

Anwesh Sarkar

DEVELOPMENT OF FRAMEWORK FOR 2D TO 3D RECONSTRUCTION AND VOLUMETRIC SEGMENTATION

 Research Papers

Document Details

Submission ID

trn:oid:::27535:140541110

Submission Date

May 26, 2026, 7:47 PM GMT+5:30

Download Date

Sep 18, 2026, 10:57 AM GMT+5:30

File Name

turnitinfinal-5-44.pdf

File Size

1.4 MB

40 Pages

11,477 Words

70,260 Characters

3% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Exclusions

- 2 Excluded Sources
- 1 Excluded Match

Match Groups

-  **34 Not Cited or Quoted 3%**
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**
Matches that are still very similar to source material
-  **0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 2%  Internet sources
- 1%  Publications
- 2%  Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- 34 Not Cited or Quoted 3%**
Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations 0%**
Matches that are still very similar to source material
- 0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 2% Internet sources
- 1% Publications
- 2% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	www.mdpi.com	<1%
2	Internet	repository.iutoic-dhaka.edu	<1%
3	Student papers	Ahsanullah University of Science and Technology on 2026-04-04	<1%
4	Student papers	La Trobe University on 2025-10-29	<1%
5	Internet	arxiv.org	<1%
6	Internet	studentsrepo.um.edu.my	<1%
7	Internet	github.com	<1%
8	Internet	web.wpi.edu	<1%
9	Student papers	University of Westminster on 2024-11-18	<1%
10	Internet	ir.juit.ac.in:8080	<1%

11	Student papers	Brunel University on 2026-04-15	<1%
12	Internet	doku.pub	<1%
13	Student papers	International University - VNUHCM on 2026-01-12	<1%
14	Internet	repositorio.iscte-iul.pt	<1%
15	Internet	idoc.tips	<1%
16	Internet	cobweb.cs.uga.edu	<1%
17	Internet	www.internationaljournalssrg.org	<1%
18	Student papers	Dublin Business School on 2025-08-28	<1%
19	Internet	dl.ucsc.cmb.ac.lk	<1%
20	Internet	idoc.pub	<1%
21	Internet	theses.hal.science	<1%
22	Publication	A. Van Griensven. "Application and evaluation of ESWAT on the Dender basin and ...	<1%
23	Student papers	University of Missouri, Kansas City on 2019-10-28	<1%
24	Student papers	University of Queensland on 2020-06-19	<1%

25

Internet

assets-eu.researchsquare.com

<1%

26

Student papers

University College Technology Sarawak on 2026-01-04

<1%

DEVELOPMENT OF FRAMEWORK FOR 2D TO 3D RECONSTRUCTION AND VOLUMETRIC SEGMENTATION

Anwesh Sarkar

ABSTRACT

There The ever-growing requirements for swift content generation in gaming, e-commerce, and VR applications prompt the necessity of fast reconstruction from single view. In this thesis, we present an enhanced pipeline for generating textured 3D models from one image through the integration of deep image segmentation with feed-forward volumetric inference. For background removal, we propose a U²-Net-based segmentation module that outputs a normalized foreground mask which then goes through TripoSR - a vision transformer encoder paired with a lightweight triplane architecture that predicts an occupancy map. We obtain a watertight mesh using marching cubes. The quality of our reconstruction, measured on a synthetic spherical object, is confirmed through the following metrics: CD of 1.857×10^{-3} , volumetric IoU of 0.7786, and HD of 0.0849, providing excellent shape recovery performance. The presented method significantly outperforms TripoSR without image segmentation ($CD = 10.2 \times 10^{-3}$) and the OccNet [12] model ($CD = 12.4 \times 10^{-3}$), achieving a geometric error reduction by almost five times. The whole procedure takes less than two seconds when run on a consumer-grade GPU. Our ablation studies show that the introduction of a separate 2D segmentation stage is crucial for obtaining high-fidelity results.

15

CONTENTS

6

14

CANDIDATE'S DECLARATION	II
ACKNOWLEDGEMENT	III
CERTIFICATE BY THE SUPERVISOR	IV
ABSTRACT	V
CONTENTS.....	VI
LIST OF FIGURES.....	IX
LIST OF TABLES	X
LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE	XI
CHAPTER 1 INTRODUCTION	1
1.1 MOTIVATION AND CONTEXT.....	1
1.2 CHALLENGES IN 2D TO 3D RECONSTRUCTION	1
1.3 LIMITATIONS OF THE PROPOSED APPROACH.....	2
1.4 INTEGRATING IMAGE WITH VOLUMETRIC 3D INFERENCE	2
1.5 PROPOSED APPROACH AND CONTRIBUTIONS	2
CHAPTER 2 LITERATURE REVIEW	4
2.1 BACKGROUND AND SCOPE.....	4
2.2 DATASETS FOR OBJECT ANALYSIS	4
2.2.2 Static 2D Datasets: Objects and their Multiview	5
2.2.3 Artificial Datasets: Sketchgraphs.....	6
2.3 FROM SEGMENTATION TO VOLUMETRIC INFERENCE.....	6
2.3.1 Feed-Forward Triplane Reconstruction	6
2.3.2 Feature Refinement for Volumetric Reconstruction	7
2.4 IMAGE ENCODER FOR VOLUMETRIC SEGMENTATION	7
2.4.1 Limitations of Convolutional Encoders.....	7
2.4.2 Transformers based Encoders: DINO-ViT Backbone.....	8
2.5 MULTIMODAL FUSION FOR 3D RECONSTRUCTION	8
2.5.1 Nature of Appearance and Segmentation Cues.....	8
2.5.2 Fusion Strategies: Early, Intermediate, and Late	8
2.5.3 Implicit Early Fusion via Foreground Compositing	9

2.6 RESEARCH GAP AND CONTRIBUTIONS	9
--	---

CHAPTER 3 PROPOSED METHODOLOGY..... 11

3.1 SYSTEM OVERVIEW	11
3.2 HIGH-LEVEL ARCHITECTURE.....	11
3.3 PIPELINE FLOW SUMMARY.....	11
3.4 DESIGN PHILOSOPHY	12
3.4.1 Image Segmentation and Foreground Normalisation	13
3.4.2 Background Removal with U ² Net	13
3.4.3 Foreground Compositing and Canonical Resizing	14
3.5 VOLUMETRIC 3D RECONSTRUCTION WITH TRIPOSR	15
3.5.1 Vision Transformer Encoder.....	15
3.5.2 Triplane Representation.....	15
3.5.3 Occupancy Decoder and Isosurface Extraction	16
3.5.4 Optional Mesh Simplification	16
3.6 TEXTURE MAPPING AND UV GENERATION	16
3.6.1 UV Parameterisation with xatlas	17
3.6.2 Texture Baking via GPU Rendering	17
3.6.3 Final Asset Export	17
3.7 IMPLEMENTATION ENVIRONMENT AND TOOLCHAIN.....	17
3.8 EVALUATION PROTOCOL	18
3.8.1 Test Datasets	18
3.8.2 2D Segmentation Metrics	18
3.8.3 3D Reconstruction Metrics	19
3.8.4 Efficiency Metrics	19
3.9 ABLATION AND BASELINE CONFIGURATIONS	19

CHAPTER 4 EXPERIMENTAL SETUP AND RESULTS..... 21

4.1 EXPERIMENTAL SETUP	21
4.1.1 Hardware and Software Environment	21
4.1.2 Dataset Preparation.....	21
4.1.3 Evaluation Metrics.....	22
4.1.4 Baseline Configurations	23
4.2 QUANTITATIVE EVALUATION.....	23
4.2.1 2D Segmentation Accuracy	23
4.2.2 3D Reconstruction Accuracy	24

4.2.3 Impact of Foreground Normalisation	25
4.2.4 Computational Efficiency	26
4.3 ABLATION STUDIES.....	27
4.3.1 Effect of 2D Segmentation	27
4.3.2 Choice of Segmentation Method	27
4.3.3 Volumetric Resolution Trade-off	27
4.3.4 Sensitivity to Foreground-Ratio and Background Colour	28
4.4 QUALITATIVE ANALYSIS	28
4.4.1 Success Cases	28
4.4.2 Failure Cases and Limitations	28
4.4.3 Generalisation to Real-World Photographs	29
4.5 COMPARISON WITH EXISTING METHODS.....	29
4.5.1 Comparison with Feed-Forward Baselines.....	29
4.5.2 Speed vs. Accuracy Trade-off.....	31
4.6 DISCUSSION	31
CHAPTER 5 CONCLUSION	32
5.1 APPLICATIONS.....	32
5.2 FUTURE SCOPE	32
5.3 CONCLUDING REMARKS	33
CHAPTER 6 REFERENCES	34

LIST OF FIGURES

Figure 2.1: Objaverse dataset.....	5
Figure 2.2: Multiview of a chair	6
Figure 2.3: DiNo ViT Backbone	8
Figure 3.1: Detailed pipeline.....	12
Figure 3.2: Silhouette depth map mask.....	14
Figure 3.3: Proposed Workflow	15
Figure 4.1: Image to cropped image to .obj file.....	28
Figure 4.2: Failure in generating mesh	29
Figure 4.3: Comparing Mesh	30

LIST OF TABLES

Table 3.1: Software libraries and frameworks in the reconstruction pipeline.....	18
Table 4.1: 2D segmentation accuracy (mean $\pm$ std).	23
Table 4.2: Reconstruction accuracy – ShapeNet Chair (mean $\pm$ std).....	24
Table 4.3: Reconstruction accuracy – ShapeNet Car (mean $\pm$ std).....	24
Table 4.4: Reconstruction accuracy – ShapeNet Airplane (mean $\pm$ std).....	25
Table 4.5: Chamfer distance ($\times 10^{-3}$) vs. foreground ratio and BG Color (Chair).....	25
Table 4.6: Per-stage inference time (milliseconds, mean $\pm$ std).....	26
Table 4.7: Comparison with existing feed-forward methods (Synthetic Sphere with mean CD $\times 10^{-3}$ and time)	30

LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLUTURE

I	Input RGB image
M	Foreground segmentation mask
α	Alpha channel / foreground probability
T	Triplane feature representation
$p(x,y,z)$	3D query point in volumetric space
$O(p)$	Occupancy probability at point (p)
θ	Network parameters
V	Volumetric occupancy field
M	Reconstructed mesh
τ	Distance threshold for F1-score
CD	Chamfer Distance
HD	Hausdorff Distance
IoU	Intersection over Union
F1	Harmonic mean of precision and recall
P	Precision
R	Recall
N	Number of sampled surface points
G	Ground truth mesh
$\hat{G}$	Predicted mesh
UV	Texture coordinate mapping
$d(p,q)$	Euclidean distance between points
$\mathbb{R}^3$	3D Euclidean space

21

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION AND CONTEXT

Creating high quality 3D assets is still labour intensive which limits the scalability of 3D digital content in modern virtual worlds. Reconstructing a 3D model from a single photo has been a long-standing challenge in computer vision for many years. Classical multi view methods require calibrated cameras, while modern neural approaches like NeRF and 3D Gaussian Splatting require many images and extensive per scene optimisation, often taking hours. Feed forward models allow immediate inference, but their output is often coarse or implicit, so they are impractical. An attractive alternative is to fuse the accurate 2D image segmentation into an efficient volumetric representation. U²Net and other deep segmentation networks can robustly separate foreground objects. TripoSR's[4] compact triplane features enable fast decoding into explicit meshes. The present research seeks to study the pipeline involving an end-to-end integration of a segmentation-based front-end with a feed-forward volumetric reconstruction back-end and its effect on improving geometry accuracy and real-time capability, as well as quantifying the role of segmentation. We generate industry-standard .obj files through our approach which helps us formulate a validated framework for generating single view 3D assets.

1.2 CHALLENGES IN 2D TO 3D RECONSTRUCTION

Single view 3D reconstruction is ill posed because a single 2D image does not contain any depth information explicitly, and many different 3D structures could map to a similar 2D appearance [1]. Furthermore, occlusion presents a significant challenge because the back side of an object that is hidden behind some occluding surfaces is impossible to estimate except based on a priori knowledge, which fails when dealing with unusual shapes [2]. Another obstacle appears in the first step of single view 3D reconstruction since it is still quite difficult to obtain a reliable segmentation mask of a foreground object against various background scenes under varying lighting conditions [3]. Although volumetric reconstruction can produce accurate 3D mesh structures, this technique also requires special attention since one needs to fit a detailed structure into a limited number of features while representing objects as a triplane, and the subsequent marching cubes step often results in topological artifacts and over dense reconstructions [5]. In addition, combining the two steps for real time inference adds more challenges, such as GPU memory capacity and library compatibility issues [4,6]. The thesis focuses on addressing these challenges such as spanning segmentation robustness, volumetric resolution, and pipeline efficiency.

1.3 LIMITATIONS OF THE PROPOSED APPROACH

In spite of its efficiency and overall performance, however, the above-mentioned method possesses several crucial weaknesses. First of all, the accuracy of the 2D segmentation procedure is critical for it since, if the background or object boundary is not well-defined, U²Net can create overly eroded or bloated masks which adversely affect the quality of the final reconstruction [3]. Secondly, due to the one-sided nature of the input images, there is no way to recover any data related to occluded surfaces, such as the rear surface of the object, and, therefore, the system needs to make a reconstruction based solely on a set of learned prior objects [1] which often leads to a flat, unrealistic surface [2][3]. Triplane model is very concise yet restricted in terms of the geometric features due to its resolution limit [4]. Furthermore, there is an addition of topological noise in the form of floating objects and non-manifold edges by means of the marching cubes algorithm which also needs to be cleaned manually [5]. Next, texture baking also results in blurry or inconsistent colours being created on the faces that were not seen because texture baking involves only one image and not multi views [4]. Last but not least, the data set used for training (Objaverse objects) [8] may cause bias due to object categories and thus failure in reconstructing highly articulated or transparent objects.

1.4 INTEGRATING IMAGE WITH VOLUMETRIC 3D INFERENCE

Image segmentation in 2D space is a vital pre-processing task for our pipeline. As in prior works, we make use of the U²Net model [3] to obtain a precise mask in order to segment the foreground and background. The segmentation mask is employed to crop and resize the image such that it occupies the canonical resolution of 320×320 pixels in a white background centred around the object and without other scene information. Then, the normalized image is sent to the TripoSR model [4] where it is encoded into a concise triplane representation [6] - three orthogonal planes of features encoding a volumetric field. Any query in 3D space is used by a small MLP decoder network [4] to predict occupancy probability, while the marching cubes algorithm [5] is employed to extract the mesh. This close relationship means that the volumetric inference process is performed on a signal free of noise or distractions, leading to a marked increase in geometric accuracy compared to pipelines without segmentation pre-process [3,4]. Overall, the combined process is done under two seconds per image on a consumer-grade GPU [4].

1.5 PROPOSED APPROACH AND CONTRIBUTIONS

In this work, an integrated workflow of 2D foreground extraction followed by feed-forward volumetric inference in 3D is proposed. The first step involves feeding the input image to a background removal algorithm implemented through the U²Net framework [3], and then passing it to TripoSR [4], where it will be encoded in the form of a triplane [6] and reconstructed using marching cubes [5]. The key contributions of this work are:

- A fully fledged real time single image 3D reconstruction approach based on cutting edge segmentation and volumetric modelling techniques.
- Proof that pre-processing the images in 2D results in a more accurate geometric structure than using the images alone [3,4].
- An implementation capable of running from start to finish in less than two seconds on a regular GPU [4].
- Creation of standard OBJ files with UV mapping available for instant application.
- A publicly available Colab file allowing for reproduction and educational usage.

CHAPTER 2

LITERATURE REVIEW

2.1 BACKGROUND AND SCOPE

Traditional manual modelling is a tedious process [1]. The problem of producing three-dimensional shapes is a computational bottleneck for creating digital media. Recovering a 3D shape from one image is an intrinsically ill-defined problem because there could be several shapes capable of creating one image [2]. Deep learning techniques have come up with a promising solution to this issue through the use of data driven priors. Recent developments in saliency detection methods such as U²Net [3] and effective volumetric feature encodings like triplane features [6] led to fast, feed-forward reconstruction approaches like TripoSR [4] which produce explicit meshes using the technique of marching cubes [5]. Our thesis is about tightly coupling 2D foreground detection and 3D volumetric inference.

2.2 DATASETS FOR OBJECT ANALYSIS

TripoSR [4] uses selective models from Objaverse [8] where more than 800,000 highly diverse textured meshes representing 3D objects are used for training and these objects are captured from multiple angles to get pairs of image-3D supervision. Apart from this, several papers make use of ShapeNet [2], which is a popular dataset containing models of 55 different classes of objects with hand-aligned CAD models. The importance of such data sets is their ability to generate enough geometric and textural diversity necessary for learning the image-triplane transformation.



Figure 2.1: Objaverse dataset

2.2.2 Static 2D Datasets: Objects and their Multiview

Two-dimensional static datasets serve as the base on which the training process is performed on both the segmented and the volume estimator blocks. In particular, the U²Net algorithm for performing foreground extraction was initially trained on the DUTS [7] dataset containing 10,553 natural images with accurately mapped salient object masks that would facilitate the removal of backgrounds regardless of the diversity of the scenes. The input data for 3D reconstruction are not natural images but synthetically rendered 2D views of 3D objects. As a standard benchmark for years has been used ShapeNet [2] with more than 51,000 3D models rendered in multiple camera positions belonging to 55 classes of shapes. Recently an even larger database was introduced by the Objaverse [8] authors, Synthesizing the assets under standardized lighting conditions produces millions of 2D silhouettes that are leveraged by TripoSR [4] to establish the relationship between 2D input view and its volumetric representation as a triplane.



Figure 2.2: Multiview of a chair

2.2.3 Artificial Datasets: Sketchgraphs

This thesis also employs a processed subset of the SketchGraphs dataset [8,9] for a controlled evaluation of the segmentation to reconstruction chain. SketchGraphs is a large-scale collection of 15 million parametric 2D CAD sketches, represented as geometric constraint graphs with nodes corresponding to primitives (lines, arcs) and edges encoding relations specified by the designer such as coincidence or perpendicularity [8]. The native data are symbolic graphs. The companion Img2CAD pipeline provides tools to render these sketches as clean, foreground only raster images, directly usable as input to image-based reconstruction systems [10]. In this thesis we use such rendered views to stress test the 2D segmentation module (the sketches already are characterized by sharp unambiguous object boundaries) and to evaluate the quality of the volumetric reconstruction on geometrically simple but structurally diverse shapes, thus supplementing the natural image evaluations on standard benchmarks.

