data, this can lead to outliers in reconstruction performance, observed in both the testing and training datasets. A notable example of this challenge can be seen in Figure 29, where a large tumor is present on the liver surface, resulting in the worst reconstruction performance across all test set categories as shown in Figure 23.

Preview for future optimizations:

In medical imaging the generation of staircase artifact through the marching cubes algorithm is a well-known challenge. For future work, exploring the capabilities of smoothing operations like Laplacian smoothing and Taubin smoothing could be beneficial. However, these techniques introduce additional processing stages and control parameters that require careful consideration [76], [77].

Additionally, utilizing a dataset that excludes tumors could prove beneficial for improved learning of the anatomical properties of healthy liver surfaces.

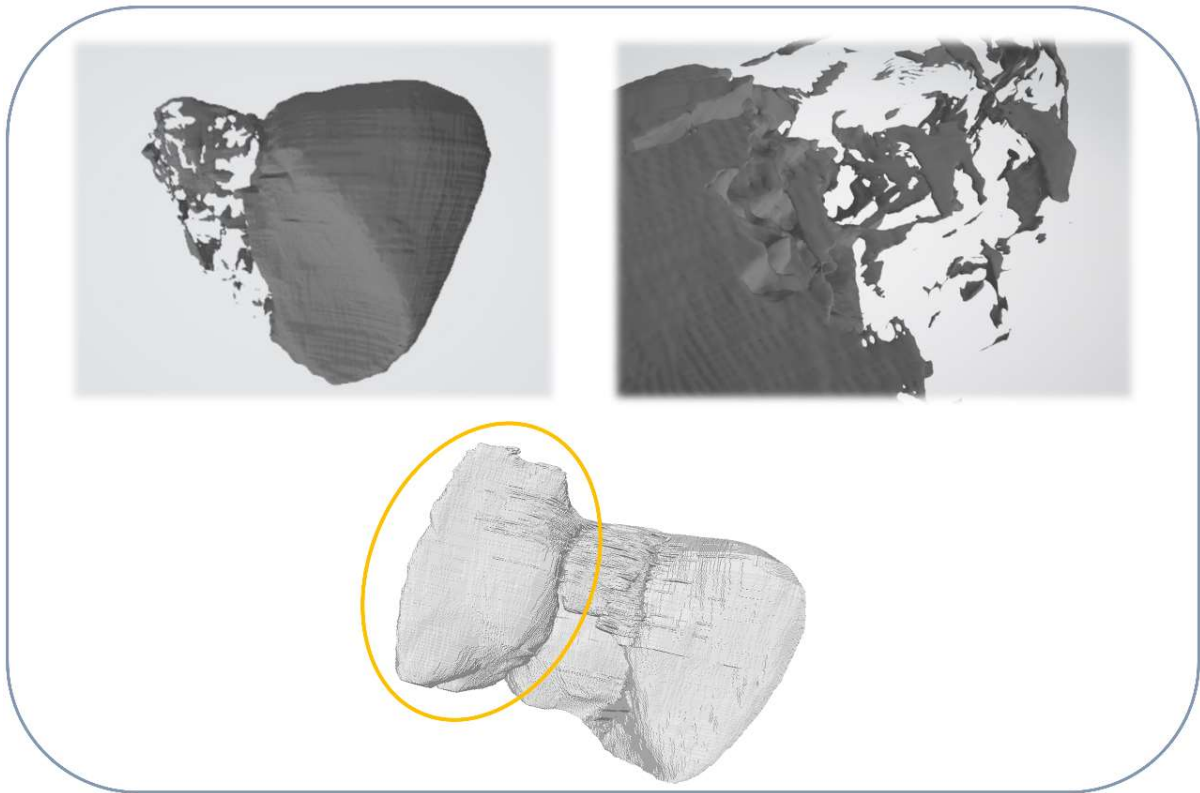


Figure 29. Illustrated at the top a mesh excluding the tumor, resulting in a incomplete mesh. At the bottom, a liver mesh generated with liver and tumor together is shown. The tumor is marked in orange.

6.4.2 Partial point cloud generation

Partial point clouds are generated using a method called ray casting. This method has proven beneficial for creating partial point clouds from a complete mesh. However, as discussed in 6.1, the 45° and 135° camera viewpoints might not reflect realistic occlusions encountered in real-world scenarios. Ray casting with a single mesh only considers self-occlusion, meaning it excludes occlusions caused by other organs. Consequently, these specific viewpoints might incorporate surfaces of the mesh that would normally be hidden behind other organs in a real-world setting.

