PUCRS

ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

FELIPE ROQUE TASONIERO

SELF-ATTENTION FOR IMPROVING THE DIFFERENTIABLE RENDERING PIPELINE IN IMAGE 3D RECONSTRUCTION

Porto Alegre

2021

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF TECHNOLOGY
COMPUTER SCIENCE GRADUATE PROGRAM

SELF-ATTENTION FOR IMPROVING THE DIFFERENTIABLE RENDERING PIPELINE IN IMAGE 3D RECONSTRUCTION

FELIPE R. TASONIERO

Master Thesis submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Prof. Dr. Rodrigo C. Barros

Porto Alegre
2021

Ficha Catalográfica

T199s Tasoniero, Felipe Roque

Self-Attention for Improving the Differentiable Rendering Pipeline in Image 3D Reconstruction / Felipe Roque Tasoniero. – 2021.
84.
Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Rodrigo Coelho Barros.

1. Deep Learning. 2. 3D Reconstruction. 3. Computer Vision. 4. Transformers. I. Barros, Rodrigo Coelho. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecária responsável: Clarissa Jesinska Selbach CRB-10/2051

Felipe Roque Tasoniero

SELF-ATTENTION FOR IMPROVING THE DIFFERENTIABLE RENDERING PIPELINE IN IMAGE 3D RECONSTRUCTION

This Master Thesis has been submitted in partial fulfillment of the requirements for the degree of Master of Computer Science, of the Graduate Program in Computer Science, School of Technology of the Pontifícia Universidade Católica do Rio Grande do Sul.

Sanctioned on the 28th October, 2021.

COMMITTEE MEMBERS:

Profª. Dra. Soraia Raupp Musse (PPGCC/PUCRS)

Prof. Dr. Claudio Rosito Jung (PPGC/UFRGS)

Prof. Dr. Rodrigo Coelho Barros (PPGCC/PUCRS - Advisor)

ACKNOWLEDGMENTS

This research was conducted within a project supported by the Brazilian Informatics Law (Law nº 8.248 of 1991) and was developed over Agreement 001/2015 between Pontifícia Universidade Católica do Rio Grande do Sul and HP Brasil Indústria e Comércio de Equipamentos Eletrônicos Ltda.

Research supported by HP Brasil Indústria e Comércio de Equipamentos Eletrônicos Ltda. using financial incentives of IPI refund reggarding the Law (Law nº 8.248 of 1991)

SELF-ATTENTION FOR IMPROVING THE DIFFERENTIABLE RENDERING PIPELINE IN IMAGE 3D RECONSTRUCTION

RESUMO

Pesquisas recentes sobre modelos de Renderização Diferenciável relacionados à reconstrução 3D de imagens utilizam modelos totalmente convolucionais para extração de *features* ou para o processamento de decodificação. Por outro lado, várias tarefas de visão computacional como reconhecimento visual, segmentação, geração de imagens e detecção de objetos tiveram grande melhoria de desempenho ao fazer uso de modelos baseados em *self-attention*, conhecidos tradicionalmente como *Transformers*. Devido a tal sucesso, neste trabalho pretendemos explorar quatro diferentes abordagens de modelos baseados em *self-attention* para reconstrução implícita de objetos 3D. Em nossa primeira abordagem, implementamos as camadas de *self-attention* da SAGAN junto as camadas convolucionais; em nossa segunda abordagem, implementamos o modelo *patchwise self-attention* para substituir completamente o codificador convolucional. Em seguida, implementamos um modelo de Transformer chamado *Pyramid Vision Transformer* para substituir o codificador convolucional do modelo DVR; finalmente, em nossa quarta abordagem, implementamos o modelo *Nyströmformer* como um otimizador para reduzir o custo computacional e para melhorar a capacidade de extração de *features*. Considerando todas as abordagens, nossos resultados mostraram que podemos alcançar resultados competitivos usando *Transformers*, bem como adicionando um otimizador para reduzir seu custo computacional. Com a aplicação do modelo de otimização e consequente redução do custo computacional, foi possível modificar o módulo referente ao decodificador de forma a melhorar os resultados de reconstrução, alcançando melhorias de até $8,5\%$ em relação aos *baselines*.

Palavras-Chave: Aprendizado Profundo, Reconstrução 3D, Visão Computacional, Transformers.

SELF-ATTENTION FOR IMPROVING THE DIFFERENTIABLE RENDERING PIPELINE IN IMAGE 3D RECONSTRUCTION

ABSTRACT

Recent studies on Differentiable Rendering models related to 3D reconstruction focus on fully convolutional-based models for data feature extraction or for the decoding process. On the other hand, computer vision tasks such as image recognition, segmentation, image generation, and object detection is benefiting largely from using fully self-attention approaches known as Transformers. Due to the recent success of the Transformer backbone models applied to computer vision, in this work we aim to explore four different approaches of self-attention-based models for implicit 3D object reconstruction from images. In our first approach, we have implemented the SAGAN Self-Attention layers together with convolutions layers; in our second approach, we have implemented a patchwise self-attention model to completely replace the convolutional encoder; next, we have implemented a Transformer model called Pyramid Vision Transformer to replace the convolutional based encoder from the DVR model; finally, we have implemented the Nyströmformer model, an optimizer to reduce the computational cost and to improve the feature extracting capability. Considering all approaches, our results have shown that we can achieve competitive results by using Transformer models, as well as adding an optimizer to reduce the computational cost. By applying the optimization model and reducing the computational cost, it was possible to modify the decoder module to increase the reconstruction results, resulting in improvements of up to 8.5% compared to the baseline approaches.

Keywords: Deep Learning, 3D Reconstruction, Computer Vision, Transformers.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

2D. – Two-dimensional

3D. – Three-dimensional

ANN. – Artificial Neural Network

CNN. – Convolutional Neural Network

DL. – Deep Learning

LSTM. – Long Short-Term Memory

ML. – Machine Learning

MLP. – Multilayer Perceptron

MVS. – Multi-View Stereo

NLP. – Natural Language Processing

RNN. – Recurrent Neural Network

ViT. – Vision Transformer

DVR. – Differentiable Volumetric Rendering

FLOP. – Floating Point Operations

SAGAN. – Self-Attention GAN

PVT. – Pyramid Vision Transformer

CONTENTS

1. INTRODUCTION

One of the long-standing goals of computer vision is recovering 3D information from the image capturing process [1]. Classic multi-view stereo (MVS) methods [6] usually match features between neighboring views for the reconstruction process, usually requiring a high degree of engineering. More recently, researchers start to investigate the performance of learning-based approaches for 3D reconstruction [10, 35, 36], with the disadvantage that most of the proposed approaches are highly dependent on ground truth meshes and/or are highly memory-consuming.

The Differentiable Rendering (DR) approach for 3D reconstruction introduces a new strategy for the learning-based reconstruction process, where there is no need to use meshes as ground truth. There are many DR approaches that use different types of data such as voxels [39], meshes [58], and point clouds [49] for reconstruction and these type of data can also be implicitly generated. The implicit 3D reconstruction can be seen as a continuous decision boundary of a classifier function that can estimate a 3D surface of an object [45], and it can reduces the memory footprint for reconstructed objects during the training process. Most of the DR approaches make use of convolutional neural networks for the encoding and decoding process, allowing the multi-view and single-view reconstruction. In this work we choose to use as a baseline, a DR model that implicitly generates a 3D object in both multi-view and single-view reconstruction process.

As an alternative to convolutional neural networks, self-attention based models have started to gain popularity in the past couple of years due to their successful results in computer vision tasks such as image recognition [15], object detection [5], image generation [70], segmentation [67], and visual question answering [55]. More recently, due to the recent success of fully self-attention based models (known in the community as “Transformers”) for tasks related to Natural Language Processing (NLP) [59, 14], many researchers started to explore these models within Computer Vision. Some recent studies have explored the self-attention modules [71] to better understand the possibility of replacing convolutional layers with self-attention. Last year, new models have emerged entirely based on the transformer architectures for image feature extraction as in Dosovitskiy *et al.* [15], which proved capable of matching the results obtained by convolutional models in Computer Vision. On the other hand, Transformers are known for having a large time and space complexity due to the attention mechanism, which is typically quadratic in the input size. To solve this problem, different approaches have emerged [4, 60, 72, 66] to reduce the quadratic complexity to an approximately-linear complexity.

Considering the lack of studies related to the usage of self-attention networks on 3D implicit reconstruction models, in this work we seek to answer the following research questions: 1) *Can we improve object reconstruction using a fully self-attention neural network?*;

and 2) *Is it possible to improve object reconstruction using attention but with a sub-quadratic computational cost?* To answer these questions, we have decided to employ four different self-attention approaches for implicit 3D object reconstruction using the DR model proposed by Niemeyer *et al.* [45] as the baseline work.

The remaining of this work is organized as follows. Chapter 2 presents the background for machine learning, deep learning, and 3D reconstruction from images. Chapter 3 shows related work on 3D reconstruction and self-attention models. In Chapter 4 we present our first and second self-attention models, while in Chapter 5 we present our third and fourth approaches. Finally, in Chapter 6, we present the conclusions of our investigations and experiments, and we point to some interesting future work possibilities.

2. BACKGROUND

This chapter presents a brief review of the theoretical foundation of Machine Leaning (ML) (Section 2.1) and Deep Learning (DL) (Section 2.2). In Section 2.2.3, we present a background on Transformers models, whereas in Section 2.3 we present both traditional and learning-based methods for 3D object reconstruction from images.

2.1 Machine Learning

Machine Learning is strongly linked to several other fields of research such as Artificial Intelligence, Probability and Statistics, Computational Complexity Theory, Information Theory, Philosophy, among others. It is considered the study of how a computer program learns through experience. A more precise definition following [43] is given as follows:

Definition: A computer program learns through an experience E in relation to a given class of tasks T and a performance measure P, if the performance for the tasks T measured by P improves with E.

To perform a given task T, a program learns a mathematical function, in an approximate way, which represents a distribution of a set of available data. A dataset is composed of instances, which consist of a set of attributes and possible associated labels. Usually, this set is divided into three parts, namely: a training set, a validation set, and a test set. The training set is used by the program as learning data to estimate a mathematical function that models the problem. The validation set is used to measure performance P and to select one of the trained models. Finally, the test set is used to analyze the generalizability of the model, and this set is never used by the models during the training stage.

Traditionally, machine learning algorithms are divided into three main categories: supervised learning, unsupervised learning, and semi-supervised learning. There is also reinforcement learning, though it is usually considered to be a totally distinct area from the others, often being approached by the literature of multi-agent systems.

2.1.1 Supervised Learning

According to [44], supervised learning aims to learn how to map input data $\mathbf{x}$ that result in output data y given a labeled set of input and output $\mathcal{D} = \{(\mathbf{x}_i, \mathrm{y}_i)_n\}$ where $\mathcal{D}$ is called the training set and $n = \{1, ..., N\}$ is the number of training examples. Also according to [44], in a standard approach, each input data $\mathbf{x}_i$ is a vector of numbers in a space of dimension D that can represent, by example, data related to a person's height and weight. These

representations are commonly called *features*, or attributes. However, in some approaches, the input data $\mathbf{x}_i$ can have a more complex object structure, such as images, sentences, time series, graphs, etc. Some common applications related to supervised learning can be data classification and regression.

2.1.2 Unsupervised Learning

Neural network models that use approaches based on unsupervised learning can generate output data without the need to use labeled input data. According to [44], the goal of unsupervised learning is to discover an "interesting structure" in a given dataset. As opposed to supervised learning, it is not said which result is desired for each input data. Instead, we formalize our task as a density estimate, that is, we build our model in the form $p(\mathbf{x}_i|\theta)$, where θ is the set of desired parameters, instead of the form $p(\mathrm{y}_i|\mathbf{x}_i, \theta)$ used in supervised learning [44]. Some of the tasks related to unsupervised learning are: discovering *clusters* and discovering latent factors.

2.1.3 Semi-supervised Learning

According to [9], semi-supervised learning is considered as an intermediate approach between supervised learning and unsupervised learning. In addition to unlabeled data, the algorithm is provided with some supervisory information, but not necessarily for all examples. Often, this information will be the targets associated with some of the examples. In this case the dataset $X = \{\mathbf{x}_i\}_{i\in\{1,...,n\}}$ can be divided into two parts: the points $X_l := (x_1, ..., x_l)$ for which the labels $Y_l := (y_1, ..., y_l)$ are given, and the points $X_u := (x_{l+1}, ..., x_{l+u})$ for which the labels are not known.

2.2 Deep Learning

Conventional Machine Learning techniques were limited in their ability to process natural data in its raw form. For decades, building a pattern recognition system or Machine Learning system required careful engineering and considerable mastery of the subject to devise a features extraction architecture that could transform this raw data into an appropriate representation to be used by the system or a vector of features in which a learning system could classify or detect input patterns [32].

Deep Learning Methods are learning methods with multiple levels of representation, obtained through the composition of non-linear modules capable of transforming the

representation at a given level into a more abstract representation [32]. Through these transformations, Deep Learning methods are able to learn extremely complex functions, making it possible to achieve state-of-the-art results for various tasks, such as image recognition [65], speech recognition [21], generation of texts [23], and image generation [54], in addition to several other tasks. Therefore, Deep Learning methods make use of Artificial Neural Network architectures with several representative layers capable of dealing with unstructured data.

Artificial Neural Networks (ANN) are algorithms used in the field of machine learning and deep learning inspired by the biological neural connections that constitute the brains of animals [41]. Its structure is composed of a set of connected nodes called artificial neurons. Each connection, similar to synapses generated in a brain, is capable of transmitting a signal to other neurons, which, upon receiving these signals, process and transmit them again to other neurons in a forward direction. In ANN, these signals correspond to representations of input data computed by some non-linear function, and the connections that transmit these signals correspond to weights capable of reducing or increasing the strength of each signal in a learning process.

The best-known ANN architecture is called Multilayer Perceptron (MLP). The focus of this architecture is to approximate a function f, for example a classifier, which maps an entry x to a category y as follows $y = f(x;\theta)$, where θ are the values of the parameters that best result in the approximate function [20]. Due to the natural advance in these models requested by more advanced tasks related to text and image processing, other types of deep models have emerged, as the Convolutional Neural Networks, Recurrent Neural Networks, and more recently the Transformers models.

2.2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN) [34] are neural networks capable of processing data with a matrix geometry. Its name indicates that the network is capable of applying mathematical operations such as convolutions [20], which are linear transformations capable of being applied to matrix data structures. The convolution operations performed in this type of architecture are given by the use of element-by-element operations through the multiplication of matrices with the same dimension. For this, weight matrices known as Kernels are used. Kernels then traverse the entire input array. Architectures like the one in Figure 2.1 perform several convolution operations, enabling the extraction of features at different levels until these feature maps reach a certain dimension. Furthermore, convolution sequences can be interspersed by activation functions, such as ReLU, and by downsizing layers, such as the Pooling layer, as in the AlexNet architecture [31].

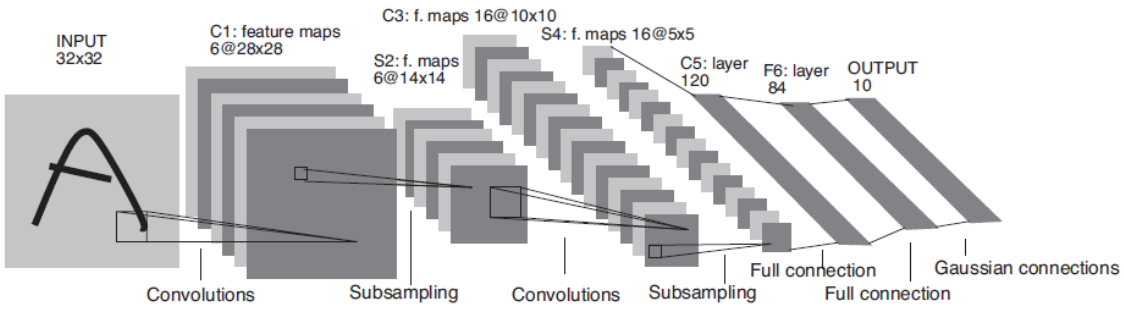

Figure 2.1 – An example of a CNN architecture (LeNet-5)[34] for classification of handwritten digits images (image from LeCun et al. [34]).

2.2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNN) [51] are a family of neural networks proposed for sequential data processing [20]. The purpose of this type of network is the sharing of parameters across the network over time. Traditionally, networks of the RNN type are loop-based networks capable of making information persist in their execution. One of the most temporary architectures in the RNN family is the Long Short-Term Memory (LSTM) [26] module.

This module can solve one of the main problems related to traditional recurring networks, which are like long-term dependencies. As opposed to a traditional RNN, LSTM modules have four internal neural networks. This type of structure allows each cell to be able to store long-term information without knowing the time, and also allows the information in each cell of type LSTM to be read, written, and erased.

2.2.3 Transformers

An alternative to RNN models is the self-attention [59] module. The essence of the self-attention modules (Figure 2.2) is that they are themselves an imitation of the mechanism of human vision. When the human vision engine detects an item, it usually does not sweep the entire scene end-to-end. Instead, it will always focus on a specific part according to the person's need. When a person notices that the object they want to pay attention to usually appears in a specific part of a scene, they will learn that, and in the future, they will look there for the object (their attention will switch to that area).

The attention mechanism of these modules is then given by the softmax operation as in Equation 2.1

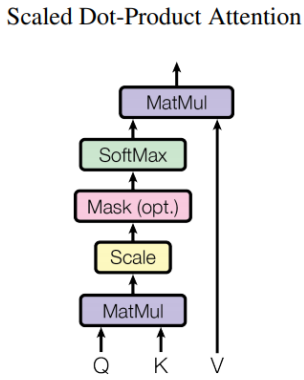

Figure 2.2 – An example of attention mechanism [59] (image from Vaswani et al. [59]).

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

where Q, K, and $V \in \mathbb{R}^{n \times m}$ are the input feature matrices, and d_k is the dimension of K. This function results in a matrix of weights that have a value distribution between 0 and 1. These values are then applied to the V matrix.

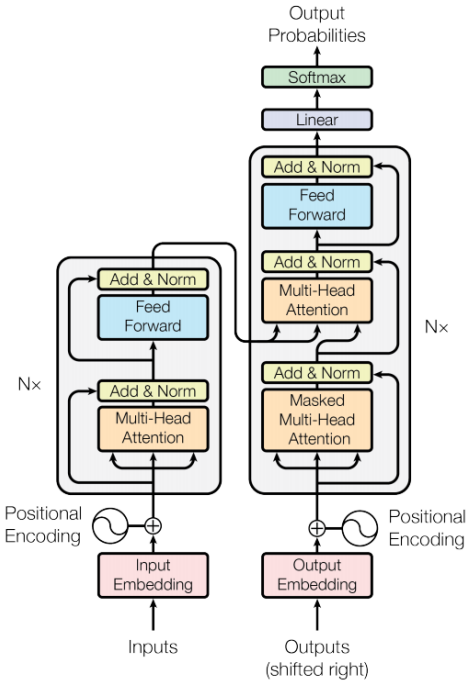

Figure 2.3 – Transformer model proposed by [59] (image from Vaswani et al. [59]).

The Transformers [59] models are considered as a sequence of attention blocks, where these blocks are composed of an attention mechanism with a residual connection followed by a linear layer with another residual connection (Figure 2.3). The Transformers model proposed by Vaswani et al. [59] has shown great promise in NLP tasks, and recently this transformer architecture was extended to computer vision tasks.

2.3 3D Reconstruction

The reconstruction of three-dimensional geometries is a technique for retrieving information about a three-dimensional structure or space based on direct measurements or through methods based on stereo images [12]. The 3D reconstruction area is directly related to the classical Computer Vision area and several types of researches have been implemented in this field for decades. Due to the evolution of 3D reconstruction techniques and also systems capable of performing large amounts of computational calculations in less time, its applications have been expanded to several areas such as engineering, architecture, and medicine, in addition to more specific applications such as 3D printers, robotics, and autonomous cars. However, currently, 3D reconstruction is not limited only to traditional methods, but several ways to perform 3D reconstruction using approaches based on Deep Learning are also being researched. Next, some traditional 3D reconstruction techniques will be discussed, as well as current learning-based methods and their characteristics.

2.3.1 Traditional Methods

Traditional 3D reconstruction methods can be divided into two main categories: Active Methods and Passive Methods [16]. The main characteristics of active methods are the use of devices to obtain data from the distance and shape of objects mechanically, that is, using equipment prepared with sensors capable of interpreting the radiance reflected by the objects, such as laser or ultrasound measuring equipment. However, due to its implementation complexity, active methods will not be discussed in this work. Passive methods, on the other hand, have as their main characteristic the non-direct interference with the objects to be reconstructed, thus, how data are obtained for the reconstruction passively is given through the use of images. A more precise definition of image-based 3D reconstruction methods according to Furukawa *et al.* [16] is given as follows:

Definition: Given a set of photographs of objects or scenes, estimate a 3D form that comes closest to explaining these photographs, assuming knowledge about materials, viewpoints, and lighting conditions.

Among the passive methods, we can cite methods based on the use of one or more images, such as the Shape by Shade, Photometric Stereo and Shape by Texture methods, and also methods based on multiple viewing angles using multiple images, such as Multi-View Stereo (MVS). The Shape by Shading method is used to reconstruct a 3D geometry by analyzing the variation in the intensity of the luminosity of an image. Due to the fact that as the normal vector varies on a three-dimensional surface, hence the brightness of this image

also varies as a function of the angle of the normal vector orientation of a given surface location with incident illumination in this region [12].