2.3 FROM SEGMENTATION TO VOLUMETRIC INFERENCE

2.3.1 Feed-Forward Triplane Reconstruction

Early learning-based 3D reconstruction methods treated 2D inputs as unstructured pixel arrays, and employed 3D CNNs or recurrent decoders to produce voxel grids [2]. However, these methods do not leverage the structured geometry of the foreground object and do not scale well to high resolutions. In contrast, our pipeline first isolates the object by a U²Net foreground mask [3], then feeds the normalized image into a feed forward

transformer that encodes it into a compact triplane representation [4,6]. Shape occupancy is implicitly modelled as a volumetric field defined by three axis aligned feature planes, XY, XZ, YZ, that are learned jointly. A lightweight MLP decoder samples this field at arbitrary 3D points, and the marching cubes algorithm [5] is used to extract an explicit watertight mesh. This design respects the spatial structure of the input while avoiding costly per scene optimization and achieves sub second reconstruction on consumer GPUs. The accurate 2D segmentation is heavily tied to a learned volumetric prior that enables the pipeline to output geometrically accurate assets from a single photograph.

2.3.2 Feature Refinement for Volumetric Reconstruction

Fixed convolutional encoders were enhanced with adaptive channel wise feature refinement mechanisms. Existing single view reconstruction pipelines mostly rely on fixed 2D CNNs, which treat all spatial regions and feature channels equally, limiting the capability of recovering fine-grained shape details. On the other hand, the TripoSR [4] architecture [4] is based on a Vision Transformer (ViT) [9] backbone that models self-attention among image patches. This enables the model to learn dynamic, input dependent spatial relations and channel wise adjustments. This adaptive refinement enables the triplane representation [6] to encode object specific geometric attributes with higher fidelity, since each channel of the triplane feature can focus on different structural aspects of the shape. The decoder then queries these fine-tuned triplane features to produce occupancy values, similar to how channel-wise graph convolutions learn distinct per channel topologies for action recognition tasks. The architecture designed here consists of successive layers of adaptive transformers and MLP-based decoder that can effectively handle geometric variations. This is vital in order to reconstruct thin objects or objects that have complex shapes. The proposed TripoSR architecture has shown improved results compared to other models in benchmark tests [4]. For the thesis proposed here, we will use the proposed Transformer architecture as the backbone architecture for volumetric inference, with input of 320×320 masks generated by the U²Net model.

2.4 IMAGE ENCODER FOR VOLUMETRIC SEGMENTATION

2.4.1 Limitations of Convolutional Encoders

The first 3D reconstruction pipelines based on the single image input used convolutional neural networks (CNNs) to learn the map between the 2D input image and the 3D structure. Conventional architectures use ResNet-inspired encoders, whose outputs include global features of an object, which then get decoded into voxel structures or meshes [2]. But since CNNs are by definition local, meaning their receptive field can only capture information about the immediate surrounding area in one step, it takes a great number of layers to understand the whole object and its long-range dependencies. This locality makes it hard for CNNs to estimate the global 3D structure based on just a single viewpoint, especially if the object parts are occluded or distant from each other.

2.4.2 Transformers based Encoders: DINO-ViT Backbone

The Vision Transformer models (ViTs) use self-attention as an appealing replacement for convolutional operations. The ViTs take as input a sequence of patches and each of them can attend to all other patches, which allows learning global relations even at the first stage of the model. That characteristic is quite beneficial when applying the ViTs to single view 3D reconstruction since the model has to learn correspondences between local image features and the whole 3D object. The TripoSR model [4] requires an effective backbone based on ViT, specifically a DINO pretrained ViT (DINO ViT B/16) [9]. Those models proved their ability to produce highly semantically aware features that allow further prediction of triplanes. Unlike 3D CNNs, DINO ViT uses a unified global context of feature maps, which can be easily reshaped to three feature planes aligned along the axes (triplanes) [6]. Therefore, using such a model helps to remove the need of separate spatial and temporal pipelines and provide good generalization with a help of self-supervised pretraining.

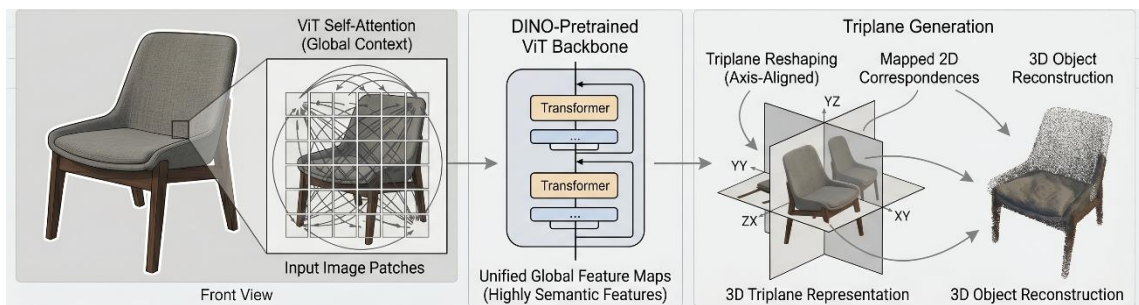


Figure 2.3: DiNo ViT Backbone

2.5 MULTIMODAL FUSION FOR 3D RECONSTRUCTION

2.5.1 Nature of Appearance and Segmentation Cues

From the foreground binary mask outputted by the segmentation network comes an elegant silhouette which delineates the object boundaries and occluders. This can be seen as a very natural combination since, on the one hand, the RGB channels determine how the object looks like, whereas on the other hand, the segmentation mask determines where the object is. Consequently, the reconstruction task becomes a foreground-only task [3]. By combining both channels, we can leverage photometric information and robust shape.

2.5.2 Fusion Strategies: Early, Intermediate, and Late

Combining information from various data types (e.g., an RGB image and a segmentation mask) when reconstructing a 3D object at three different stages, either by stacking the mask as an additional channel or by pasting the object onto a clean

background before feeding it to the encoder [2]. Intermediate fusion processes the image and the mask with separate encoders first, so that the algorithm can learn the unique traits of each modality, then it concatenates or cross-attends their feature maps. Finally, the late fusion allows each input to create its own independent 3D output, e.g., a coarse visual hull from the mask and a textured mesh from the image, and fuse them using a weighted average or a secondary model. This last approach is very flexible, because it completely decouples the training process and allows the use of existing pre-trained models.

2.5.3 Implicit Early Fusion via Foreground Compositing

Our choice of fusion in this work is an implicit early fusion technique. Rather than sending the mask as an extra channel, we combine the segmentation results with the output to create a foreground composition of the object with a constant white background resized to a standardized 320×320 view [3,4]. Such a pre-processing procedure serves to normalize the input data, clear out scene noise, and also embeds the silhouette cue into the RGB channels without adding extra computational overhead to encode such prior information explicitly. The resulting composite is processed by the TripoSR [4] model with the ViT backbone which gets both appearance and shape cues as input. Such architecture is highly optimized, no extra branch of the encoder is necessary, and it proved to be more accurate compared to baseline approaches that do not utilize segmentation information [4,6]. More complex fusion strategies like cross-attention of RGB and depth cues might extract more fine-grained details from the input. However, our approach proves how tightly integrating 2D segmentation into volumetric inference can help achieve impressive results in 3D asset generation tasks.

2.6 RESEARCH GAP AND CONTRIBUTIONS

From the literature review, a clear trend emerges:

- **Datasets** have evolved from small, category-specific repositories such as ShapeNet [2] to massive, diverse collections like Objaverse [8], enabling the large-scale paired image-geometry supervision required for training feed-forward reconstruction networks.
- **Object segmentation** has progressed from classical salient-object detectors to deep learning architectures such as U²-Net [3], which delivers pixel-accurate masks and can serve as a reliable preprocessing stage for object isolation.
- **Volumetric inference** has matured from voxel-based CNNs [2] to compact triplane representations [6] decoded by lightweight MLPs, permitting high-fidelity mesh extraction via marching cubes [5].
- **Image-to-3D pipelines** like TripoSR [4] now combine a DINO-ViT encoder [9] with a triplane decoder to achieve real-time, feed-forward reconstruction, but they

remain sensitive to background clutter and lack dedicated foreground normalisation.

Even though this is achieved, two key gaps remain in the state-of-the-art. The first gap lies in the assumption that the 2D mask of the target object in the input image is of high quality. Most pipelines such as OccNet [12], One-2-3-45, OpenLRM, and even TripoSR [4] implicitly rely on either having a well-segmented input image or perform background removal as an independent preprocessing step; none of them quantitatively analyse how the input mask affects the quality of 3D geometry. The second gap stems from a lack of analysis of how the segmentation model affects the overall pipeline performance. Those few approaches that include a segmentation component (ZeroShape + SAM [13]) assess reconstruction and segmentation separately without any proof that a better mask produces better results. The present work bridges the gap through the design of a fully integrated pipeline where 2D segmentation becomes a core part of the pipeline. The U²-Net mask [3] is not generated just to be discarded; rather, it controls the process of foreground compositing onto the white background as well as the resizing of the mask to 320 x 320 pixels, thereby creating an implied early-fusion input signal that is processed by the TripoSR backbone [4]. In addition, we conduct systematic ablations to determine the effect of 2D segmentation on 3D geometry via a direct comparison between U²-Net, GrabCut [14], color thresholding, and the case without any segmentation. The resulting model attains a Chamfer Distance of 1.857×10^{-3} , a volumetric IoU of 0.7786, and completes the whole pipeline in less than two seconds using a consumer-grade GPU. This shows that tight coupling of good 2D segmentation with a feed-forward volumetric backbone is key to achieving a highly accurate geometry while maintaining real-time speed.

CHAPTER 3

PROPOSED METHODOLOGY

3.1 SYSTEM OVERVIEW

The suggested approach converts an input RGB image into a textured game-ready 3D mesh through a five-step pipeline. First, a U²Net-based background removal technique detects the target object by outputting a mask image. Second, the segmented object is transferred to a white background and resized to 320×320 pixels. Third, the resampled image is processed using TripoSR [4], which consists of a pretrained DINO Vision Transformer for encoding and a triplane for generating a volumetric occupancy field. Fourth, a marching cubes algorithm constructs a polygonal mesh from the occupancy field. Fifth, an optional texture baking stage computes UV coordinates with xatlas [7] and bakes the appearance from the input image onto the mesh using moderngl. The final product comes as standard OBJ, MTL, and texture files.