Preview for future optimizations:

In future works, it would be beneficial to explore more complex and realistic datasets that incorporate various occlusion scenarios present in real-world situations. This could involve using multiple viewpoints or even dynamic viewpoints that simulate a surgeon's movements during an operation.

6.5 Training process

As described under 4.1.2, the training process for the PoinTr network involves randomly selecting one partial point cloud from 250 available views in each epoch for reconstruction. Since the entire training dataset consists of 250 partial views with a fixed number of 2048 points each, there are only minor variations in point cloud density. This limited variation might not be ideal for achieving a robust reconstruction model. Additionally, the random selection process does not guarantee that all available data gets utilized during training.

Furthermore, as discussed in section 6.3, points within the generated partial point clouds do not correspond exactly to the positions of points in the ground truth data. Raycasting also inherently introduces continuous changes in point cloud density across the surface. Due to these factors, the chamfer distance might not be the most suitable metric for evaluating reconstruction performance, especially when training data with overall varying point cloud densities is used. However, it remains a valuable loss function during training due to its efficiency and focus on minimizing point-to-point distances.

Preview for future optimizations:

For future works, it would be beneficial to consider using training data with variations in the number of points per partial point cloud. This variation can help the model learn and generalize better to unseen data with different point densities, leading to a more robust reconstruction process. Additionally, replacing the random selection of one point cloud per epoch with a method that randomly picks a point from the entire dataset, ensuring each point is chosen only once and all data has been processed during training, could improve the utilization of the training data. Finally, adopting a density-independent chamfer distance metric could be more effective in evaluating reconstruction performance when dealing with variations in point cloud density.

7 Conclusion

This work presented a workflow for generating a reconstruction dataset from CT data. This dataset includes complete and partial point clouds representing multiple viewpoints, along with a trained reconstruction network. The workflow offers theoretical applicability to other organs, provided labeled CT data exists.

The key finding of this work is the possibility of using the reconstruction network as a pre-processing step to enhance registration accuracy and efficiency. By generating a more complete point cloud representation of the organ, the reconstruction process could provide a better foundation for subsequent registration algorithms.

While the reconstruction results are promising, there are limitations to consider. State-of-the-art deep learning algorithms designed specifically for partial registration tasks are constantly improving. As these algorithms become more sophisticated, the need for a separate pre-processing step to address self-occlusion with reconstruction networks might become less critical.

However, the potential of this reconstruction workflow extends beyond registration. In the exciting realm of future augmented reality applications for open liver surgery, real-time organ reconstruction offers significant advantages. By dynamically reconstructing the operated organ during surgery, AR could provide surgeons with invaluable visualizations of the operative field.

8 Acknowledgment

First and foremost, I am incredibly grateful to Prof. Dr. Stefanie Remmele for allowing me to conduct this research in her laboratory. Her exceptional guidance, support, and insightful feedback throughout this process have been invaluable.

I would also like to express my sincere gratitude to Serouj Khajarian and Michael Schwimmbeck for their positive collaboration and the valuable feedback they provided throughout this project.

My deepest thanks go to Katharina and Wolfgang Bode, who generously dedicated significant time to meticulously proofreading and correcting my thesis.

A special thanks goes to Prof. Dr. Jaud, who ignited my passion for learning in my very first semester. His influence undoubtedly played a significant role in leading me to this point.

Finally, as I mark the completion of my Bachelor's studies with this thesis, I would like to express my sincere gratitude to Katharina Bode and my family for their unwavering support throughout my academic journey.