Assuming a knowing distant light source, an observer, and assuming that the object surface is a uniform albedo, the variation in radiation is considered as a local function of the surface orientation,

$$I(x, y) = R(p(x, y), q(x, y)) \tag{2.2}$$

where $R(p, q)$ is the reflectance map, that is, a proportion of the incident radiation flux and the radiation reflected by the surface, and $(p, q) = (z_x, z_y)$ are the depth maps. In order to be able to reconstruct the surface, it is then necessary to discover the values of the orientation fields (p, q) and carry out the integration of these values in space. According to [12], considering that the surface is diffuse (Lambertian) it is possible to assume that the reflectance map is the scalar product of the normal surface. $\hat{\mathbf{n}} = (p, q, 1)/\sqrt{1 + p^2 + q^2}$ and the direction of the light source $\boldsymbol{\upsilon} = (\upsilon_x, \upsilon_y, \upsilon_z)$,

$$R(p, q) = max\left(0, \rho\frac{p\upsilon_x + q\upsilon_y + \upsilon_z}{\sqrt{1 + p^2 + q^2}}\right) \tag{2.3}$$

where ρ is the surface reflectance factor (albedo). So, as the values related to light source and observer are known, it is possible to estimate through the Equations (2.2-2.3), the values of (p, q) using, for example, the nonlinear least-squares method.

The Photometric Stereo method is very similar to the previous method, but its difference is due to the possibility of using multiple light sources that can be selectively turned on or off, thus being considered a more reliable method compared to the Shading from Shape method [12]. The Shape by Texture method, on the other hand, requires several processing steps, including the extraction of repeated patterns or the measurement of local frequencies to calculate local deformations, and a subsequent step to infer the local orientation of the surface [12].

However, the previous three methods only have the ability to reconstruct a three-dimensional projection of a single view of the image, making the reconstruction process more complex. One of the most used solutions today for a complete reconstruction of a three-dimensional object or scene is the MVS. From a set of images in different views, it is possible to generate a correspondence between pixels of each image using a MVS approach. According to Furukawa *et al.* [16], given a pixel in an image, finding the corresponding pixel in another image requires two ingredients:

- An efficient way to generate possible candidate pixels in other images.
- A measure to determine the probability that a given candidate is the right one.

Given then a pixel in an image, we can consider that assuming an optical ray that passes through that pixel and the center of the camera in an image, the corresponding pixel in another image can only also be over the projection of this optical ray, as can be seen in Figure 2.4. The MVS approach has the advantage of being easy to implement some of its standard algorithms, in addition to rebuilding objects with excellent quality according to the number of views used in the process. However, there are still some limitations of this method. According to Furukawa *et al.* [16], some cases where the reconstruction process is flawed are: non-Lambertian surfaces (e.g. metallic or transparent surfaces), surfaces with little texture, and surfaces with very fine structures. These disadvantages are due to the difficulty in finding correspondences between pixels between pairs of images.

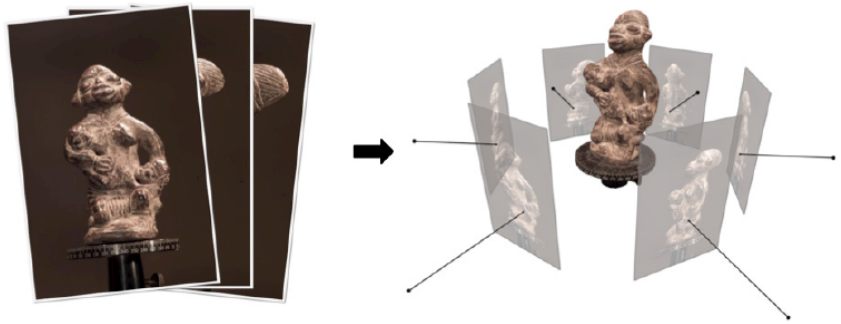

Figure 2.4 – 3D reconstruction using the MVS method. Given a set of images (left), it is possible to estimate the three-dimensional geometry (right) of the object or scene of these images, finding the correspondence between the pixels of each image [16]

2.3.2 Learning-based Methods

The use of methods based on machine learning for three-dimensional reconstruction of objects or scenes is based largely on the intrinsic capacity of human beings to be able to infer dimensions and geometries of these objects using only their vision and their prior knowledge of other objects or similar scenes. In addition, the growing amount of generated data sets and the advancement of new technologies capable of assisting in computing a large volume of information enabled research to be carried out in the field of 3D reconstruction using traditional approaches in the area of deep learning.

Although this type of approach is very recent compared to traditional methods, several studies have shown very promising results. The first works relating to this type of approach date back to 2015 [22]. Since then, the number of publications has increased significantly, always bringing new features and improvements related to the reconstruction of three-dimensional models, and due to this increase, companies such as NVIDIA and Facebook have also brought advances in this area, such as libraries Kaolin [28] and PyTorch 3D

[48], capable of helping to speed up the 3D reconstruction process. This advance is mainly due to the ability of this type of approach to infer a complete volumetric representation of the object or scene without the need to use a lot of information as input. With the use of deep neural networks, according to [22] we can formulate the learning-based 3D reconstruction problem as follows:

Definition: Let $\mathbf{I} = \{I_k, k = 1, ..., n\}$ be a set of one or more RGB images of one or more objects X contained in those images. The 3D reconstruction is performed through the process of learning a predictor f_θ capable of inferring a shape $\hat{X}$ that is as close as possible to a unknown shape X. In other words, the function f_θ is the minimizer of the error function $\mathcal{L}(\mathbf{I}) = d(f_\theta(\mathbf{I}), X)$, where θ is the parameter set of f, and $d(.,.)$ is a measure of distance between the target shape X and the reconstructed form $f(\mathbf{I})$.

There are in the literature different architectures and data structures used for 3D reconstruction using deep neural networks and traditional methods. Currently, many researchers are looking for improvements in learning based 3D reconstruction using only RGB images, due to its ease of use in different types of neural networks and the great availability of data to be used in training these networks. However, there are other types of structures used, as indicated in Figure 2.5. These structures can be used both as input and output in some models of neural networks, but the use of these types of structures as input ends up bringing great difficulties in terms of memory and adaptation in several existing algorithms.

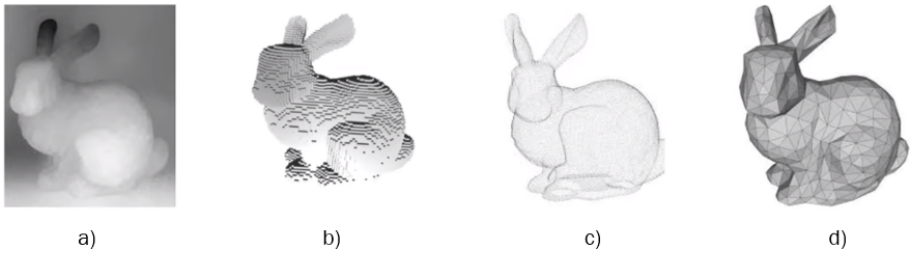

Figure 2.5 – Representation of data structures used in learning-based 3D reconstruction methods: a) depth-maps, b) voxels, c) point clouds, and d) meshes.

Depth-Maps type structures are images that contains information about the distance of the surface of an object from a camera viewpoint. Depth-Maps are commonly used as input data because in addition to the information of the RGB channels of the images, this type of structure has depth information, which in turn helps in the quality of reconstructed 3D models, but its disadvantage is the limited amount of data available for training. Voxel-type structures are considered three-dimensional representations of pixels. This type of structure has the advantage of being easy to be used both as input and output data in many deep neural network architectures, but it has a high memory consumption for reconstructing 3D models with a high degree of structural detail. The Point-Clouds type structure are a collection of points, positioned in a 3D space, that represents the surface of and object. This type of structure is little used in learning-based 3D reconstruction because it is difficult to imple-

ment in many neural network architectures, and also has a high memory cost. Its advantage lies in the quality of the surface representation of the 3D model. Finally, the mesh-type structure is a collection of vertices and faces that can represent the boundary of an object, and has the advantage of its low memory consumption, but there are still many difficulties in using this type of structure as an input in several neural network architectures. Due to these reasons, this work will present approaches that use RGB images as input to learning-based 3D reconstruction models and don't need 3D models as labels for training process.

Another important factor to be considered is the amount of data used for training deep neural networks for 3D reconstruction. There are currently several datasets available, such as those in Table 2.1. Datasets with a large volume of information such as images and 3D models referring to these images help in the training of models based on supervised learning, which are the most used in 3D reconstruction research. In addition, there are specific datasets for certain types of tasks, such as data for reconstructing human bodies and data for reconstructing certain types of objects.

Table 2.1 – Most popular datasets used in learning-based models for 3D reconstruction.

	N° Img.	**N° Categ.**	**Obj. per Img.**	**3D Ground Truth**
Pascal 3D [64]	30,899	12	Multiple	Yes
ShapeNet [8]	51,300	55	Single	Yes
ModelNet [63]	127,915	662	Single	No
Pix-3D [56]	9,531	9	Single	Yes

It is also possible to separate 3D reconstruction methods based on learning by the type of neural network architecture according to the data structure used. According to Gao *et al.* [17], 3D reconstruction methods from RGB images are capable of using most types of deep neural network architectures, such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). Other types of data structures like Point Clouds and Meshes end up being restricted to a few types of architectures, such as Variational Autoencoders, and Graph CNNs. In addition, methods that use RGB images for 3D reconstruction also benefit from the ability to use more than one type of network architecture in a single model, making use of the advantages of each architecture to solve problems related to 3D reconstruction, such as thin details in the 3D structures or voids contained in these 3D shapes. Finally, recent work has proposed a new method for 3D reconstruction, known as Differentiable Rendering. The main advantage of this method is that it doesn't need to use a 3D object as the ground truth for learning-based models. Kato *et al.* [29] presents a differentiable rendering model that can take different types of data as shape and camera parameters as input and outputs an RGB image or Depth image. Also, as presented in Kato *et al.* [29] a differentiable renderer computes the gradients of the output with respect to the input parameters to optimize a loss function, but the computation of the gradients should be accurate enough to propagate meaningful information required to minimize the objective function.

3. RELATED WORK

In this section, we briefly present different strategies that we could find in the literature related to the following topics: self-attention-based networks used in the 3D reconstruction pipeline for computer vision tasks; different transformer backbone models for vision tasks; and finally, transformer models that aim to optimize the computational cost of the attention mechanism.

Self-Attention in 3D Reconstruction: studies that present self-attention-based models for 3D reconstruction are the work from Salvi *et al.* [52], and from Lin *et al.* [37]. In Salvi *et al.* [52], the authors explore the usage of the self-attention module proposed by Zhang *et al.* [70] inside the encoder of the model proposed by Mescheder *et al.* [42] for 3D object reconstruction. On the other hand, in Lin *et al.* [37], the authors present a new method for 3D hand reconstruction in the wild, using a self-attention-based encoder-decoder network.

Vision Transformer Backbone: the work proposed by Carion et al. [7] and by Dosovitskiy et al. [15] present new learning models based on self-attention, but with similar architectures to the Transformers models used in NLP tasks, and in addition, they compared those models with Convolutional Neural Networks. In Carion et al. [7], a Transformer model called DETR is proposed for object detection. This model has as characteristics an encoder-decoder type architecture, in which the encoder is in charge of extracting information from a set of features processed by a Convolutional Neural Network, and the decoder is in charge of using the information extracted by the encoder in order to detect a certain number of objects in an image

Unlike the model proposed by [7], the work published by Dosovitskiy et al. [15] presents a Vision Transformer (ViT) architecture capable of being used as a backbone in neural network models for image classification. The ViT model has a structure supported only by an encoder, having as its main characteristic the use of image patches as input and a positional encoding system to ensure the extraction of information from a patch referring to its position concerning the original feature. The experiments performed by Dosovitskiy et al. [15] were related to image classification for different datasets, comparing the proposed ViT model with ResNet models.

In [68] a new Transformer model is presented, called Token-To-Token (T2T), and it is based on the model proposed by Dosovitskiy et al. [15]. The T2T model aims to add one more step of dividing the network input patches, further reducing the total size of the input feature sequence in the self-attention modules. In addition, other encoder modifications were made. According to [68], the backbone of the ViT model proposed by Dosovitskiy et al. [15] had a much larger number of parameters than some convolutional models. The T2T model has a backbone structure similar to ViT, but with 2 more layers and with smaller

hidden dimensions. This modification made the structure of the T2T model comparable to other convolutional models in terms of number of parameters.

As in Yuan et al. [68], the work proposed by Touvron et al. [57] was also based on the model proposed by Dosovitskiy et al. [15], but a new learning technique was used to improve the results in image classification, and also to produce a more efficient model, that is, a smaller model with less computational cost. The approach used by Touvron et al. [57] was the implementation of a training process known as Knowledge Distillation.

In the work proposed by Wang et al. [61] and by Jaegle et al. [27], new Transformers models that could be used as a backbone for various tasks in the field of Computer Vision were proposed. In Jaegle et al. [27] a Transformer model is proposed, and its structure is based mainly on two components: 1) a cross-attention module, where a byte array (e.g., pixels of an image) and a latent array are mapped to a latent array, and 2) a Transformer tower that also maps a latent array to a latent array. In the experiments performed by Jaegle et al. [27], analyses were performed in several tasks using the developed Transformers model.

The work proposed by Raghu et al. [46] aimed to explore the ability of Transformer models to learn image representations as in convolutional models. In Raghu et al. [46], the internal representations of each layer generated by the ViT model and by the ResNet model were analyzed individually. The results proposed by Raghu et al. [46] showed that the representations computed by the ViT model in the initial layers were different from those computed by the ResNet model and that the ViT model was able to propagate more information in the initial layers than the ResNet model. Furthermore, analyzing the results obtained regarding the transport of local and global information by the models, it was observed that the ViT model accesses more global information from the input data in the first layers of the network than the ResNet model.

Transformer Optimization: some recent work have proposed different approaches for transformer optimization, i.e., reduce its computational cost related to the self-attention mechanism. The Longformer [4] and Big Bird [69] models have a similar approach to reducing the spatial and temporal complexity of self-attention mechanisms. Longformer [4] has a sliding local window attention mechanism, but it depends on the depth of the neural network to "recover" some information from initial layers. In addition, they show that adding a global attention mechanism could still reduce the complexity of the mechanism, but it requires some engineering for choosing special attention points for self-attention computation. The Big Bird [69] model is similar to Longformer [4], but they add a new mechanism called random attention.

The Linformer [60] model has reduced the computation by approximating the self-attention mechanism to a low-rank matrix, i.e., a matrix projection in a lower dimension that contains similar information, and more recently the Performer [72] model has approximated a self-attention mechanism by decomposing the softmax computation using a kernel function that approximate the inner product.

4. SELF-ATTENTION BASED 3D RECONSTRUCTION APPROACH

Inspired by the proposed approach of Salvi *et al.* [52] that uses the self-attention module in conjunction with a CNN for implicit 3D reconstruction, as well as by the advantage of new approaches of fully self-attention-based neural networks in Computer Vision, in this chapter we present the following content: Section 4.1 presents the baseline model for 3D reconstruction that we use as our pipeline for all our experiments; Section 4.2 presents our first approach using self-attention layers in conjunction with convolutional layers, while Section 4.3 presents a full self-attention encoder. Finally, in Section 4.4 all the experimental setup is presented, and in Section 4.5 we report our results for these two proposed approaches.

4.1 Baseline

In this work, we have used the DVR (Differentiable Volumetric Rendering) model proposed by Niemeyer *et al.* [45] as our baseline for 3D reconstruction. DVR is an implicit object mesh and texture reconstruction model that uses the volumetric rendering process to predict the object depth map from an image. The decision to use the DVR model is mainly due to: 1) the use of volumetric rendering based on the rasterization process, which makes the process inherently differentiable, that is, it is possible to use only images and information of the positions of objects to have an end-to-end learning process, no longer being necessary to use meshes as supervision; 2) there is no need to condition the representation of texture in geometry, i.e., the model shares parameters of both geometry and texture; and 3) the DVR model makes it possible to reconstruct 3D objects for single or multi-view images using the same architecture.

As in Mescheder *et al.* [42], the model proposed by Niemeyer *et al.* [45] represents the 3D shape of an object implicitly given by an occupancy function as follow:

$$f_\theta : \mathbb{R}^3 \times \mathcal{X} \rightarrow [0, 1] \tag{4.1}$$

where for each point $\mathbf{p} \in \mathbb{R}^3$ in a tridimensional space, the occupancy function $f_\theta(\mathbf{p}, \mathbf{z})$, where $\mathbf{z}$ are the input images, gives for each point a probability between 0 and 1, similarly to a classification neural network. The surface of an object is determined by $f_\theta = \tau$ where $\tau \in [0, 1]$ is a pre-defined parameter that determines the surface level of an object by indicating if a given value is located inside or outside of an object. Once the surface points are defined, it is possible to extract the object mesh using the marching cube algorithm [40].

Similarly, the DVR [45] model is also capable of extracting the texture information of an object as follow:

$$\mathbf{t}_\theta : \mathbb{R}^3 \times \mathcal{X} \rightarrow \mathbb{R}^3 \tag{4.2}$$

which returns the information of the colors from RGB channels for each point $\mathbf{p} \in \mathbb{R}^3$ in a tridimensional space. The values of the texture $\mathbf{t}_\theta$ of an object are given by the object surface ($f_\theta = \tau$). Both occupancy probabilities f_θ and texture field $\mathbf{t}_\theta$ are given by the same neural network.

The DVR [45] pipeline initially processes an image by an encoder where its features are extracted, and in parallel, an initial set of N points relative to random positions in a three-dimensional cartesian space is sampled from latent space. Then, both the set of cartesian points and the image features are processed by a decoder, where the output is a set of N cartesian points with information related to the occupancy function and a set of N points with information related to texture.

The encoder-decoder architecture of the DVR [45] model is illustrated in Figure 4.1, where the encoder is a ResNet [24] neural network and the decoder is a fully-connected neural network with residual connections. The original encoder is made of five ResNet-18 [24] layers and a fully-connected layer with a dimension of 256, and the output of the encoder are passed through five fully connected blocks with residual connections using a ReLU[19] activation function and a hidden dimension of 512 for 2.5D supervision or 128 for 2D supervision. The output of the final fully-connected block is then passed through one fully-connected layer with an output of dimension 4, one of them for the occupancy probabilities and the other three for the RGB color values.

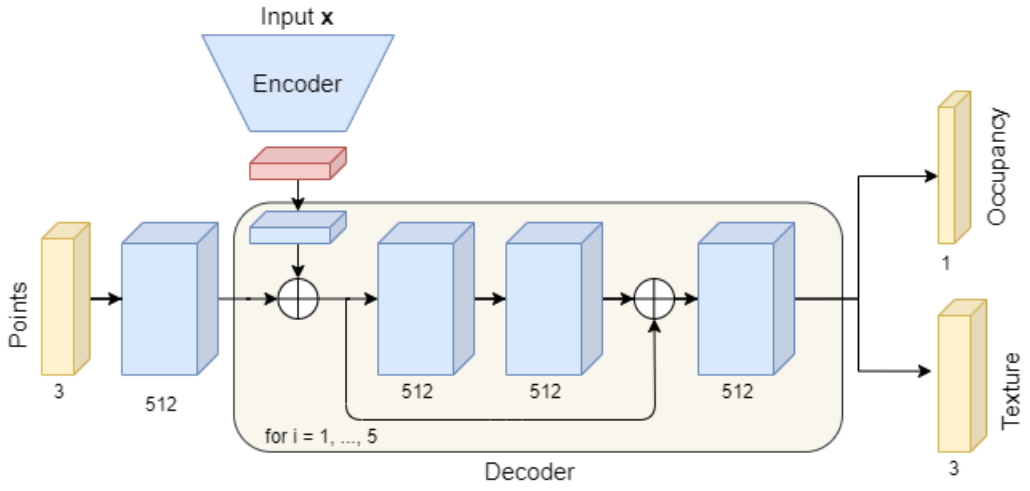

Figure 4.1 – The encoder-decoder pipeline of the DVR model [45] for texture and occupancy points extraction. The original encoder is composed of a ResNet-18 model that has an output of 256-dimension following by a linear block with a 512-dimension for the 2.5D supervision process (RGB and depth supervision), or a linear block with a 128-dimension for 2D supervision process (only RGB supervision). The decoder is composed of fully-connected layers with residual connections. The output is a single points matrix with a depth dimension of 4, one for ocuppancy points and the other three for texture points.

Next, with the generated occupancy and texture information, the network starts to predict the depth map of an object. As described in Niemeyer *et al.* [45], by sampling points as pixels in a 2D plane given by a camera matrix information, lines are sampled using the ray casting method from the camera location through each sampled point in the plane. For each ray, they search for points on the ray that indicates the change from free space to occupied space by evaluating the occupancy prediction f_θ. Those changes on each sampled ray can be used to find the depth values corresponding to a given camera matrix. In the next step, each surface depth point is unprojected and evaluated using the texture field information at the given 3D location. The resulting 2D rendering is compared to the ground truth image. If the ground truth depth maps are available, it is possible to define a loss directly on the predicted surface depth.

The learning process proposed by Niemeyer *et al.* [45] just needs the information from the 2D image, allowing the network to learn by the given reconstruction loss:

$$\mathcal{L}(\hat{\mathbf{I}}, \mathbf{I}) = \sum_{u} ||\hat{\mathbf{I}}_\mathbf{u} - \mathbf{I}_\mathbf{u}|| \tag{4.3}$$

where $\mathbf{I}$ is the observed image, $\hat{\mathbf{I}}$ is the rendered image, $\mathbf{u}$ indicates the image pixel and $||.||$ is the photo-consistency measure as $l_1 - norm$. To minimize the loss function, Niemeyer *et al.* [45] proposed a approach for the rendering process given $\hat{\mathbf{I}}$ and $\mathbf{u}$, and also find a closed analytic expression to compute the gradients related to network parameters θ.

