Politecnico di Torino

Politecnico di Torino

Master's degree course in Data Science and Engineering

A.a. 2024/2025

Graduation Session July 2025

Depth Estimation from technical drawings and 3D Mesh Reconstruction with Deep Learning

Supervisor: Paolo Garza

candidate: Emanuele De Leo

Contents

Chapter 1

Introduction

This thesis addresses the challenge of automatically generating 3D models from images, a fundamental problem in the field of computer vision. The task of inferring three-dimensional geometry from two-dimensional representations is particularly relevant in industries such as fashion, where design workflows frequently involve technical drawings and schematic illustrations. Brands often supply these files as the primary design assets, which must then be converted into accurate and detailed 3D models for use in digital production.

During my internship, I collaborated on a project for a fashion company that was specialized in transforming technical design files into 3D digital assets. The core objective was to accelerate the creation of high-quality assets from technical files provided by clients, thus reducing manual modeling time and enabling a more scalable workflow. To tackle this problem, I explored a variety of methods for the conversion of 2D to 3D, to 3D, focusing in particular on deep learning-based predictive models. My work involved experimenting with both convolutional neural networks (CNNs), known for their strength in capturing spatial features from images, and vision transformers (ViTs), which offer a global attention approach to visual representation learning. These architectures were evaluated for their ability to infer depth and reconstruct 3D geometry from minimal visual features. By combining practical insights from the fashion design domain with state-of-the-art techniques in machine learning, this thesis investigates scalable solutions for automating the generation of 3D objects, also referred to as meshes, in digital production.

1.1 Context

The initial work was focused on automating the generation of 3D models from design files, which typically consist of DWG files that contain simple drawings of objects from multiple perspectives, along with annotations (text notes or measures). Automating this process can significantly streamline a faster workflow for 3D artists who might have to model hundreds of products in a limited time period. By automating the conversion of 2D drawings into 3D meshes, artists can save valuable time, decreasing the likelihood of manual errors and reducing their work in refining, translating, and combining these meshes. Additionally, more powerful techniques extend beyond fashion design and could be applied to other fields where 2D-to-3D conversion is necessary, potentially enabling scalable solutions for 3D mesh construction. The reconstruction of a complete 3D asset from a single image requires not only robust depth estimation but also a strong understanding of object geometry, prior knowledge of shape structures, and the ability to infer occluded or non-visible parts of the object in the image. Although current deep learning methods already allow for the reconstruction of approximate 3D objects from real-world images, the challenge of reconstructing 3D models from schematic and line-based representations, such as simple CAD figures, remains an active and complex problem. Successfully automating this task would not only benefit sectors like fashion and, in general, industrial design, but could also support scalable 3D model generation in fields such as manufacturing, gaming, and virtual reality.

1.2 Goal

The main objective of this work is to integrate predictive deep learning models into workflows capable of quickly generating useful 3D meshes from 2D technical drawings and general images. This integration aims to support the automated reconstruction of three-dimensional assets that can be used in the digital industry.

Two different tasks are explored: the depth-based 3D mesh reconstruction from technical drawings, which involves developing a pipeline to predict depth and generate 3D meshes from an image representing the drawings, and the end-to-end mesh generation from a single image, which means directly generating a detailed 3D mesh from a single image.

Depth-Based Reconstruction from Technical Drawings

This approach focuses on predicting depth maps from schematic 2D views (e.g., front, side, top), translating simplified, line-based illustrations into coherent 3D geometries, preserving their proportions and design intent. These depth maps should encode the spatial structure of the final product and are used to generate 3D meshes through surface reconstruction techniques. This task is difficult because of the lack of detail in the input figures, which are essential black and white representations. This method is especially useful when only technical drawings are available, as often happens in CAD models or design sketches.

End-to-End Mesh Generation from a Single Image

The secondary approach involves the direct generation of a complete 3D mesh from a single image that represents the object. This approach can also be applicable on complex images such as photos, but it does not guarantee the generation of acceptable objects from poor representations, such as in the case of technical raw figures. The goal here is to produce complete 3D assets, including surface geometry and visual texture, using architectures capable of learning complex features from an image, and using them during the reconstruction of the volume. This enables the creation of accurate, visually enriched 3D assets from minimal input, paving the way for scalable and automated digital content creation. A key advantage is that it produces nearly finished models that require only minor adjustments, saving time for designers and 3D artists. Instead of building shapes from scratch, they can start from a solid base and focus on refining details, materials, or proportions. This makes the workflow faster and more accessible, especially in fields such as product visualization, game development, and virtual prototyping.

1.3 Structure

This paper is structured into two main sections: Depth estimation from Technical Drawings and 3D Mesh Reconstruction.

The first part focuses on the task of depth estimation from a single representation of a design technical file, for which a dedicated neural network has been specifically designed and trained. This section explores the model architecture, training process, and evaluation metrics in detail, presenting extensive experiments and results to validate the proposed approach. The depth estimation work does not end with the prediction of depth maps; as

an additional step, these maps are further processed to generate 3D representations, demonstrating the practical potential of the estimated depth in spatial reconstruction tasks.

The second part addresses the problem of reconstructing a 3D mesh from a single image using existing state-of-the-art methods. These models are analyzed and compared on the basis of their performance.

Although both tasks involve inferring 3D information from 2D input, they are treated independently in this work in each chapter, with the depth estimation technique representing the primary contribution.

Chapter 2

Related works

In this chapter, we explore the existing literature that contributed to the development of methods to achieve the goals of depth estimation and 3D mesh reconstruction. Covers recent works and how those works are applied in real-world scenarios.

2.1 Monocular Depth Estimation

Monocular depth estimation is a computer vision task that involves predicting the depth of objects from a single image, that is, predicting the distance of surfaces from a single camera position.

A depth estimation model takes an RGB image as input and predicts a depth mask by assigning a depth value for each pixel of the image. This method involves extracting relevant local and global features from the input to highlight several relationships in the image [figure 2.1 presented an example of depth estimation].

Local features refer to fine-grained visual features found in small regions of the image, such as edges, textures, and boundaries. These features are crucial for understanding depth discontinuities (e.g., the border between a foreground object and the background) and for preserving spatial details to construct a coherent 3D shape in a predicted depth map. Global features, on the other hand, capture broader contextual information throughout the entire image. These include the overall layout of the scene, object co-occurrence, perspective features (such as vanishing points), and semantic relationships (e.g., recognizing that a ceiling is typically farther away than a table). Global features help the model infer depth even in textureless or ambiguous regions, where local cues alone are insufficient due to the lack of details and, in some cases, also the lack of color (e.g., black and white drawings).

Feature extraction can be performed using an encoder that integrates convolutional or transformer neural networks; at the end, a decoder constructs a depth mask using the extracted features as a guideline. In the case of CNN, the encoder progressively downsamples the input image by applying convolutions to extract features at multiple scales. In the encoding stage, downsampling enables a progressive increase of the receptive field, but the feature resolution obtained with this process can be low for RGB images with fine-grained textures or intricate details. Many high-level features, including local features, are lost in this process and can thus be hard to recover. This fact is not particularly important in some tasks (it depends on the quality needs), but in the field of 3D modeling, especially in the context of fashion design, it is critical. In this case, the process of 3D modeling requires a very detailed depth mask to efficiently construct a coherent and regular three-dimensional object that meets the standards of 3D fashion design.

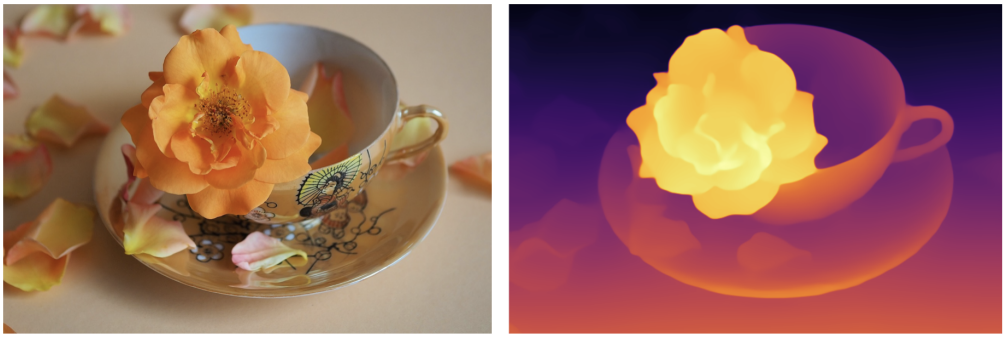

Figure 2.1: An example of a depth mask generated by the Depth Anything model starting from a complex input image

Partial solutions to this problem have been implemented that require different exploitation of resources. In this work, I designed Depthy, a CNN architecture with a Resnet-50 as encoder. This architecture was studied not only to predict depth in images, but to generate it, because the input consists of technical not colored drawings with lack of tridimensional details. On the other hand, there exist the so-called foundation models. These models integrate a vision transformer as the backbone, enabling the network to outperform predictions on different tasks. In this case, a type of these models was chosen to perform the depth estimation task. The drawback in implementing these architectures is better feature extraction without the need to down-sample the input image. This fact determines a high feature resolution (both local and global features), leading to a more detailed and precise depth mask. Both techniques exposed above include a step consisting of training

the models on popular datasets accurately designed for depth estimation and image segmentation. These datasets typically consist of RGB images paired with corresponding ground-truth depth maps, either densely annotated or sparsely sampled, and are acquired using RGB-D sensors, stereo rigs, or other sensors. In the case of CNN, the training is feasible, while in the other case the training can be huge due to both the amount of data and resources required. In any case, vision transformer-based architectures pretrained on datasets with millions of images are inherently efficient and exhibit strong generalization capabilities.

CNN-Based Monocular Depth Estimation

Various depth estimation techniques are mostly based on convolutional neural networks due to the strong performance of the learned representations. CNN-based models typically formulate depth estimation as a per-pixel regression, that is then solved using fully convolutional networks. In this case, monocular depth estimation aims to map an input image I with dimension H×W×C to an output pixel-wise depth map Yˆ with dimension H×Wx1, with H, W being the height and width of the image and C the number of RGB channels. CNN-based methods typically establish this mapping by learning a neural network F such that Y = F(I). To learn the parameters of such networks, direct supervision losses L(Y) are applied to the output Y to approximate the depth map of the ground truth Y. Self-supervision losses are applied to the reconstructed image that is synthesized to approximate the target image I [In Figure 2.2 a typical CNN architecture is presented.]

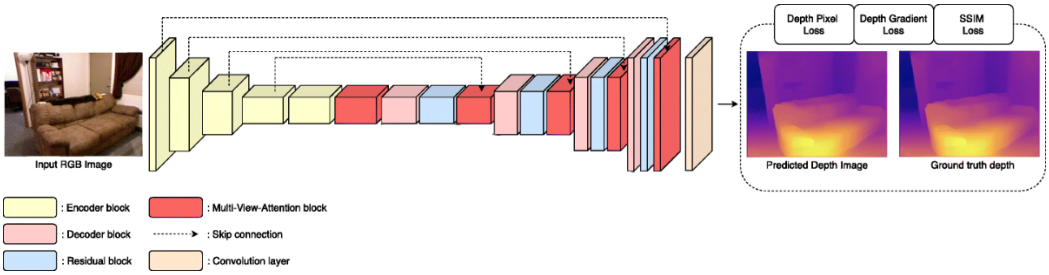

Figure 2.2: A classical representation of a CNN architecture designed for monocular depth estimation

VIT-Based Monocular Depth Estimation

Recent architectures to do monocular depth estimation rely on Vision Transformers (ViTs), often referred to as foundation models. These models leverage self-attention mechanisms to capture useful dependencies and extract rich contextual features from input images.

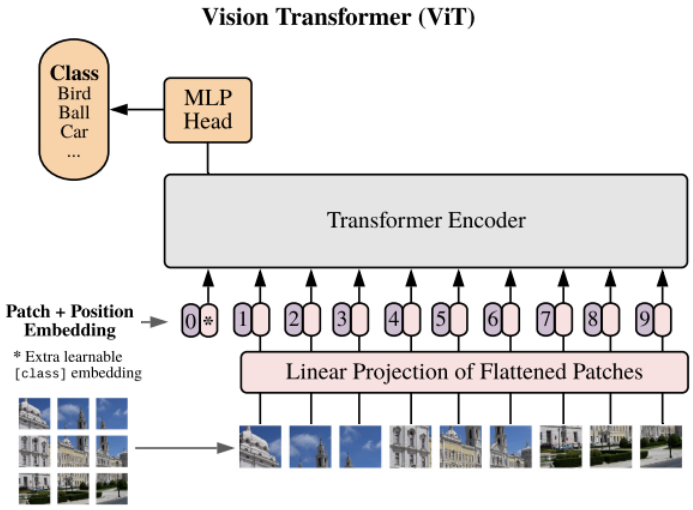

Figure 2.3: A representation of a vision transformer architecture

By processing images as sequences of patches, ViTs are able to learn global representations that are particularly beneficial for tasks like depth estimation, where understanding the spatial structure of the entire scene is crucial. The resulting embeddings typically lead to more accurate and detailed depth maps compared to traditional CNN-based encoders. An example of a foundation model is the Depth Anything model, one of the most popular models to perform depth estimation on a wide range of different images. Depth Anything is very powerful, especially for the particular technique used to train it, which consists in augmenting the dataset with self-made synthesized labels. Specifically, during training, a pretrained teacher model is used to generate depth annotations for a wide range of images (around 62M different images from eight public datasets); the resulting labeled images are then jointly learned with other labeled images from a student model. This procedure was proven to increase the ability of the model to predict depth values. The model has a typical encoder-decoder structure that integrates a Vision Transformer as encoder and a Dense Prediction Transformer as

decoder (DPT). Practically, tokens from various stages of the vision transformer are assembled into image-like representations at various resolutions, and progressively they are combined into full-resolution predictions using a convolutional decoder. The use of the Depth Anything model in this work should be intended as a guide to compare its outputs with the results from another more compact and efficient network [In Figure 2.3 the vision transformer architecture is presented].