9 References

- [1] M. Yamamoto and S. Ariizumi, "Glissonean pedicle approach in liver surgery," *Ann. Gastroenterol. Surg.*, vol. 2, no. 2, pp. 124–128, Mar. 2018, doi: 10.1002/ags3.12062.
- [2] N. Kokudo, N. Takemura, K. Ito, and F. Mihara, "The history of liver surgery: Achievements over the past 50 years," *Ann. Gastroenterol. Surg.*, vol. 4, no. 2, pp. 109–117, Mar. 2020, doi: 10.1002/ags3.12322.
- [3] B. May and D. Madoff, "Portal Vein Embolization: Rationale, Technique, and Current Application," *Semin. Interv. Radiol.*, vol. 29, no. 02, pp. 081–089, Jun. 2012, doi: 10.1055/s-0032-1312568.
- [4] N. Golse, A. Petit, M. Lewin, E. Vibert, and S. Cotin, "Augmented Reality during Open Liver Surgery Using a Markerless Non-rigid Registration System," *J. Gastrointest. Surg.*, vol. 25, no. 3, pp. 662–671, Mar. 2021, doi: 10.1007/s11605-020-04519-4.
- [5] V. M. Banz *et al.*, "Intraoperative image-guided navigation system: development and applicability in 65 patients undergoing liver surgery," *Langenbecks Arch. Surg.*, vol. 401, no. 4, pp. 495–502, Jun. 2016, doi: 10.1007/s00423-016-1417-0.
- [6] Z. Zhao *et al.*, "Augmented reality technology in image-guided therapy: State-of-the-art review," *Proc. Inst. Mech. Eng. [H]*, vol. 235, no. 12, pp. 1386–1398, Dec. 2021, doi: 10.1177/09544119211034357.
- [7] "AR/VR headset shipments by market 2021-2026," Statista. Accessed: Mar. 23, 2024. [Online]. Available: <https://bibaccess.fh-landshut.de:2128/statistics/1301629/ar-vr-headset-shipments-by-market/>
- [8] R. Londei *et al.*, "Intra-operative augmented reality in distal locking," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 10, no. 9, pp. 1395–1403, Sep. 2015, doi: 10.1007/s11548-015-1169-2.
- [9] A. Elmi-Terander *et al.*, "Surgical Navigation Technology Based on Augmented Reality and Integrated 3D Intraoperative Imaging: A Spine Cadaveric Feasibility and Accuracy Study," *Spine*, vol. 41, no. 21, pp. E1303–E1311, Nov. 2016, doi: 10.1097/BRS.0000000000001830.
- [10] L. Ma, T. Huang, J. Wang, and H. Liao, "Visualization, registration and tracking techniques for augmented reality guided surgery: a review," *Phys. Med. Biol.*, vol. 68, no. 4, p. 04TR02, Feb. 2023, doi: 10.1088/1361-6560/acaf23.
- [11] R. Tang *et al.*, "Augmented reality technology for preoperative planning and intraoperative navigation during hepatobiliary surgery: A review of current methods," *Hepatobiliary Pancreat. Dis. Int.*, vol. 17, no. 2, pp. 101–112, Apr. 2018, doi: 10.1016/j.hbpd.2018.02.002.
- [12] F. Fan, B. Kreher, H. Keil, A. Maier, and Y. Huang, "Fiducial marker recovery and detection from severely truncated data in navigation-assisted spine surgery," *Med. Phys.*, vol. 49, no. 5, pp. 2914–2930, 2022, doi: 10.1002/mp.15617.
- [13] T. Czerniawski, M. Nahangi, C. Haas, and S. Walbridge, "Pipe spool recognition in cluttered point clouds using a curvature-based shape descriptor," *Autom. Constr.*, vol. 71, pp. 346–358, Nov. 2016, doi: 10.1016/j.autcon.2016.08.011.
- [14] Y. Guo, F. Sohel, M. Bennamoun, J. Wan, and M. Lu, "A novel local surface feature for 3D object recognition under clutter and occlusion," *Inf. Sci.*, vol. 293, pp. 196–213, Feb. 2015, doi: 10.1016/j.ins.2014.09.015.
- [15] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015, doi: 10.1109/TRO.2015.2463671.
- [16] Q. H. Gao, T. R. Wan, W. Tang, and L. Chen, "Object registration in semi-cluttered and partial-occluded scenes for augmented reality," *Multimed. Tools Appl.*, vol. 78, no. 11, pp. 15079–15099, Jun. 2019, doi: 10.1007/s11042-018-6905-5.
- [17] H. Ghaednia *et al.*, "Augmented and virtual reality in spine surgery, current applications and future potentials," *Spine J.*, vol. 21, no. 10, pp. 1617–1625, Oct. 2021, doi: 10.1016/j.spinee.2021.03.018.
- [18] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep Learning for 3D Point Clouds: A Survey." *arXiv*, Jun. 23, 2020. Accessed: Mar. 07, 2024. [Online]. Available: <http://arxiv.org/abs/1912.12033>