To obtain the gradients of $\mathcal{L}$ with respect to network parameters θ, it is first used the multivariate chain rule:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \sum_{u} \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{I}}_\mathbf{u}} \cdot \frac{\partial \hat{\mathbf{I}}_\mathbf{u}}{\partial \theta} \tag{4.4}$$

and by exploiting $\hat{\mathbf{I}}_\mathbf{u} = \mathbf{t}_\theta(\hat{\mathbf{p}})$ as presented in Niemeyer *et al.* [45], it is possible to obtain

$$\frac{\partial \hat{\mathbf{I}}_\mathbf{u}}{\partial \theta} = \frac{\partial \mathbf{t}_\theta(\hat{\mathbf{p}})}{\partial \theta} + \frac{\partial \mathbf{t}_\theta(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \cdot \frac{\partial \hat{\mathbf{p}}}{\partial \theta} \tag{4.5}$$

where $\hat{\mathbf{p}}$ denotes the first ray casting point of intersection with the isosurface $\{\mathbf{p} \in \mathbb{R}^3 | f_\theta(\mathbf{p}) = \tau\}$. For a camera located at $\mathbf{r}_0$, the ray can be described, for any pixel $\mathbf{u}$, by $\mathbf{r}(d) = \mathbf{r}_0 + d\mathbf{w}$, where $\mathbf{w}$ is the vector connecting $\mathbf{r}_0$ and $\mathbf{u}$. Since $\hat{\mathbf{p}}$ lies on $\mathbf{r}$ as described by Niemeyer *et al.* [45], there exist a depth value $\hat{d}$, such that $\hat{\mathbf{p}} = \mathbf{r}(\hat{d})$, and $\frac{\partial \hat{\mathbf{p}}}{\partial \theta}$ can be reformulated as:

$$\frac{\partial \hat{\mathbf{p}}}{\partial \theta} = \frac{\partial \mathbf{r}(\hat{d})}{\partial \theta} = \mathbf{w} \frac{\partial \hat{d}}{\partial \theta} \tag{4.6}$$

Then, for computing the gradient of the surface depth $\hat{d}$ *wrt.* θ, Niemeyer *et al.* [45] exploited the *Implicit Differentiation* as in Walter Rudin *et al.* [50]. So, by differentiating

$f_\theta(\hat{\mathbf{p}}) = \tau$ on both sides with respect to θ, Niemeyer *et al.* [45] have demonstrated that it is possible to obtain:

$$\begin{aligned} \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \theta} + \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \cdot \frac{\partial \hat{\mathbf{p}}}{\partial \theta} = 0 \\ \Leftrightarrow \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \theta} + \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \cdot \mathbf{w} \frac{\partial \hat{d}}{\partial \theta} = 0 \end{aligned} \tag{4.7}$$

Finally, from rearranging the Eq 4.7 it is possible to reach the closed form expression for the gradient of the surface depth $\hat{d}$:

$$\frac{\partial \hat{d}}{\partial \theta} = -\left(\frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \cdot \mathbf{w} \right)^{-1} \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \theta} \tag{4.8}$$

Niemeyer *et al.* [45] remark that calculating the gradient of the surface depth $\hat{d}$ *wrt.* the network parameters θ only involves calculating the gradient of f_θ at $\hat{\mathbf{p}}$ *wrt.* the network parameters θ and the surface point $\hat{\mathbf{p}}$. As a result, they do not need to store intermediate results like volumetric data generated by voxel-based approaches for computing the gradients, resulting in a lower memory consumption than other related works.

Finally, the forward and backward pass for the surface depth prediction for the automatic differentiation proposed by Niemeyer *et al.* [45] can be described. In the forward pass the surface depth $\hat{d}$ can be determined by finding the first occupancy change on the ray $\mathbf{r}$. To do that, the occupancy network are evaluated at n equally-spaced points on the ray $\{\mathbf{p}_j^{ray}\}_{j=1}^{n}$, and using a step size of Δs, the coordinates of the points in a cartesian space can be expressed as:

$$\mathbf{p}_j^{ray} = \mathbf{r}(j\Delta s + s_0), \tag{4.9}$$

where s_0 denotes the closest possible surface point. According to Niemeyer *et al.* [45], it is possible to find the smallest j for which f_θ changes from free space ($f_\theta < \tau$) to occupied space ($f_\theta \geq \tau$) as follow:

$$j = \underset{j'}{\operatorname{argmin}} \left(f_\theta(\mathbf{p}_{j'+1}^{ray}) \geq \tau > f_\theta(\mathbf{p}_{j'}^{ray}) \right) \tag{4.10}$$

Then, to obtain the approximation of the surface depth, the secant method is applied to the interval $[j\Delta s + s_0, (j+1)\Delta s + s_0]$ as proposed by Niemeyer *et al.* [45].

In the backward pass, the input to the backward pass $\lambda = \frac{\partial \mathcal{L}}{\partial \hat{d}}$ is the gradient of the loss with regard to a single surface depth prediction. The output of the backward pass is $\lambda \frac{\partial \hat{d}}{\partial \theta}$, which can be computed using Eq. 4.8. The backward pass is implemented for the hole batch of depth values $\hat{d}$. Niemeyer *et al.* [45] implement this efficiently by rewriting $\lambda \frac{\partial \hat{d}}{\partial \theta}$ as:

$$\lambda \frac{\partial \hat{d}}{\partial \theta} = \mu \frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \theta} \tag{4.11}$$

and

$$\mu = -\left(\frac{\partial f_\theta(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \cdot \mathbf{w} \right)^{-1} \lambda \tag{4.12}$$

where the right term in Eq. 4.11 corresponds to a normal backward operation applied to the neural network f_θ and Eq. 4.12 indicates an element-wise scalar multiplication for all elements in the batch of depth values as indicated by Niemeyer *et al.* [45].

During training, N images are given $\{\mathbf{I}_k\}_{k=1}^N$ together with corresponding camera intrinsics, extrinsics, and object masks $\{\mathbf{M}_k{}_{k=1}^N\}$. The model can also incorporate depth information $\{\mathbf{D}_k{}_{k=1}^N\}$, if it is available. For training f_θ and $\mathbf{t}_\theta$, an image $\mathbf{I}_k$ and N_p points $\mathbf{u}$ are random sampled on the image plane. There are three distinguish cases: first, P_0 denote the set of points $\mathbf{u}$ that lie inside the object mask $\mathbf{M}_k$ and for which the occupancy network predicts a finite surface depth $\hat{d}$. For these points it is possible to define a loss $\mathcal{L}_{RGB}(\theta)$ directly on the predicted image $\hat{\mathbf{I}}_k$. Moreover, P_1 denote the points $\mathbf{u}$ which lie outside the object mask $\mathbf{M}_k$. It is possible to define a loss $\mathcal{L}_{freespace}(\theta)$ that encourages the network to remove spurious geometry along corresponding rays. Finally, P_2 denote the set of points $\mathbf{u}$ which lie inside the object mask $\mathbf{M}_k$, but for which the occupancy network does not predict a finite surface depth $\hat{d}$. It is also possible to define a loss $\mathcal{L}_{occupancy}(\theta)$ that encourages the network to produce a finite surface depth. The depth loss $\mathcal{L}_{depth}(\theta)$ can be used when the depth information of an object are given. The normal loss $\mathcal{L}_{normal}(\theta)$ can also be used on the surface points to incorporate the prior belief that surfaces are predominantly smooth. It acts as a geometric regularizer and can be in particular useful for real-world scenarios where the reconstruction problem might be less constrained and noisier.

The final loss for a sampled view is given by the sum of the components $\mathcal{L}(\theta)$ as follow:

$$\begin{aligned} \mathcal{L}(\theta) = \lambda_0 \mathcal{L}_{RGB}(\theta) \\ + \lambda_1 \mathcal{L}_{depth}(\theta) \\ + \lambda_2 \mathcal{L}_{normal}(\theta) \\ + \lambda_3 \mathcal{L}_{freespace}(\theta) \\ + \lambda_4 \mathcal{L}_{occupancy}(\theta) \end{aligned} \tag{4.13}$$

where λ_i are the loss weight, and $i = 0, ..., 4$. The values of the loss weights proposed by Niemeyer *et al.* [45] are: $\lambda_0 = \lambda_3 = \lambda_4 = 1$, $\lambda_1 = 0$ for 2D supervision or $\lambda_1 = 10$ for 2.5D supervision, and $\lambda_2 = 0.1$. The photo-consistency loss $\mathcal{L}_{RGB}(\theta)$ for each point P_0 can be defined as:

$$\mathcal{L}_{RGB}(\theta) = \sum_{\mathbf{u} \in P_0} ||\xi(\mathbf{I_u}) - \xi(\hat{\mathbf{I}}_\mathbf{u})|| \tag{4.14}$$

where $\xi(\cdot)$ computes image features and $||\cdot||$ defines a robust error metric. As showed in Niemeyer *et al.* [45], the RGB-values are used and (optionally) image gradients as features and an l_1-loss for $||\cdot||$. The depth loss can be directly incorporated as l_1-loss on the predict surface depth as:

$$\mathcal{L}_{depth}(\theta) = \sum_{\mathbf{u} \in P_0} |d - \hat{d}|_1 \tag{4.15}$$

where d indicates the ground truth depth value of the sampled image point $\mathbf{u}$ and $\hat{d}$ denotes the predicted surface depth for pixel $\mathbf{u}$.

To penalize if a point $\mathbf{u}$ lies outside the object mask but the predicted surface depth $\hat{d}$ is finite, Niemeyer *et al.* [45] proposed the freespace loss that can be used as:

$$\mathcal{L}_{freespace}(\theta) = \sum_{\mathbf{u} \in P_1} BCE(f_\theta(\hat{\mathbf{p}}), 0) \tag{4.16}$$

where BCE is the binary cross-entropy. When no surface depth is predicted, the freespace loss is applied to a randomly sampled point on the ray, and if a point u lies inside the object mask but the predicted surface depth $\hat{d}$ is infinite, the network falsely predicts no surface points on ray $\mathbf{r}$. To encourage predicting occupied space on this ray, Niemeyer *et al.* [45] proposed to sample the depth value of the first point on the ray which lies inside all object masks (depth of the visual hull) d and define

$$\mathcal{L}_{occupancy}(\theta) = \sum_{\mathbf{u} \in P_2} BCE(f_\theta(\mathbf{r}(d), 1) \tag{4.17}$$

where, intuitively, $\mathcal{L}_{occupancy}(\theta)$ encourages the network to occupy space along the respective rays which can then be used by $\mathcal{L}_{RGB}(\theta)$ and $\mathcal{L}_{depth}(\theta)$ to refine the initial occupancy. Optionally, the representation proposed by Niemeyer *et al.* [45] incorporate a smoothness prior by regularizing surface normals. This is useful especially for real-world data as training with 2D or 2.5D supervision includes unconstrained areas where this prior enforces more natural shapes. This loss is defined as:

$$\mathcal{L}_{normal}(\theta) = \sum_{\mathbf{u} \in P_0} ||\mathbf{n}(\hat{\mathbf{p}}_\mathbf{u}) - \mathbf{n}(\mathbf{q}_\mathbf{u})||_2 \tag{4.18}$$

where $\mathbf{n}(\cdot)$ denotes the normal vector, $\hat{\mathbf{p}}_\mathbf{u}$ the predicted surface point and $\mathbf{q}_\mathbf{u}$ a randomly sampled neighbor of $\hat{\mathbf{p}}_\mathbf{u}$.

To evaluate the model, Niemeyer *et al.* [45] have used the accuracy and completeness values which are the (mean) Euclidean distances from the prediction to the ground

truth and from the ground truth to the prediction, respectively. After the training process, the DVR network predicts the occupancy probability for an object image input and reconstructs the object by using the marching cubes algorithm. Then, the predicted surface can be evaluated using the ground truth mesh available in the ShapeNet dataset. For the evaluation process, $100,000$ points are sampled from the prediction and the ground truth mesh. The accuracy and completeness equations presented in Niemeyer *et al.* [45] are described as follow:

$$\text{Accuracy}(S_{pred}|S_{GT}) := \frac{1}{S_{pred}} \int_{\mathbf{p}\in S_{pred}} \min_{\mathbf{q}\in S_{GT}} ||\mathbf{p}-\mathbf{q}||_2 d\mathbf{p} \tag{4.19}$$

$$\text{Completeness}(S_{pred}|S_{GT}) := \frac{1}{S_{GT}} \int_{\mathbf{p}\in S_{GT}} \min_{\mathbf{q}\in S_{pred}} ||\mathbf{p}-\mathbf{q}||_2 d\mathbf{p} \tag{4.20}$$

where more formally, let us define the ground truth and predicted surface of a 3D shape as S_{GT} and S_{pred}, respectively. Since there is an inherent trade-off between accuracy and completeness, analyzing them separately provides only little useful information. For instance, it is possible to achieve a good completeness score (at the expense of accuracy) by predicting a lot of spurious geometry. Similarly, a good accuracy score can be achieved (at the expense of completeness) by predicting only a small fraction of the object where the method is most certain. For all experiments, Niemeyer *et al.* [45] report the Chamfer-L_1 [42] distance which is the mean of the two entities:

$$\begin{aligned}\text{Chamfer}-\text{L}_1(S_{pred}|S_{GT}) := \frac{1}{2}(&\text{Accuracy}(S_{pred}|S_{GT}) \\ &+ \text{Completeness}(S_{pred}|S_{GT}))\end{aligned} \tag{4.21}$$

4.2 Self-Attention Layers Approach

Our first approach is to use self-attention layers in a different domain in a similar fashion as proposed by Salvi *et al.* [52]. To do that, we have implemented the SAGAN self-attention module proposed by Zhang *et al.* [70]. By using self-attention layers in conjunction with convolutional layers, we could leverage the model to better extract local and global information [62, 3]. The self-attention GAN (SAGAN) is a Generative Adversarial Network (GAN) that allows attention-driven, long-range dependency modeling for image generation tasks. To do so, they developed a self-attention module that is complementary to convolutions and helps with modeling long-range, multi-level dependencies across image regions, as illustrated in Figure 4.2.

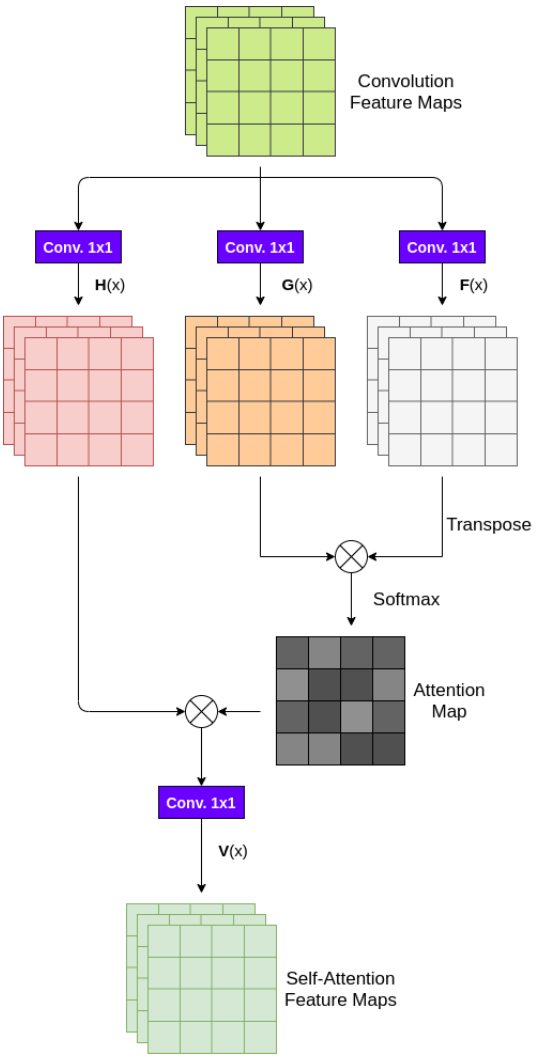

Figure 4.2 – Illustration of SAGAN Self-Attention module [70] used in conjunction with convolutional layers in our experiments.

The SAGAN self-attention module can be easily linked to a CNN layer. As described in Zhang *et al.* [70], the features from the previous layer $\mathbf{x} \in \mathbb{R}^{C \times N}$, where C is the number of channels and N is the number of feature locations of features, are first transformed into two feature spaces **F**, **G** to calculate the attention, where $\mathbf{F(x)} = \mathbf{W_F x}$, $\mathbf{G(x)} = \mathbf{W_G x}$. Then, to extract the attention map from those feature spaces, the Softmax function is applied to the features multiplication of the transposed **F(x)** and **G(x)**, as follow:

$$\zeta_{j,i} = \frac{exp(s_{i,j})}{\sum_{i=1}^{N} exp(s_{i,j})}, \tag{4.22}$$

where $s_{i,j} = \mathbf{F}(\mathbf{x}_i)^T \mathbf{G}(\mathbf{x}_j)$, $\zeta_{i,j}$ indicates the extent to which the model attends to the i^{th} location when synthesizing the j^{th} region. Then, the output of the self-attention module is described as follow:

$$\mathbf{o} = \mathbf{V}\left(\sum_{i=1}^{N} \zeta_{j,i}\mathbf{H}(\mathbf{x}_i)\right), \tag{4.23}$$

where $\mathbf{o} \in \mathbb{R}^{C\times N}$, $\mathbf{V(x)} = \mathbf{W_V x}$, $\mathbf{H(x)} = \mathbf{W_H x}$, and $\mathbf{W_F}$, $\mathbf{W_G}$, $\mathbf{W_H}$, and $\mathbf{W_V}$ are the learned weight matrices.

As a result, Zhang *et al.* further multiply the output of the attention layer by a scale parameter and add back the input feature map, so the final output is given by,

$$\mathbf{y_i} = \eta\mathbf{o_i} + \mathbf{x_i}, \tag{4.24}$$

where η is a learnable parameter that allows the network to first rely on the cues in the local neighborhood and then gradually learn to assign more weight to the non-local evidence [70]. We hypothesize that by applying the SAGAN self-attention module in different parts of the encoder from DVR, we could improve the baseline model results in a similar way as in Salvi *et al.* [52] and Ramachandran *et al.* [47], where applying the self-attention layers at the final layers of the encoder could leverage the encoder's capability of feature extraction.

4.3 Patchwise Self-Attention Approach

Patchwise Self-Attention is a model proposed by Zhao *et al.* [71], where the aggregation operations are performed by self-attention blocks and transformation operations are performed by an element-wise perceptron layer. It was observed in Zhao *et al.* [71] that traditional convolutional networks are divided into two types of functions: 1) aggregation functions, where convolutional operations are performed between map resources and kernels, and 2) transformation functions, where they are performed as operations of linear mapping and operations with non-linear functions. From these observations, Zhao *et al.* has proposed the usage of vectorized self-attention blocks as aggregation operations.

The vectorized self-attention block proposed by Zhao *et al.* is described as follows:

$$\mathbf{y}_i = \sum_{j\in\mathcal{R}(i)} \alpha(\mathbf{x}_{\mathcal{R}(i)})_j \odot \beta(\mathbf{x}_j), \tag{4.25}$$

where $\odot$ is the Hadamard product, $\mathcal{R}(i)$ is the local footprint of the aggregation and $\mathbf{x}_{\mathcal{R}(i)}$ is the patch of feature vectors in $\mathcal{R}(i)$. $\alpha(\mathbf{x}_{\mathcal{R}(i)})$ is a adaptive weight tensor of the same spatial dimensionality as the patch $\mathbf{x}_{\mathcal{R}(i)}$, and $\alpha(\mathbf{x}_{\mathcal{R}(i)})_j$ is the vector at location *j* in this tensor, corresponding spatially to the vector $\mathbf{x}_j$ in $\mathbf{x}_{\mathcal{R}(i)}$. The function β produces the feature vectors $\beta(\mathbf{x}_j)$ that are aggregated by α. The vectorized self-attention block is not permutation-invariant or cardinality-invariant, i.e., the weight computation $\alpha(\mathbf{x}_{\mathcal{R}(i)})$ can index the feature vectors

$\mathbf{x}_j$ individually, by location, and can intermix information from feature vectors from different locations within the footprint.

The $\alpha(\mathbf{x}_{\mathcal{R}(i)})$ can be decomposed as follows:

$$\alpha(\mathbf{x}_{\mathcal{R}(i)}) = \gamma(\delta(\mathbf{x}_{\mathcal{R}(i)})), \tag{4.26}$$

where γ is a function that performs the transformation operations and $\delta(\mathbf{x}_{\mathcal{R}(i)})$ is a concatenation function that combines the feature vectors $\mathbf{x}_j$ from the patch $\mathbf{x}_{\mathcal{R}(i)}$ and can be described as follows:

$$\delta(\mathbf{x}_{\mathcal{R}(i)}) = [\varphi(\mathbf{x}_i), [\psi(\mathbf{x}_j)]_{\forall j \in \mathcal{R}(i)}], \tag{4.27}$$

where φ and ψ are trainable transformations such as linear mappings and have matching output dimensionality. The Vectorized Self-Attention Block is illustrated in Figure 4.3.