2.2 Vision Transformer

A Vision Transformer is a neural network architecture that was inspired by the success of transformers in natural language processing. It allows encoding relevant information from images in order to improve the performance and the quality of extracted features in the predictions. In the case of a vision transformer, an image is split into fixed-size patches, each of them is then linearly embedded, positional encoding is added, and the result is fed to a standard transformer encoder. The encoder processes the patch embeddings into multiple layers, each containing a multihead self-attention module. The multi-head self-attention layer is able to capture relationships between the different patches. The goal is to take an average over the features of the patches, weighting them depending on their values. In other words, we want to automatically decide which part we want to look at more than others. This mechanism reflects the same logic used in natural language processing (NLP) models, where self-attention is applied to the sequence of words in a sentence to construct a contextual response. Just as a language model receives a sentence as a prompt and computes attention scores to understand the contextual relationships between words, so that it can generate coherent and meaningful responses, a vision transformer applies attention across different areas of the images to find relationships between them.

After the self-attention block, the output is passed through a feedforward neural network, typically consisting of two fully connected layers with a nonlinear activation function in between (such as ReLU).

The final step is crucial since it introduces non-linearity, allowing the model to learn complex relationships in the parts of the image. These components are surrounded by residual connections and layer normalization, which improve gradient flow and training stability. The self-attention mechanism plays a central role in ViT. It enables the model to dynamically weight patch features based on their contextual relevance, effectively allowing it to "attend" to different parts of the image depending on the task. This global

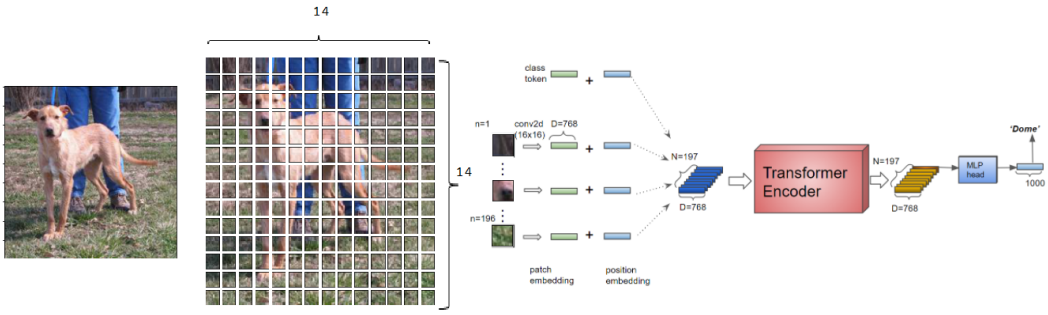

Figure 2.4: How a vision transformer works

attention mechanism gives to ViT a strong advantage in tasks that require capturing long-range dependencies and global context, such as semantic segmentation, depth estimation and 3D shape reconstruction. Compared to convolutional networks, which inductively bias the model toward local patterns through weight sharing and locality, Vision Transformers are more data hungry, more efficient and effective. With sufficient data, they often learn more expressive and generalized features, leading to better predictions that can be important in sensitive scenarios [In figure 2.4 the processing of a vision transformer architecture is presented].

2.3 3D Mesh reconstruction

A mesh is a three-dimensional entity composed of polygonal units, usually triangles or quadrilaterals, organized in a data structure that contains the coordinates of the vertices and the faces that connect these vertices in the 3D plane. Each face defines a small part of the surface and, when combined, they form the complete shape of the object [In figure 2.5 an example of a mesh is presented].

Meshes are a fundamental part of 3D generation and are widely used in digital fields such as gaming, cinematography, augmented reality, and virtual prototyping. In computer vision and graphics, 3D reconstruction refers to the process of recovering the shape and appearance of an object starting from 2D inputs, such as RGB images.

Traditional 3D mesh reconstruction methods rely on multi-view geometry, where multiple images taken from different perspectives of an object are used to estimate its depth and structure based on geometric constraints. However, these approaches often require a large number of views and highly controlled

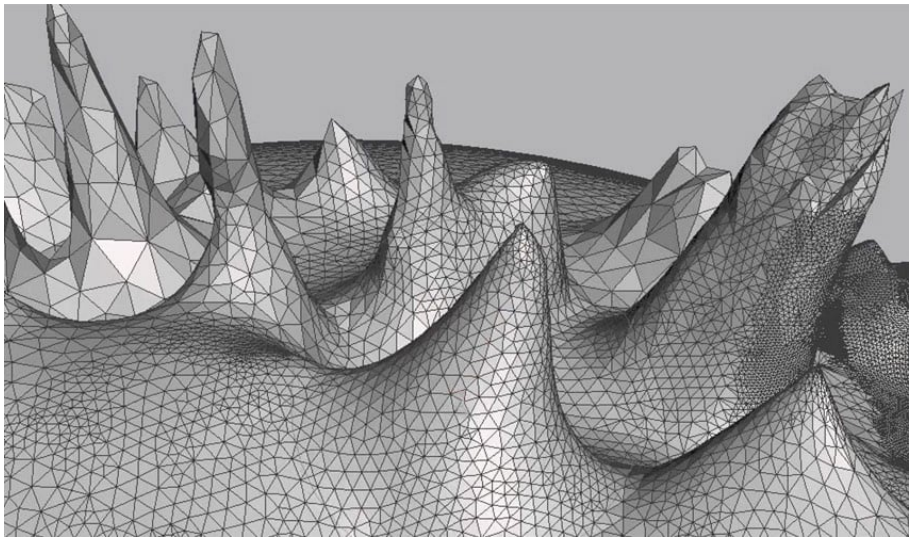

Figure 2.5: Example of a mesh with triangular faces

acquisition setups to produce accurate results. Recent learning-based methods are capable of reconstructing complex 3D scenes from a discrete set of views, even from a single image, using deep neural networks trained on large-scale datasets to infer depth and structure. Among these, neural radiance fields (NeRF) have emerged as a particularly effective utility technique in this field. NeRF can support the inference of a 3D scene by learning a continuous volumetric representation from multiple views, approximating how color and density vary along light rays passing through the scene. Given a sparse set of input images, NeRF can synthesize novel views with fine-grained detail. Its strength lies in its ability to generalize across viewpoints and to capture subtle variations in geometry and lighting, making it a cornerstone in recent research on view synthesis and neural rendering. However, NeRF models are often computationally intensive to train and render, which limits their applicability in real-time or resource-constrained environments. To address these limitations, several lightweight versions, such as Tiny NeRF, have been introduced, but they typically offer reduced quality compared to the original models [In figure 2.6 a schema of 3D mesh reconstruction from an image is presented].

Deep learning models based on convolutional neural networks (CNNs) or transformers have shown the ability to reconstruct 3D objects quickly and accurately. CNNs are often trained on large datasets and use specific loss functions to learn internal patterns used to make predictions on data coming from the real-world. On the other hand, transformer-based models

can generalize better to real-world input and are able to capture long-range dependencies between image parts.

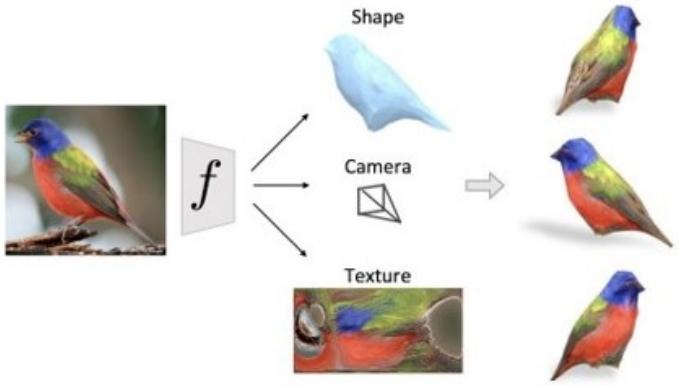

Figure 2.6: Schema of 3D reconstruction from a 2D sketch.

Today, there are large datasets with millions of 3D meshes, such as ShapeNet, which are used to train models capable of generating 3D objects. These datasets often contain not only the mesh structure (vertices and faces), but also extra information such as surface normals, textures, materials, and metadata like object categories or tags. This allows the models to learn rich and varied features and generate more realistic and detailed 3D output.

2.3.1 NeRF (Neural Radiance Field)

Neural Radiance Field (NeRF) is a technique that enables the synthesis of novel views of complex 3D scenes by learning the geometry and appearance of the scene from a discrete set of posed images, allowing interpolation between views and novel view synthesis to fill in unobserved regions. It represents a 3D scene using a continuous function parameterized by the weights of a multilayer perceptron. The input consists of 5D coordinates composed of 3D positions (x, y, z) and a viewing directions (azimuthal and polar angles) with respect to the scene. This allows the network to encode how the appearance of the scene changes with respect to different camera positions.

The output is a volume density and an RGB color. To enable the multilayer perceptron to represent high-frequency details, NeRF maps the input

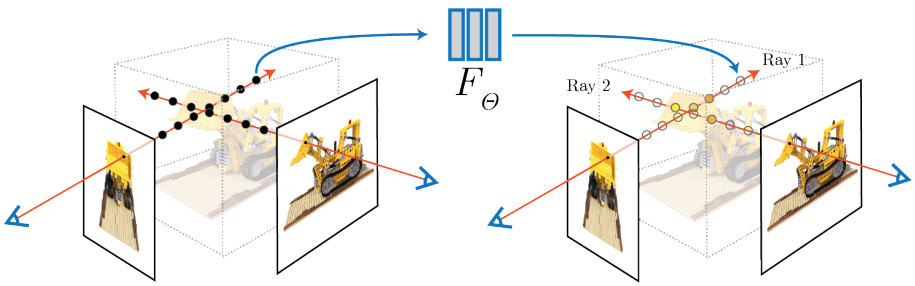

Figure 2.7: In this image is represented the method used by NeRF to synthesize novel views from a 3D scene

5D coordinates to a higher-dimensional space by applying a complex sinusoidal function. This allows the network to map the coordinates from 3D to a high-dimensional space, allowing to capture high-frequency details. Volume rendering is done following these steps: for each pixel in the image, a ray is sent r(t) = o + td from the origin o, along the viewing direction d [in figure 2.7 the NeRF methodology is presented]. N sample points are then generated along the ray and, for each of them, the multilayer perceptron predicts the density and the color. The MLP is typically composed of eight fully connected layers with 256 neurons each, with skip connections and a separation between density prediction and color prediction [in figure 2.8 a multilayer perceptron is presented]. The final color C(r) for the pixel is computed by summing the contributions from all the samples as follows:

$$C(r) = \sum_{i=1}^{N} T_i \cdot \alpha_i \cdot \mathbf{c}_i \tag{2.1}$$

where:

- $\mathbf{c}_i$ is the RGB color predicted by the MLP at the i-th sample point,
- $\alpha_i = 1 - \exp(-\sigma_i \delta_i)$ is the opacity at sample i,
- σ_i is the predicted volume density at that point,
- δ_i is the distance between consecutive sample points along the ray,
- $T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$ is the accumulated transmittance, representing the probability that the ray has not been occluded before reaching point i.

The multilayer perceptron is trained to minimize the difference between the rendered pixel colors C(r) and the ground truth pixel colors Cgt(r) from the training image.

In some 3D reconstruction architectures, NeRF is integrated with triplane representations to efficiently sample features in space. The continuous density field predicted by NeRF can then be converted into an explicit mesh representation using methods like Marching Cubes.

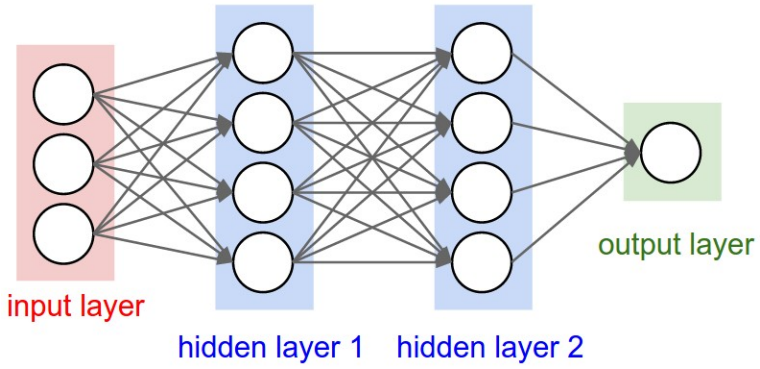

Figure 2.8: A standard multilayer perceptron

2.3.2 Large Reconstruction Model

A Large Reconstruction Model (LRM) predicts a 3D mesh from an object in an image, including its texture. LRMs combine scalable transformer encoders with specialized 3D decoders and neural radiance fields to infer high-fidelity meshes and textures from a single scene. A LRM has hundreds of millions of learnable parameters to predict a neural radiance field (NeRF) from the input image in order to construct a detailed 3D model of the scene.