- [19] A. Dai, C. R. Qi, and M. Nießner, “Shape Completion using 3D-Encoder-Predictor CNNs and Shape Synthesis.” arXiv, Apr. 11, 2017. Accessed: Mar. 11, 2024. [Online]. Available: <http://arxiv.org/abs/1612.00101>
- [20] A. Sharma, O. Grau, and M. Fritz, “VConv-DAE: Deep Volumetric Shape Learning Without Object Labels.” arXiv, Sep. 09, 2016. Accessed: Mar. 11, 2024. [Online]. Available: <http://arxiv.org/abs/1604.03755>
- [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.” arXiv, Sep. 22, 2017. Accessed: Mar. 11, 2024. [Online]. Available: <http://arxiv.org/abs/1709.07599>
- [22] B. Fei *et al.*, “Comprehensive Review of Deep Learning-Based 3D Point Cloud Completion Processing and Analysis,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 22862–22883, Dec. 2022, doi: 10.1109/TITS.2022.3195555.
- [23] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation.” arXiv, Apr. 10, 2017. Accessed: Mar. 11, 2024. [Online]. Available: <http://arxiv.org/abs/1612.00593>
- [24] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space.” arXiv, Jun. 07, 2017. Accessed: Mar. 11, 2024. [Online]. Available: <http://arxiv.org/abs/1706.02413>
- [25] P. Mandikal and R. V. Babu, “Dense 3D Point Cloud Reconstruction Using a Deep Pyramid Network.” arXiv, Jan. 25, 2019. Accessed: Mar. 11, 2024. [Online]. Available: <http://arxiv.org/abs/1901.08906>
- [26] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, “PCN: Point Completion Network.” arXiv, Sep. 26, 2019. Accessed: Mar. 11, 2024. [Online]. Available: <http://arxiv.org/abs/1808.00671>
- [27] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, “PU-Net: Point Cloud Upsampling Network.” arXiv, Mar. 26, 2018. Accessed: Mar. 11, 2024. [Online]. Available: <http://arxiv.org/abs/1801.06761>
- [28] J. Wang, Y. Cui, D. Guo, J. Li, Q. Liu, and C. Shen, “PointAttN: You Only Need Attention for Point Cloud Completion.” arXiv, Mar. 16, 2022. Accessed: Mar. 11, 2024. [Online]. Available: <http://arxiv.org/abs/2203.08485>
- [29] Y. Yang, C. Feng, Y. Shen, and D. Tian, “FoldingNet: Point Cloud Auto-encoder via Deep Grid Deformation.” arXiv, Apr. 02, 2018. Accessed: Nov. 26, 2023. [Online]. Available: <http://arxiv.org/abs/1712.07262>
- [30] X. Zhang *et al.*, “View-Guided Point Cloud Completion,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA: IEEE, Jun. 2021, pp. 15885–15894. doi: 10.1109/CVPR46437.2021.01563.
- [31] H. Xie, H. Yao, S. Zhou, J. Mao, S. Zhang, and W. Sun, “GRNet: Gridding Residual Network for Dense Point Cloud Completion.” arXiv, Jul. 20, 2020. Accessed: Nov. 09, 2023. [Online]. Available: <http://arxiv.org/abs/2006.03761>
- [32] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic Graph CNN for Learning on Point Clouds.” arXiv, Jun. 11, 2019. Accessed: Mar. 12, 2024. [Online]. Available: <http://arxiv.org/abs/1801.07829>
- [33] Y. Zhang, D. Huang, and Y. Wang, “PC-RGNN: Point Cloud Completion and Graph Neural Network for 3D Object Detection.” arXiv, Dec. 21, 2020. Accessed: Mar. 12, 2024. [Online]. Available: <http://arxiv.org/abs/2012.10412>
- [34] L. Pan *et al.*, “Multi-View Partial (MVP) Point Cloud Challenge 2021 on Completion and Registration: Methods and Results.” arXiv, Dec. 22, 2021. Accessed: Nov. 17, 2023. [Online]. Available: <http://arxiv.org/abs/2112.12053>
- [35] M. Sarmad, H. J. Lee, and Y. M. Kim, “RL-GAN-Net: A Reinforcement Learning Agent Controlled GAN Network for Real-Time Point Cloud Shape Completion.” arXiv, Apr. 28, 2019. Accessed: Mar. 12, 2024. [Online]. Available: <http://arxiv.org/abs/1904.12304>
- [36] C. Xie, C. Wang, B. Zhang, H. Yang, D. Chen, and F. Wen, “Style-based Point Generator with Adversarial Rendering for Point Cloud Completion.” arXiv, May 12, 2021. Accessed: Nov. 09, 2023. [Online]. Available: <http://arxiv.org/abs/2103.02535>