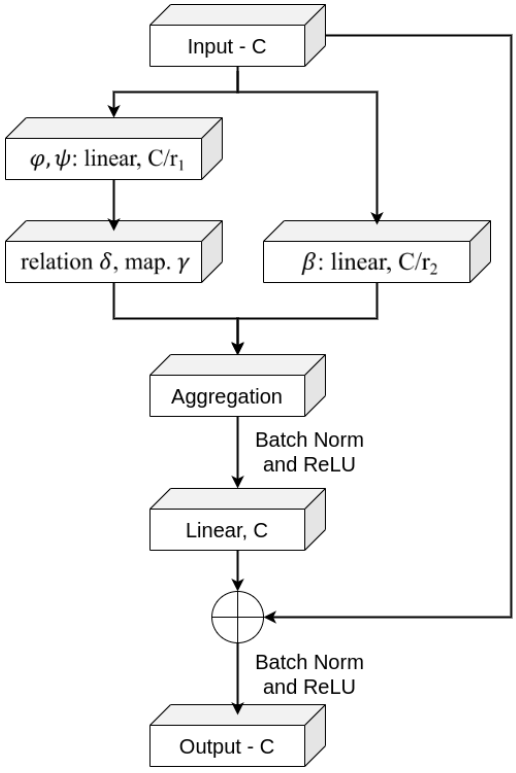

Figure 4.3 – The vectorized self-attention block [71]. *C* denote the channel dimension, $\oplus$ denotes the direct sum, the left branch calculates the attention weights $\alpha = \gamma(\delta(\mathbf{x}))$, by computing the function δ (via the mappings φ and ψ) and a subsequent mapping γ, while the right branch transforms features using a linear mapping β. r_1 and r_2 denote the factors by which both branches reduce channel dimension for efficient processing.

In Zhao *et al.*, the Patchwise Self-Attention network architectures are called SANX, where X refers to the number of vectorized self-Attention blocks. They have proposed three types of SANX architectures - SAN10, SAN15, and SAN19. Those architectures are correspondent to ResNet26 [25], ResNet38 [24], and ResNet50 [24] respectively. All three

SANX proposed by Zhao *et al.* perform better than the correspondent ResNet architectures in image recognition tasks. Besides that, all three Patchwise Self-Attention architectures have fewer than 35% network parameters and 35% fewer Floating Point Operations (FLOP) than all corresponding ResNet architectures. Considering that self-attention modules are highly computational expensive, our intuition is that we could obtain similar results with less memory consumption, allowing future improvements as hyperparameters optimization or architecture modifications.

4.4 Experiments

For our implementation using the SAGAN Self-Attention module as in Section 4.2, we have modified the original DVR encoder to fit the self-attention module after each ResNet layer as illustrated in Figure 4.4. We have decided to experiment with the SAGAN self-attention module only for 2.5D single-view supervision and 2D multi-view supervision, considering that we can reproduce the same behavior from 2D multi-view supervision to 2.5D multi-view supervision. For both types of supervision, we have modified the encoder in three ways as presented in Salvi *et al.*[52]. First, we have trained the DVR network by applying the self-attention module after each ResNet layer inside the encoder. Then, for the second experiment, we have trained the DVR network by applying the self-attention module just after the ResNet layers 1 and 2, and for the last experiment, we have trained the DVR network by applying the self-attention module just after the ResNet layers 3 and 4.

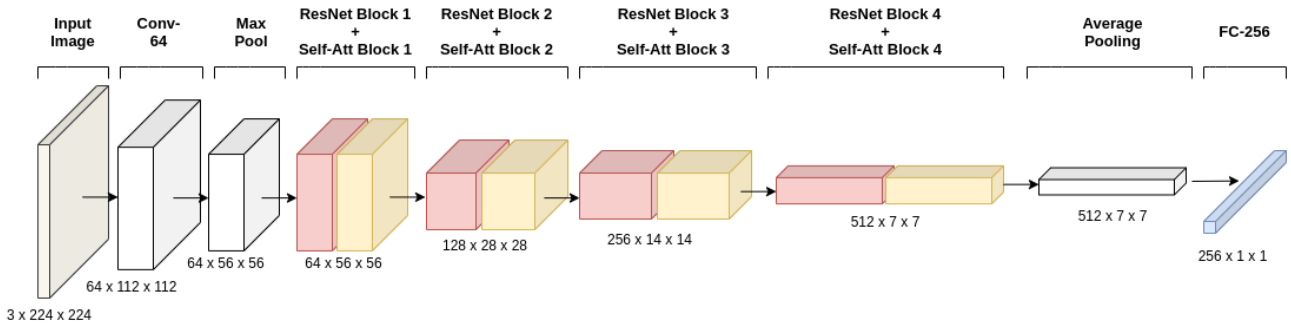

Figure 4.4 – The representation of the original encoder from DVR [45] with the SAGAN self-attention module (yellow blocks) after each ResNet layer (red blocks)

Then, for our implementation of the Patchwise Self-Attention network illustrated in Figure 4.5, where we have decided to remove the ResNet encoder from the original implementation of the DVR model, and experiment with the SAN10 Patchwise self-attention network explained in Section 4.3 as an encoder. The SAN10 model implemented in this work was not pre-trained. We have decided to experiment with the SAN10 architecture for 2D multi-view supervision, and 2.5D multi-view supervision. We did not experiment with the SAN10 for 2.5D multi-view supervision due to its high computational cost and consequently

a reduction of the batch size. For these experiments, we have decided to use the same architecture. We have only modified the final transition layer from the original Patchwise implementation, from [1000 $\times$ 1 $\times$ 1] dimension to [256 $\times$ 1 $\times$ 1] dimension to fit the DVR network.

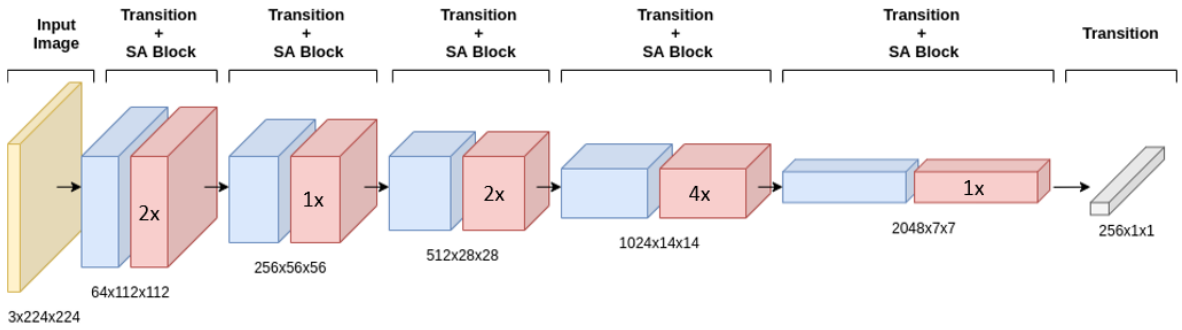

Figure 4.5 – The illustration of Patchwise Self-Attention (SAN10) architecture as an encoder, where the blue blocks correspond to transition blocks and red blocks correspond to SA Block. The number inside each red block corresponds to the amount of the respective block are connected after the transition block.

We have followed the training protocol as in Niemeyer *et al.* [45], starting with a ray sampling accuracy of $n = 16$ which is iteratively increased to $n = 128$ by doubling n after 200, 600, and 1000 thousand iterations. We also set $\tau = 0.5$ for all experiments according to the baseline model. We train on a single NVIDIA GTX 1080 GPU with a sample of 1024 random points. We use the Adam optimizer [30] with learning rate $lr = 10^{-4}$. We have trained the DVR model using the ResNet18, SAGAN self-attention module, and SAN10 for 260 thousand epochs in 2D multi-view supervision, and for 2.5D single-view supervision training process we have decided to train the ResNet18 and SAGAN self-attention module for 260 epochs, and the SAN10 for 500 epochs considering that the network was not pre-trained and the lack of multiple object views. The dataset that was used for training and evaluation steps is the ShapeNet [8], which consists of 13 object classes with 24 images of resolution 256^2, camera matrix, depth maps, and object masks per object which are used for supervision as in Niemeyer *et al.* [45]. For multi-view supervision, we have used a batch size of 16 images for 2D supervision and a batch size of 4 images for 2.5D supervision, and for single-view supervision, we have used a batch size of 16 images in 2.5D supervision.

4.5 Results

As described in Section 4.4, we have experimented with the baseline DVR model using a ResNet18 encoder, and four other self-attention variations for both multi-view and

single-view supervision. In Table 4.1 it is presented the results for the 2D multi-view supervision training process. It is possible to observe that for this type of supervision, we improve the results as expected using the SAGAN self-attention layers in conjunction with convolutional layers, but differently from Salvi *et al.* our best results were using the SAGAN self-attention layers after all convolutional layers inside the encoder. The two other approaches using the SAGAN self-attention layers had a similar mean result as the ResNet18 approach. For the fully self-attention model SAN10 used in our experiments, we could not improve the mean results, but on the other hand, using the SAN10 model it was possible to see an improvement in object classes as Airplane, Car, and Rifle in comparison with the ResNet18 model, that could indicate a possible capacity of improvement on object classes that have more structural details.

Table 4.1 – The Chamfer-L_1 distance values for each object class relative to multi-view supervision reconstruction. The SAN10 is the Patchwise Self-Attention approach, and the SA-X in the SAGAN self-attention approach where X is "All" for the self-Attention layer applied after all conv layers, X is "12" for the self-attention layer applied after the first and second conv layers, and X is "34" for the self-attention layer applied after the third and fourth conv layers.

	Multi-view Supervision				
			2D		
Object Class	SAN10	ResNet18	SA-all	SA-12	SA-34
airplane	0.0248	0.0257	0.0247	**0.0245**	0.0312
bench	0.0330	0.0254	0.0268	0.0250	**0.0248**
cabinet	0.0337	0.0248	**0.0236**	0.0247	0.0275
car	0.0237	0.0252	0.0214	0.0228	**0.0212**
chair	0.0336	**0.0292**	0.0305	0.0306	0.0303
display	0.0355	0.0283	**0.0279**	0.0295	0.0286
lamp	0.0540	0.0441	**0.0418**	0.0484	0.0538
loudspeaker	0.0395	0.0324	**0.0313**	0.0325	0.0342
rifle	0.0215	0.0255	**0.0175**	0.0293	0.0192
sofa	0.0308	**0.0262**	**0.0262**	0.0265	0.0272
table	0.0341	0.0329	0.0313	0.0322	**0.0309**
telephone	0.0246	0.0198	0.0187	**0.0174**	0.0191
vessel	0.0284	0.0264	**0.0239**	0.0272	0.0253
mean	0.0321	0.0281	**0.0266**	0.0285	0.0287

The results of our experiments for 2.5D single-view supervision are indicated in Table 4.2. Differently from multi-view supervision, for our experiments using the SAGAN self-attention layers we could observe a similar improvement using these layers after the first and second convolutional layers than using after the third and fourth convolutional layers. Also, using the SAGAN self-attention layers after all convolutional layers could still improve the results for reconstruction.

Considering that for the single-view supervision training process we have less view of the same object, and our fully self-attention implementation was not pre-trained, we have

Table 4.2 – The Chamfer-L_1 distance values for each object class relative to single-view supervision reconstruction. The SAN10* is the Patchwise Self-Attention approach trained for 500 epochs, and the SA-X in the SAGAN self-attention approach where X is "All" for the self-attention layer applied after all conv layers, X is "12" for the self-attention layer applied after the first and second conv layers, and X is "34" for the self-attention layer applied after the third and fourth conv layers.

	Single-view Supervision				
		2.5D			
Object Class	ResNet18	SA-all	SA-12	SA-34	SAN10*
airplane	0.0402	0.0307	0.0364	**0.0306**	0,0303
bench	0.0443	0.0479	**0.0426**	0.0459	0,0458
cabinet	0.0407	0.0410	**0.0399**	0.0419	0,0412
car	**0.0266**	0.0317	0.0286	0.0290	0,0289
chair	**0.0487**	0.0514	0.0493	0.0499	0,0487
display	0.0558	0.0536	0.0556	**0.0529**	0,0586
lamp	0.0715	0.0766	**0.0702**	0.0791	0,0702
loudspeaker	0.0548	0.0551	**0.0531**	0.0555	0,0547
rifle	0.0804	0.0278	0.0380	**0.0273**	0,0286
sofa	**0.0427**	0.0467	0.0442	0.0444	0,0463
table	**0.0524**	0.0546	0.0531	0.0556	0,0540
telephone	0.0331	**0.0315**	0.0316	0.0329	0,0380
vessel	0.0414	0.0394	0.0402	**0.0389**	0,0377
mean	0.0487	0.0452	**0.0448**	0.0449	0,0448

decided to train the SAN10 model for 500 epochs. It is possible to observe in Table 4.2 that the SAN10 model has achieved a similar improvement than the SA-12 model, but it took almost double the training time. Also, it was possible to observe that some object classes had a bigger difference in results between the SAN10 model and the SA-12 model, e.g., the bench and airplane classes. We believe that this behavior could have resulted from the weight initialization, since the SA-12 has used some pre-trained layers from the ResNet18 model, and the SAN-10* weights were initialized from a normal distribution. The pre-trained model could leverage the network to learn from previous images features, giving it the capability to extract new information in less time.

In Figure 4.6 and Figure 4.7 it is possible to observe our qualitative results from our SAGAN self-attention and Patchwise self-attention approaches. In our multi-view supervision experiments, it is possible to observe that SAN10 has produced a reasonable result for some object classes in comparison with other approaches. Differently from multi-view supervision, in the single-view supervision process, it was possible to observe that SAN10* has not produced good results in comparison with other approaches for some object classes such as table and airplane.

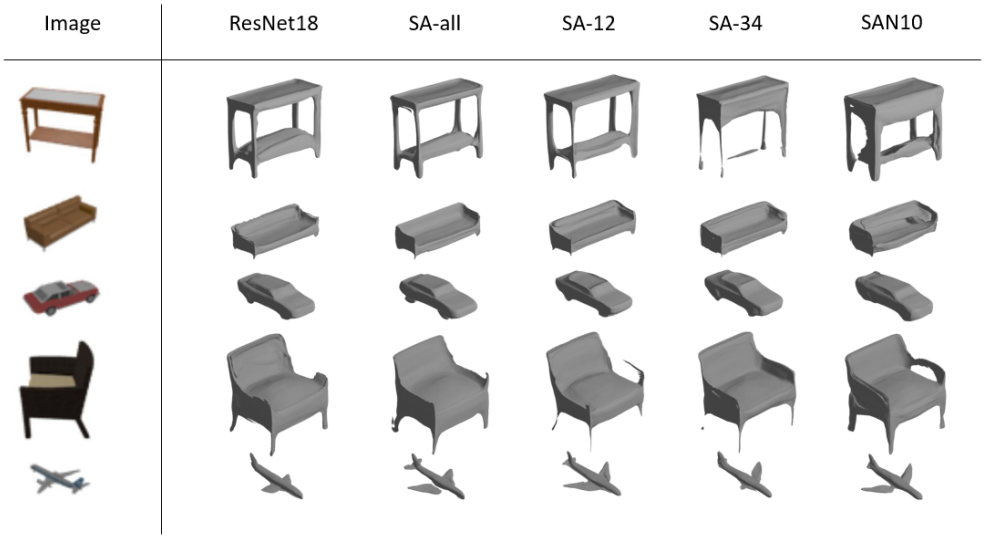

Figure 4.6 – Qualitative results for 2D multi-view supervision reconstruction using SA-X, where X indicates the position of the SAGAN self-attention module after a ResNet layer, and SAN10 is the Patchwise Self-Attention network.

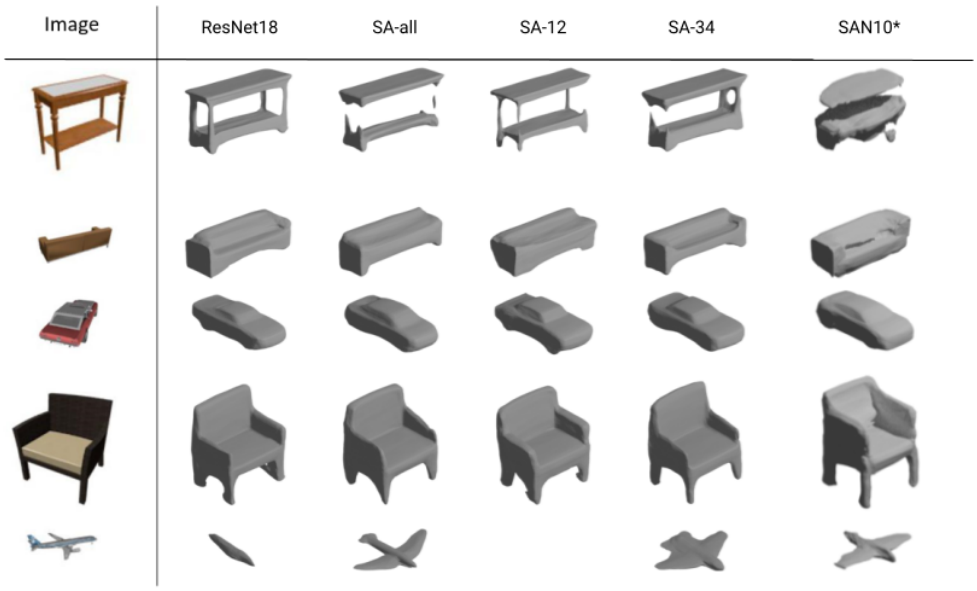

Figure 4.7 – Qualitative results for 2.5D single-view supervision reconstruction using SA-X, where X indicates the position of the SAGAN self-attention module after a ResNet layer, and SAN10* is the Patchwise Self-Attention network trained for 500 epochs.

5. 3D RECONSTRUCTION BASED ON VISION TRANSFORMER

In computer vision tasks, convolutional architectures remain dominant [33, 31, 24]. Inspired by the Transformers used in NLP, multiple studies try combining CNN-like architectures with self-attention [62, 7], and even replace the convolutions entirely [47].

We have investigated the performance of a Transformer model operating directly over images, with the fewest possible modifications. Differently from the previous approaches presented in Chapter 4, the transformer model known as “Vision Transformer” splits the images into patches and provides the sequence of linear embeddings of these patches as input to the model. Image patches are treated the same way as tokens (words) in NLP applications. Thus, we use the transformer as an image feature-extraction module in the 3D reconstruction training process.

Transformers models used as backbone in Computer Vision [7, 15, 68, 27, 46, 57] usually employ the COCO [38] and ImageNet [13] datasets, i.e., there are insight and results regarding those specific datasets. In this work, however, we use images of isolated and centralized objects with a white background, i.e., the ShapeNet dataset [8]. We hypothesize that even with this type of image, it will be possible to take advantage of the same benefits that Transformer models obtained in other datasets, like their capability of extracting object structure and position information. Also, according to Ranghy *et al.* [46], transformer models are able to propagate more information in the initial layers than ResNet models, i.e., they can access more global information from the input data in the shallow layers. Our intuition is that we could also modify the transformer layers to focus on data regions and consequently improve the ability of the model in extracting features from input data.

This chapter comprises the following sections: in Section 5.1, we present our approach using a transformer-based model that is used as a backbone for vision tasks, and in Section 5.2 we present an optimization approach that uses the Nyströmformer model. In Section 5.3 we report our experiments using both original and optimized transformer models. Finally, in Sections 5.4 and 5.5 we present the quantitative and qualitative results.

5.1 Pyramid Vision Transformer

The Pyramid Vision Transformer (PVT) [61] is a transformer model for Computer Vision tasks that was based on the transformer proposed by Dosovitskiy *et al.* [15]. The PVT model was developed as a backbone model for many dense prediction tasks as image classification, segmentation, object detection, and many more.

Differently from previous transformers, the PVT model provides advantages such as: 1) it introduces a progressive shrinking pyramid to reduce the feature sequence length

when the depth of the model is increased, significantly reducing the computational burden; 2) it can serve as a backbone for many downstream tasks like image-level prediction and pixel-level prediction; and 3) it has improved the performance of image classification and object detection in comparison with convolution models like ResNet and previous transformers like ViT [15] and DETR [7].

In this work, we propose the usage of PVT [61] as the encoder of DVR [45]. Since the original DVR encoder is composed of a ResNet-18 [24] model, we hypothesize that due to the spacial reduction mechanism and the previous results of the PVT model and other previous transformer models in different vision tasks, we could improve the results of the 3D object reconstruction by switching the convolution-based encoder for a transformer-based encoder. In addition, we also believe that the structure proposed by PVT, which comprises self-attention layers followed by linear layers, and with the addition of residual connections, can help the model to extract both high-level and low-level features throughout the model.

The PVT backbone mainly comprises four compression stages as illustrated in Figure 5.1. Each of the stages shares a similar architecture, which consists of a patch embedding layer and multiple transformer blocks. The compression stage can be seen in Figure 5.2.

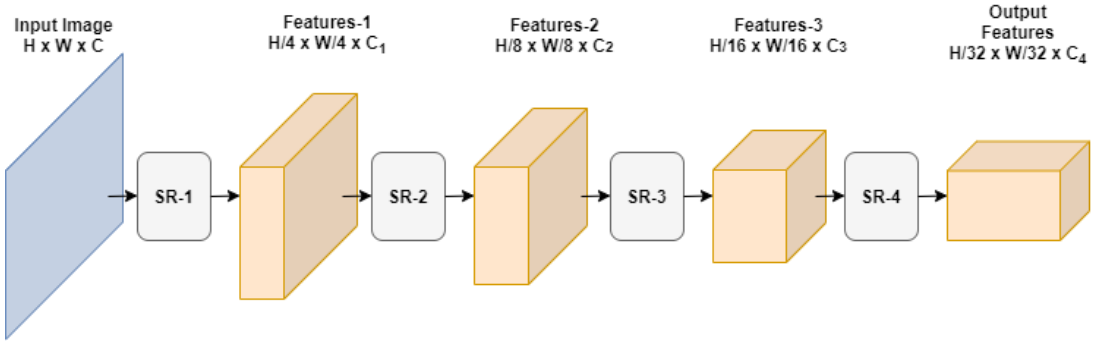

Figure 5.1 – The Pyramid Vision Transformer (PVT) architecture proposed by Wang *et al.* [61], which consists of four spatial reduction stages {SR-1, SR-2, SR-3, and SR-4} for feature extraction, sharing a similar architecture that comprises a patch embedding layer and transformer blocks.