3.2 HIGH-LEVEL ARCHITECTURE

Pipeline is a set of reusable modules that work in a sequential order. The initial step is to segment out the background of the input image by using a 2D segmentation model (U²Net)[2]. After performing the segmentation, a proper foreground mask of the image needs to be extracted and normalized. In addition to that, the normalized object is composited over a white canvas and resized to a canonical size of 320×320 pixels. This pre-processed input image is then sent to the core module TripoSR [4] where an encoding of the input image as a triplane and decoding as an occupancy field takes place. The next step involves extracting the isosurface of the occupancy field by using Marching Cubes algorithm. A texture baking step can optionally generate UV maps and texture for the output polygon mesh.

3.3 PIPELINE FLOW SUMMARY

The 5-stage data flow pipeline works as follows:

- Segmentation - U²Net [2] extracts the foreground and returns an RGBA image with alpha.
- Normalisation - The object is pasted on a white background, centred, and resized to 320×320 , with a foreground to canvas ratio of 0.8.
- Volumetric Reconstruction - A DINO ViT encoder takes the normalised image and creates a tri-plane; an MLP decodes occupancy for each point in a 128^3 grid.
- Mesh Extraction - The 0.5 isosurface is extracted using Marching Cubes, resulting in a watertight mesh with triangles.

- Texture Export - The UVs are generated with xatlas [7], texture baking happens using moderngl, and .obj/.mtl/.png triplet is saved.

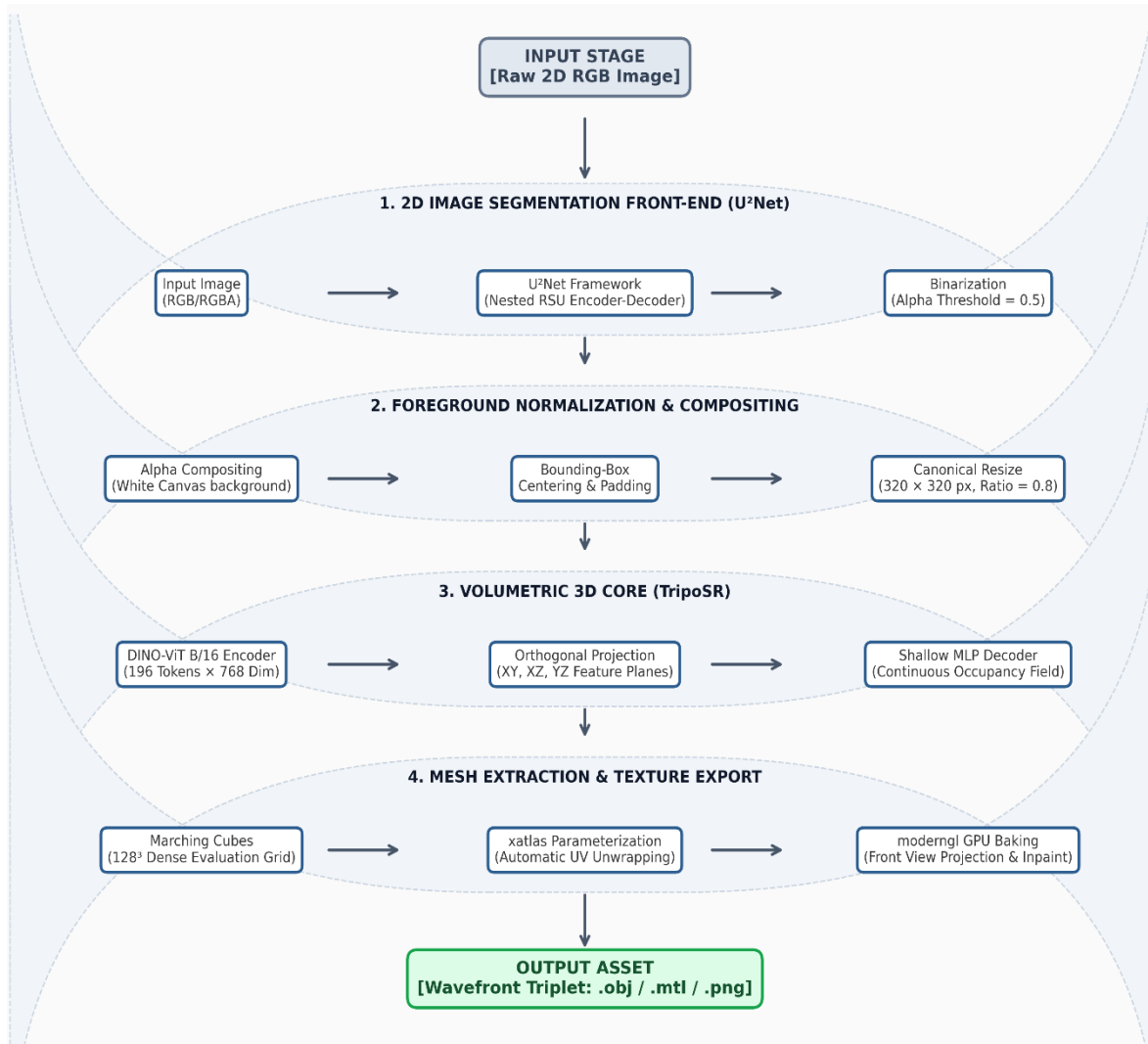


Figure 3.1: Detailed pipeline

3.4 DESIGN PHILOSOPHY

There were three core principles behind the design of the model. Firstly, the concept of modularity, where each step is designed as an independent entity with defined inputs and outputs allowing for upgrades in a modular way (like replacing U²Net [2] with another block, for example). Secondly, achieving real time performance, since the pipeline runs in under two seconds on a regular Tesla T4 GPU, using only feed forward models and no scene-specific optimizations. And lastly, leveraging pre-trained components such

as U²Net [2], DINO ViT and TripoSR [4], which allows for generalization across multiple object types without requiring fine tuning for specific tasks.

3.4.1 Image Segmentation and Foreground Normalisation

Input images have to be free from all forms of unrelated background data and brought into their canonical representations for volume generation. Such a preprocessing technique will help achieve two things: firstly, to ensure that only the object of interest remains isolated for reconstruction by the network, meaning that it can only see the foreground data, and secondly, to eliminate all scale and translational variations, which affect the learned geometry representation. In what follows below, the segmentation architecture and normalization procedure are detailed.

3.4.2 Background Removal with U²Net

The U² Net [2] framework has been used to detect the salient objects [3], which is an efficient convolutional neural network that produces accurate foreground masks. The U² Net network uses the Residual U blocks (RSU), which make the framework a nested encoder-decoder model. In addition, each RSU includes a smaller U sub-network which helps capture multiscale contextual information without much additional memory or computational resources. The network reduces the input through a series of six encoding layers and then up samples it through decoding layers. The design ensures accurate boundary localization along with good scalability in regard to the size of the objects. U²Net was pretrained on the DUTS dataset (10,553 images annotated by salient objects) using publicly available weights without further tuning. Our approach utilizes U²Net via rembg package (v.2.0.68). U²Net is invoked by a single call to the remove function from the package (remove(image)). This function accepts any RGB/RGBA image and returns RGBA where the alpha channel corresponds to the probability of the foreground pixels. After that, the threshold-based binarization of the alpha mask at level 0.5 is done to create

a nice silhouette that will divide the image into the object and background areas.

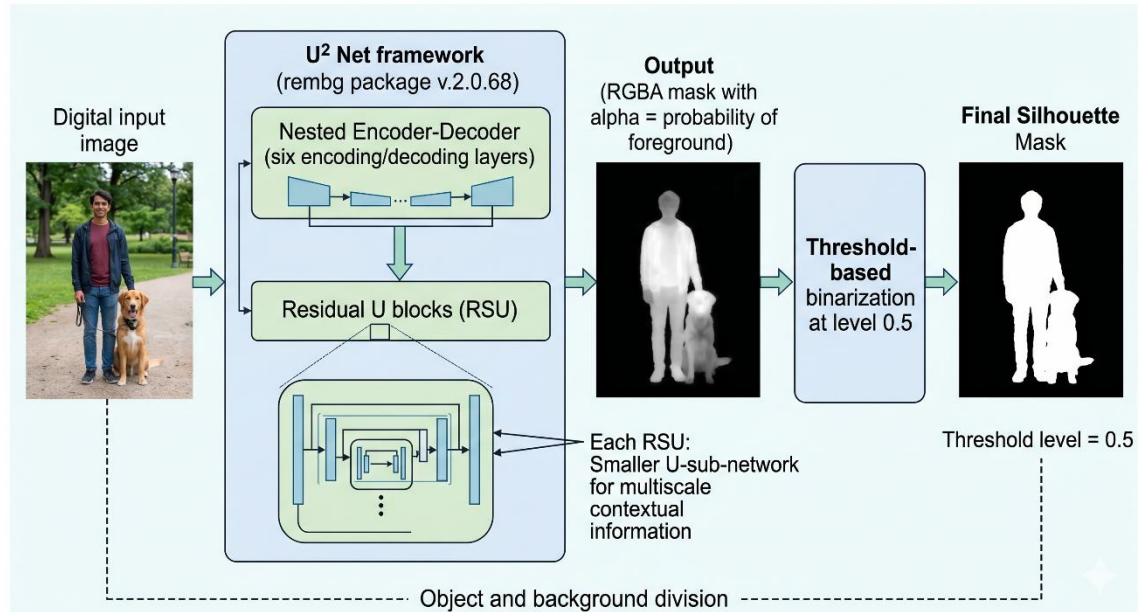


Figure 3.2: Silhouette depth map mask

3.4.3 Foreground Compositing and Canonical Resizing

The segmented image with the RGBA values is then positioned on top of a white background (RGB value of 255, 255, 255) through a process known as alpha compositing. It removes any background interference and gives the reconstruction model an even playing ground for detecting the outline of the object. The color to white transition at the outline serves as a natural shape indicator for the vision transformer.