- [37] A. Vaswani *et al.*, “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2017. Accessed: Mar. 11, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
- [38] X. Yu, Y. Rao, Z. Wang, Z. Liu, J. Lu, and J. Zhou, “PoinTr: Diverse Point Cloud Completion with Geometry-Aware Transformers.” arXiv, Aug. 19, 2021. doi: 10.48550/arXiv.2108.08839.
- [39] P. Xiang *et al.*, “SnowflakeNet: Point Cloud Completion by Snowflake Point Deconvolution with Skip-Transformer,” in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Montreal, QC, Canada: IEEE, Oct. 2021, pp. 5479–5489. doi: 10.1109/ICCV48922.2021.00545.
- [40] S. Li, P. Gao, X. Tan, and M. Wei, “ProxyFormer: Proxy Alignment Assisted Point Cloud Completion with Missing Part Sensitive Transformer,” in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vancouver, BC, Canada: IEEE, Jun. 2023, pp. 9466–9475. doi: 10.1109/CVPR52729.2023.00913.
- [41] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI: IEEE, Jun. 2012, pp. 3354–3361. doi: 10.1109/CVPR.2012.6248074.
- [42] A. X. Chang *et al.*, “ShapeNet: An Information-Rich 3D Model Repository.” arXiv, Dec. 09, 2015. Accessed: Mar. 08, 2024. [Online]. Available: <http://arxiv.org/abs/1512.03012>
- [43] K. Mo *et al.*, “PartNet: A Large-scale Benchmark for Fine-grained and Hierarchical Part-level 3D Object Understanding.” arXiv, Dec. 06, 2018. Accessed: Mar. 08, 2024. [Online]. Available: <http://arxiv.org/abs/1812.02713>
- [44] R. Modrzejewski, T. Collins, B. Seeliger, A. Bartoli, A. Hostettler, and J. Marescaux, “An in vivo porcine dataset and evaluation methodology to measure soft-body laparoscopic liver registration accuracy with an extended algorithm that handles collisions,” *Int. J. Comput. Assist. Radiol. Surg.*, vol. 14, no. 7, pp. 1237–1245, Jul. 2019, doi: 10.1007/s11548-019-02001-4.
- [45] M. Gao, N. Ruan, J. Shi, and W. Zhou, “Deep Neural Network for 3D Shape Classification Based on Mesh Feature,” *Sensors*, vol. 22, no. 18, p. 7040, Sep. 2022, doi: 10.3390/s22187040.
- [46] B. Yang, N. Haala, and Z. Dong, “Progress and perspectives of point cloud intelligence,” *Geo-Spat. Inf. Sci.*, vol. 26, no. 2, pp. 189–205, Apr. 2023, doi: 10.1080/10095020.2023.2175478.
- [47] Brno University of Technology (BUT) *et al.*, “Voxel size and calibration for CT measurements with a small field of view,” *E-J. Nondestruct. Test.*, vol. 24, no. 3, Mar. 2019, doi: 10.58286/23714.
- [48] U. Wickramasinghe, E. Remelli, G. Knott, and P. Fua, “Voxel2Mesh: 3D Mesh Model Generation from Volumetric Data.” arXiv, Apr. 01, 2020. Accessed: Mar. 09, 2024. [Online]. Available: <http://arxiv.org/abs/1912.03681>
- [49] W. E. Lorensen and H. E. Cline, “Marching cubes: A high resolution 3D surface construction algorithm”.
- [50] X. Wang, S. Gao, M. Wang, and Z. Duan, “An Marching Cube Algorithm Based on Edge Growth”.
- [51] “Datei:Marching Cubes Kubus.png,” *Wikipedia*. Mar. 18, 2013. Accessed: Mar. 09, 2024. [Online]. Available: https://de.wikipedia.org/w/index.php?title=Datei:Marching_Cubes_Kubus.png&oldid=115558080
- [52] “Datei:Marching Cubes Beispiel.png,” *Wikipedia*. Mar. 18, 2013. Accessed: Mar. 09, 2024. [Online]. Available: https://de.wikipedia.org/w/index.php?title=Datei:Marching_Cubes_Beiispiel.png&oldid=115558086
- [53] Henrik, *This diagram illustrates the ray tracing algorithm for rendering an image*. 2008. Accessed: Mar. 13, 2024. [Online]. Available: https://commons.wikimedia.org/wiki/File:Ray_trace_diagram.svg
- [54] A. Appel, “Some techniques for shading machine renderings of solids,” in *Proceedings of the April 30--May 2, 1968, spring joint computer conference on - AFIPS '68 (Spring)*, Atlantic City, New Jersey: ACM Press, 1968, p. 37. doi: 10.1145/1468075.1468082.
- [55] S. D. Roth, “Ray casting for modeling solids,” *Comput. Graph. Image Process.*, vol. 18, no. 2, pp. 109–144, Feb. 1982, doi: 10.1016/0146-664X(82)90169-1.