In Wang *et al.* [61], four different PVT architecture were proposed with different sizes: PVT-Tiny, PVT-Small, PVT-Medium, and PVT-Large. According to the authors, all of them follow the two rules of ResNet [24]: 1) the deeper the network is, the larger the hidden dimensions, while the output resolution shrinks, and 2) the major computation resources are concentrated in the third stage. Due to the high computational cost of DVR [45], and to allow a fair comparison, we decided to use the PVT-Tiny model, which shows a similar architecture to a ResNet-18 model.

The feature extraction pipeline (Figure 5.1) of PVT starts with a given input image with size of $H \times W \times 3$, where H is the height and W is the width of the image, and it is divided into $\frac{H}{4} \times \frac{W}{4}$ patches, where each patch has a size of $4 \times 4 \times 3$. Then, each patch is

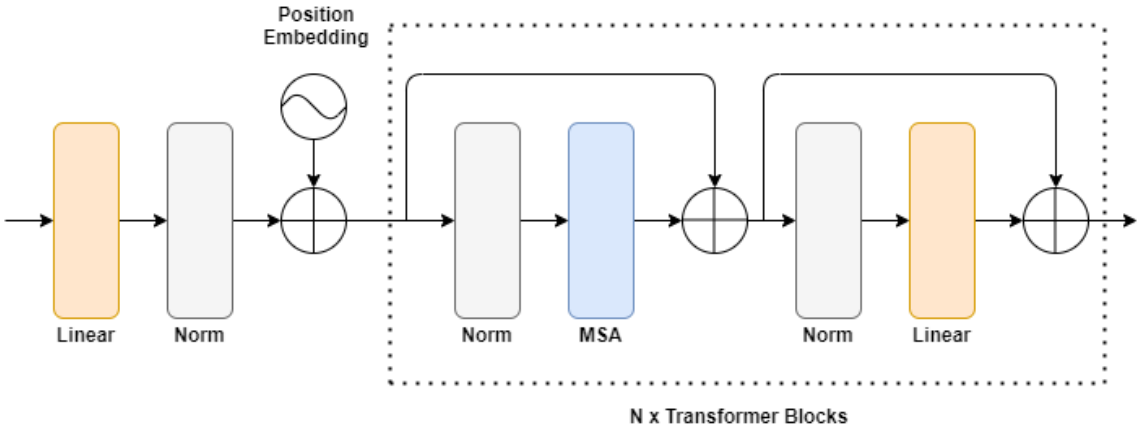

Figure 5.2 – Feature compression stage as proposed by Wang *et al.* [61]. Here, the input feature patches are embedded and then they pass through a positional encoding step. The encoded features pass through $N\times$ transformer blocks where N is the number of blocks.

flatten and projected to a embedding dimension C_i-dim, where i is the corresponding stage. The output shape from the linear layer is $\frac{H_{i-1}}{P_i} \times \frac{W_{i-1}}{P_i} \times C_i$, where P_i is the patch size of the corresponding stage. Then, the embedded patches alongside with the position embeddings pass through a transformer encoder (Figure 5.2) with N layers and the output is reshaped to a feature map with size $\frac{H_{i-1}}{P_i} \times \frac{W_{i-1}}{P_i} \times C_i$. In the same way, using the feature maps from prior stages, it is possible to obtain the feature maps for the next stage.

The spatial reduction occurs inside the self-attention layer. According to Wang *et al.* [61], the attention layer receives query Q, key K, and value V as input, and the spatial reduction operation is applied to K and V as follows:

$$SR(\mathbf{x}) = Norm(Reshape(\mathbf{x}, R_i)) \tag{5.1}$$

where R_i is the reduction ratio in stage i, the $Reshape(\mathbf{x}, R_i)$ operation reshapes the input $\mathbf{x} \in \mathbb{R}^{(H_i W_i)\times C_i}$ into $\frac{H_i W_i}{R_i^2} \times (R_i^2 C_i)$, and $Norm(.)$ is the layer normalization procedure [2]. The spatial reduction operation can largely reduce the memory overhead.

5.2 Nyströmformer Optimization

One of the main disadvantages related to transformers models is their high computational cost due to their quadratic complexity over the input due to the self-attention procedure. Therefore, models such as Longformer [4], Linformer [60], Performer [72], Big Bird [69], and Nyströmformer [66] emerged with the intention of reducing this complexity to an approximately-linear complexity.

Despite the success of these models in reducing the temporal and spatial complexity of the self-attention mechanisms, the Nyströmformer model [66] proved not only to be

more effective in reducing the complexity of the self-attention mechanism to large dimensions, but also to be more efficient in producing approximate information that could help other models to obtain better results in tasks related to NLP and Computer Vision.

The basic idea is to use landmarks $\widetilde{K}$ and $\widetilde{Q}$ from key K and query Q to derive an efficient Nyström approximation without accessing the full QK^T to approximately calculate the full softmax matrix S. The approximation of S can be written as

$$\widehat{S} = softmax\left(\frac{Q\widetilde{K}^T}{\sqrt{d_q}}\right) Z^* softmax\left(\frac{\widetilde{Q}K^T}{\sqrt{d_q}}\right) \tag{5.2}$$

where Z^* is an approximation of

$$A_S = softmax\left(\frac{\widetilde{Q}\widetilde{K}^T}{\sqrt{d_q}}\right) \tag{5.3}$$

by calculating the Moore-Penrose pseudoinverse as described in [66]. In addition, Nyströmformer can approximate the full softmax matrix by selecting all landmarks from query and key. According to Xiong *et al.* [66], the landmark points can be selected by simply using segment-means, *e.g.*, for input queries $Q = [q_1; ...; q_n]$, the n queries are separated into m segments, and $l = \frac{n}{m}$ is the landmark points for Q, assuming that n is divisible by m for simplicity, then the landmarks can be computed as:

$$\widetilde{q}_j = \sum_{i=(j-1)\times l+1}^{(j-1)\times l+m} \frac{q_i}{m} \tag{5.4}$$

where $j = 1, ..., m$, and the same approach can be used to compute the input keys $K = [k_1; ...; k_n]$ landmarks. As showed by the authors, 64 landmarks is often sufficient to ensure a good approximation, and given the size o input length, the Nyströmformer can reduce the amount of memory used more than 10 fold, and the running time more than 5 fold, in comparison to a standard Transformer model that has no sequence length reduction.

One of our hypotheses is that by applying the Nyströmformer as an approximation of the softmax on the self-attention layers of the transformer model, it could be possible to reduce the memory consumption even though PVT [61] already has a spatial reduction mechanism within its architecture.

We also believe that by applying the Nyströmformer optimization we could still preserve the results of the implementation of the original PVT model, due to the characteristics of the images from ShapeNet dataset used in this work. Since the ShapeNet dataset contains images with a single, centralized object and a white background as presented in Section 4.1, we hypothesize that by using the number of landmarks indicated in [66] we could have a good approximation due to the fact that the information of the images is always located in similar regions and the amount of relevant information are similar as well. Note

that the segment-means for landmarks computation is similar to local average pooling used in NLP [53] and vision [18] tasks, giving a good approximation that associates features with the same importance to all locations in a small window.

5.3 Experiments

Our experiments using a transformer model follows a similar approach from our self-attention based neural network experiments in Chapter 4. We decided to use the PVT backbone model as an encoder inside the DVR model. Considering the architectures proposed by Wang *et al.* [61], we implemented the PVT-Tiny architecture that has a similar structure to the ResNet-18 model used in the DVR baseline.

PVT-Tiny (Figure 5.1) comprises two transformer blocks inside each spatial reduction stage $SR - i$ with $i = 1, ..., 4$, where the patch size P_i of each stage is {4, 2, 2, 2}, the channel dimension C_i is {64, 128, 320, 512}, the reduction factor R_i is {8, 4, 2, 1}, the number of heads N_i of the self-attention layers in each spatial reduction stage is {1, 2, 5, 8}, and the features dimension after each reduction is reduced by a factor of 2 as illustrated in Figure 5.1. Then, we add a linear layer at the end of the PVT-Tiny model for spatial reduction to fit into the DVR model. The last linear layer has a dimension of $[256 \times 1 \times 1]$. Differently from our implementation of SAN10, we decided to experiment with the pre-trained PVT-Tiny model and the standard PVT-Tiny model (not pre-trained). The pre-trained PVT-Tiny model was trained for an image classification task using the ImageNet dataset [13] in a similar way as ResNet in [24].

Since PVT-Tiny has a small memory footprint in comparison with our previous self-attention based neural network implementations, we decided to increase the batch size from 16 to 32 images in the training process, and from 4 to 8 images in the validation process for both 2D and 2.5D in multi-view supervision, and for 2.5D in single-view supervision. This results in a faster training process without reducing the generalization capability of the network, and also could help the network in finding better optima of the objective function [20].

Increasing the batch size also increases memory consumption. For the 2D supervision training process, the increase in memory consumption was still below our maximum memory available, but for the 2.5D supervision training process it was necessary to reduce the hidden dimension of the decoder from 512 to 128. We believe that 128 as the size of the decoder hidden dimension can still produce good results, since the same dimension is used in the multi-view supervision training process.

Then, for our optimization experiments, we have decided to implement Nyströmformer inside the PVT-Tiny model, as a softmax approximation for the self-attention layers. Due to the spatial reduction in the PVT model, we decided not to implement the Nyström-

former in the last PVT spatial reduction stage $SR - 4$, since the number of landmarks proposed by the authors is not sufficiently smaller than the dimension of the features in this stage.

For our experiments using the Nyströmformer, we decided to implement it only within the 2.5D single-view supervision training process. As an optimization process, we believe that the results of using the Nyströmformer in one of the supervised processes could have the same behavior in other supervised processes. We have used the Nyströmformer in three different ways: only in the first layer; in the first and second layers; and in the first, second, and third layers.

Then, considering the computational reduction provided by the Nyströmformer implementation, we have decided to experiment with the addition and removal of decoder blocks for our best Nyströmformer model. Since each decoder block from the DVR model receives information from both the last decoder block and encoder output feature, we believe that adding more blocks in the decoder could improve the capability of the model in retaining information from different object classes. On the other hand, by removing a block from the decoder, we expect a reduction of the capability of the model to retain information from extracted features, but could result in a reduction of memory consumption that can be used in other future types of investigations, such as batch size increment or increasing of the hidden dimensions from the decoder.

We also have decided to increase the number of epochs for training the model with an additional decoder block, since the increment of the number of parameters in the decoder module could result in more difficulty in finding good local minima. For all other model hyperparameters, we have decided to use the same values from our last experiments presented in Section 4.4.

5.4 Quantitative Results

In this section we present our results using the $Chamfer - L_1$ metric as in Niemeyer *et al.* [45]. The results from our multi-view supervision experiments are presented in Table 5.1. Considering our previous self-attention-based approaches and previous transformer related work, it is possible to observe that we could achieve better results using a pre-trained transformer model instead of a non pre-trained model, as one could expect. Also, in both 2D and 2.5D experiments, the mean results were similar to the ResNet18 model, but considering each object class individually, it is possible to observe that in 2D experiments our PVT implementation has improved the results of classes with fewer structural details, while in 2.5D experiments our PVT implementation has a more general similarity between object classes.

Table 5.1 – The Chamfer-L_1 distance values for each object class relative to multi-view supervision reconstruction training for 500 epochs. For our experiments in this type of supervision, we have used the pre-trained Pyramid Vision Transformer model (PVT-T(PT)) and a non pre-trained Pyramid Vision Transformer model (PVT-T(S)).

	Multi-view Supervision					
	2D			2.5D		
Object Class	ResNet18	PVT-T(PT)	PVT-T(S)	ResNet18	PVT-T(PT)	PVT-T(S)
airplane	**0.02062**	0.02586	0.02479	0.01838	**0.01832**	0.02237
bench	0.02544	**0.02494**	0.02929	**0.02200**	0.02237	0.02438
cabinet	0.02540	**0.02447**	0.02619	**0.01952**	0.02028	0.02172
car	**0.02038**	0.02258	0.02172	0.02012	**0.01998**	0.02295
chair	**0.02760**	0.02973	0.03415	**0.02672**	0.02890	0.03250
display	0.02889	**0.02543**	0.02799	0.03055	**0.02867**	0.03093
lamp	0.04304	**0.04147**	0.04839	0.04468	**0.03993**	0.04686
loudspeaker	0.03144	**0.02998**	0.03275	0.03129	**0.03051**	0.03213
rifle	0.01853	**0.01844**	0.02182	0.01689	**0.01642**	0.01786
sofa	0.02590	**0.02405**	0.03185	**0.02464**	0.02593	0.02771
table	**0.02836**	0.03043	0.03077	**0.02417**	0.02459	0.02673
telephone	0.01822	**0.01714**	0.01839	0.01551	**0.01533**	0.01771
vessel	**0.02474**	0.02478	0.03077	**0.02565**	0.02687	0.03077
mean	**0.02604**	0.02610	0.02914	0.02462	**0.02447**	0.02728

For our single-view supervision experiments, it is possible to observe the quantitative results in Tables 5.2 and 5.3. As presented in Section 5.3, beyond our implementation of the PVT model, we have decided to experiment with the Nyströmformer model within the single-view supervision process, as well as with the addition and removal of decoder blocks. In Table 5.2, it is possible to observe that using only the pre-trained PVT model we could achieve better results than the ResNet18 model, and as we expected, we could still improve the results using the Nyströmformer as an optimization approach. Our experiments demonstrate that we can reach better results using the Nyströmformer only in the first transformer stage, indicating that the Nyströmformer can improve the capability of the network in extracting information from a specific data region.

In Table 5.3, it is possible to observe that adding and removing one block from the decoder does not significantly change the mean of the results. Considering our hypothesis in Section 5.3, it is possible to observe that adding one linear block in the decoder increases the training time to achieve the best result. Our experiment of adding one linear block took twice as long as our previous experiments, but training for $1,000$ epochs result in an improvement of 8.25% in relation to the ResNet18 model, and surprisingly achieving a better result using much less memory than the experiments proposed in [45].

Table 5.2 – The Chamfer-L_1 distance values for each object class relative to 2.5D single-view supervision reconstruction training for 500 epochs. For our experiments in this type of supervision, we have used the pre-trained Pyramid Vision Transformer model (PVT-T(PT)), a non pre-trained Pyramid Vision Transformer model (PVT-T(S)), and a pre-trained PVT-T model with a Nyströmformer approximation (PVT-TNX), where X means the stages where Nyströmformer was applied as explained in Section 5.3.

	Single-view Supervision 2.5D					
Object Class	ResNet18	PVT-T(PT)	PVT-T(S)	PVT-TN1	PVT-TN2	PVT-TN3
airplane	0.03153	0.02949	0.03360	**0.02694**	0.02906	0.02894
bench	0.04407	0.04168	0.05100	**0.04086**	0.04460	0.04462
cabinet	0.04097	0.03940	0.04393	**0.03917**	0.04073	0.04158
car	0.02861	0.02848	0.03152	**0.02779**	0.02811	0.02977
chair	**0.04809**	0.04876	0.05158	0.04814	0.04824	0.04932
display	0.05399	**0.05137**	0.06219	0.05290	0.05628	0.05460
lamp	0.07226	0.07241	0.07016	**0.06323**	0.06367	0.06994
loudspeaker	0.05448	**0.05181**	0.05684	0.05184	0.05302	0.05428
rifle	0.03154	0.02731	0.03411	**0.02453**	0.02558	0.02615
sofa	0.04407	0.04385	0.05016	**0.04289**	0.04541	0.04647
table	0.05158	0.05114	0.05837	**0.05108**	0.05269	0.05458
telephone	0.03269	0.02989	0.03982	0.03264	**0.02980**	0.03164
vessel	0.03808	0.03635	0.04539	**0.03534**	0.03613	0.03785
mean	0.04400	0.04246	0.04836	**0.04133**	0.04256	0.04383

Table 5.3 – The Chamfer-L_1 distance values for each object class relative to 2.5D single-view supervision reconstruction training for 500 epochs, using the pre-trained PVT-T model with a Nyströmformer approximation (PVT-TNX), where X means the stages where Nyströmformer was applied as explained in Section 5.3. We also experiment the removal and addition of blocks of the decoder, indicated by PVT-TN1-4B and PVT-TN1-6B.

	Single-view Supervision - 2.5D				
	500 Epochs				1000 Epochs
Object Class	ResNet18	PVT-TN1	PVT-TN1-4B	PVT-TN1-6B	PVT-TN1-6B
airplane	0.03153	**0.02694**	0,02812	0,02845	0,02713
bench	0.04407	**0.04086**	0,04305	0,04255	**0,04055**
cabinet	0.04097	**0.03917**	0,03976	0,03936	**0,03897**
car	0.02861	**0.02779**	0,02816	0,02783	**0,02628**
chair	0.04809	0.04814	0,04786	**0,04749**	**0,04535**
display	0.05399	0.05290	**0,05152**	0,05253	**0,05081**
lamp	0.07226	0.06323	0,06599	**0,06125**	**0,05989**
loudspeaker	0.05448	0.05184	0,05293	**0,05151**	**0,05051**
rifle	0.03154	**0.02453**	0,02591	0,02587	0,02509
sofa	0.04407	**0.04289**	0,04403	0,04325	**0,04212**
table	0.05158	0.05108	0,05209	**0,05068**	**0,04968**
telephone	0.03269	0.03264	**0,03025**	0,03163	0,03058
vessel	0.03808	**0.03534**	0,03608	0,03701	0,03622
mean	0.04400	**0.04133**	0,04198	0,04149	**0,04024**

5.5 Qualitative Results

In this section, we perform a qualitative analysis of the results obtained through the experiments indicated in Section 5.3. Because often the numerical results obtained may not express the quality of many reconstructed models, and also that a visual comparison aiming to indicate the quality of an object can often be considered subjective, we have conducted a qualitative study through the application of a questionnaire that was able to extract information about the quality of the reconstructed models as well as the characteristics of the objects that could indicate how close they are to their original images. The questions of the full questionnaire are presented in the Appendix A.

We decided to analyze the reconstructed objects using two of the best models used in our experiments (PVT-TN1 and PVT-TN1-6B) and also the ResNet18 model. The reconstructed objects used in this questionnaire were randomly selected from 26 reconstructed objects for each class. Then, questions were asked requesting that a grade from 1 to 5 be indicated for each reconstructed object, where 1 indicates a strongly poor reconstruction quality and 5 indicates a strongly good reconstruction quality. Then, for each class of objects, we asked which structural parts of the objects would indicate how similar they are to their original image. In total, 47 participants answered the questionnaire.

The results from our questionnaire on the reconstructed objects are presented in Figures 5.3 to 5.26. It is possible to observe that unanimously for all questions the transformer models showed better reconstruction quality compared to the ResNet18 model. Also, only in questions regarding the *sofa* object class the PVT-TN1-6B model has better results in all questions. For the *plane*, *car*, and *chair* object classes, the PVT-TN1-6B model showed better results in comparison to the PVT-TN1 model in $2/3$ of the questions. Furthermore, by extracting the top-5 words from the answers to the questions on the structural parts (Figures 5.27 to 5.30), it was possible to observe that classes with more structural details as *plane* and *chair* have a structural attribute with much higher importance than other attributes, and classes like *car* and *sofa* have a similarity in the importance of their structural attributes. These words are important to corroborate the qualitative results that were obtained.

We also show our reconstruction results from multi-view supervision experiments in Figure 5.31. It is possible to observe that for both 2D and 2.5D supervision, all models have a similar reconstruction result, except for the chair reconstructed using the ResNet18 model in 2.5D supervision, which has a small change in overall dimensions of the object compared to other models. The similarity of the reconstructed objects also corroborates the numerical results obtained in our experiments.

Figure 5.3 – The reconstructed objects presented in Question 1 in the questionnaire regarding the *plane* object class.

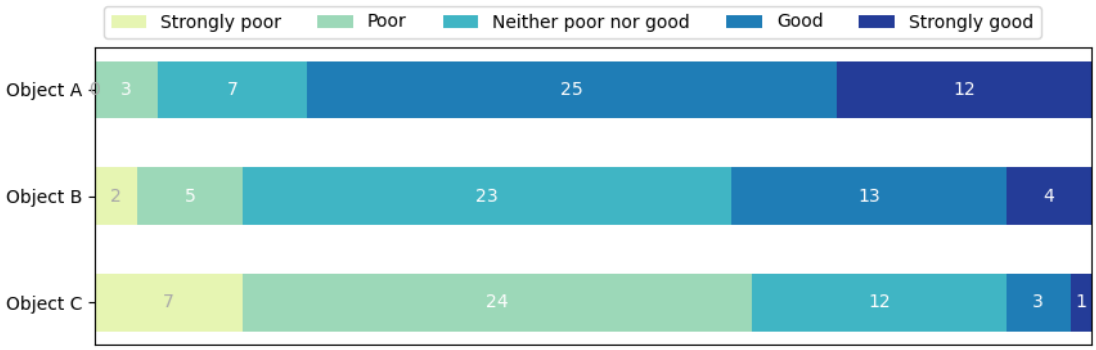

Figure 5.4 – Results of the Question 1 regarding the *plane* object class.