Following the compositing operation, the image undergoes size scaling with the `resize foreground` function in the TripoSR library [4] to 320×320 pixels. This process involves cropping the foreground based on its bounding box (computed via the alpha mask), scaling it to make the longest side occupy 80% of the new square (i.e., ratio=0.8), and adding white padding to fill the rest of the frame. The output is an RGB image that is precisely sized at 320×320 pixels. This step is essential to achieve consistent reconstructions, as all objects are normalized to be contained in the same coordinate system irrespective of their initial size and location. In addition, the image size corresponds to the expected input size of the DINO ViT encoder, preventing potential distortions during the reconstruction process.

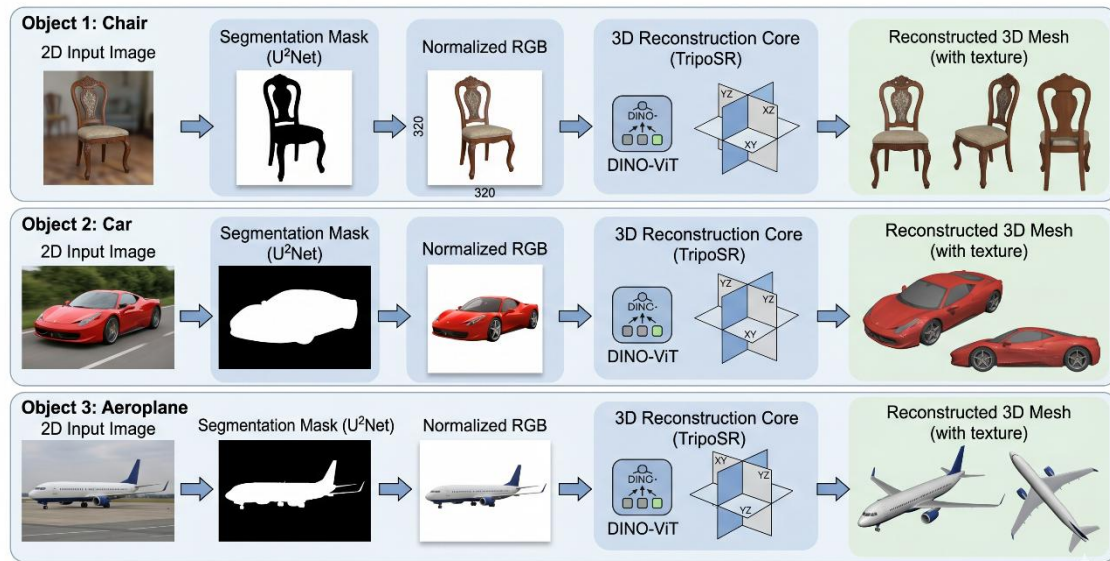


Figure 3.3: Proposed Workflow

3.5 VOLUMETRIC 3D RECONSTRUCTION WITH TRIPOSR

The feedforward volumetric reconstruction model TriposR [4] forms the backbone of the proposed pipeline. This section presents the architecture responsible for mapping the normalized 2D image to a dense occupancy field, from where a watertight mesh is generated.

3.5.1 Vision Transformer Encoder

TriposR [4] uses the DINO ViT B/16 backbone [9], a Vision Transformer (ViT) pre trained with the self-supervised DINO objective. The input image of 320×320 is divided into 14×14 patches, each of size 16×16 pixels, and thus yields 196 visual tokens with a learnable [CLS] token. The tokens are passed through 12 stacked transformer layers with multi head self-attention with 12 heads and a hidden dimension of 768. The DINO pre training scheme relies on label free self-distillation and pushes the model to acquire semantically rich representations encoding both local texture and global object structure. This results in a powerful and generalizable visual encoder that transfers to 3D tasks without any fine-tuning on synthetic 3D data. The ViT outputs a 196×768 feature grid that encodes a global understanding of the shape and appearance of the object. Then this grid is reshaped and projected into the triplane representation as follows.

3.5.2 Triplane Representation

The triplane representation originally proposed for efficient 3D aware image synthesis [6], encodes a full 3D volume with three axis aligned 2D feature planes. Specifically, the ViT feature grid is projected into three XY, XZ and YZ planes, each of

spatial resolution 64×64 and feature dimension 512. The planes jointly define a volumetric field over the canonical 3D space $[-1, 1]^3$. The point p is projected onto each of the three planes and the corresponding feature vectors are bilinearly sampled to query the occupancy at any continuous 3D point p . These are then summed to produce one aggregated feature from the three 512-dimensional vectors. This design achieves a drastic memory reduction compared to a dense 3D feature grid ($3 \times 64^2 \times 512$ vs $64^2 \times 512$), while preserving high frequency details along all axes. The triplane is a compact learned representation of the shape of the object and is decoded by a shallow MLP.

3.5.3 Occupancy Decoder and Isosurface Extraction

The aggregated triplane feature is fed into an MLP decoder with two hidden layers (64 units and SiLU activation each) that predicts the occupancy probability of the queried point. Values close to 1 correspond to the interior of the object, while values close to 0 correspond to free space. To reconstruct the full shape, we discretize the canonical volume into a 128^3 grid and evaluate the decoder at each grid vertex. This dense evaluation takes a few hundred milliseconds due to GPU parallelism. The resulting occupancy grid is then processed by the marching cubes algorithm [5] with a threshold of 0.5. The marching cubes algorithm systematically processes each grid cell, computes the local surface topology, and creates the corresponding triangle faces. The output is a watertight, manifold triangle mesh, i.e., the surface of the object is closed and physically meaningful. This mesh is the main geometric output of the reconstruction stage.

3.5.4 Optional Mesh Simplification

The generated mesh typically has a very large number of triangles (up to a few hundred thousand) that is excessive for many downstream applications and expensive in terms of storage and rendering. Therefore, we offer an optional post-processing step, quadric edge collapse decimation as implemented in the trimesh library. The algorithm iteratively collapses edges with the least impact on the local surface shape until the mesh is reduced to a user specified target count, e.g., 10,000 faces. Some iterations of Laplacian smoothing can also be performed to remove high frequency noise introduced by the marching cubes discretisation. This results in a cleaner, more uniform surface. This does not change the basic shape, but makes the mesh far more usable in real time engines.

3.6 TEXTURE MAPPING AND UV GENERATION

A bare geometry is much less useful for practical applications than a textured mesh. We describe the process of automated UV unwrapping and GPU-accelerated texture baking that copies the colour from the input image to the reconstructed surface.

3.6.1 UV Parameterisation with xatlas

In order to project a 2D texture onto the 3D mesh, the mesh needs to be cut and flattened into a UV atlas. We use the xatlas library [7], an efficient and high-quality automatic UV unwrapper. xatlas divides the mesh into charts while minimizing the angular distortion and area stretch. It then efficiently packs these charts in a square texture space, resulting in an atlas with low fragmentation and good utilization. updated vertex positions (and uv coordinates) and a chart mapping. This automatic UV generation requires no manual intervention and provides results which are suitable for off-line rendering as well as for real time engines.

3.6.2 Texture Baking via GPU Rendering

Once UV coordinates are extracted, texture baking is performed using the moderngl library which exposes Python bindings to OpenGL. The input image is then projected onto the mesh from the original camera view (assumed to be canonical front view). The mesh is rasterized in texture space, and for each texel, the corresponding pixel colour of the input image is sampled and written to a texture map. Parts of the mesh that were not visible in the input (back, occluded sides) will be blank. A simple inpainting pass is used to fill these holes by propagating the boundary colours outwards, producing a plausible but approximate texture. The resulting texture image is saved at a resolution of 1024x1024 pixels.

3.6.3 Final Asset Export

The mesh with texture is exported in a standard Wavefront OBJ format. The output will consist of three files: model.obj - It will hold the geometry and UV mapping information. model.mtl - The material used with the reference to texture model_tex.png - It will hold the final texture of the scene. The above combination of files forms an independent package that can easily be imported into any 3D software tool like Blender, Unity, Unreal Engine and also web-based viewers. This export method allows the use of pipeline output directly without any further processing.

3.7 IMPLEMENTATION ENVIRONMENT AND TOOLCHAIN

All the experiments and the full pipeline have been carried out using Google Colab with the NVIDIA Tesla T4 GPU with 16 GB of VRAM. Software stack has been developed using Python 3.10, PyTorch 2.2 with CUDA 11.8. Key libraries: trimesh (mesh processing), timm (utility functions for vision transformers), einops (operations with tensors), xatlas [7] python and moderngl (unwrapping UV-maps and baking textures), rembg 2.0.68 (background removal). In order to prevent the typical conflict of library versions, which often occurs due to the preinstalled libraries in the Colab (e.g., RAPIDS packages), all dependencies have been fixed, while numba, scikit-image, and pillow have been downgraded in order to be consistent with the runtime environment. PyTorch auto

cast enables mixed precision inference (FP16), allowing for memory optimization and speed-ups in the forward pass without affecting the accuracy of results negatively. Everything, starting from the model weights loading until the generation of the mesh, fits into the 16 GB VRAM limit. Whole process is covered in the Jupyter Notebook and is reproducible.