- [56] K. Pietroszek, "Raycasting in Virtual Reality," in *Encyclopedia of Computer Graphics and Games*, N. Lee, Ed., Cham: Springer International Publishing, 2018, pp. 1–3. doi: 10.1007/978-3-319-08234-9_180-1.
- [57] C.-H. Lee and Y. G. Shin, "Antialiasing in Raycasting Digital Terrain Models".
- [58] G. F. Gusmão, C. R. H. Barbosa, and A. B. Raposo, "Development and Validation of LiDAR Sensor Simulators Based on Parallel Raycasting," *Sensors*, vol. 20, no. 24, Dec. 2020, doi: 10.3390/s20247186.
- [59] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 2414–2423. doi: 10.1109/CVPR.2016.265.
- [60] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, "What Does BERT Look At? An Analysis of BERT's Attention." arXiv, Jun. 10, 2019. Accessed: Mar. 22, 2024. [Online]. Available: <http://arxiv.org/abs/1906.04341>
- [61] J. Lee, R. Tang, and J. Lin, "What Would Elsa Do? Freezing Layers During Transformer Fine-Tuning." arXiv, Nov. 08, 2019. Accessed: Mar. 14, 2024. [Online]. Available: <http://arxiv.org/abs/1911.03090>
- [62] D. Vucetic, M. Tayaranian, M. Ziaeeafard, J. J. Clark, B. H. Meyer, and W. J. Gross, "Efficient Fine-Tuning of BERT Models on the Edge," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2022, pp. 1838–1842. doi: 10.1109/ISCAS48785.2022.9937567.
- [63] J. Pfeiffer *et al.*, "AdapterHub: A Framework for Adapting Transformers," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online: Association for Computational Linguistics, 2020, pp. 46–54. doi: 10.18653/v1/2020.emnlp-demos.7.
- [64] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, "Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning." arXiv, Dec. 22, 2020. Accessed: Mar. 22, 2024. [Online]. Available: <http://arxiv.org/abs/2012.13255>
- [65] B. Rister, D. Yi, K. Shivakumar, T. Nobashi, and D. L. Rubin, "CT-ORG, a new dataset for multiple organ segmentation in computed tomography," *Sci. Data*, vol. 7, no. 1, p. 381, Nov. 2020, doi: 10.1038/s41597-020-00715-8.
- [66] P. Bilic *et al.*, "The Liver Tumor Segmentation Benchmark (LiTS)," *Med. Image Anal.*, vol. 84, p. 102680, Feb. 2023, doi: 10.1016/j.media.2022.102680.
- [67] M. Pfeiffer *et al.*, "Non-Rigid Volume to Surface Registration using a Data-Driven Biomechanical Model." arXiv, May 29, 2020. doi: 10.48550/arXiv.2005.14695.
- [68] C. S. Bamji *et al.*, "From: C. Bamji *et al.*, '1Mpixel 65nm BSI 320MHz Demodulated TOF Image Sensor with 3.5µm Global Shutter Pixels and Analog Binning,' ISSCC Deg. Tech. Papers, pp. 94-95, Feb 2018. IEEE Explore Link: <https://ieeexplore.ieee.org/document/8310200/>," 2018.
- [69] "GitHub - yuxumin/PoinTr: [ICCV 2021 Oral] PoinTr: Diverse Point Cloud Completion with Geometry-Aware Transformers." Accessed: Apr. 04, 2024. [Online]. Available: <https://github.com/yuxumin/PoinTr>
- [70] B. Koo *et al.*, "Automatic, global registration in laparoscopic liver surgery," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 17, no. 1, pp. 167–176, Jan. 2022, doi: 10.1007/s11548-021-02518-7.
- [71] A. Neelakantan *et al.*, "Adding Gradient Noise Improves Learning for Very Deep Networks." arXiv, Nov. 20, 2015. Accessed: Apr. 09, 2024. [Online]. Available: <http://arxiv.org/abs/1511.06807>
- [72] S. Khajarian, S. Remmele, and O. Amft, "Image-based Live Tracking and Registration for AR-Guided Liver Surgery Using HoloLens2: A Phantom Study," in *2023 IEEE EMBS International Conference on Biomedical and Health Informatics (BHI)*, Pittsburgh, PA, USA: IEEE, Oct. 2023, pp. 1–4. doi: 10.1109/BHI58575.2023.10313488.
- [73] T. Wu, L. Pan, J. Zhang, T. Wang, Z. Liu, and D. Lin, "Density-aware Chamfer Distance as a Comprehensive Metric for Point Cloud Completion".
- [74] F. Lin *et al.*, "Hyperbolic Chamfer Distance for Point Cloud Completion".
- [75] R. Palomar, F. A. Cheikh, B. Edwin, A. Beghdadhi, and O. J. Elle, "Surface reconstruction for planning and navigation of liver resections," *Comput. Med. Imaging Graph.*, vol. 53, pp. 30–42, Oct. 2016, doi: 10.1016/j.compmedimag.2016.07.003.