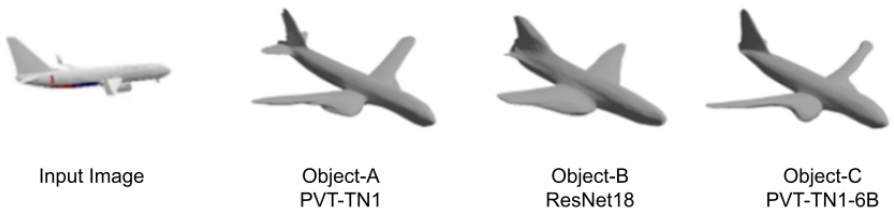

Figure 5.5 – The reconstructed objects presented in Question 2 in the questionnaire regarding the *plane* object class.

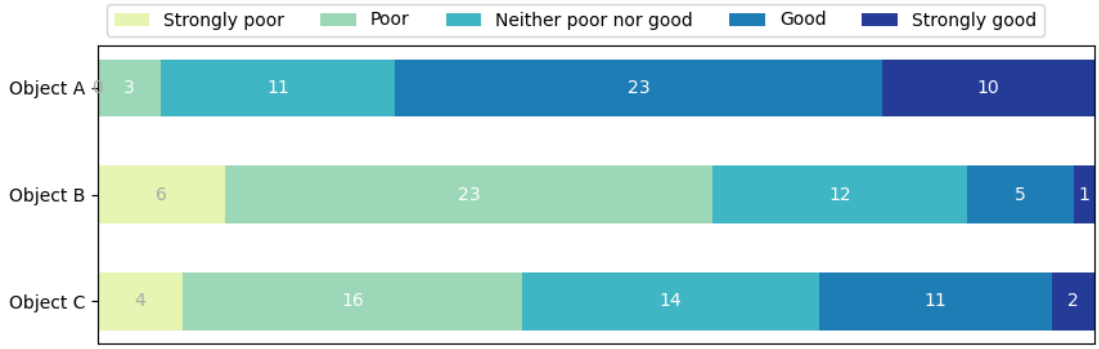

Figure 5.6 – Results of the Question 2 regarding the *plane* object class.

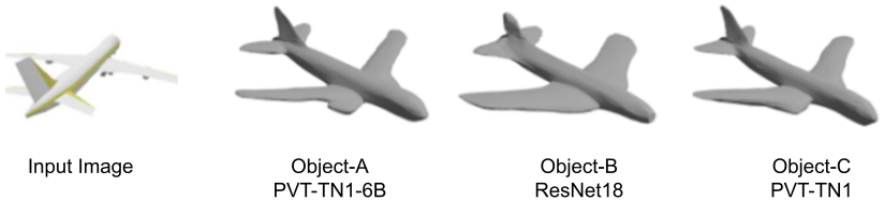

Figure 5.7 – The reconstructed objects presented in Question 3 in the questionnaire regarding the *plane* object class.

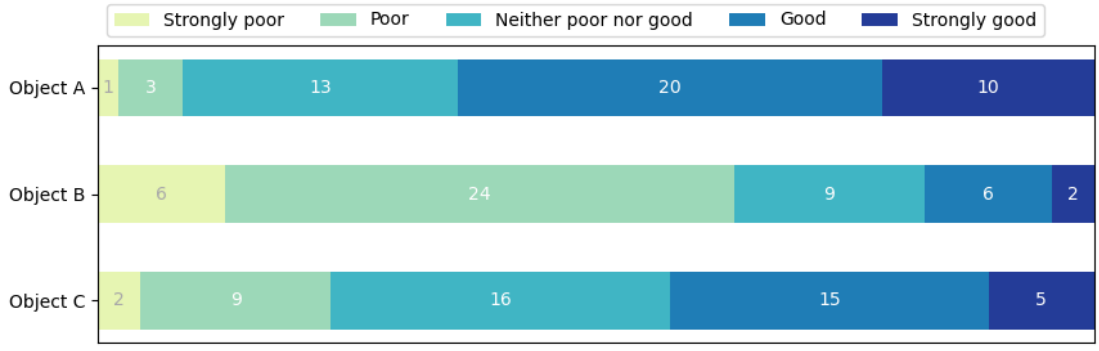

Figure 5.8 – Results of the Question 3 regarding the *plane* object class.

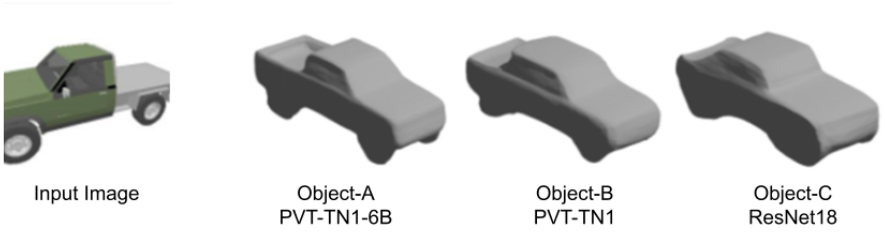

Figure 5.9 – The reconstructed objects presented in Question 1 in the questionnaire regarding the *car* object class.

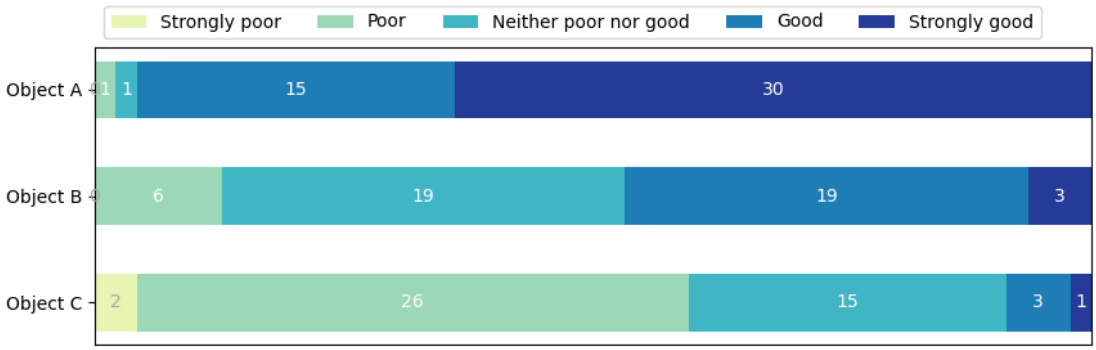

Figure 5.10 – Results of the Question 1 regarding the *car* object class.

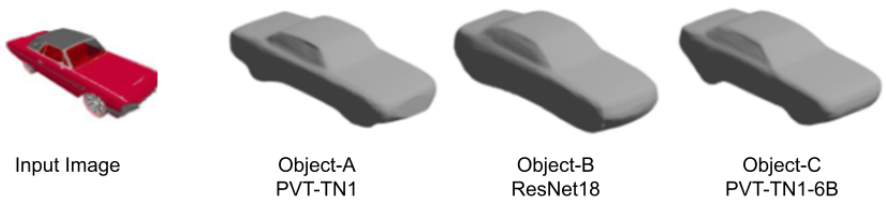

Figure 5.11 – The reconstructed objects presented in Question 2 in the questionnaire regarding the *car* object class.

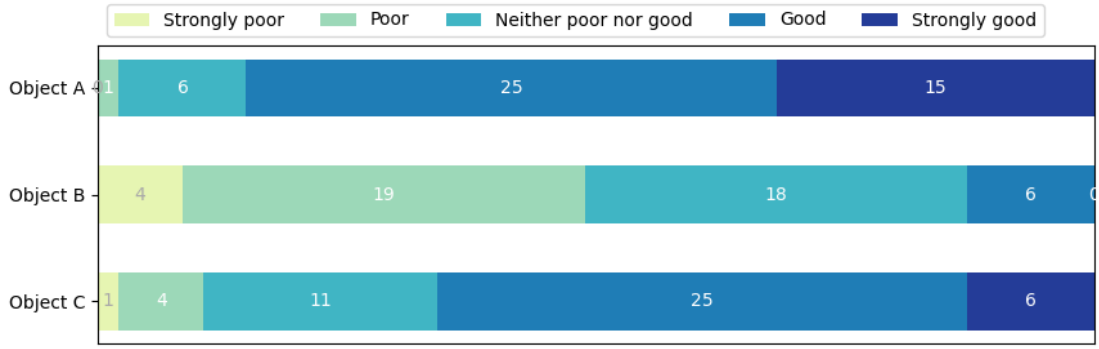

Figure 5.12 – Results of the Question 2 regarding the *car* object class.

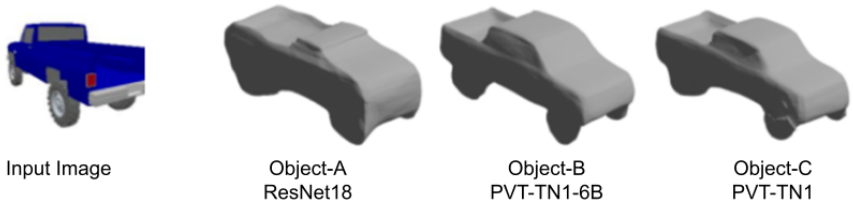

Figure 5.13 – The reconstructed objects presented in Question 3 in the questionnaire regarding the *car* object class.

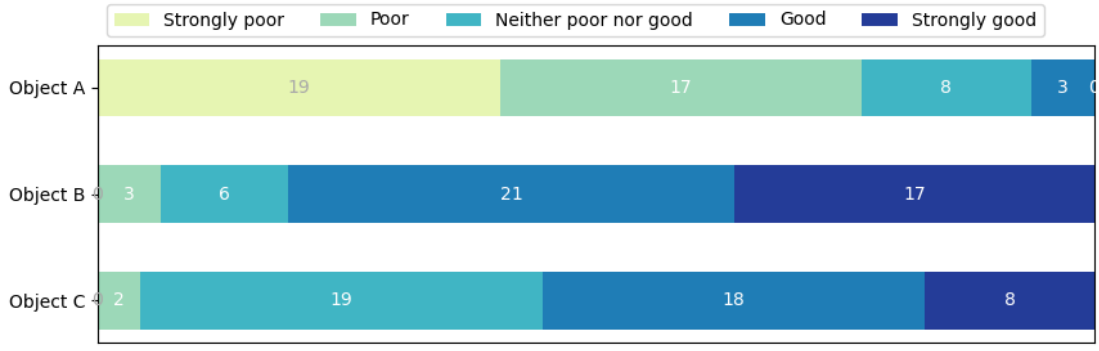

Figure 5.14 – Results of the Question 3 regarding the *car* object class.

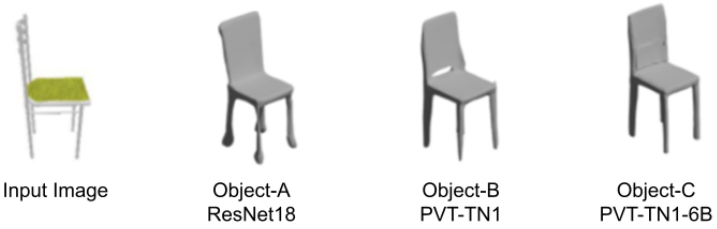

Figure 5.15 – The reconstructed objects presented in Question 1 in the questionnaire regarding the *chair* object class.

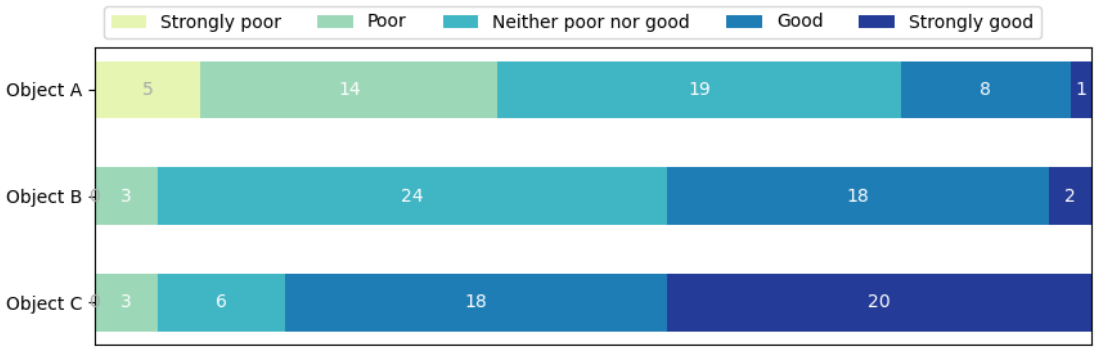

Figure 5.16 – Results of the Question 1 regarding the *chair* object class.

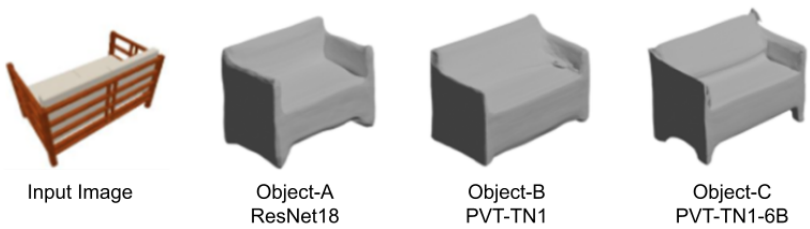

Figure 5.17 – The reconstructed objects presented in Question 2 in the questionnaire regarding the *chair* object class.

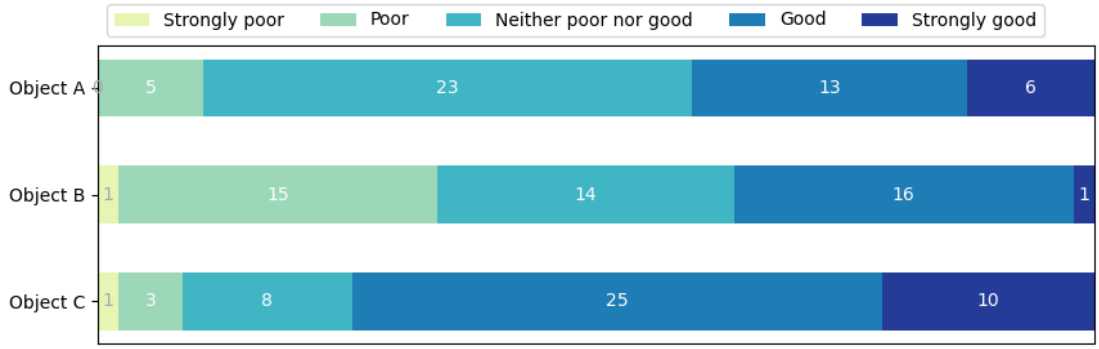

Figure 5.18 – Results of the Question 2 regarding the *chair* object class.

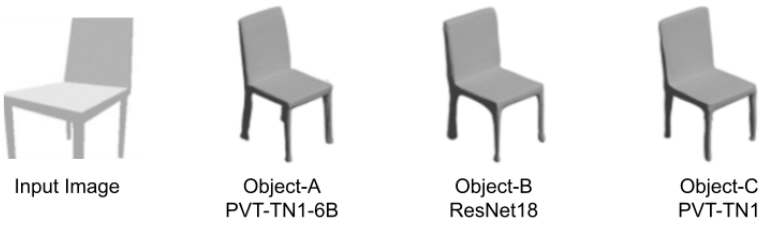

Figure 5.19 – The reconstructed objects presented in Question 3 in the questionnaire regarding the *chair* object class.

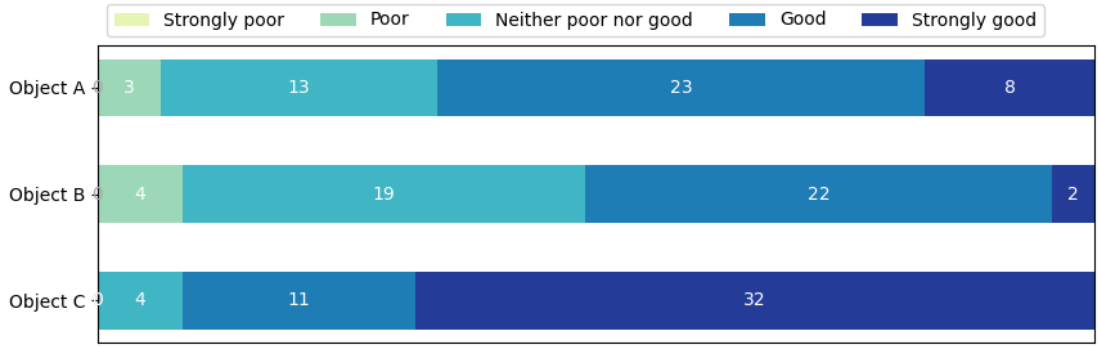

Figure 5.20 – Results of the Question 3 regarding the *chair* object class.

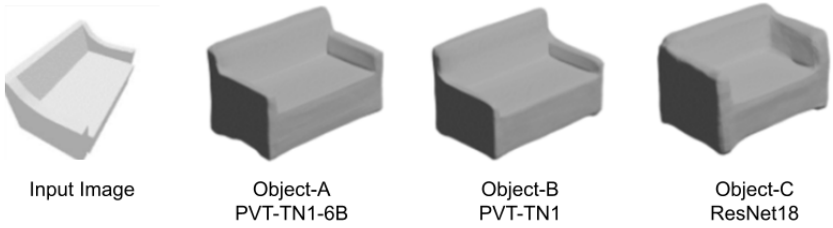

Figure 5.21 – The reconstructed objects presented in Question 1 in the questionnaire regarding the *sofa* object class.

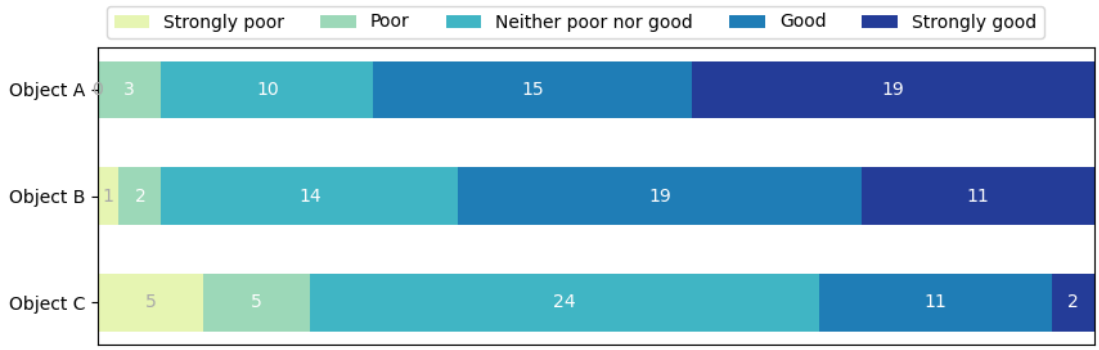

Figure 5.22 – Results of the Question 1 regarding the *sofa* object class.

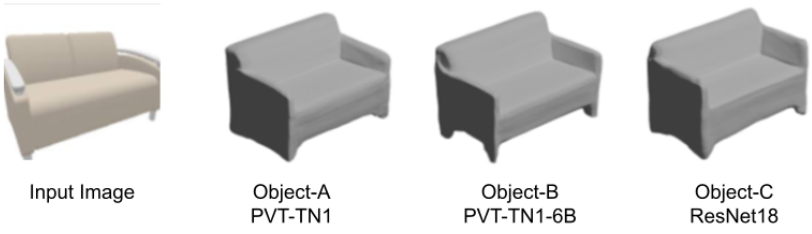

Figure 5.23 – The reconstructed objects presented in Question 2 in the questionnaire regarding the *sofa* object class.

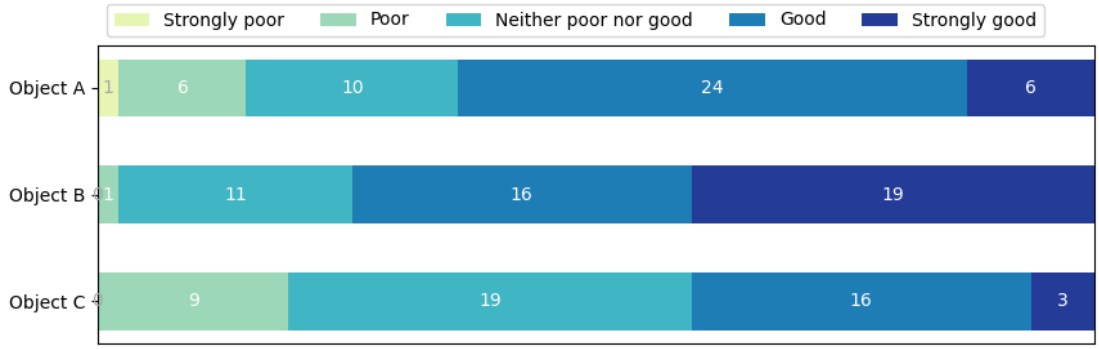

Figure 5.24 – Results of the Question 2 regarding the *sofa* object class.

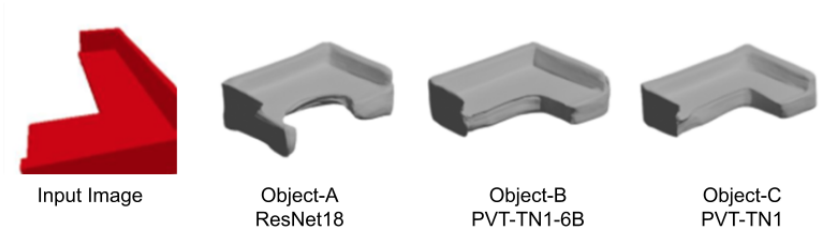

Figure 5.25 – The reconstructed objects presented in Question 3 in the questionnaire regarding the *sofa* object class.