A classical LRM integrates a pretrained Vision Transformer (ViT) that ingests the input image and outputs multiscale feature maps. By attending across global and local contexts, the encoder captures both coarse shape cues and fine details (e.g., surface patterns), providing a rich latent representation for 3D mesh reconstruction. The encoded features are projected into three orthogonal 2D feature planes, commonly referred to as the XY, YZ, and ZX. Each plane encodes volumetric information along one axis, enabling an efficient intermediate representation that bridges 2D image features and 3D volumetric fields without the full memory footprint of a voxel grid. The result of this concatenation is a useful triplane that can be treated as a

starting scene to make a prediction of the volume. NeRF module samples points in 3D space by querying the triplane features. The model predicts per-point density and RGB color vectors, which are then volume-rendered into surface geometry and texture. This neural rendering stage produces high-resolution details, including subtle shading and illumination effects. From the continuous radiance field, a mesh is extracted often via marching cubes on the inferred density field. UV coordinates for texture mapping are derived either directly from the radiance field or via a learned texture decoder, resulting in a fully textured 3D mesh ready for downstream use.

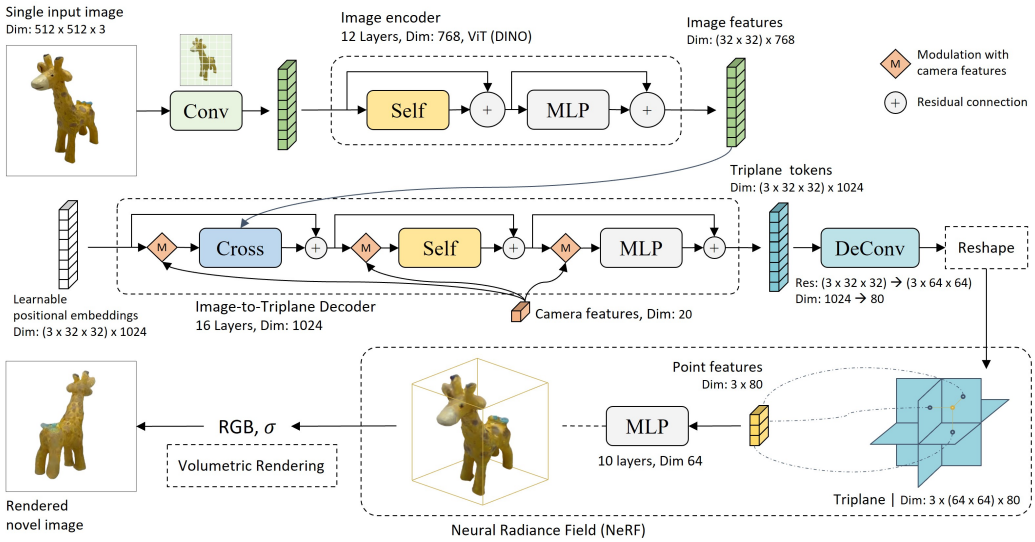

Figure 2.9: The tripoSR architecture and how it works

TripoSR

TripoSR is an example of a large reconstruction model. It has a vision transformer-based architecture, and it can create a high-quality 3D mesh from a single 2D image [in figure 2.11 an example of model outcome is presented]. TripoSR addresses the task of 3D mesh reconstruction by using an image-to-triplane decoder and a triplane-based neural radiance field (NeRF) [in figure 2.9 the full architecture is reported]. TripoSR can efficiently generate 3D shape, texture, and lighting information of an object in an image. The encoder is a pretrained vision transformer model (DINO) that converts an image into vectors encoding global and local features of the processed

image. The information is fed into the two decoders to predict the 3D mesh with texture. The image-to-triplane decoder transforms the vectors into a triplane-NeRF representation, which is a compact 3D representation. This decoder integrates self-attention layers and cross-attention layers. The self-attention layer allows to learn relationships between different regions of the triplane, while the cross-attention layer allows for attending to the vectors from the encoder, incorporating these features into the triplane representation. Finally, the NeRF model consists of a stack of multilayer perceptrons, which predict the color and density of a 3D point of the triplane. The result is a 3D textured mesh that respects the details of the object in the input image. The training of this architecture was done by applying particular loss functions that reflect the perceptual similarity between the rendered views and the input images. Among them, LLPIPS is the perceptual loss based on LPIPS (Learned Perceptual Image Patch Similarity), and it was used to ensure that the reconstructed object preserves fine-grained visual details that are meaningful to human perception.

Parameter		Value
Image Tokenizer	image resolution	512×512
	patch size	16
	# attention layers	12
	# feature channels	768
Triplane Tokenizer	# tokens	$32 \times 32 \times 3$
	# channels	16
Backbone	# channels	1024
	attention layers	16
	# attention heads	16
	attention head dim	64
	cross attention dim	768
Triplane Upsampler	factor	2
	# input channels	1024
	# output channels	40
	output shape	$64 \times 64 \times 40$
NeRF MLP	width	64
	# layers	10
	activation	SiLU
Renderer	# samples per ray	128
	radius	0.87
	density activation	exp
	density bias	-1.0
Training	learning rate	4e−4
	optimizer	AdamW
	lr scheduler	CosineAnnealingLR
	# warm-up steps	2,000
	λ_{LPIPS}	2.0
	λ_{mask}	0.05

Figure 2.10: All the details of the tripoSR with pretrained encoder and NeRF models

The total loss function used during training is:

$$\mathcal{L}_{\text{recon}} = \frac{1}{V} \sum_{v=1}^{V} \left(\mathcal{L}_{\text{MSE}}(\hat{I}_v, I_v^{\text{GT}}) + \lambda_{\text{LPIPS}} \, \mathcal{L}_{\text{LPIPS}}(\hat{I}_v, I_v^{\text{GT}}) + \lambda_{\text{mask}} \, \mathcal{L}_{\text{mask}}(\hat{M}_v, M_v^{\text{GT}}) \right) \tag{2.2}$$

where:

- V is the number of views
- $\hat{I}_v$ is the rendered image of the v-th view
- I_v^{GT} is the corresponding ground-truth image
- $\mathcal{L}_{\text{MSE}}$ is the Mean Squared Error loss
- $\mathcal{L}_{\text{LPIPS}}$ is the perceptual loss based on LPIPS
- λ_{LPIPS} e λ_{mask} are weights which balance the loss components

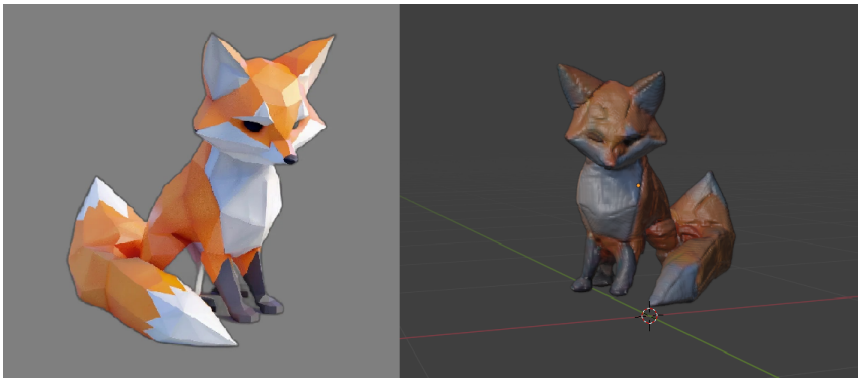

Figure 2.11: An example of the conversion from 2D image to 3D objects using the tripoSR model

The MSE loss minimizes the root mean square difference between rendered images $\hat{I}_v$ and the ground-truth images I_v^{GT}:

$$\mathcal{L}_{\text{MSE}}(\hat{I}_v, I_v^{\text{GT}}) = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{I}_v^{(i)} - I_v^{\text{GT},(i)} \right)^2 \tag{2.3}$$

where N is the number of pixels in the image.

The perceptual loss (LPIPS) compares high-level features between images, improving the visual fidelity of the reconstruction:

$$\mathcal{L}_{\text{LPIPS}}(\hat{I}_v, I_v^{\text{GT}}) = \sum_l \frac{1}{H_l W_l} \sum_{h=1}^{H_l} \sum_{w=1}^{W_l} \left\| \phi_l(\hat{I}_v)_{h,w} - \phi_l(I_v^{\text{GT}})_{h,w} \right\|_2^2 \tag{2.4}$$

where:

- $\phi_l(\cdot)$ represents the features extracted from the l-th layer of a pretrained neural network
- H_l e W_l are the height and width of the feature map to the layer l.
- $(\cdot)_{h,w}$ indicates the value at the position (h, w).

The loss function of the mask is defined as:

$$\mathcal{L}_{\text{mask}}(\hat{M}_v, M_v^{\text{GT}}) = \text{BCE}(\hat{M}_v, M_v^{\text{GT}}) \tag{2.5}$$

where $\hat{M}_v$ e M_v^{GT} are the rendered and ground-truth masks respectively of the v-th view

An important parameter of the model is the foreground ratio, which refers to the threshold used to distinguish the foreground from the background in the image during the 3D reconstruction process. Basically, this parameter determines which part of the image will be considered to generate the 3D model. A value of 0.85 indicates that 85 percent of the image information, based on a certain criterion (such as depth, intensity, or other attributes), will be classified as foreground and used for reconstruction, while the remaining 15 percent will be considered as background and ignored in the process [in figure 2.10 the values of the parameters for the default model are reported from the original paper.].

Chapter 3

Methodology

3.1 Description

The primary objective was not to create fully detailed 3D models directly from the design files, since the required level of detail for the final products was particularly high, but rather to streamline and accelerate the initial stages of the modeling process [in figure 3.1 an example of design file is reported]. To achieve this, I experimented various depth estimation techniques and I designed a convolutional neural network tailored to predict depth maps from black and white technical drawings with lack of details.

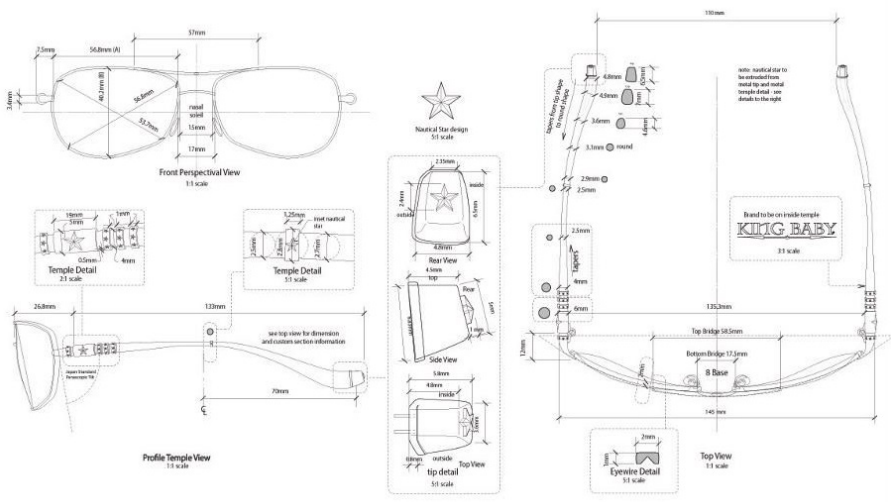

Figure 3.1: an example of sketches in a DWG file

The first step of the pipeline involved extracting the relevant figures from the DWG files. These files were preprocessed and converted to the DXF format, which is more manageable and structured for automated analysis. This type of file is organized into multiple layers, each containing different types of geometric entities. Standard entities such as text, lines, and circles were filtered out, while curves and polylines, more specific to the geometry of the object, were retained. Once cleaned, the files were converted to images, allowing figure-level manipulation [in figure 3.2 an example of a cleaned DWG file is reported]. Each figure typically represented a different view of the final 3D object. Initially, during the tests, these views were segmented and prepared for depth estimation. In the end, the chosen methodology integrated both segmentation and depth estimation, making this crucial step automatic. It is important to note that the process of predicting depth from these images should not be interpreted as a conventional depth estimation task. Unlike natural images, these technical drawings lack inherent perspective or shading cues. Therefore, the task can be seen as an agnostic generation of plausible depth values for simple and schematic representations. Alternatively, once the individual figures were extracted, they could be manually colored by a designer to suggest specific viewpoints and lighting. This preprocessing step can enable a more accurate reconstruction, though it introduced additional manual effort.