- [76] J. Vollmer, R. Mencl, and H. Müller, “Improved Laplacian Smoothing of Noisy Surface Meshes,” *Comput. Graph. Forum*, vol. 18, no. 3, pp. 131–138, Sep. 1999, doi: 10.1111/1467-8659.00334.
- [77] G. Taubin, “Curve and surface smoothing without shrinkage,” in *Proceedings of IEEE International Conference on Computer Vision*, Cambridge, MA, USA: IEEE Comput. Soc. Press, 1995, pp. 852–857. doi: 10.1109/ICCV.1995.466848.

10 List of figures

| | |
|--|----|
| Figure 1. Illustrates the projected shipment forecast for augmented reality (AR) and virtual reality (VR) headsets globally, segmented by market (consumer and commercial) from 2021 to 2026 [7]. | 7 |
| Figure 2. Illustrates the distribution of 103 research papers focusing on augmented reality (AR) applications in various surgical procedures [10]. | 8 |
| Figure 3. Illustrates a surgeon's view of a partially exposed liver during open liver surgery. Image: Acquisition from the AliAS data study, Lakumed Hospital Landshut-Achdorf. The study was approved by the Ethics Committee of the Technical University of Munich. BfArM study register number: DRKS00032826. | 9 |
| Figure 4. Illustrates different point cloud completion techniques. | 10 |
| Figure 5. Illustrates different representations of a rabbit as point cloud, voxel grid, and polygon mesh [45]. | 12 |
| Figure 6. Illustrates a logical cube, a key element in the marching cubes algorithm for 3D surface reconstruction. The algorithm analyzes a grid of data points (voxels) at the cube's vertices. Each voxel in the grid can be assigned a value based on a specific property, such as density or intensity [51]. | 12 |
| Figure 7. Illustrates example surfaces reconstructed from a logical cube. The figure illustrates how different configurations of vertex values within a cube can result in various triangulated surfaces [52]. | 13 |
| Figure 8. Shows the concept of ray casting in computer graphics. It shows a camera, a light source, a scene object and an image with the projected object [53]. | 14 |
| Figure 9. Illustrates the transformer architecture like proposed in "Attention is all you need" utilizing cross attention and self-attention [37]. | 15 |
| Figure 10. Illustrates a multi-head attention layer, a core component in transformer models used for various natural language processing (NLP) tasks such as machine translation and text summarization. The multi-head attention mechanism allows the model to focus on the most relevant parts of an input sequence for each word or token, enabling it to capture long-range dependencies within the sequence. | 16 |
| Figure 11. PoinTr pipeline: A partial point cloud is downsampled to extract local features around center points. Positional encoding is added before feeding these features to a transformer encoder-decoder for point proxy prediction. Finally, a two-stage refinement process with an MLP and FoldingNet completes the point cloud [38]. | 17 |
| Figure 12. Illustrates the methodological framework employed in this work. The left side (blue) depicts the data generation process, where high-fidelity 3D object models (meshes) and extracted point clouds are created. Subsequently, partial meshes and point clouds with added noise were generated to simulate real-world scenarios. The right side (green boxes) showcases the training and validation process for a PoinTr network and the generated data. | 19 |
| Figure 13. Illustrates the data generation process. Top left: Annotated CT data. Top right: Generation of meshes through marching cubes algorithm. Bottom left: Generation of complete point cloud through poisson disk sampling. Bottom right: Generation of partial views through raycasting. | 20 |
| Figure 14. Illustrates the noise generation process. For each point where a ray intersects an object, a Gaussian-sampled value with a standard deviation of 0.2 %, of the camera distance along the ray direction is added to the intersection point. | 22 |
| Figure 15. Ray casting results without noise (left) and with noise added (right). | 23 |
| Figure 16. Calculating camera distance for full liver coverage. Left side: Calculating of maximum mesh distance, Right side: Camera distance calculation based on maximum mesh distance. | 24 |