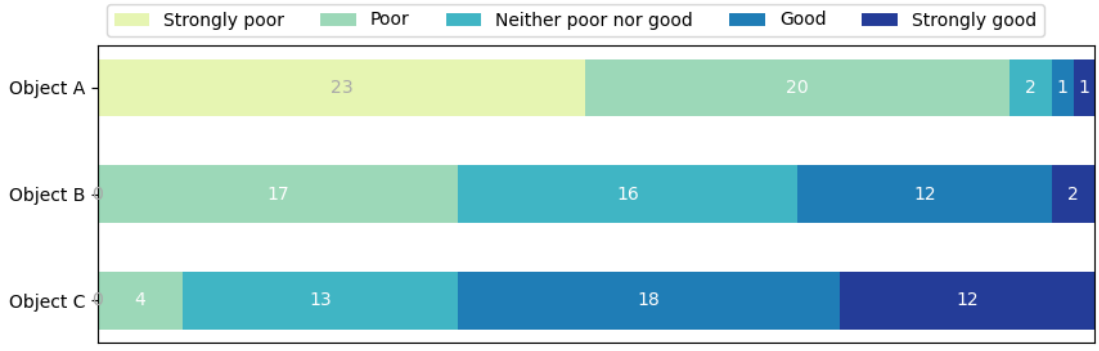

Figure 5.26 – Results of the Question 3 regarding the *sofa* object class.

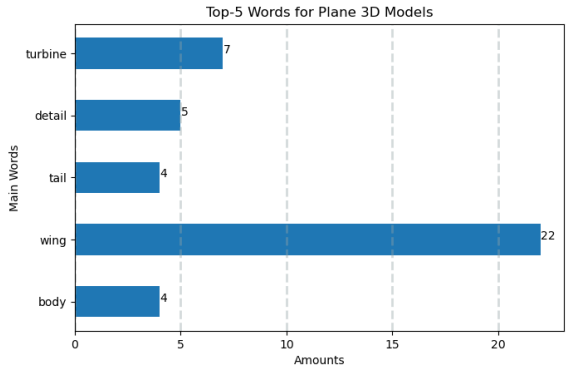

Figure 5.27 – The five most cited words in the question about 3D plane models.

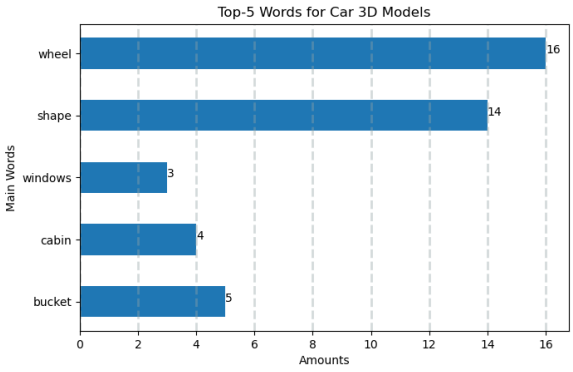

Figure 5.28 – The five most cited words in the question about 3D car models.

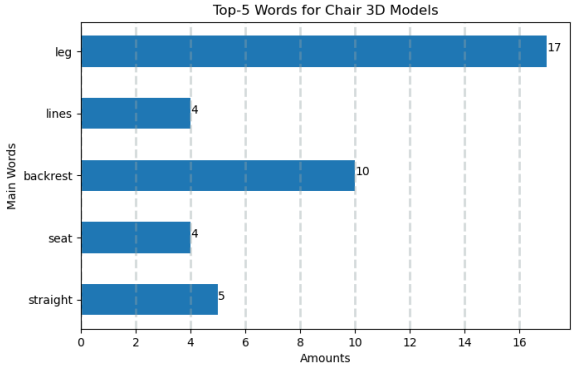

Figure 5.29 – The five most cited words in the question about 3D chair models.

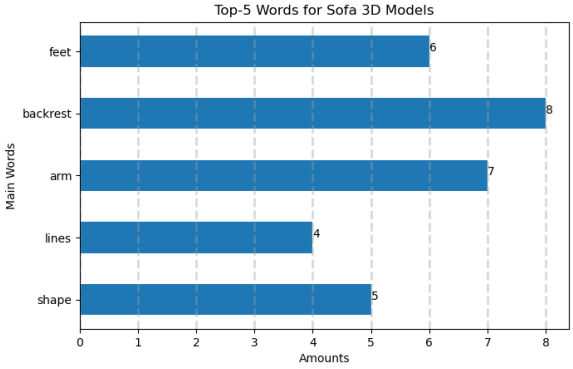

Figure 5.30 – The five most cited words in the question about 3D sofa models.

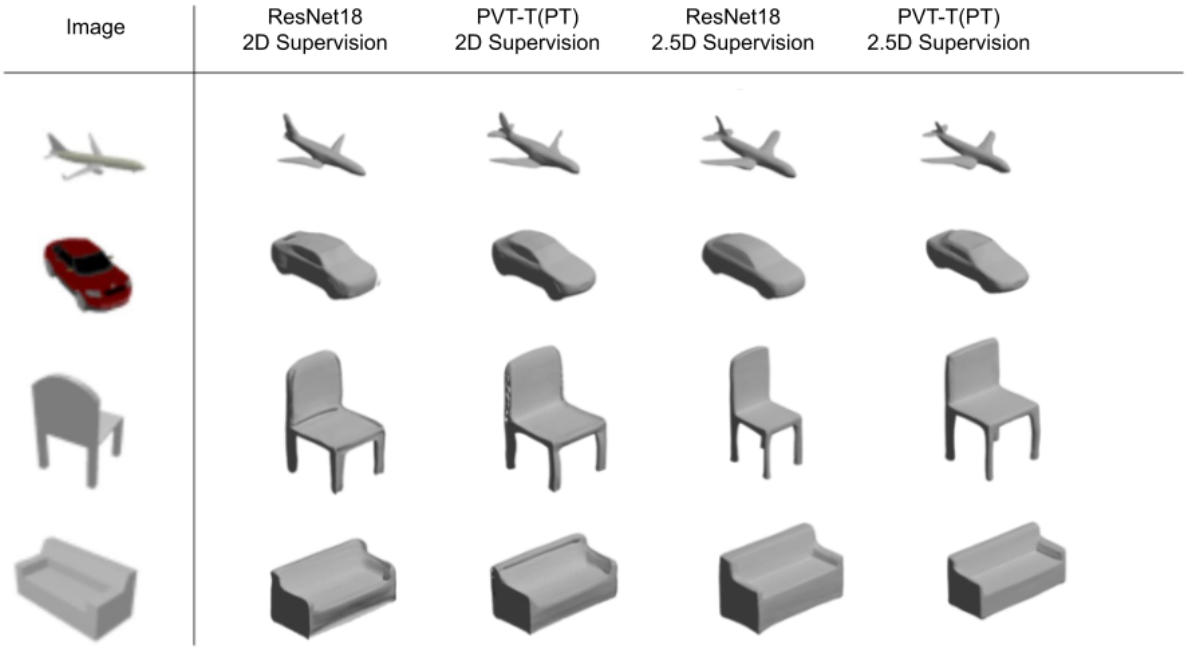

Figure 5.31 – Qualitative results for 2D and 2.5D multi-view supervision reconstruction using a ResNet18 model and a PVT-T(PT) model.

6. CONCLUSIONS

In this work, we have present four different self-attention-based approaches for implicit 3D object reconstruction. We first introduce the usage of self-attention layers in conjunction with convolutional layers, as well as the usage of a fully self-attention model that is considered an approximation of convolutions. Our experiments show that, as presented by Salvi *et al.* [52], the usage of the SAGAN self-attention module inside the decoder produces improvements on implicit 3D object reconstruction, even in scenarios without 3D supervision. The usage of the SAGAN self-attention module also shows that we can produce improvements on different object classes based on the position of the self-attention module inside the encoder.

The Patchwise Self-Attention network does not show any improvements for the number of training iterations that we have trained the DVR model, but according to Cordonnier *et al.* [11], different architectures of self-attention models could take more training time to reach the same results as ResNet network, including non pre-trained models.

Considering the results on the first two approaches, we have experimented with the usage of a backbone transformer model considering its recent advantages on computer vision tasks, like image classification, object detection, and image segmentation. By applying the Pyramid Vision Transformer model, it was possible to observe that the pre-trained transformer model has produced better or similar results than the baseline.

We have also experimented with the Nyströmformer model as an approximation of the softmax operation within self-attention modules in the single-view supervision training process, considering that this type of approximation could have a similar result in other types of supervision. We believe that by using this approximation one can achieve a reduction of computational cost while improving the results of the reconstruction due to its capability of focusing on more representative regions of the features. By applying the Nyströmformer model in different stages of the PVT model, it was possible to observe that we could improve the results by more than 6% concerning the baseline model, while our implementation of a pre-trained PVT model resulted in an improvement of 3.5% to the baseline model.

Since the Nyströmformer model could also reduce memory consumption, we have decided to use the reduced memory to add one more linear block from the decoder, as well as to remove one linear block from the decoder to compare the capability of the decoder blocks to retain information from objects. Our results have shown that adding one more linear block to the decoder can still produce improvements in the reconstruction process, but it was necessary to train the model with a larger number of epochs. Our results for adding a linear block inside the decoder have shown an improvement of 8.5% compared to the baseline, and this result is still better than the results of Niemeyer *et al.* [45], where they have trained the model using more computational resources. We also applied a questionnaire to qualitatively

evaluate the reconstructed objects via the PVT-TN1-6B, PVT-TN1, and ResNet18 models used in the single-view supervision process. Through this questionnaire, it was possible to corroborate the quantitative results previously demonstrated, thus building a strong case on the advantages of using approaches based on both the Pyramid Vision Transformer and the Nyströmformer model, showing an impressive improvement in reconstruction results.

Looking at the overall results, we believe that self-attention-based models such as Transformers can be used to fully replace convolutional models as they can extract and retain relevant information from input data in a similar way to convolutional models. To deal with their increased computational cost, we showed that optimization strategies can generate models with lower computational cost and possibly greater learning capability for tasks related to Computer Vision.

Considering the results obtained in this thesis and the recent advances in the use of transformer models in Computer Vision, we intend in the future to keep investigating the use of self-attention-based models as a decoder module for 3D reconstruction tasks. We plan to investigate how to replace the current linear blocks by self-attention blocks inside the decoder, and also to investigate how self-attention layers compare to linear layers. In addition, we also intend to provide new strategies to optimize transformer models, since many models for 3D reconstruction have a high computational cost mainly due to the amount of computations that need to be performed by the decoder module.

REFERENCES

[1] Andrew, A. M. “Multiple view geometry in computer vision”, *Kybernetes*, vol. 30, Dec 2001, pp. 1333–1341.

[2] Ba, J. L.; Kiros, J. R.; Hinton, G. E. “Layer normalization”. Source: https://arxiv.org/pdf/1607.06450.pdf, Jan 2022.

[3] Bello, I.; Zoph, B.; Vaswani, A.; Shlens, J.; Le, Q. V. “Attention augmented convolutional networks”. In: IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 3286–3295.

[4] Beltagy, I.; Peters, M. E.; Cohan, A. “Longformer: The long-document transformer”. Source: https://arxiv.org/pdf/2004.05150.pdf, Jan 2022.

[5] Bhattacharyya, P.; Huang, C.; Czarnecki, K. “Sa-det3d: Self-attention based context-aware 3d object detection”. Source: https://arxiv.org/pdf/2101.02672.pdf, Jan 2022.

[6] Bleyer, M.; Rhemann, C.; Rother, C. “Patchmatch stereo - stereo matching with slanted support windows”. In: British Machine Vision Conference (BMVC), 2011, pp. 1–11.

[7] Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. “End-to-end object detection with transformers”. In: European Conference on Computer Vision (ECCV), 2020, pp. 213–229.

[8] Chang, A. X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; Xiao, J.; Yi, L.; Yu, F. “Shapenet: An information-rich 3d model repository”, Technical Report, Stanford University - Princeton University - Toyota Technological Institute at Chicago, 2015, 1–11p.

[9] Chapelle, O.; Schölkopf, B.; Zien, A. “Semi-Supervised Learning”. The MIT Press, 2006, 542p.

[10] Choy, C. B.; Xu, D.; Gwak, J.; Chen, K.; Savarese, S. “3d-r2n2: A unified approach for single and multi-view 3d object reconstruction”. In: European Conference on Computer Vision (ECCV), 2016, pp. 628–644.

[11] Cordonnier, J.-B.; Loukas, A.; Jaggi, M. “On the relationship between self-attention and convolutional layers”. In: International Conference on Learning Representations (ICLR), 2020, pp. 1–18.

[12] Cyganek, B.; Siebert, J. P. “An introduction to 3D computer vision techniques and algorithms”. John Wiley & Sons, 2011, 520p.

[13] Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; Fei-Fei, L. “Imagenet: A large-scale hierarchical image database”. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009, pp. 248–255.

[14] Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), 2019, pp. 4171–4186.

[15] Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; Houlsby, N. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: International Conference on Learning Representations (ICLR), 2021, pp. 1–22.

[16] Furukawa, Y.; Hernández, C. “Multi-view stereo: A tutorial”, *Foundations and Trends® in Computer Graphics and Vision*, vol. 9, Jun 2015, pp. 1–148.

[17] Gao, Z.; Li, E.; Yang, G.; Wang, Z.; Tian, Y.; Liang, Z.; Guo, R.; Li, S. “Object reconstruction with deep learning: A survey”. In: IEEE 9th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), 2019, pp. 643–648.

[18] Gao, Z.; Wang, L.; Wu, G. “Lip: Local importance-based pooling”. In: IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 3355–3364.

[19] Glorot, X.; Bordes, A.; Bengio, Y. “Deep sparse rectifier neural networks”. In: 14th International Conference on Artificial Intelligence and Statistics (AISTATS), 2011, pp. 315–323.

[20] Goodfellow, I.; Bengio, Y.; Courville, A. “Deep Learning”. MIT Press, 2016, 1–775p.

[21] Han, K. J.; Prieto, R.; Ma, T. “State-of-the-art speech recognition using multi-stream self-attention with dilated 1d convolutions”. In: IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), 2019, pp. 54–61.

[22] Han, X.; Laga, H.; Bennamoun, M. “Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, Nov 2019, pp. 1–27.

[23] He, J.; Spokoyny, D.; Neubig, G.; Berg-Kirkpatrick, T. “Lagging inference networks and posterior collapse in variational autoencoders”. In: International Conference on Learning Representations (ICLR), 2019, pp. 1–15.

[24] He, K.; Zhang, X.; Ren, S.; Sun, J. “Deep residual learning for image recognition”. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[25] He, K.; Zhang, X.; Ren, S.; Sun, J. “Identity mappings in deep residual networks”. In: European Conference on Computer Vision (ECCV), 2016, pp. 630–645.

[26] Hochreiter, S.; Schmidhuber, J. “Long short-term memory”, *Neural Computation*, vol. 9, Nov 1997, pp. 1735–1780.

[27] Jaegle, A.; Gimeno, F.; Brock, A.; Vinyals, O.; Zisserman, A.; Carreira, J. “Perceiver: General perception with iterative attention”. In: 38th International Conference on Machine Learning (ICML), 2021, pp. 4651–4664.

[28] Jatavallabhula, K. M.; Smith, E.; Lafleche, J.-F.; Tsang, C. F.; Rozantsev, A.; Chen, W.; Xiang, T.; Lebaredian, R.; Fidler, S. “Kaolin: A pytorch library for accelerating 3d deep learning research”. Source: https://arxiv.org/pdf/1911.05063.pdf, Jan 2022.

[29] Kato, H.; Beker, D.; Morariu, M.; Ando, T.; Matsuoka, T.; Kehl, W.; Gaidon, A. “Differentiable rendering: A survey”. Source: https://arxiv.org/pdf/2006.12057.pdf, Jan 2022.

[30] Kingma, D. P.; Ba, J. “Adam: A method for stochastic optimization”. Source: https://arxiv.org/pdf/1412.6980v5.pdf, Jan 2022.

[31] Krizhevsky, A.; Sutskever, I.; Hinton, G. E. “Imagenet classification with deep convolutional neural networks”. In: Advances in Neural Information Processing Systems (NIPS), 2012, pp. 1097–1105.

[32] LeCun, Y.; Bengio, Y.; Hinton, G. “Deep learning”, *Nature*, vol. 521, May 2015, pp. 436–444.

[33] LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; Jackel, L. D. “Backpropagation applied to handwritten zip code recognition”, *Neural Computation*, vol. 1, Dec 1989, pp. 541–551.

[34] LeCun, Y.; Haffner, P.; Bottou, L.; Bengio, Y. “Object recognition with gradient-based learning”. Springer, 1999, 1–26p.

[35] Li, K.; Pham, T.; Zhan, H.; Reid, I. “Efficient dense point cloud object reconstruction using deformation vector fields”. In: European Conference on Computer Vision (ECCV), 2018, pp. 497–513.

[36] Liao, Y.; Donne, S.; Geiger, A. “Deep marching cubes: Learning explicit surface representations”. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 2916–2925.

[37] Lin, K.; Wang, L.; Liu, Z. “End-to-end human pose and mesh reconstruction with transformers”. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 1954–1963.

[38] Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C. L. “Microsoft coco: Common objects in context”. In: European Conference on Computer Vision (ECCV), 2014, pp. 740–755.

[39] Liu, H.-T. D.; Tao, M.; Jacobson, A. “Paparazzi: surface editing by way of multi-view image processing.”, *ACM Transactions on Graphics*, vol. 37, Sep 2018, pp. 1–11.

[40] Lorensen, W. E.; Cline, H. E. “Marching cubes: A high resolution 3d surface construction algorithm”, *ACM SIGGRAPH Computer Graphics*, vol. 21, Jul 1987, pp. 163–169.

[41] McCulloch, W. S.; Pitts, W. “A logical calculus of the ideas immanent in nervous activity”, *The Bulletin of Mathematical Biophysics*, vol. 5, Dec 1943, pp. 115–133.

[42] Mescheder, L.; Oechsle, M.; Niemeyer, M.; Nowozin, S.; Geiger, A. “Occupancy networks: Learning 3d reconstruction in function space”. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4460–4470.

[43] Mitchell, T. M. “Machine Learning”. McGraw-Hill International Editions, 1997, 432p.

[44] Murphy, K. P. “Machine learning: a probabilistic perspective”. MIT press, 2012, 1104p.

[45] Niemeyer, M.; Mescheder, L.; Oechsle, M.; Geiger, A. “Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision”. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 3504–3515.

[46] Raghu, M.; Unterthiner, T.; Kornblith, S.; Zhang, C.; Dosovitskiy, A. “Do vision transformers see like convolutional neural networks?” Source: https://arxiv.org/pdf/2108.08810.pdf, Jan 2022.

[47] Ramachandran, P.; Parmar, N.; Vaswani, A.; Bello, I.; Levskaya, A.; Shlens, J. “Stand-alone self-attention in vision models”. In: Advances in Neural Information Processing Systems (NIPS), 2019, pp. 1–13.

[48] Ravi, N.; Reizenstein, J.; Novotny, D. R.; Gordon, T.; Lo, W.-Y.; Johnson, J. C.; Gkioxari, G. “Accelerating 3d deep learning with pytorch3d”. In: SIGGRAPH Asia Courses, 2020, pp. 1–18.

[49] Roveri, R.; Öztireli, A. C.; Pandele, I.; Gross, M. “Pointpronets: Consolidation of point clouds with convolutional neural networks”. In: Computer Graphics Forum (Eurographics), 2018, pp. 87–99.

[50] Rudin, W.; et al.. “Principles of mathematical analysis”. McGraw-hill New York, 1964, 352p.

[51] Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. “Learning representations by back-propagating errors”, *Nature*, vol. 323, Oct 1986, pp. 533–536.

[52] Salvi, A.; Gavenski, N.; Pooch, E.; Tasoniero, F.; Barros, R. “Attention-based 3d object reconstruction from a single image”. In: International Joint Conference on Neural Networks (IJCNN), 2020, pp. 1–8.

[53] Shen, D.; Wang, G.; Wang, W.; Min, M. R.; Su, Q.; Zhang, Y.; Li, C.; Henao, R.; Carin, L. “Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms”. In: 56th Annual Meeting of the Association for Computational Linguistics (ACL), 2018, pp. 440–450.

[54] Song, Y.; Ermon, S. “Generative modeling by estimating gradients of the data distribution”. In: Advances in Neural Information Processing Systems (NIPS), 2019, pp. 11918–11930.

[55] Su, W.; Zhu, X.; Cao, Y.; Li, B.; Lu, L.; Wei, F.; Dai, J. “Vl-bert: Pre-training of generic visual-linguistic representations”. In: International Conference on Learning Representations (ICLR), 2020, pp. 1–14.

[56] Sun, X.; Wu, J.; Zhang, X.; Zhang, Z.; Zhang, C.; Xue, T.; Tenenbaum, J. B.; Freeman, W. T. “Pix3d: Dataset and methods for single-image 3d shape modeling”. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 2974–2983.

[57] Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; Jégou, H. “Training data-efficient image transformers and distillation through attention”. In: International Conference on Machine Learning (ICML), 2021, pp. 10347–10357.

[58] Tulsiani, S.; Zhou, T.; Efros, A. A.; Malik, J. “Multi-view supervision for single-view reconstruction via differentiable ray consistency”. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2626–2634.

[59] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; Polosukhin, I. “Attention is all you need”. In: Advances in Neural Information Processing Systems (NIPS), 2017, pp. 5998–6008.

[60] Wang, S.; Li, B. Z.; Khabsa, M.; Fang, H.; Ma, H. “Linformer: Self-attention with linear complexity”. Source: https://arxiv.org/pdf/2006.04768.pdf, Jan 2022.

[61] Wang, W.; Xie, E.; Li, X.; Fan, D.-P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; Shao, L. “Pyramid vision transformer: A versatile backbone for dense prediction without convolutions”. In: IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 568–578.