As mentioned above, the central component of the pipeline was a convolutional neural network architecture, that I called Depthy, specifically designed to generate depth maps from particular technical drawings, extended also on other black and white representations.

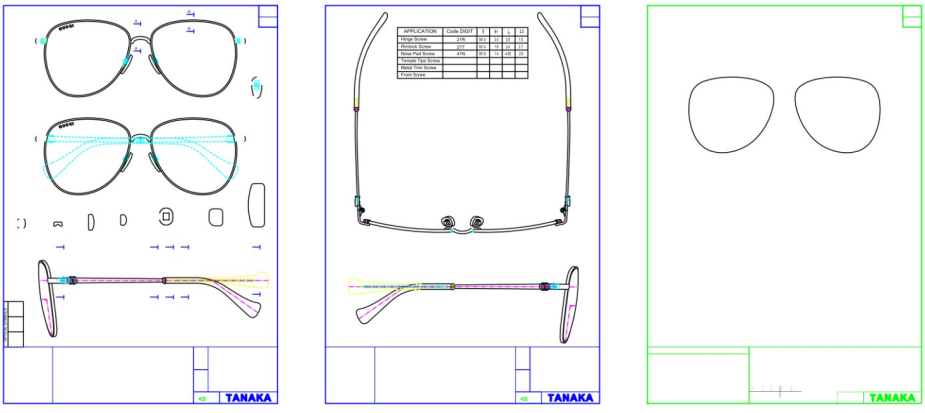

Figure 3.2: an example of a cleaned DWG file converted to image

CNNs are specifically designed to learn spatial patterns and visual representations, such as edges, textures, shapes, and complex object structures, enabling robust predictions on real-world images. ResNet-50 is a 50-layers convolutional neural network that excels at capturing multiscale features across different levels of abstraction. Its key innovation lies in its residual connections, which create direct pathways that allow information to flow from earlier layers to later layers in the network. ResNet-50 enables the training of significantly deeper networks while maintaining gradient flow throughout the entire architecture. This design allows the network to learn both low-level features (such as edges and textures) in the initial layers and high-level semantic features (such as areas of the object) in the deeper layers, resulting in a comprehensive hierarchical feature representation suitable for complex computer vision tasks [in figure 3.3 an outcome from the network is reported].

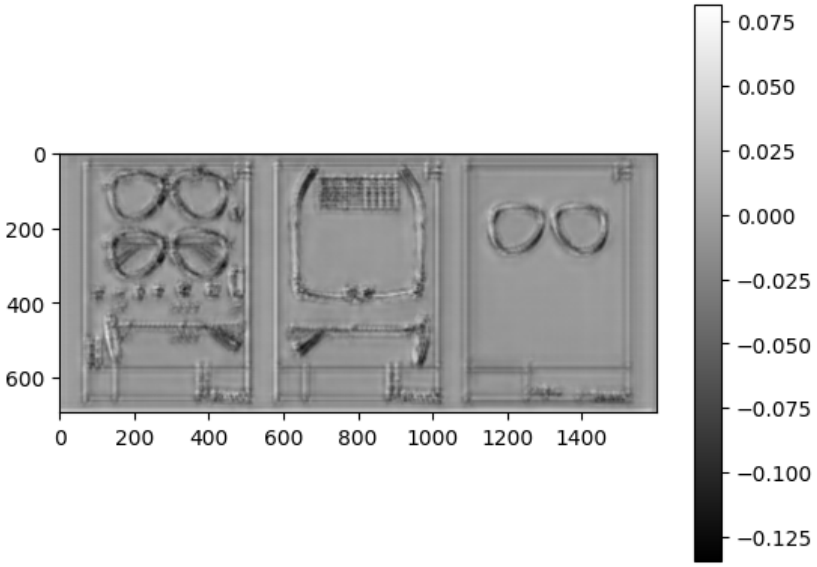

Figure 3.3: Depth map predicted by the Depthy model for the DWG presented above

The decoder consists of a Pixel Shuffle Decoder (PSD) block for upsampling and a Squeeze-and-Excitation (SE) block for channel-wise attention. Residual connections further enhance feature refinement, and a final convolutional layer generates the single-channel depth map. This map encodes per-pixel relative depth, where the value at each pixel denotes the estimated distance from the virtual camera.

To effectively train this architecture, a combination of complementary loss

functions was used, each contributing to a different aspect of depth map quality. Rather than relying solely on traditional pixel-wise metrics, the training objective was designed to balance visual accuracy, robustness, and perceptual coherence. For instance, while basic regression losses like MSE provided a strong baseline for minimizing the overall discrepancy between predictions and ground truth, they tended to over-penalize outliers and lacked sensitivity to structural details. Greater emphasis was placed on the visual quality of the outputs, as a lower loss does not necessarily guarantee that the generated outputs are visually acceptable or meaningful. To address these limitations, more adaptive losses, such as the BerHu, were introduced, which dynamically adjust the penalty applied to large residuals. This helped maintain stable gradients even in the presence of depth discontinuities or noise, a common challenge in real-world scenes. Additionally, a gradient-based loss term was added to encourage consistency in local depth variations, which is particularly beneficial for preserving edges and fine structures that are often smoothed out during naive regression. The perceptual performance of the model was further refined using SSIM-based supervision, which prioritizes structural similarity over absolute numerical correctness. This was especially useful in ensuring that the predicted depth maps retained coherent geometric patterns and spatial layouts, which are critical when these maps are later used for tasks such as 3D mesh reconstruction or novel view synthesis. The combined effect of these loss terms was to guide the network not only towards accurate depth estimation but also towards representations that are robust and visually plausible.

The conversion from 2D depth maps to 3D meshes involved projecting each pixel of the image into a 3D space, effectively reconstructing the corresponding surface geometry based on the depth value (intensity) of the pixel in the mask [in figure 3.4 this methodology is reported]. This was achieved by directly assigning the coordinates (x, y) of the pixel and using the corresponding depth value as the z coordinate to create the triangular faces of the mesh. To refine this process, a depth-to-width ratio was applied to control the scale of the 3D reconstruction. The grayscale depth map was interpreted as a 2D matrix, rotated to align with the 3D coordinate system, and normalized to the range [0, 1]. The normalized values were then scaled using the depth-to-width ratio and image dimensions to define the final depth range. A Gaussian filter was subsequently applied to the depth map to smooth the surface, reducing noise, and enhancing realism. The mesh was constructed by dividing the depth grid into triangular cells, each defined by two triangles, where the vertex coordinates were derived from the spatial positions and depth values of the pixels.

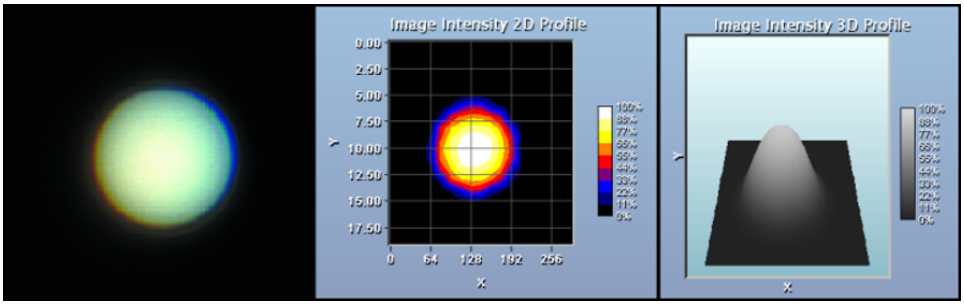

Figure 3.4: depth map to 3D mesh conversion

As mentioned above, many design files in the company already include real 2D representations of the final product. This motivated an exploration of advanced mesh reconstruction techniques capable of producing complete 3D models, including surface textures, from a single colored image. One of the most effective architectures in the domain of 3D mesh reconstruction is TripoSR, a state-of-the-art model that integrates a transformer-based encoder and neural radiance fields (NeRF). TripoSR is capable of extracting high-level features from 2D images and generating textured 3D models by having extraordinary performance in reconstructing hidden areas of the final object [in figure 3.5 an outcome from the model is reported].

To evaluate TripoSR's performance, I conducted a series of experiments using images from the company catalog and compared the reconstructed models with ground-truth 3D objects. The assessment was based on widely adopted 3D reconstruction metrics, including the Chamfer distance and the F-score, along with recall and precision scores. The results demonstrated excellent performance across a variety of object types and geometries.

Figure 3.5: Example of a final rendered object from TripoSR

3.2 Depthy

Depth estimation from black-and-white technical drawings is a challenging task due to the absence of texture, color, light information and the reliance on structural cues. The proposed architecture was designed to extract and process high-level structural features while leveraging multiscale information and attention mechanisms to produce accurate depth predictions.

3.2.1 Network architecture

Depthy is a fully convolutional network that combines an encoder-decoder architecture with attention mechanisms and multiscale feature fusion. This architecture was built to preserve as many details as possible in the decoding stage, using a well-established neural network (Resnet-50) as encoder. The model integrates different modules in the decoding stage. Enhanced pixel shuffle decoder (PSD) is utilized for efficient upsampling, incorporating a pixel shuffle operation to enhance spatial resolution and a Squeeze-and-Excitation (SE) block to selectively emphasize important channels. Additionally, the features from the encoder are refined through the Atrous Spatial Pyramid Pooling (ASPP) module, which leverage dilated convolutions to extract multi-scale contextual information.

3.2.2 Dataset

The most popular datasets for depth estimation are KITTI and NYUv2.

The KITTI dataset consists of outdoor driving scenes captured using stereo cameras and a LiDAR sensor, making it ideal for evaluating depth estimation methods in autonomous driving scenarios. It provides high-resolution images along with accurate depth ground truth generated from LiDAR, and includes various environmental conditions such as different lighting, motion, and weather scenarios.

On the other hand, the NYUv2 dataset focuses on indoor scenes and was collected using sensors. It includes over 400,000 RGB images from 464 indoor scenes across 26 different scene types, such as bedrooms, living rooms, and offices. The dataset provides dense depth maps aligned with RGB images and is widely used for benchmarking depth estimation models in indoor environments [in figure 3.6 a subset of the dataset is reported].

These datasets offer complementary challenges: KITTI emphasizes sparse, long-range depth, while NYUv2 offers dense, short-range depth. For train-

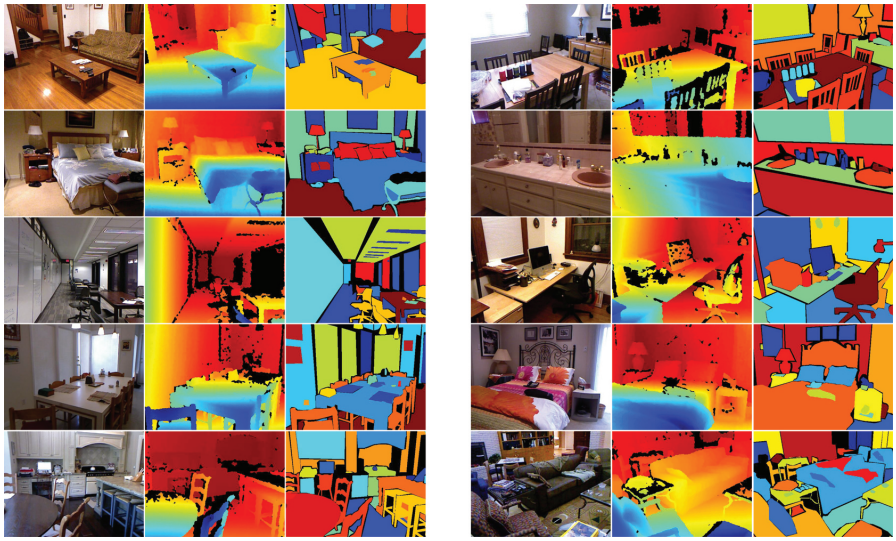

Figure 3.6: examples of images taken from the NyU 2 dataset, along with the corresponding depth mask

ing purposes, the NYUv2 dataset was selected as the most appropriate for the current task. Its densely annotated depth maps include a wide variety of everyday indoor objects with diverse shapes, sizes, and spatial configurations. This richness provides a strong foundation for training the network to efficiently segment objects and accurately predict depth values, particularly in technical images where high precision is required for depth reconstruction and scene understanding starting from raw black and white figures which lack particular details.

3.2.3 Implementation

The Depthy model was implemented with a pretrained Resnet-50 to predict depths in input technical images. To improve performance, multiple tests were carried out. These tests involved training the architecture with the NYU2 dataset, that contains detailed depth masks along with ground truth RGB images. As simplicity, the initial tests involved the use of Mean Squares Error (MSE) as loss function to test the network capabilities; then, a complex combination of losses was used. These losses were Berhu loss, gradient loss and SSIM.

First, a pretrained encoder is used to capture base features from the input drawings. The encoder processes the input drawing and it produces feature maps at decreasing spatial resolutions. This process involves an initial convo-

lution, batch normalization, and ReLU activation to process the input. Then other four layers progressively reduce the spatial resolution while increasing the semantic richness of the features [in figure 3.7 the Resnet-50 architecture is reported].