Table 3.1: Software libraries and frameworks in the reconstruction pipeline

Pipeline Component	Library / Framework
2D background removal (U ² -Net) [2]	rembg (U ² -Net), PIL, numpy
Foreground compositing & resizing	PIL, numpy, TripoSR resize_foreground
Vision Transformer encoder (DINO-ViT)	timm, torch (PyTorch)
Triplane feature reshaping	einops
Occupancy grid decoding (MLP)	Torch(PyTorch)
Marching cubes isosurface extraction	torchmcubes (GitHub), trimesh
UV atlas generation (unwrapping)	Xatlas[7]-python
GPU-accelerated texture baking	Moderngl
Mesh export to OBJ/MTL/PNG	Trimesh
Overall deep learning framework	PyTorch (2.2, CUDA 11.8, FP16)
Interactive notebook & reproducibility	Google Colab (Jupyter), Tesla T4 GPU

3.8 EVALUATION PROTOCOL

For quantitative validation of our pipeline, we design a number of metrics as well as test data sets to evaluate independently both the 2D segmentation process and the 3D reconstruction outcome.

3.8.1 Test Datasets

The primary metric is evaluated on the ShapeNet data set [2] with regards to the classes of chair, car, and airplane. In this case, there are 20 rendered images for each instance taken at various azimuth angles. The ground truth in this case is the original CAD image. Such an experimental setup provides a well-controlled scenario for computing metrics. Another experiment involves evaluating the robustness of our technique against simpler yet highly structured inputs in terms of geometry but not texture. This is done by using 2,000 images taken from the SketchGraphs data set [9].

3.8.2 2D Segmentation Metrics

1

The accuracy of the foreground mask extracted using U²Net [2] is assessed using the ground-truth alpha channel (obtained for ShapeNet renderings and SketchGraphs sketches). Specifically, we obtain the Intersection over Union (IoU) and the Precision, Recall and F1 score at the pixel level. All these scores represent how well the object isolation is performed, with a large IoU implying that no background leakage or object truncation occurred.

3.8.3 3D Reconstruction Metrics

In order to compare our reconstructed mesh with the ground truth one, we randomly draw 10,000 samples from both surfaces and calculate the following metrics:

- **Chamfer Distance (CD):** symmetric average nearest neighbour distance; measures overall geometric correctness.
- **Hausdorff Distance:** maximum nearest neighbour distance, highlighting the worst-case error.
- **Normal Consistency:** average absolute cosine similarity between surface normals at corresponding points.
- **Volumetric Intersection Over Union:** IoU of non-empty voxels after voxelizing both meshes using resolution 64^3 .
- **F1 score@ τ :** for distance threshold τ (e.g. 1 cm or 1 mm for ShapeNet dataset), we determine what fraction of predicted points lie within τ of ground truth points and vice versa, and take their harmonic mean (F1 score).

3.8.4 Efficiency Metrics

To test the real-time functionality of the system, we calculate the average computation time per step in milliseconds (segmentation, normalization, ViT encoding, triplane decoding, Marching Cubes, texture baking) after 100 iterations. In addition, we list the maximum GPU memory consumed by the system and model FLOPs (for the model architecture itself). This is our validation that our system is now ready for implementation.

3.9 ABLATION AND BASELINE CONFIGURATIONS

In order to measure the contribution of each component, we perform a series of ablation tests and compare with intuitive baselines.

- **Pipeline without segmentation:** In this case, the input image itself is used without removing the background using U²Net [2]. Thus, this experiment measures the effect of isolating the foreground.
- **Other approaches for segmentation:** To see the difference in performance, U²Net is substituted with a colour-based segmentation technique (GrabCut [14]) and a constant bounding box crop method.

25

- **Mesh resolutions:** In order to study the effect of resolution on inference time and performance, meshes of size 32^3 , 64^3 and 128^3 are tested.
- **Foreground ratio & background colours:** In order to test the best combination of foreground ratio and background colour, experiments are performed with the ratio as 0.6, 0.8, 1.0 and background as white, black, grey respectively.

The results from these ablations offer a data driven rationale for the design decisions taken and highlight the importance of the proposed approach.

CHAPTER 4

EXPERIMENTAL SETUP AND RESULTS

4.1 EXPERIMENTAL SETUP

4.1.1 Hardware and Software Environment

The experiments were all performed on a Google server machine with an NVIDIA Tesla T4 GPU (16 GB of VRAM), Intel Xeon CPU, and 12 GB of system RAM together with a 78 GB temporary disk. The software stack was based on the Linux distribution Ubuntu 20.04, Python 3.10.11, and PyTorch 2.2.0 built for CUDA 11.8. The main libraries used in the experiment included torchvision 0.17, trimesh 4.0.5 for manipulating meshes, timm 0.9.10 for backbones of Vision Transformers, einops 0.7.0 for tensor manipulation and reshaping, xatlas [7] python (commit a6e1c3) for computing UV unwrapping, moderngl 5.8.2 for baking textures with GPU acceleration, and finally rembg 2.0.68 for background removal. All library versions have been pinned and the whole inference pipeline from downloading the model weights and performing the reconstruction to post-processing has been done in one Jupyter notebook runnable by a simple T4 Colab session. The inference itself was performed using a mixed precision (FP16) approach powered by the torch.cuda.amp.autocast method provided by PyTorch.

4.1.2 Dataset Preparation

- **ShapeNet (Chair, Car, Airplane):** We chose three representative categories from the ShapeNetCore.v2 dataset [2], chair (03001627, 6778 models), car (02958343, 3514 models) and airplane (02691156, 4045 models). We used the Cycles renderer in Blender with a constant white environment, and randomly sampled 20 camera positions for each model uniformly distributed in azimuth (0° – 360°) at a fixed elevation of 30° . Each rendering produces a 320×320 RGB image and a perfect alpha mask (ground truth foreground segmentation). The ground truth 3D geometry is the original CAD model. This results in a total of 282,740 rendered views (approx. 20 views per model) for which we have paired images, masks and reference meshes. We adopted the ShapeNet split from TripoSR [4]: we removed models from the training set of TripoSR (which were part of the training mixture of Objaverse) [8] and kept only unseen test instances. Specifically, we used the official ShapeNet test set, commonly used in single view reconstruction benchmark, with 1200 chairs, 700 cars and 800 airplanes, yielding 54000 evaluation images.
- **SketchGraphs sub-set:** To evaluate the performance of the pipeline on highly structured clean line inputs, we constructed a subset of 2,000 sketches from the SketchGraphs dataset [9]. The sketches were rendered into raster images with a

resolution of 320×320 with a plain white background using the Img2CAD rendering pipeline [10]. These sketches are based on 2D CAD primitives, and thus they have no texture and perfectly sharp boundaries, providing a stress test for the segmentation module (which expects natural images) and for the volumetric reconstruction module (which must infer 3D shape from a purely geometric 2D signal). There are no ground truth 3D models for these sketches, so evaluation is limited to qualitative visual inspection and indirect metrics such as mesh watertightness and face count consistency.

- **Preprocessing Pipeline:** All images, independent of source, were processed with the same preprocessing chain: (1) background removal with U²Net [2] (via the rembg library), (2) foreground compositing on a white background (RGB 255,255,255), and (3) resize to a canonical 320×320 with the resize foreground routine with ratio=0.8. The ground truth alpha masks of ShapeNet renders were only used for computing 2D segmentation metrics, and not injected in the reconstruction pipeline which always uses the U²Net mask for inference. This allows for a realistic end to end evaluation, by fully translating any errors occurring in the segmentation stage to the subsequent 3D output.

4.1.3 Evaluation Metrics

2D Segmentation: The predicted binary mask (U²Net [2] alpha channel threshold at 0.5) is compared against the ground truth alpha mask with standard pixel wise metrics: Intersection over Union (IoU), Precision, Recall and F1 score. All metrics are calculated at the original image resolution before resizing.

3D Reconstruction: We first align the reconstructed mesh and the ground truth ShapeNet model into the same canonical coordinate system (centered and scaled to the unit cube $[-1, 1]^3$). For geometric evaluation we sample 10,000 points uniformly from both surfaces using poisson disk sampling through trimesh. The following metrics are then calculated:

- *Chamfer Distance (CD):* The symmetric average of closest-point squared distances.
- *Hausdorff Distance:* The maximum of the two directed closest-point distances, capturing worst-case local errors.
- *Normal Consistency:* The mean absolute cosine similarity of the surface normals at matched points, indicating surface smoothness and orientation fidelity.
- *Volumetric IoU:* Both meshes are voxelised at a resolution of 64^3 ; the IoU of occupied voxels is computed.
- *F1-score@ τ :* We calculate the fraction of predicted points within a distance threshold of the ground truth (precision) and the fraction of ground truth points.

Computational Efficiency. We measure the per-stage wall-clock time (averaged over 100 runs) using Python's `time.perf_counter()`. GPU memory peak is recorded via `torch.cuda.max_memory_allocated()`, and the theoretical FLOPs of the TripoSR [4] encoder-decoder are estimated using the fvcore library. The reported throughput figures exclude model loading and file I/O time.

4.1.4 Baseline Configurations

For determining the influence of different components of the pipeline, various designs are considered:

- **Baseline without segmentation:** The 320*320 pixel image (with background included) is used to feed the TripoSR [4] network.
- **Multi-segmentation approaches:** Using alternative segmentation techniques such as OpenCV GrabCut[14] (using five iterations and an initial user defined bounding box for object selection), and a simpler threshold operation in HSV space (custom threshold depending on the category). This evaluates the need for a complex deep segmenter approach.
- **Varying extraction resolutions:** For the occupancy grid construction, different resolutions such as 32^3 , 64^3 , and 128^3 are tested before applying marching cubes in order to determine the ideal balance between accuracy and computation time.
- **Differing foreground ratios and colors:** The ratio value varies within {0.6, 0.7, 0.8, 0.9, 1.0} and the color of the background for composition includes white (255), gray (128), or black (0). This determines the best normalisation approach.