| | |
|---|----|
| Figure 17. Training data generation. The flowchart on the left illustrates how camera positions are generated in each step by varying two angles and utilizing golden angle sampling. The angles used for this are azimuth and pole distance angle as depicted on the right. | 24 |
| Figure 18. Test data generation. The figure shows three different views to reflect different surgeon views in open liver surgery..... | 25 |
| Figure 19. Flow chart of PoinTr training process. This flowchart illustrates the PoinTr training process. The blue boxes represent the standard training pipeline, while the green box highlight adaptations made specifically for training with ray-casted data. | 27 |
| Figure 20. Illustrates area percentage by distance and angle. This figure illustrates the percentage of visible area with respect to the complete object. The distribution of these values is shown for the two camera distances 1 m (at the top) and 2 m (at the bottom) for the camera positions at 45° (blue), 90° (orange) and 135° (green). | 31 |
| Figure 21. Illustrates the number of points in each partial point cloud relative to the visible area of the associated partial mesh. The distribution of these values is shown for the two camera-distances: 1 m (at the top) and 2 m (at the bottom) for the camera positions at 45° (blue), 90° (orange), and 135° (green). | 32 |
| Figure 22. Illustrates the complete mesh (top) and the corresponding partial views generated from different viewpoints (45°, 90°, and 135°, from left to right)..... | 33 |
| Figure 23. Shows the model's performance (reconstruction error) using boxplots. The turquoise boxes represent the clean model, while the orange boxes represent the model trained on noisy data. The x-axis categories represent camera distance (1 m and 2 m) with and without additional noise in the test data. Stars indicate statistically significant differences ($\alpha = 0.05$) between clean and noisy data performance for each category..... | 34 |
| Figure 24. Illustrates the main effects of the clean model. This plot depicts the main effects of individual categories, on the reconstruction performance of the model trained on clean data, as identified through a design of experiments analysis. Each category (from left to right: view point at 45°, 90°, 135°; camera distance at 1 m, 2 m; noise-level with noise, without noise) represented on the x-axis is investigated independently to determine its impact on the model's output. The vertical axis represents the main effect, visualized through a change of the chamfer distance, which signifies the average change in reconstruction error (lower is better)..... | 35 |
| Figure 25. Illustrates the main effects of the noisy model. This plot depicts the main effects of individual categories, on the reconstruction performance of the model trained on clean data, as identified through a design of experiments analysis. Each category (from left to right: view point at 45°, 90°, 135°; camera distance at 1 m, 2 m; noise-level with noise, without noise) represented on the x-axis is investigated independently to determine its impact on the model's output. The vertical axis represents the main effect, visualized through a change of the chamfer distance, which signifies the average change in reconstruction error (lower is better)..... | 36 |
| Figure 26. Illustrates the influence of object geometry on point cloud density captured by a virtual sensor (camera). The left side showcases a 135° viewpoint, highlighting surfaces in orange that are nearly parallel to the sensor rays in this view. The right side depicts the mesh and camera from a perpendicular viewpoint, emphasizing the near-parallel orientation of these marked surfaces relative to the sensor rays. | 40 |
| Figure 27. Illustrates a mesh generated from the LiTS dataset using CT scans with low voxel resolution. The low-resolution results in a mesh with a stepped surface, where the smooth anatomical features of the liver are not accurately captured. | 42 |

Figure 28. Illustrates planar surfaces with a stepped structure, highlighted in orange, extend over the visible mesh surface. These artifacts likely arise due to the presence of stepped surfaces within the training data. 43

Figure 29. Illustrated at the top a mesh excluding the tumor, resulting in a incomplete mesh. At the bottom, a liver mesh generated with liver and tumor together is shown. The tumor is marked in orange. 44

11 List of tables

| | |
|--|----|
| Table 1. Details of virtual pinhole camera parameters for ray casting scene | 21 |
| Table 2. Hyperparameters of the PoinTr network used in training process and reconstruction. | 27 |
| Table 3. Full factorial experiment design table. This table presents a full factorial design experiment, outlining all combinations of tested parameters to comprehensively analyze their potential influence on the reconstruction process. | 29 |
| Table 4. Traceability of the research question to related experiments and results. | 29 |
| Table 5. Overview of materials used for both training and test of the PoinTr framework..... | 30 |
| Table 6. Lists mean and standard deviation for area in percentage and points per area of all test data categories. | 32 |
| Table 7. Paired t-test results: Clean vs. Noisy Model Performance (n, t, p-value; $p < 0.05$ indicates statistical significance)..... | 34 |
| Table 8. This table visualizes the best reconstruction (lowest chamfer distance) for the test data category with camera distance 1 m and camera position 45° . The left side shows the ground truth data: the point cloud representation at the top and the mesh representation at the bottom. The right side displays the reconstruction results in two different views: the reconstructed point cloud visualized in blue points and a partially mesh that represents the partial point cloud that the network receives as input for reconstructing the complete object..... | 37 |
| Table 9. This table visualizes the worst reconstruction (highest chamfer distance) for the test data category with camera distance 1 m and camera position 45° . The left side shows the ground truth data: the point cloud representation at the top and the mesh representation at the bottom. The right side displays the reconstruction results in two different views: the reconstructed point cloud visualized in blue points and a partially mesh that represents the partial point cloud that the network receives as input for reconstructing the complete object..... | 38 |
| Table 10. Average points per area, number of points and average visible area for ground truth, training data, and testing data at 1 m and 2 m. | 41 |