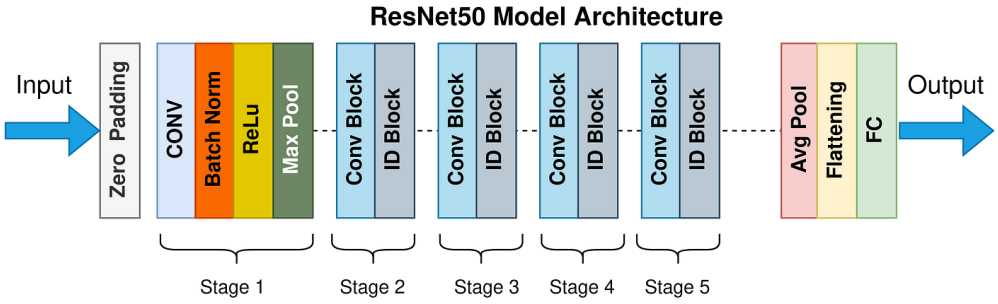

Figure 3.7: the resent-50 architecture

After that, the ASPP module is used to capture additional contextual information. The ASPP module enhances the global context by capturing features at multiple scales. It uses dilated convolutions with different dilation rates (1, 6, 12, 18) to extract features at varying receptive fields. A global average pooling branch is included to encode global contextual information. The outputs from all branches are concatenated and passed through a 1×1 convolution to produce a unified feature map. The ASPP module plays a crucial role in enhancing the skip connections from the encoder. By applying dilated convolutions, this module is used to extract multi-scale contextual information, enriching the features passed to the decoder. The use of batch normalization and ReLU activation further ensures that the processed features remain well-structured and contribute effectively to the depth reconstruction [in figure 3.8 the ASPP module is reported].

The resulting features are then fed to a decoder that progressively upsamples them to reconstruct a depth map. To achieve this, it relies on a Pixel Shuffle Decoder (PSD) module, which integrates two key components. The first is the Pixel Shuffle, a sub-pixel convolution technique that enables efficient upsampling by redistributing low-resolution feature maps into a higher-resolution output [in figure 3.9 the pixel shuffle operation is reported]. The second is the Squeeze-and-Excitation (SE) Block, which introduces channel-wise attention, enhancing the most relevant features while suppressing those that are less significant by looking at the values in each channel [in figure 3.10 the functionality of the SE block is explained].

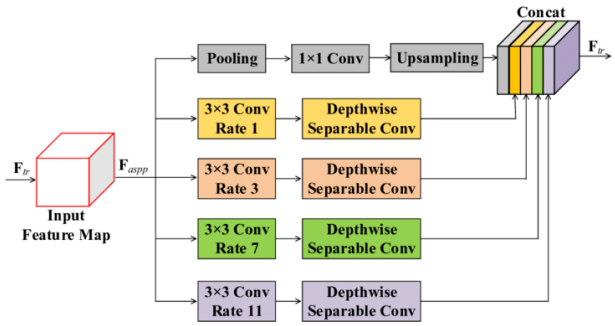

Figure 3.8: Atrous Spatial Pyramid Pooling (ASPP) module

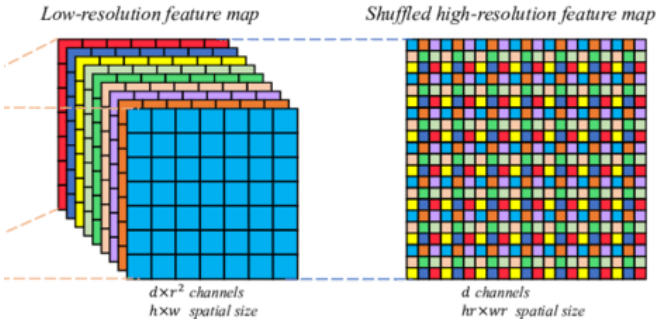

Figure 3.9: Pixel Shuffle decoding

Embedding this process of refinement on the decoding stage, feature fusion is enhanced, and more details of the object are preserved.

Once the feature maps have been upsampled, they are further refined through a series of convolutional layers followed by ReLU activation. This step ensures that the reconstructed depth information maintains clarity and precision. Additionally, to preserve fine-grained details from the original image, the architecture incorporates residual connections. These connections are not directly fused but first processed through EESP modules, which refine the features before merging them into the decoder.

Finally, the depth map is generated through a series of well-defined steps. The output from the fourth stage of the decoder is combined with the residual connections, ensuring that essential details are retained. This merged feature map is then passed through a 3×3 convolution, which reduces it to a single-channel representation suitable for depth estimation. As a final step, the depth map is upsampled using bilinear interpolation, restoring it to the original input resolution and ensuring a smooth, high-quality output.

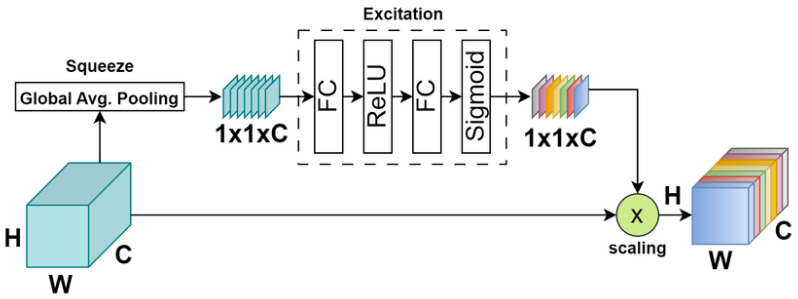

Figure 3.10: Squeeze-and-Excitation (SE) Block

3.3 Tools and techniques

I used a Python library designed to manipulate STL objects (3D meshes) that is NumpySTL. Thanks to this library, I wrote custom methods to translate, rotate, re-scale and combine meshes. Moreover, I used this library and the Scipy library for the mesh creation from depth images. In the first test case, the 2D to 3D conversion was done by mapping the 2D coordinates of the grey-scaled input image to 3D adding z-values to some faces with respect to the intensity of the pixels in the grey-scaled image; then, these new 3D coordinates were used to generate the vertices and the faces that composed the final mesh. I obviously used Blender as guideline during the full project.

The CNN was designed using the python library Pytorch and it was deployed on a virtual environment due to the requested resources.

3.3.1 Loss Functions

The training of the Depthy model was carried out with different combinations of popular loss functions used in fields like segmentation, depth estimation and 3D mesh reconstruction. Moreover, various weights for these loss functions were added, since they were tailored for different aspects of the depth map generation.

MSE

The Mean Squared Error (MSE) loss measures the difference between the predicted depth map $\hat{D}$ and the ground truth depth map D. This loss function penalizes large errors by averaging the squared differences over all pixels:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(D_i - \hat{D}_i)^2$$

where N is the total number of pixels in the depth map. By minimizing this loss, the model learns to generate depth estimates that closely align with the actual depth values.

Berhu loss

The BerHu loss is a robust alternative to the Mean Squared Error (MSE) loss, effectively balancing small and large errors. Compared to MSE, the BerHu loss offers significant advantages, particularly in handling large errors and maintaining stable gradients during training. One of its key strengths is its robustness to outliers.

$$L(D, \hat{D}) = \begin{cases} |D_i - \hat{D}_i|, & \text{if } |D_i - \hat{D}_i| \leq c \\ \frac{(D_i - \hat{D}_i)^2 + c^2}{2c}, & \text{if } |D_i - \hat{D}_i| > c \end{cases}$$

where c is a threshold, typically set as a fraction of the maximum absolute error in a batch.

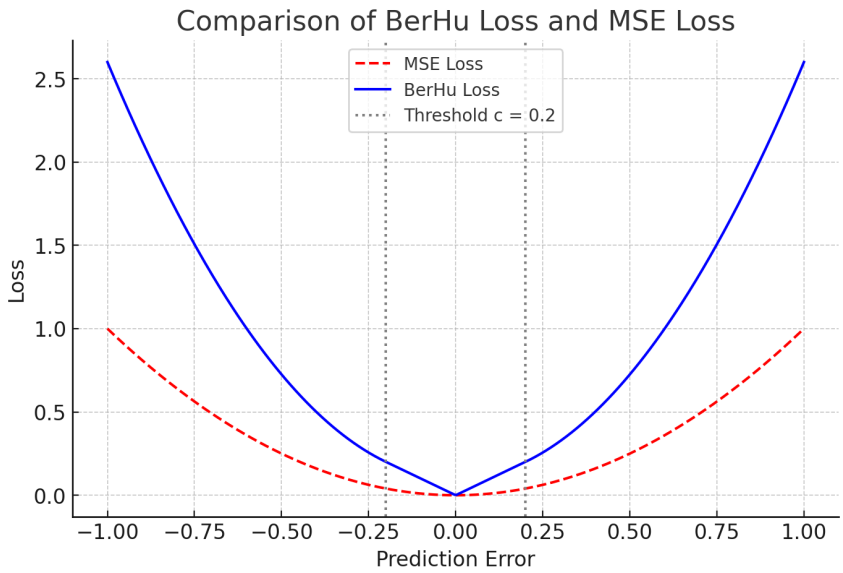

Figure 3.11: Comparison between Berhu and MSE functions

While MSE penalizes large errors quadratically, making the optimization process highly sensitive to extreme values, the BerHu loss adopts a more adaptive approach. For small errors, it applies a linear penalty, whereas for larger errors, it transitions to an L2-like formulation. This dynamic adjustment mitigates the excessive influence of outliers on the training process [in

figure 3.11 the comparison between Berhu and MSE is plotted]. Another important advantage of BerHu loss is its ability to maintain stable gradients. When dealing with small errors, it functions like the L1 loss, ensuring that the gradients remain meaningful and do not go down excessively. This helps prevent the vanishing gradient problem.

Gradient loss

The Gradient Loss is designed to preserve spatial details by enforcing consistency in the depth gradients between the predicted depth map $\hat{D}$ and the ground truth D. It is defined as:

$$L_{\text{grad}} = \sum_{i,j} \left(|\nabla_x D_{i,j} - \nabla_x \hat{D}_{i,j}| + |\nabla_y D_{i,j} - \nabla_y \hat{D}_{i,j}| \right)$$

where ∇_x and ∇_y represent the finite difference gradients along the horizontal and vertical directions, respectively:

$$\nabla_x D_{i,j} = D_{i+1,j} - D_{i,j}, \quad \nabla_y D_{i,j} = D_{i,j+1} - D_{i,j}$$

By minimizing this loss, the model learns to generate depth maps with smooth yet sharp transitions, improving the preservation of fine details and edges in the reconstructed scene.

SSIM

The Structural Similarity Index Measure (SSIM) is widely used in deep learning for image-based tasks, including mesh reconstruction, as it prioritizes perceptual quality over pixel wise differences. Unlike MSE, which treats all pixel errors equally, SSIM captures structural information by considering luminance, contrast, and texture similarities between the predicted depth map $\hat{D}$ and the ground truth D.

The SSIM index is computed in this way:

$$\text{SSIM}(D, \hat{D}) = \frac{(2\mu_D \mu_{\hat{D}} + C_1)(2\sigma_{D\hat{D}} + C_2)}{(\mu_D^2 + \mu_{\hat{D}}^2 + C_1)(\sigma_D^2 + \sigma_{\hat{D}}^2 + C_2)}$$

where μ_D and $\mu_{\hat{D}}$ are the mean values, σ_D^2 and $\sigma_{\hat{D}}^2$ are the variances, and $\sigma_{D\hat{D}}$ is the covariance. C_1 and C_2 are small constants for numerical stability.

The SSIM loss is then defined in this way:

$$L_{\text{SSIM}} = 1 - \text{SSIM}(D, \hat{D})$$

By maximizing structural similarity, SSIM loss is particularly effective in deep learning techniques such as NeRF and depth estimation, where preserving geometric details and perceptual quality is crucial.

3.3.2 Metrics in 3D mesh reconstruction

The metrics used for the evaluation of the outcomes from the 3D mesh reconstruction model are the Chamfer distance and the F-score.

The Chamfer distance

The Chamfer distance quantifies the similarity between two sets of points. This distance is often used as a loss function in point cloud generation tasks to minimize the distance between ground-truth and outcome point sets. In this case, the Chamfer distance was computed between the output mesh and the ground-truth mesh, corresponding to the real object in the input image. To compute the Chamfer distance between two meshes effectively, we should take only subsets of points for each mesh, since the real number of points is huge. So, N points were uniformly sampled for each mesh and the distance was calculated [in figure 3.12 the representation of a calculation of the distance is reported].