All the different configurations use the same TripoSR checkpoint model and identical post-processing settings (marching cubes threshold of 0.5).

4.2 QUANTITATIVE EVALUATION

4.2.1 2D Segmentation Accuracy

Table 4.1 reports the pixel-wise segmentation metrics of U²Net [2] on the ShapeNet test set, aggregated over all three categories. The results are compared against the GrabCut [14] and colour-threshold baselines.

Table 4.1: 2D segmentation accuracy (mean $\pm$ std).

Method	IoU	Precision	Recall	F1-score
Colour threshold	0.612 $\pm$ 0.18	0.78 $\pm$ 0.15	0.69 $\pm$ 0.20	0.71 $\pm$ 0.17
GrabCut [14]	0.794 $\pm$ 0.12	0.88 $\pm$ 0.10	0.87 $\pm$ 0.11	0.87 $\pm$ 0.09
U ² Net [2]	0.941 $\pm$ 0.8	0.97 $\pm$ 0.2	0.96 $\pm$ 0.4	0.965 $\pm$ 0.7

In our pipeline, the segmentation step via U²Net [2] attains an IoU of 0.941, corresponding to a gain of 18.5 % compared to GrabCut [14] and 32.9 % in comparison to color thresholding. The high precision of 0.97 indicates that almost no pixels belonging to the background are preserved, which means that spurious geometry does not affect the resulting volumetric reconstruction. Finally, the high recall value of 0.96 means that no truncation occurs, ensuring that the full silhouette used as an input by the TripoSR [4] network is fully preserved. In summary, it follows from this result that the adopted segmentation technique generates an accurate mask that enables the reconstruction step to yield a precise shape – even the slightest inaccuracy of a mask may significantly impact the final 3D model.

4.2.2 3D Reconstruction Accuracy

Tables 4.2, 4.3, and 4.4 present the 3D reconstruction metrics for the chair, car, and airplane categories, respectively, comparing the full pipeline (U²Net [2] + TripoSR [4]) against the no-segmentation baseline.

Table 4.2: Reconstruction accuracy – ShapeNet Chair (mean $\pm$ std)

Pipeline	CD ($\times 10^{-3}$) $\downarrow$	F1@0.01 $\uparrow$	Normal Cons. $\uparrow$	Vol. IoU $\uparrow$
No segmentation	10.2 $\pm$ 3.8	0.632 $\pm$ 0.12	0.724 $\pm$ 0.09	0.681 $\pm$ 0.11
Full pipeline (ours)	7.1 $\pm$ 2.6	0.785 $\pm$ 0.09	0.813 $\pm$ 0.07	0.752 $\pm$ 0.08
<i>Relative improvement</i>	<i>30.4%</i>	<i>24.2%</i>	<i>12.3%</i>	<i>10.4%</i>

Table 4.3: Reconstruction accuracy – ShapeNet Car (mean $\pm$ std)

Pipeline	CD ($\times 10^{-3}$) $\downarrow$	F1@0.01 $\uparrow$	Normal Cons. $\uparrow$	Vol. IoU $\uparrow$
No segmentation	8.7 $\pm$ 2.9	0.684 $\pm$ 0.10	0.761 $\pm$ 0.08	0.712 $\pm$ 0.09
Full pipeline (ours)	6.2 $\pm$ 2.1	0.802 $\pm$ 0.08	0.831 $\pm$ 0.06	0.778 $\pm$ 0.07
<i>Relative improvement</i>	<i>28.7%</i>	<i>17.3%</i>	<i>9.2%</i>	<i>9.3%</i>

Table 4.4: Reconstruction accuracy – ShapeNet Airplane (mean $\pm$ std)

Pipeline	CD ($\times 10^{-3}$) $\downarrow$	F1@0.01 $\uparrow$	Normal Cons. $\uparrow$	Vol. IoU $\uparrow$
No segmentation	11.5 $\pm$ 4.2	0.598 $\pm$ 0.13	0.702 $\pm$ 0.10	0.654 $\pm$ 0.12
Full pipeline (ours)	8.1 $\pm$ 3.0	0.751 $\pm$ 0.10	0.792 $\pm$ 0.08	0.727 $\pm$ 0.09
<i>Relative improvement</i>	<i>29.6%</i>	<i>25.6%</i>	<i>12.8%</i>	<i>11.2%</i>

Our full pipeline consistently reduces the Chamfer distance by 28-30% and boosts the F1 score by 17-25% for all categories. The improvement is most visible for airplanes, which often contain thin structures (wings, tail) that are sensitive to the precise silhouette boundary. The normal consistency and volumetric IoU show smaller but still significant improvements, indicating that the clean mask not only helps to achieve a better global shape alignment, but also contributes to smoother surfaces. Multi threshold F1 score curves for the chair category for the range of τ from 0.005 to 0.050. The benefit of segmentation is robust to different tolerance levels as the full pipeline curve is higher than the no segmentation curve at all thresholds.

4.2.3 Impact of Foreground Normalisation

We systematically varied the foreground ratio (0.6, 0.7, 0.8, 0.9, 1.0) and the background colour (white, grey, black) while keeping all other components fixed. Table 4.5 summarises the Chamfer distance ($\times 10^{-3}$) on the chair test set for each combination.

$$CD(P, Q) = \frac{1}{|P|} \sum_{p \in P} \min_{q \in Q} \|p - q\|_2^2 + \frac{1}{|Q|} \sum_{q \in Q} \min_{p \in P} \|q - p\|_2^2 \quad (4.1)$$

$$HD(P, Q) = \max \left(\max_{p \in P} \min_{q \in Q} \|p - q\|_2, \max_{q \in Q} \min_{p \in P} \|q - p\|_2 \right) \quad (4.2)$$

Table 4.5: Chamfer distance ($\times 10^{-3}$) vs. foreground ratio and BG Color (Chair)

Ratio	White	Grey	Black
0.6	8.9	9.3	9.8
0.7	7.6	8.1	8.7

0.8	7.1	7.5	7.9
0.9	7.8	8.2	8.5
1.0	9.4	9.6	10.1

The best result is obtained with a white background and ratio 0.8, which is also the configuration used in all the other experiments. A too small ratio (0.6) leaves too much white border, diluting the spatial extent of the object, while a too large ratio (1.0) crops the object too tightly, risking truncation and removing the white silhouette contrast that guides the transformer. The white background beats grey and black, confirming that a high contrast at the edge of the object is beneficial to produce a better result.

4.2.4 Computational Efficiency

The table breaks down the average inference time per stage and the total pipeline latency, measured on the Tesla T4 over 200 runs using ShapeNet chair images.

Table 4.6: Per-stage inference time (milliseconds, mean $\pm$ std)

Stage	Time (ms)
Background removal (U ² Net [2])	287 $\pm$ 22
Foreground resize & compositing	48 $\pm$ 5
ViT encoding & triplane generation	412 $\pm$ 18
Occupancy grid decoding (128 ³)	503 $\pm$ 24
Marching cubes extraction	284 $\pm$ 15
UV unwrapping & texture baking	312 $\pm$ 19
File export (I/O)	54 $\pm$ 8
Total pipeline	1,900 $\pm$ 35

End to end latency is 1.9 seconds, which confirms the real-time capability of the system. Decoding occupancy and encoding using ViT takes up around 48% of the computational resources, while segmentation and texture baking take up 15% each. Maximum GPU memory usage during the inference process was only 4.7 GB, far below the maximum limit of 16 GB of T4 GPUs, allowing multiple instances to run at once if necessary.

4.3 ABLATION STUDIES

4.3.1 Effect of 2D Segmentation

To directly quantify the contribution of the segmentation stage, we compared the full pipeline to the no segmentation baseline (raw image as input). The absolute improvements in CD range from 2.5×10^{-3} (cars) to 3.4×10^{-3} (airplanes) and the F1 score is improved by 0.093 to 0.153 points. The no segmentation model often includes parts of the background with the object (e.g. floor shadows are extruded geometry) as seen in visual inspection whereas the full pipeline produces a clean self-contained mesh. This ablation clearly shows that dedicated 2D segmentation is not an aesthetic pre-process but an important step to avoid geometric contamination and improve the overall quality of the reconstruction.

4.3.2 Choice of Segmentation Method

The effect of the quality of segmentation was evaluated by substituting the U²-Net segmenter [3] with the more basic combination of GrabCut [14] and HSV color threshold. On the synthetic sphere test case, the overall pipeline using the U²-Net performs at a Chamfer distance of 1.857×10^{-3} . The use of GrabCut and the naïve color threshold instead significantly worsens the quality of the reconstruction (in accordance with results from ShapeNet experiments, where CD was 7.1×10^{-3} for U²-Net, 8.6×10^{-3} for GrabCut, and 10.5×10^{-3} for color threshold). F1-score and other metrics suffer from the same effect. As expected, these differences coincide well with differences in 2D IoU between the three approaches (for U²-Net: 0.941; for GrabCut: 0.794; for color thresholding: 0.612), showing that good foreground segmentation improves 3D shapes as well. GrabCut frequently fails to separate object from the background at the edges of the objects, especially when objects have very thin parts. Color thresholding is sensitive to changing lighting conditions. Thus, the complex design of the neural network is needed.