[62] Wang, X.; Girshick, R.; Gupta, A.; He, K. “Non-local neural networks”. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 7794–7803.

[63] Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; Xiao, J. “3d shapenets: A deep representation for volumetric shapes”. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1912–1920.

[64] Xiang, Y.; Mottaghi, R.; Savarese, S. “Beyond pascal: A benchmark for 3d object detection in the wild”. In: IEEE Winter Conference on Applications of Computer Vision (WACV), 2014, pp. 75–82.

[65] Xie, Q.; Luong, M.-T.; Hovy, E.; Le, Q. V. “Self-training with noisy student improves imagenet classification”. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 10687–10698.

[66] Xiong, Y.; Zeng, Z.; Chakraborty, R.; Tan, M.; Fung, G.; Li, Y.; Singh, V. “Nyströmformer: A nyström-based algorithm for approximating self-attention”, *AAAI Conference on Artificial Intelligence*, vol. 35, May 2021, pp. 14138–14148.

[67] Ye, L.; Rochan, M.; Liu, Z.; Wang, Y. “Cross-modal self-attention network for referring image segmentation”. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 10502–10511.

[68] Yuan, L.; Chen, Y.; Wang, T.; Yu, W.; Shi, Y.; Jiang, Z.-H.; Tay, F. E.; Feng, J.; Yan, S. “Tokens-to-token vit: Training vision transformers from scratch on imagenet”. In: IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 558–567.

[69] Zaheer, M.; Guruganesh, G.; Dubey, K. A.; Ainslie, J.; Alberti, C.; Ontanon, S.; Pham, P.; Ravula, A.; Wang, Q.; Yang, L.; et al.. “Big bird: Transformers for longer sequences.” In: Advances in Neural Information Processing Systems (NIPS), 2020, pp. 17283–17297.

[70] Zhang, H.; Goodfellow, I.; Metaxas, D.; Odena, A. “Self-attention generative adversarial networks”. In: International Conference on Machine Learning (ICML), 2019, pp. 7354–7363.

[71] Zhao, H.; Jia, J.; Koltun, V. “Exploring self-attention for image recognition”. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 10076–10085.

[72] Zheng, S.; Lu, J.; Zhao, H.; Zhu, X.; Luo, Z.; Wang, Y.; Fu, Y.; Feng, J.; Xiang, T.; Torr, P. H.; Zhang, L. “Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers”. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 6881–6890.

APPENDIX A – QUESTIONNAIRE

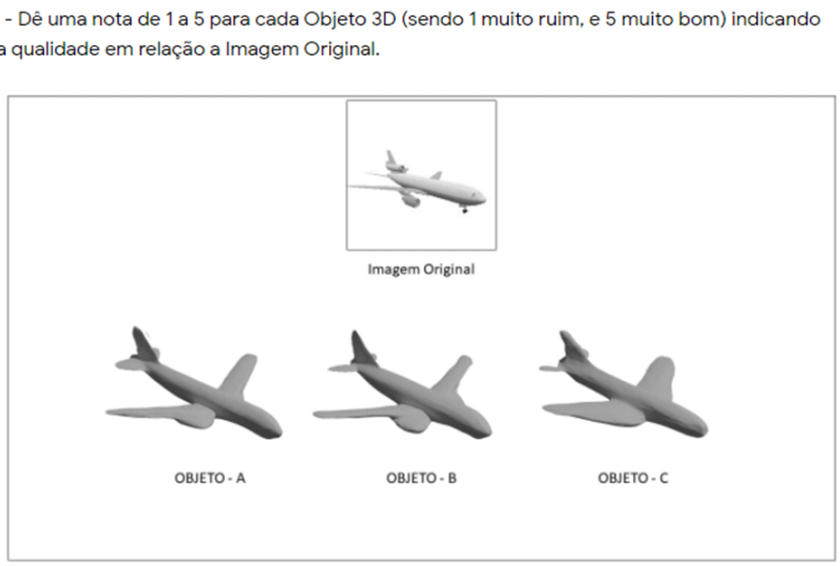

Figure A.1 – Question 1 for reconstruction related to the Plane object class. Question: "Q1 - Rate each 3D Object from 1 to 5 (where 1 is strongly poor, and 5 is strongly good) indicating their quality considering the Original Image."

Figure A.2 – Question 2 for reconstruction related to the Plane object class. Question: "Q2 - Rate each 3D Object from 1 to 5 (where 1 is strongly poor, and 5 is strongly good) indicating their quality considering the Original Image."

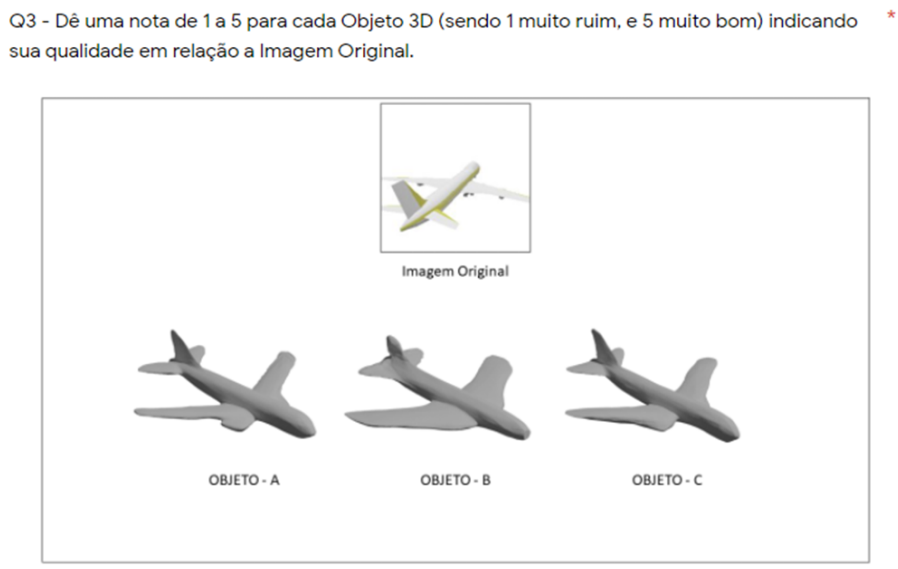

Figure A.3 – Question 3 for reconstruction related to the Plane object class. Question: "Q3 - Rate each 3D Object from 1 to 5 (where 1 is strongly poor, and 5 is strongly good) indicating their quality considering the Original Image."

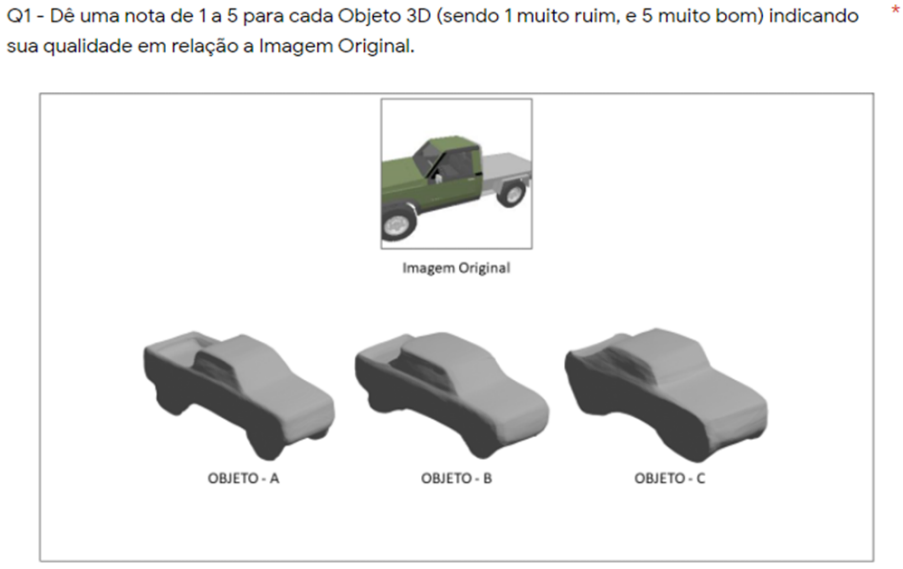

Figure A.4 – Question 1 for reconstruction related to the Car object class. Question: "Q3 - Rate each 3D Object from 1 to 5 (where 1 is strongly poor, and 5 is strongly good) indicating their quality considering the Original Image."

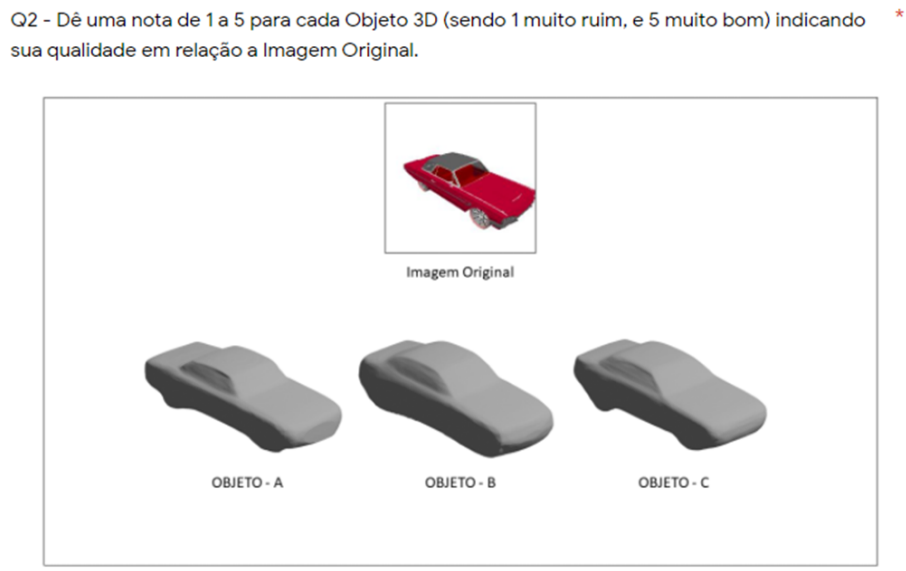

Figure A.5 – Question 2 for reconstruction related to the Car object class. Question: "Q3 - Rate each 3D Object from 1 to 5 (where 1 is strongly poor, and 5 is strongly good) indicating their quality considering the Original Image."

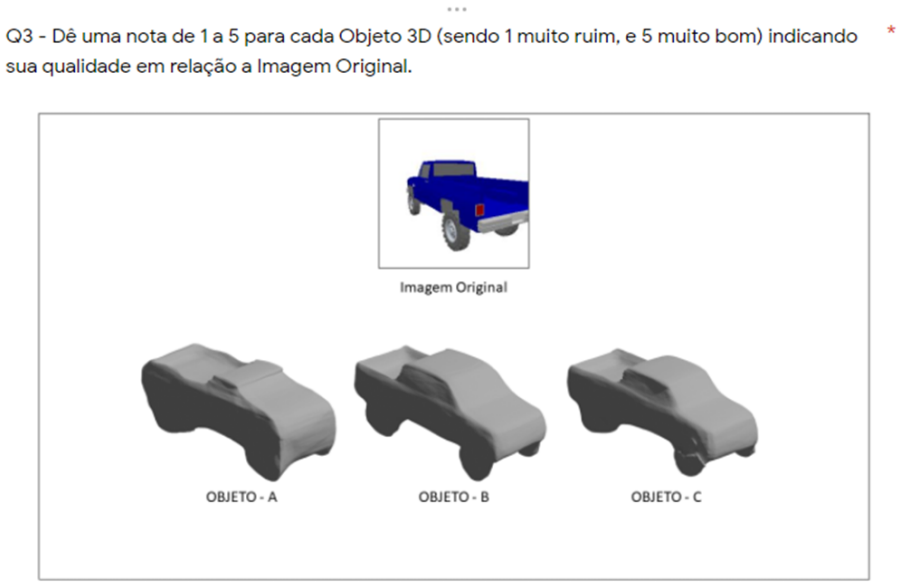

Figure A.6 – Question 3 for reconstruction related to the Car object class. Question: "Q3 - Rate each 3D Object from 1 to 5 (where 1 is strongly poor, and 5 is strongly good) indicating their quality considering the Original Image."

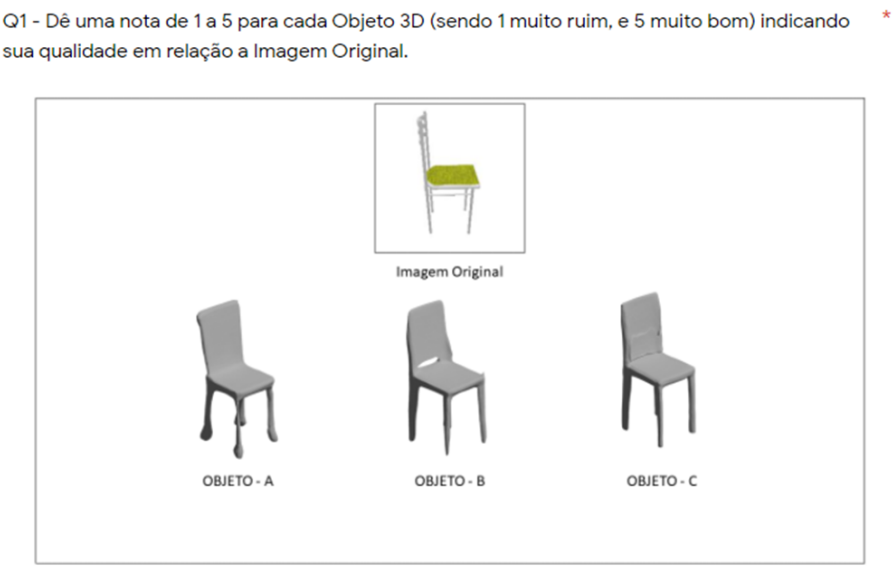

Figure A.7 – Question 1 for reconstruction related to the Chair object class. Question: "Q3 - Rate each 3D Object from 1 to 5 (where 1 is strongly poor, and 5 is strongly good) indicating their quality considering the Original Image."

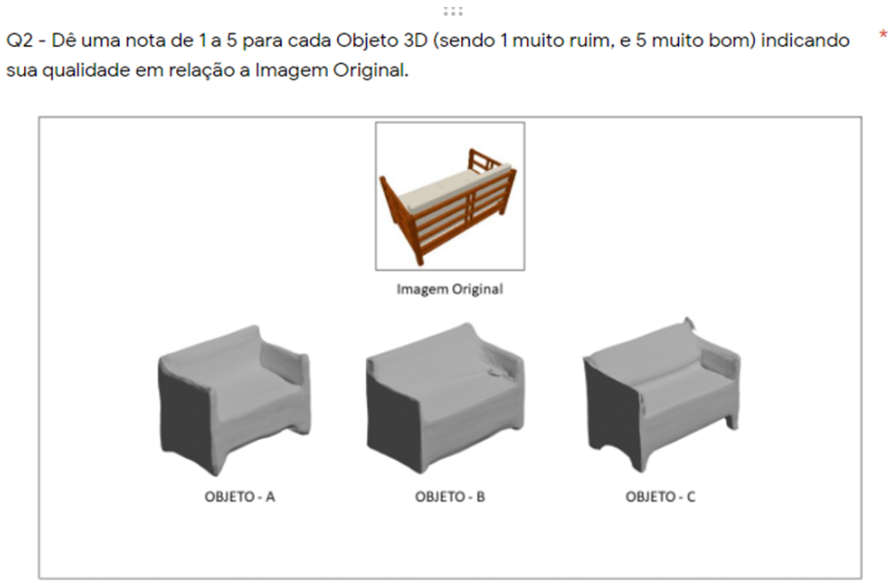

Figure A.8 – Question 2 for reconstruction related to the Chair object class. Question: "Q3 - Rate each 3D Object from 1 to 5 (where 1 is strongly poor, and 5 is strongly good) indicating their quality considering the Original Image."

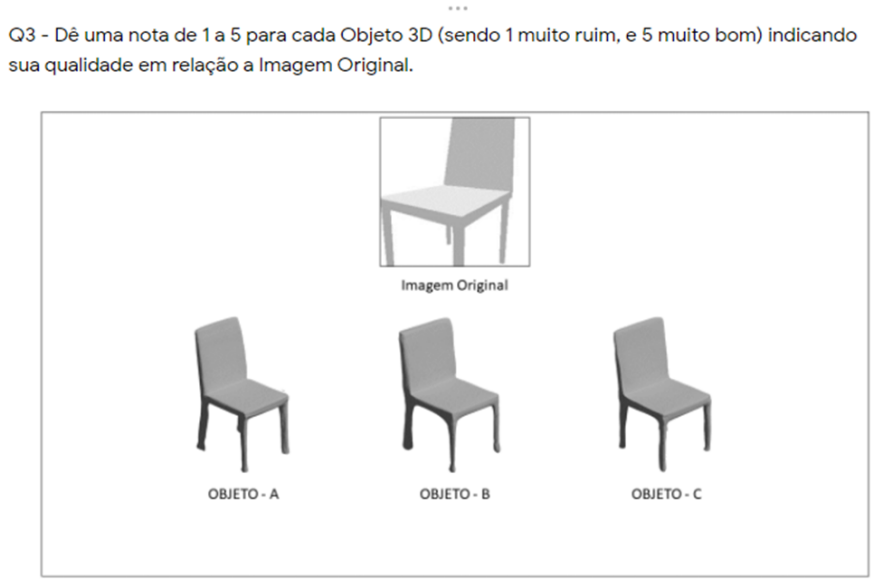

Figure A.9 – Question 3 for reconstruction related to the Chair object class. Question: "Q3 - Rate each 3D Object from 1 to 5 (where 1 is strongly poor, and 5 is strongly good) indicating their quality considering the Original Image."

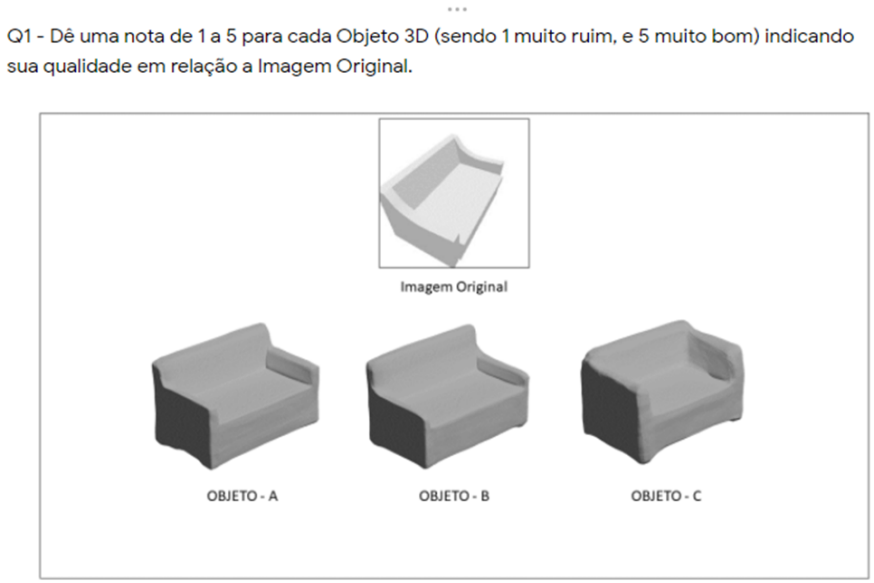

Figure A.10 – Question 1 for reconstruction related to the Sofa object class. Question: "Q3 - Rate each 3D Object from 1 to 5 (where 1 is strongly poor, and 5 is strongly good) indicating their quality considering the Original Image."

Figure A.11 – Question 2 for reconstruction related to the Sofa object class. Question: "Q3 - Rate each 3D Object from 1 to 5 (where 1 is strongly poor, and 5 is strongly good) indicating their quality considering the Original Image."

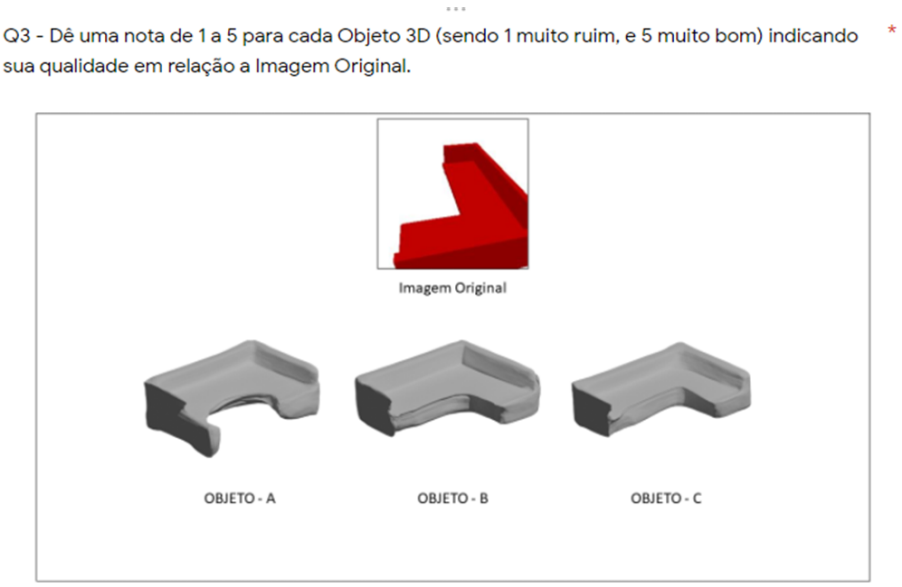

Figure A.12 – Question 3 for reconstruction related to the Sofa object class. Question: "Q3 - Rate each 3D Object from 1 to 5 (where 1 is strongly poor, and 5 is strongly good) indicating their quality considering the Original Image."

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Graduação
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: prograd@pucrs.br
Site: www.pucrs.br