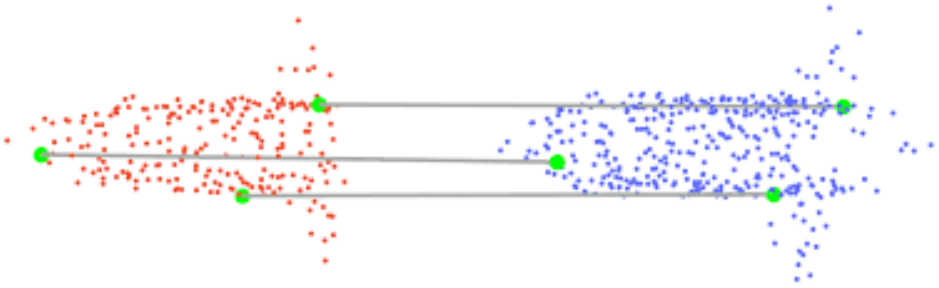

Figure 3.12: Calculation of the Chamfer distance

Considering the two sets of vertices P and Q, the calculation involves finding the nearest neighbors in Q for each point p in P (and vice versa) using the Euclidean Distance, then summing the two terms.

$$\text{Chamfer}(P,Q) = \sum_{p \in P} \min_{q \in Q} \|p - q\|^2 + \sum_{q \in Q} \min_{p \in P} \|q - p\|^2$$

Values between 0.01 and 0.1 are optimal, between 0.1 and 0.5 may indicate an acceptable level of similarity, while values greater than 1 may indicate a large discrepancy between the meshes.

The F-score

The F-score is a metric used to evaluate the similarity between two 3D surfaces by combining precision and recall, providing a more balanced view of how well the predicted mesh aligns with the ground truth. It is particularly useful when the comparison between point clouds needs to penalize both missing points and outliers [in figure 3.13 a visual representation of these metrics is reported].

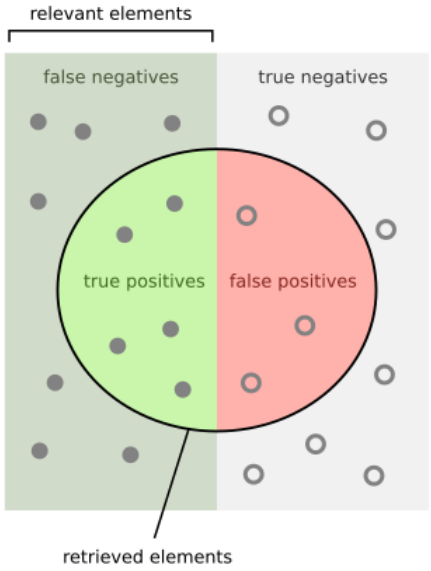

Figure 3.13: Calculation of the F-score value

To compute the F-score, two point clouds are obtained by uniformly sampling points from the predicted mesh and the ground truth mesh. A threshold distance τ is defined to determine whether a predicted point is considered close enough to a ground truth point (and vice versa). Precision measures the proportion of predicted points that lie within τ of any ground truth point, while recall measures the proportion of ground truth points that are matched by any predicted point within the same distance. The F-score is then calculated as the harmonic mean of precision and recall:

$$\text{F-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Typically, the threshold τ is set to a fixed value such as 0.01 (depending on the scale of the object). Higher F-score values indicate better overlap between the reconstructed and ground truth shapes. While the Chamfer distance penalizes positional deviation, the F-score emphasizes point-wise correspondence and completeness, making it a complementary metric for evaluating reconstruction quality.

Chapter 4

Experiments and results

The experiments involved predicting an acceptable depth map from different views of an object using the CNN cited and reconstructing a 3D mesh, implementing a foundation model. Although the initial goal of this work was to operate on technical images featuring various types of eyewear, the experiments for both tasks were extended to a broader set of design files and images from different domains. This shift was motivated by the need to test the proposed methodology on more complex objects, which pose greater challenges in terms of volumetric reconstruction, given, for example, the significant differences in structure and geometric complexity between a car and a pair of sunglasses. Moreover, other tests were conducted on general technical images from different web sources. In the first task, the experiments

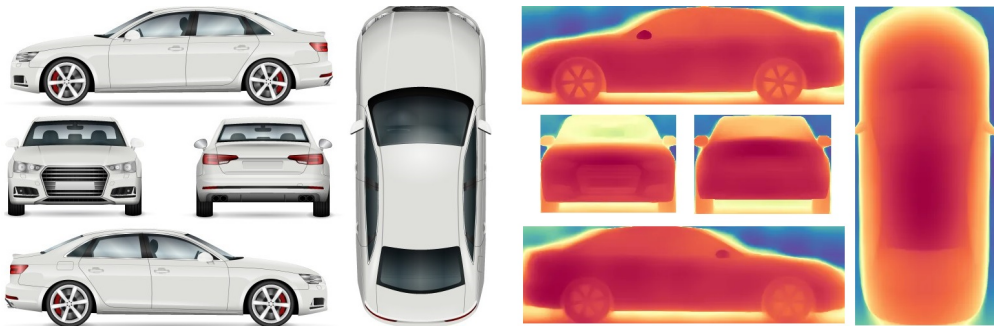

Figure 4.1: output depth masks from the Depth Anything model for multiple colored views of a car

were conducted on some drawings of objects' views in technical DWG files. The initial experiments required the prediction of depth masks from the colored views by using the Depth Anything model, that is a state-of-art model to make depth estimation. I passed different images of cars' drawings to the

Depth Anything model [in figure 4.1 the drawings along with masks are reported], then I converted the resulting masks to meshes [in figure 4.2 the the resulting meshes are reported]. This model allowed to obtain very detailed masks that could be used as input for the 3D mesh generation algorithm in a sort of best case scenario, but the limitation remained to be the manage of not colored figures and the process involving doing depth estimation on images with multiple objects; in fact the model required an additional automatic segmentation step to capture the single figures, that in the case of technical drawings is a challenging task. The resulting depth images of the single views where then converted to 3D objects as explained in the methodology section.

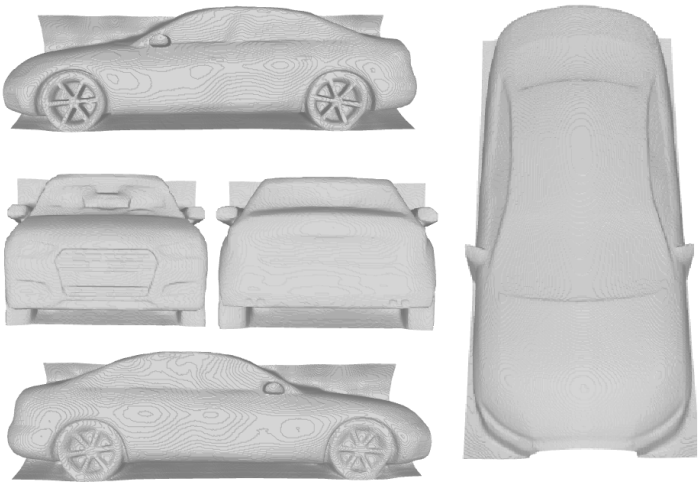

Figure 4.2: the result of the 2D to 3D conversion for the outcome depth masks from the Depth Anything model

After completing these initial experiments, I moved on to the development of a custom CNN architecture, with the goal of replicating and potentially improving the performance observed before. The motivation for designing a custom model arose from several limitations observed in the previous tests. In particular, the Depth Anything model demonstrated excellent results on well-colored images, but its performance significantly degraded when applied to uncolored technical drawings or images containing multiple figures due to the complete absence of information. The need for additional steps, such as segmentation and depth generation, highlighted the necessity of a specialized, flexible, resource-constrained and fast solution.

The results of the exploration of the literature in the field of depth estimation and of the development of possible CNN solutions led to the design of

Depthy, a neural network architecture tailored for technical drawings. The main focus in designing the network was on the decoding stage, since the Resnet-50 achieved good results on encoding features for performing both image segmentation and depth estimation.

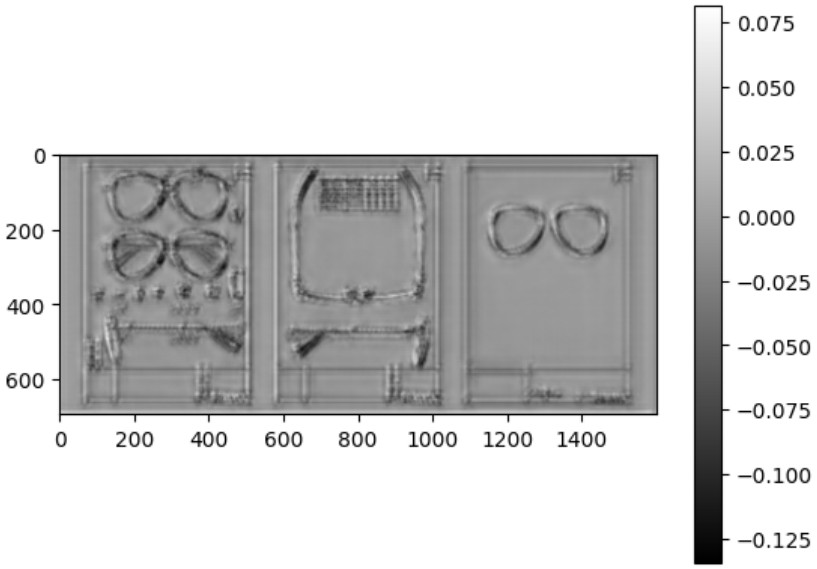

Figure 4.3: Greyscaled depth map predicted by the Depthy model for the DWG presented above ready to be converted to meshes

The initial experiments involved training the architecture on the NYU v2 dataset using the MSE loss. The goal was to perform a relatively simple and general training procedure, without over-optimizing the network for that specific dataset, since the model was later intended to be applied to a different domain, that is the domain of technical images, which differ significantly in terms of visual structure and distribution. Before training, the dataset undergoes a series of transformations to improve model generalization and performance.

These preprocessing steps are applied to each image using a composition of transformations. The images are first rescaled to standardize their input size. Then, some specific data augmentation techniques are applied. First, random horizontal flipping and random rotations, helping the model learn features that are invariant to minor spatial changes. After the transformations, the images are converted into tensors. Finally, the images are normalized using the mean and the standard deviation values computed from

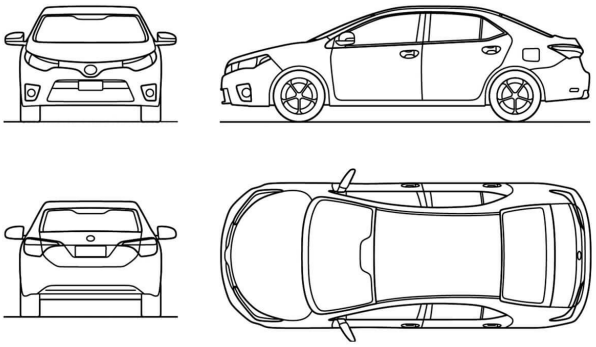

Figure 4.4: a test image reporting different drawings that are views of a car with lack of details and color

the ImageNet dataset. This normalization ensures that the input values are on a similar scale.

The network was trained with a batch size equal to 10 to balance memory usage and training stability. After few epochs, the network seemed to have not acceptable performance in segmenting the figures and the quality of the depth values was not acceptable, because some essential details were lost, resulting in simple clouds of depth values. This was normal, since structural information was not preserved in the decoding stage. So, the network needed to be upgraded with additional layers and trained by minimizing more complex loss functions. So, I integrated enhanced modules and trained the network with a more complex loss function that was a combination of Berhu, gradient and SSIM losses [in figure 4.4 an input image in the cars' domain is presented, while the outcome for this image is reported in figure 4.5].

These losses were properly weighted during the different training epochs. In this way, I was able to enforce the attention on the edges to make the model able to construct a more detailed depth mask. To improve convergence and generalization during training, I employed a learning rate scheduling strategy (ReduceLROnPlateau).

The scheduler used is a conservative yet effective mechanism, as it adjusts the learning rate only when the monitored metric stalls, rather than at fixed intervals. In our case, the validation loss is monitored. If no improvement is observed over two consecutive epochs (patience=2), the learning rate is halved (factor=0.5). This approach allows the optimizer to explore the loss landscape with a sufficiently high learning rate in the early phases, and only slows down when the model approaches a plateau, encouraging fine-tuning

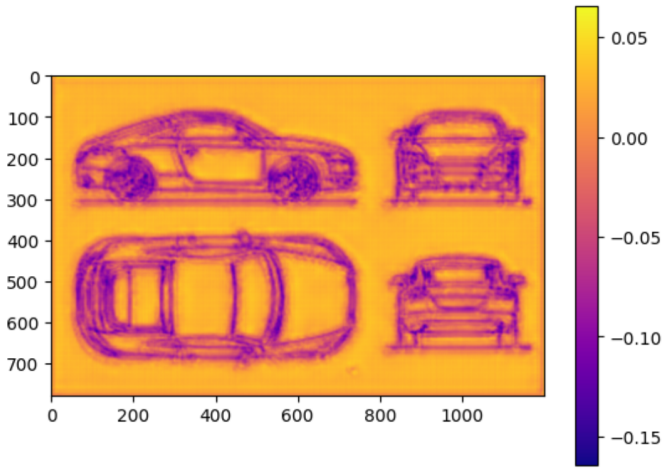

Figure 4.5: output depth mask from Depthy model with pretrained encoder

rather than premature convergence.

Such a strategy reduces the risk of overfitting early on, since it avoids lowering the learning rate too soon and helps the model escape shallow local minima. Moreover, the reduction factor of 0.5 strikes a good balance, and it is used to adjust the learning dynamics without being overly aggressive, which could otherwise hinder further improvements. To complement this adaptive scheduling, an early stopping mechanism is used with a maximum patience of five epochs (max patience=5). This setup allows natural fluctuations in the validation loss without halting the training prematurely, ensuring that short-term noise does not interfere with long-term learning trends. Additionally, the optimizer used in this setup is Adam, which incorporates weight decay for effective regularization.

The network was trained on both the pretrained and non-pretrained configurations for the encoder, trying different types of losses, by also weighting them [in figure 4.3 an outcome of the network is presented]. The pretrained configuration offered a good quality in the results for simple car drawings. The main lack observed was in the consistency of the depth across empty areas of the object (e.g., the difference in depth between the roof and the bonnet of a car in the upside-down prediction). The pretrained network was additionally trained with different values of SSIM and Berhu losses. For any loss configuration, after a few epochs, the network was overfitted. This behavior was also observed when training the network from scratch (without pretrained encoder). The most acceptable outcomes were obtained in the first 3 epochs for both scenarios [in figure 4.6 an outcome of the network af-

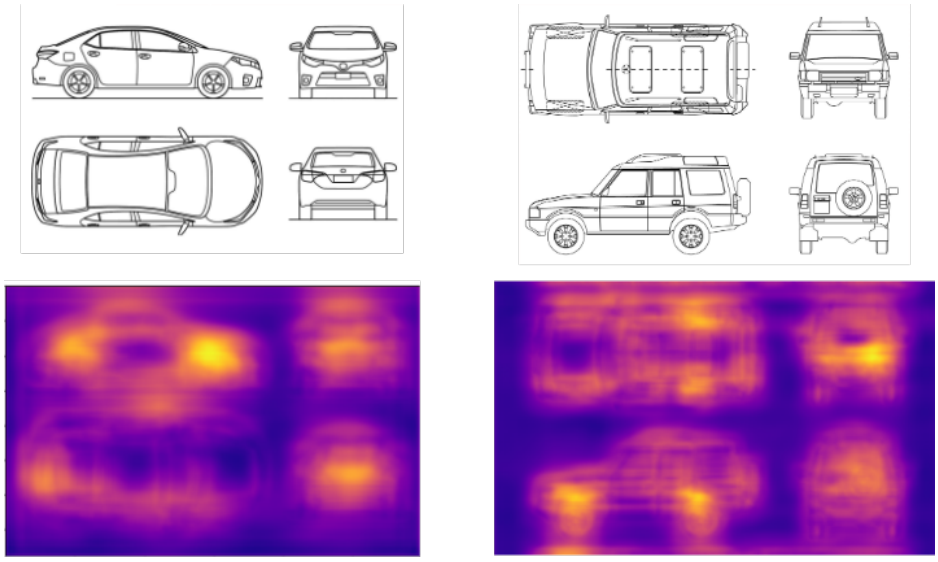

Figure 4.6: output depth mask from Depthy model after training on NyU 2 dataset with 2 epochs using SSIM=1, Berhu=0.6 and gradient=0.2

ter training it for 2 epochs is presented, while in figure 4.7 a training history is reported for a specific configuration of loss functions].

In this step, the problem of predicting the depth values for these drawings by learning from a depth estimation dataset arose. In this scenario, the training was supposed not to be strong to obtain useful results. After a few epochs, a vanishing of structural details was observed by testing the network step by step, which was reasonable. This fact justifies the need for more controllable training with the scheduling techniques explained above. Moreover, to improve the preservation of fine geometric structures and enhance the sharpness of object boundaries in the predicted depth maps, an additional loss term was introduced. Specifically, a gradient loss was incorporated to penalize discrepancies in the spatial gradients between the predicted and ground-truth depth maps. This encourages the network to better capture depth discontinuities and retain structural details, which are often smoothed out or lost when relying solely on photometric losses such as BerHu and SSIM.

In the case of the end-to-end 3D mesh reconstruction task, some available 3D assets with their single 2D representation have been collected from the company catalog (these models were used as ground-truth images for testing).

A pretrained TripoSR was used to predict the volume of objects from single input images (with the configuration shown in the image). I tuned the

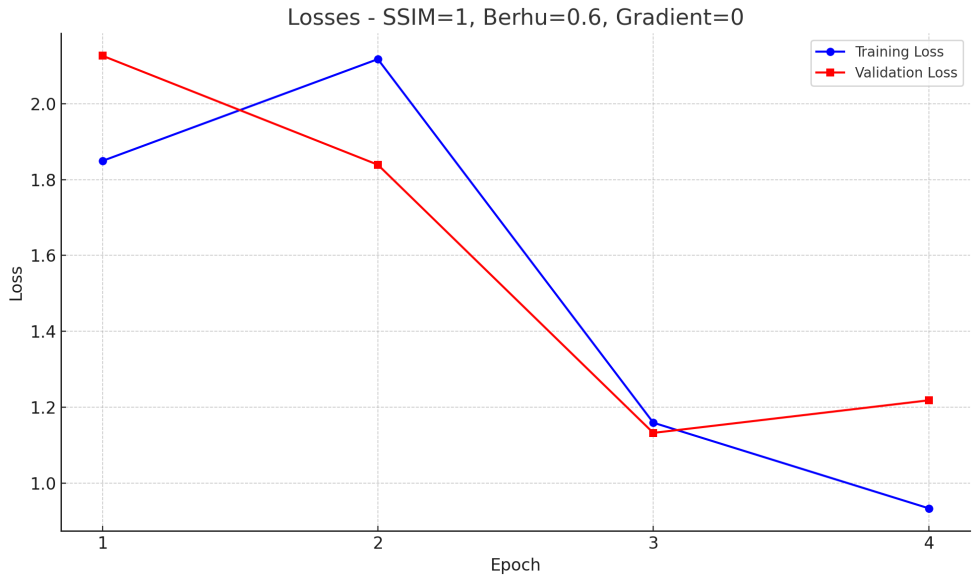

Figure 4.7: training history with SSIM=1 and Berhu=0.6

model with a foreground ratio equal to 0.75 and a marching cubes resolution equal to 320, then I passed 2D images of assets from the company catalog. The model could efficiently reconstruct the "hidden" part of the representation, returning a 3D object with an acceptable texture. The fidelity of the reconstruction was quantified by calculating the Chamfer distance and the F-score between the output and the ground-truth object [in figure 4.8 a comparison between the outcome and the ground-truth is plotted]. These metrics are approximate, as they were computed on a subset of points sampled from the meshes. In this case, 10000 points were randomly sampled in both meshes, then the chamfer distance and the F-score value were computed for these two subsets.

4.1 Interpretation of results

4.1.1 Principal results

The primary outcome of the experiments is related to the effectiveness of the proposed architecture in predicting depth values from single images, even under particularly challenging scenarios such as technical line drawings and multi-object figures. Unlike pretrained models such as Depth Anything,

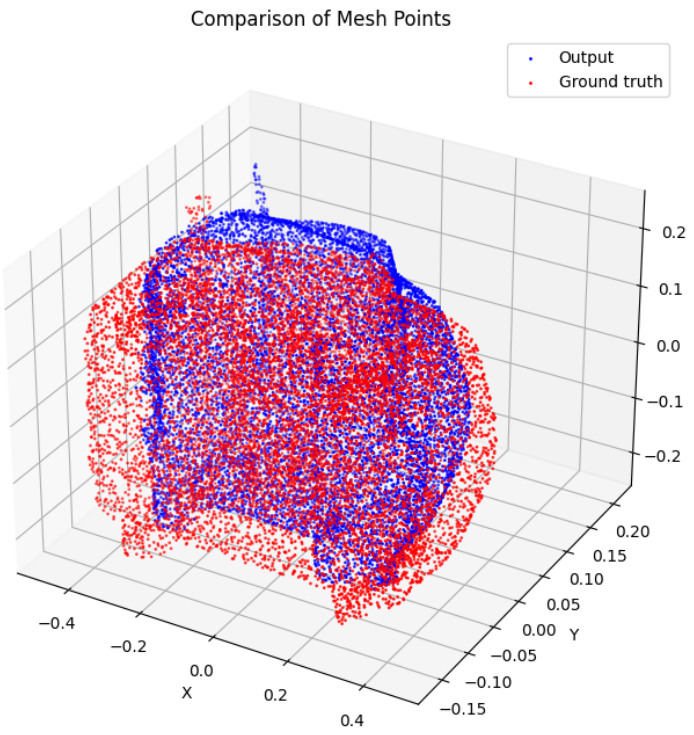

Figure 4.8: Comparison of sampled points between ground truth mesh and the output from tripoSR

which demonstrated notable limitations when applied to uncolored or non-photorealistic images, the proposed CNN showed an increased robustness in these non-traditional domains [examples of outcomes in different domains are reported in figures 4.9 and 4.10].

One of the most significant findings is that the CNN does not require extensive training to generalize well across various input types. Interestingly, the model does not exhibit typical signs of overfitting, even when trained for a relatively low number of epochs. This suggests that the network quickly learns the key features necessary for depth prediction, although the output initially lacks finer structural details, such as sharp object boundaries or complex surface contours. These missing details are not due to overfitting, but rather to the inherent difficulty of the task. In technical drawings and sparse sketches, crucial spatial cues are often absent or minimal. This observation justifies the integration of edge-aware loss components, such as gradient and SSIM terms, which significantly improved the reconstruction quality. Moreover, there were some benefits in training the pretrained networks for few

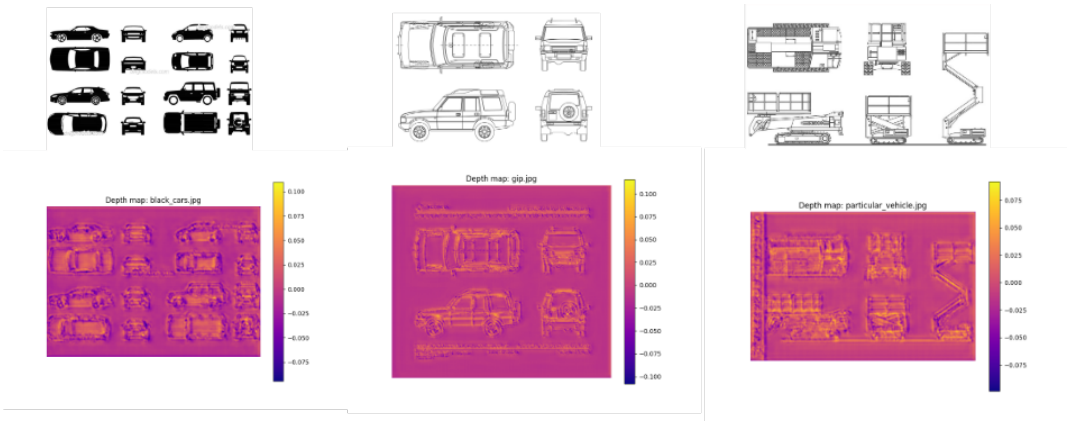

Figure 4.9: results on images with vehicles

epochs to refine the outcomes, both for structural preservation and coherent depth reconstruction (as much as possible).

Another key result is the validation of the architectural choices made in the decoder section of the network. While standard ResNet-based encoders are sufficient to capture semantic features, the decoder plays a pivotal role in reassembling a spatially coherent depth map. Enhanced decoding strategies and a properly tuned loss function proved essential in preserving structural elements and avoiding depth value dispersion, which would otherwise result in noisy or overly smoothed depth maps. Taken together, these results demonstrate the feasibility of CNN-based methods in handling difficult input modalities, such as technical or synthetic drawings, for depth estimation.

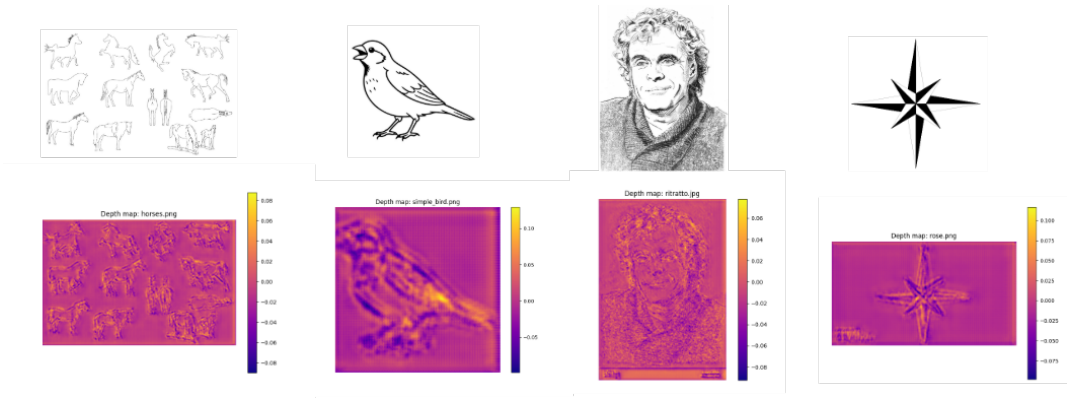

Figure 4.10: results on general images

4.1.2 Secondary results

The secondary findings are centered on the evaluation of transformer-based architectures for the task of 3D mesh reconstruction from single-view 2D images. In particular, the use of TripoSR showed state-of-the-art performance in reconstructing full volumetric representations from a single image input. The model proved effective across a wide range of categories, including vehicles, architectural elements, and stylized objects, even in cases where significant occlusion or missing parts were present [in figure 4.11 an example of an input image along with outcome and ground-truth meshes is presented]. Quantitative evaluation using Chamfer Distance and F-score confirmed the high fidelity of the reconstructions compared to the available ground truth meshes. The chosen configuration parameters, such as a foreground ratio of 0.75 and marching cubes resolution of 320 were effective in producing detailed and consistent 3D surfaces, indicating a good balance between computational efficiency and geometric precision. These results underscore the ability of transformer-based models to implicitly learn complex spatial priors and geometrical relationships from training data, enabling plausible inference of invisible object parts in the 3D domain. Notably, the results support the hypothesis that transformer-based architectures, unlike classical CNNs, are more adept at learning global context and symmetry patterns—features that are crucial for high-quality volumetric reconstruction.

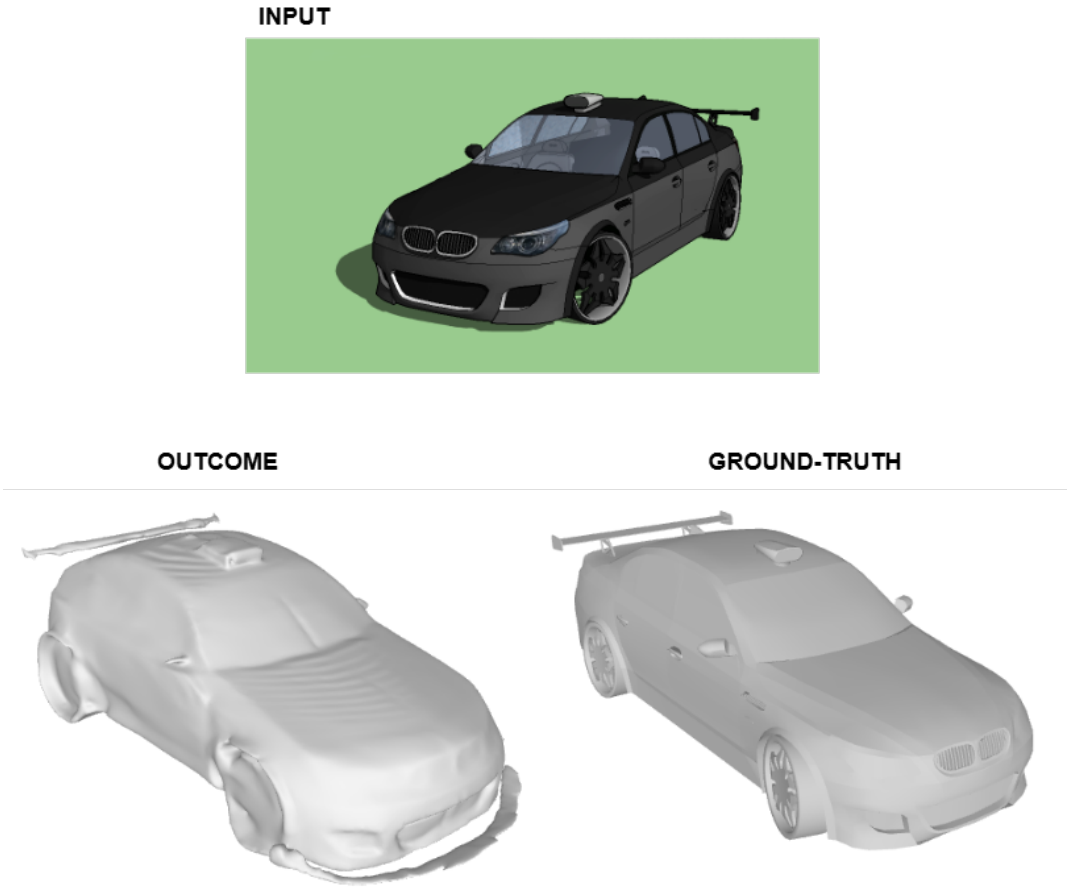

Figure 4.11: Example of an input image representing a car drawing along with the outcome from TripoSR and the corresponding ground-truth mesh

Chapter 5

Discussion

The aim of this work is to deploy 3D rendering techniques in different scenarios; among them there are the cases of 3D reconstruction from 2D images and design files. With minimum resources everyone can be able to generate acceptable 3D objects from 2D images.

5.1 Comparison with the literature

Different methods exist that allow the generation of 3D meshes by input features. Among these, two are mentioned: the voxel-based and the point-cloud-based. These models often have an architecture that integrates convolutional layers.

Voxel-based methods represent 3D objects as a grid of small cubes (or "voxels") in space, similar to 3D pixels. Each voxel contains information about whether it is part of the object (occupied) or empty space. The 3D object is reconstructed by filling in or removing voxels within this grid, often using volumetric data derived from images. This approach can approximate both shape and surface details.

In point-cloud-based reconstruction, points are positioned in 3D space to capture the shape and contours of an object without needing a grid structure. These points are later processed to form a surface or mesh. However, both voxel-based and point-cloud-based methods present notable limitations. Voxel-based approaches, despite their ability to model volume, suffer from a high memory and computational cost, especially as the resolution of the voxel grid increases. This trade-off between resolution and efficiency often leads to coarse reconstructions or heavy computational demands. Moreover, the discretization of space into voxels can result in a loss of fine surface details. Point-cloud-based methods, on the other hand, are more memory-efficient

and flexible, but they struggle with the lack of inherent structure. Since point clouds do not define explicit surface connectivity, additional processing is required to infer topology and build consistent meshes. This makes it challenging to recover complex geometries or fine-grained surface features, especially when the point distribution is sparse or noisy. Furthermore, learning meaningful features from unordered point sets can be difficult, which affects the accuracy and robustness of the reconstruction.

Recent advancements in 3D mesh reconstruction have aimed to address the limitations of voxel-based and point-cloud-based methods, which struggle with computational inefficiency, resolution constraints, or sparsity issues. Notably, implicit surface representation techniques have gained traction for their ability to generate high-fidelity, continuous surfaces. Among these, transformer-based models like TripoSR have achieved state-of-the-art results by leveraging global contextual information with local geometric details. TripoSR leverages implicit surface representations combined with transformer-based architectures to directly predict high-fidelity 3D surfaces from 2D inputs. This allows for the generation of detailed, continuous surfaces without the resolution constraints of voxels or the sparsity issues of point clouds. This model excels particularly in capturing fine geometric details and complex topologies, producing watertight and visually coherent meshes that require minimal post-processing. Its ability to fuse global contextual information from the input image(s) with local surface geometry makes it more robust to variations in pose, lighting, and occlusions. Furthermore, thanks to its transformer backbone and signed distance function representation, it generalizes better across diverse object categories and real-world scenarios. These advantages make TripoSR and similar models state-of-the-art solutions for accurate and efficient 3D mesh reconstruction.

Building upon these advancements, MeshFormer introduces a novel paradigm by integrating explicit 3D-native priors into the architecture. Unlike traditional triplane-based or voxel-centric methods, MeshFormer represents features using 3D sparse voxels and combines transformers with 3D convolutions to better encode projective correspondences between 2D and 3D structures. This design allows for efficient training and high-quality textured meshes with fine-grained geometric details. A key innovation of MeshFormer is its use of normal maps as additional input guidance, predicted via 2D diffusion models or acquired through photometric techniques. These maps provide critical clues for surface refinement, enabling the generation of detailed geometry. Furthermore, MeshFormer adopts a unified, single-stage training process by combining Signed Distance Function (SDF) supervision with differentiable surface rendering, bypassing the need for complex multi-stage pipelines. This approach not only enhances training efficiency but also produces meshes with

superior texture and geometric fidelity compared to existing methods.

By incorporating explicit 3D biases and normal guidance, MeshFormer demonstrates significant improvement in mesh quality and reconstruction speed, positioning itself as a promising alternative for open-world sparse-view tasks. Its integration with 2D diffusion models further paves the way for applications like single-image-to-3D and text-to-3D generation, highlighting its versatility and practical relevance.

5.2 Limitations

First, the main limitation of this work concerns the infeasibility of directly generating a complete 3D mesh from the technical drawings with high fidelity, because there are few sides (basically 4-5). The chosen strategy was to bring these views as 3D meshes, maintaining the lines and details of the original figure as much as possible. The main limitation lies in the training of the proposed architecture, as the chosen dataset (NYU-v2) consists of RGB-D images of indoor environments, which significantly differ from the technical black-and-white line drawings used in this work. This domain gap may limit the generalization capabilities of the model when applied to schematic representations lacking texture, color, and perspective cues. Although transfer learning from real-world RGB images can provide a strong initialization, the absence of domain-specific features in the training data might affect the precision and consistency of the predicted depth maps when dealing with simplified and abstract figures. Future improvements could involve the use of synthetic datasets specifically designed for technical drawing interpretation or the creation of a custom dataset that better reflects the target input domain.

Relating to the secondary objective, the main limitation in using a model like TripoSR can be the quality of the representation and the type of object (although it adapts to many objects), as well as the fact that the training of these models is not accessible to everyone. The requested details level is high in this field, but in real-world applications, it could not be a problem. In both cases the main limitation is the need for post-production techniques of a 3D artist.

Chapter 6

Conclusions

Recent advancements in computer vision have significantly enhanced the capabilities of both monocular depth estimation and 3D mesh reconstruction. These developments have broadened the applicability of 3D modeling across various domains, including industrial design, gaming, robotics, and augmented reality. This thesis has independently explored these two tasks, examining state-of-the-art methodologies and proposing tailored implementations to address specific challenges inherent to each domain.

6.1 Future Perspectives

The fields of monocular depth estimation and 3D mesh reconstruction are rapidly evolving, with several notable advancements shaping their trajectories.

In monocular depth estimation, the introduction of Depth Anything v2 represents a significant evolution. This model leverages synthetic data for training and employs large-scale pseudo-labeled real images to enhance its predictive capabilities. Compared to its predecessor, Depth Anything v2 offers finer and more robust depth predictions, demonstrating improved generalization across diverse scenes and conditions. Future research may focus on integrating other transformer-based modules and incorporating uncertainty estimation techniques to produce confidence-aware depth maps, particularly valuable in safety-critical applications.

Regarding 3D mesh reconstruction, recent innovations have addressed the challenges of extracting accurate geometric representations from neural implicit models. NeRFMeshing introduces a compact and flexible architecture like TripoSR that distills volumetric 3D representations into geometrically accurate meshes, enabling real-time rendering. Additionally, methods like Deli-

cate Textured Mesh Recovery employ adaptive surface refinement techniques to generate high-quality textured meshes from images. Another notable development is SuGaR (Surface-Aligned Gaussian Splatting), which optimizes millions of tiny particles to align with surfaces, facilitating efficient 3D mesh reconstruction and high-quality rendering. Another architecture designed to extend the capability of TripoSR is the SF3D (Stable Fast 3D Mesh Reconstruction with UV-unwrapping and Illumination Disentanglement), that is a powerful and efficient model for fast feedforward 3D mesh reconstruction from a single image.

Looking ahead, the integration of these advanced techniques promises to further enhance the fidelity and efficiency of 3D modeling processes. Continued research into hybrid models and real-time rendering capabilities will be pivotal in expanding the accessibility and applicability of 3D modeling technologies across various workflows in digital production.

Bibliography

[1] *Attention is all you need*. https://arxiv.org/abs/1706.03762. [2 Aug 2023].

[2] *Deep Learning-based Depth Estimation Methods from Monocular Image and Videos: A Comprehensive Survey*. https://arxiv.org/abs/2406.19675. [28 Jun 2024].

[3] *Depth Anything V2*. https://arxiv.org/abs/2406.09414. [13 Jun 2024].

[4] *Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data*. https://arxiv.org/abs/2401.10891. [19 Jan 2024].

[5] *LRM: Large Reconstruction Model for Single Image to 3D*. https://arxiv.org/abs/2311.04400. [9 Mar 2024].

[6] *MeshXL: Neural Coordinate Field for Generative 3D Foundation Models*. https://arxiv.org/abs/2405.20853. [31 May 2024].

[7] *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. https://arxiv.org/abs/2003.08934. [19 Mar 2020].

[8] *Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network*. https://arxiv.org/abs/1609.05158v2. [16 Sep 2016].

[9] *Stable Fast 3D Mesh Reconstruction with UV-unwrapping and Illumination Disentanglement*. https://arxiv.org/abs/2408.00653. [1 Aug 2024].

[10] *TripoSR: Fast 3D Object Reconstruction from a Single Image*. https://arxiv.org/abs/2403.02151. [4 Mar 2024].

[11] *Visualization of Convolutional Neural Networks for Monocular Depth Estimation*. https://arxiv.org/abs/1904.03380. [6 Apr 2019].

[12] *Voxel Structure-based Mesh Reconstruction from a 3D Point Cloud*. https://arxiv.org/abs/2104.10622. [21 Apr 2021].