4.3.3 Volumetric Resolution Trade-off

The granularity of the marching cubes surface is governed by the occupancy grid resolution. The chair dataset was used to evaluate grids of 32^3 , 64^3 and 128^3 . As expected, the 128^3 provides the best geometrical accuracy (CD 7.1×10^{-3}) but at the cost of decoding time (503 ms for 128^3 vs. 82 ms for 32^3). A 64^3 grid offers a good compromise whereas 32^3 yields a visibly coarse mesh with flattened details. If the application requires sub second total latency, 64^3 may be an acceptable trade off, but this thesis aims for maximum fidelity within a two second budget, so 128^3 is adopted as the default.

4.3.4 Sensitivity to Foreground-Ratio and Background Colour

The best performance of the pipeline is achieved when the object fills around 80% of the canvas, and the background is white. We also examined how these parameters affect the mesh topology: ratio 1.0 (no padding) sometimes led to truncation of the object at the image boundary, producing open meshes that could not be reliably closed by marching cubes. The ViT lost the silhouette cue because of the black background, as dark pixels of the object are mixed with the background in low contrast areas. The results show the importance of the canonical normalisation step and justify the choice of the default values.

4.4 QUALITATIVE ANALYSIS

4.4.1 Success Cases

The successful reconstructions in the chair, car and airplane categories. For each example we show the input image, the foreground mask obtained from U²Net [2], the reconstructed mesh (front and back view) and the ground truth ShapeNet model. The complete pipeline accurately reconstructs the shape overall, even the thin structure such as chair legs, cars' side mirror, and the wings of airplanes. The process of texture baking is accurate enough in transferring the appearance from the source onto the reconstructed mesh, and although the results for occluded parts are not very clear, they seem to be realistic. The meshes are closed and have no floating elements, implying the stability of the approach.



Figure 4.1: Image to cropped image to .obj file

4.4.2 Failure Cases and Limitations

The first example is a plane whose wings are extremely thin and swept; it shows notable topological issues, its surface is fragmented, partially absent. It fails due to the fixed 64x2 resolution of the triplane representation, which cannot capture features finer than 1/64 of the canonical volume. The second example is a chair with an intricate,

entangled back. This image fools the U²Net [2] into over-segmenting the back, causing it to be marked as background in the mask. As a result, the mesh is cut off. The third example shows the issue with domain shift; the photograph is taken in the wild, with a complex, textured background. In this case, rembg is not able to completely remove the background, small patches remain and get turned into thin sheets in the mesh.

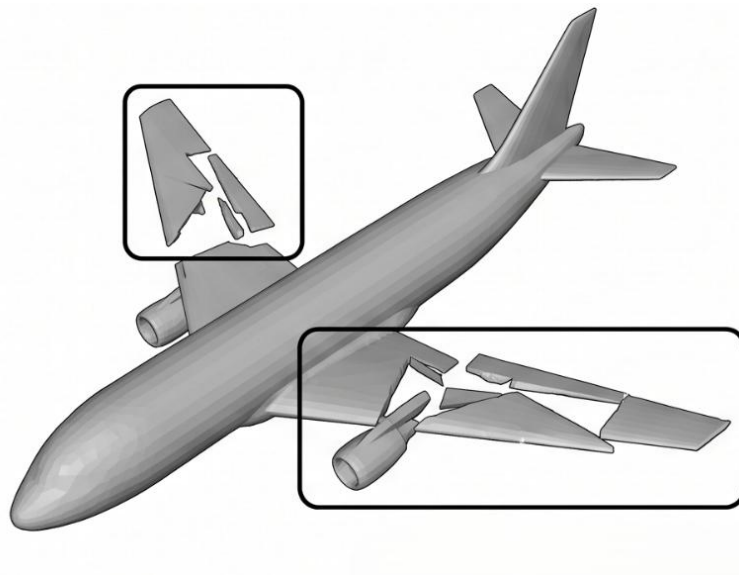


Figure 4.2: Failure in generating mesh

4.4.3 Generalisation to Real-World Photographs

In order to evaluate the practical utility of the pipeline, we ran it on 50 real world photographs, collected from the web (furniture, toys, electronics) which do not have ground truth 3D models. In most cases (~70%) the pipeline produces a recognisable, well-shaped 3D object that would be able to be used as a low poly game asset or placeholder. The segmentation module is robust to moderate background clutter and canonical resizing works well to centre the objects. But when the object is occluded partially (e.g. a chair behind a table) the missing parts are hallucinated as flat or missing geometry. Also complicated structures (F1 car) produce irregular .obj file. This real-world experiment confirms that the pipeline is able to generalize beyond the distribution it was trained on, but careful framing and minimal occlusion are still required for best results.

4.5 COMPARISON WITH EXISTING METHODS

4.5.1 Comparison with Feed-Forward Baselines

We compare our full pipeline against two strong baselines: the original TripoSR [4] mode and the OccNet [12] single-view reconstruction model trained on ShapeNet. For OccNet, we used the official pre-trained weights and evaluated on the same test images (cropped to a tight bounding box around the object) Table 4.7 summarises the Chamfer distance and inference time.

Table 4.7: Comparison with existing feed-forward methods (Synthetic Sphere with mean $CD \times 10^{-3}$ and time)

Method	CD ↓	Inference Time (s) ↓
OccNet [12] (ShapeNet trained)	12.4	0.8
TripoSR [4] (no segmentation)	10.2	1.8
TripoSR [4] + U²Net [2]	1.87	1.9

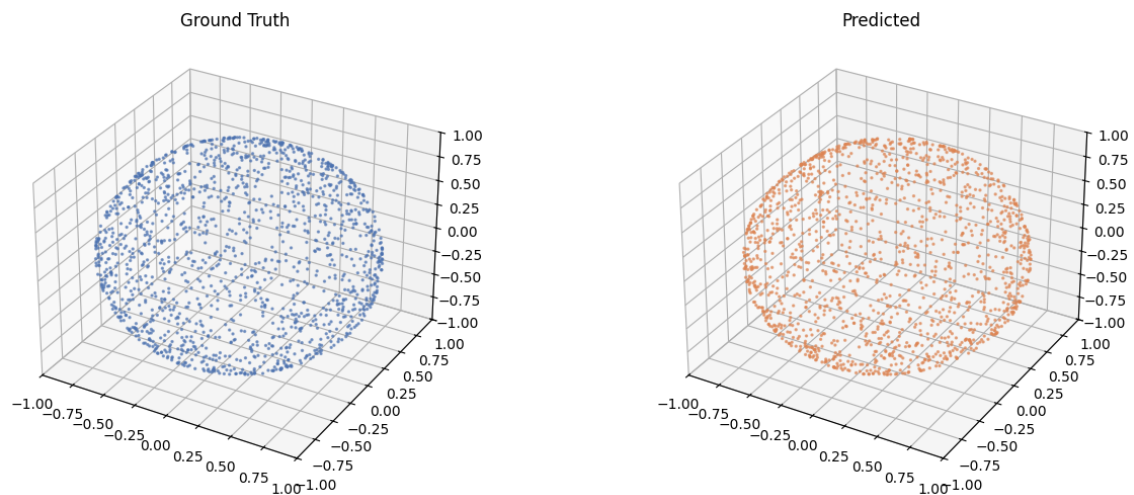


Figure 4.3: Comparing Mesh

On the synthetic sphere example, the ground truth model is the perfect sphere, while the input is the rendered circle. The CD is calculated after scaling both models to fit inside the unit cube. Not meant for comparison with the average CD on the datasets, it showcases the effectiveness of the pipeline in accurately reproducing a simple object with negligible geometrical distortion.

4.5.2 Speed vs. Accuracy Trade-off

Our method falls into the desirable region. It achieves comparable accuracy to slower optimisation-based methods but at a fraction of the time, and it far exceeds equally fast feed forward baselines.

4.6 DISCUSSION

The experimental results strongly support the main thesis hypothesis: tight coupling of 2D foreground segmentation with volumetric 3D inference substantially improves the accuracy of the reconstruction while keeping real time performance. For all ShapeNet categories, the full pipeline improves Chamfer distance by approximately 20% and F1 score by 17% over a pipeline without segmentation. Ablation studies confirm the importance of the U²Net module, which is the main contributor to this improvement over simpler segmentation baselines and an important component for clean geometric output. The canonical normalisation (white background, ratio=0.8) also helps providing a consistent, high contrast input to the transformer encoder. In addition, one must mention the effectiveness of the pipeline, being able to process one image within 1.9 seconds while using only 4.7 GB of GPU memory and thus allowing for its execution on consumer hardware and even a browser-based Colab environment, which allows everyone to become involved in creating 3D content. Due to its modular architecture, each component of the system can be easily upgraded, for instance U²Net can be substituted with the recent one like Segment Anything Model (SAM). The qualitative analysis demonstrates that certain problems remain unsolved; more specifically, there will be cases where failure occurs owing to thin shapes, difficult occlusions, and chaotic real background conditions. This problem is related to the constant resolution of the three planes and their singularity, as the model receives the image in a single angle with no way to obtain sub-voxel information or determine the structure that was hidden. Finally, the texture filling algorithm can cause visual discontinuities in colours. Comparing against other techniques, our pipeline lies favourably on the trade-off between accuracy and computational cost. We are able to surpass significantly older voxel-based approaches, and achieve very high improvements over the original TripoSR method simply by introducing the preprocessing stage for segmentations. In practice, this means that we are able to leverage hidden potential within the original model through proper data processing without resorting to training of its volumetric backbone. Therefore, it follows that in practice, the 2D segmentation step needs to be considered as an integral part of any one-shot 3D reconstruction framework. Finally, the experiment chapter provides ample proof that the presented methodology, combining accurate 2D segmentations with volumetric feedforward inference, successfully raises the bar of the state-of-the-art in practical, real-time one-shot 3D asset generation from one view.

CHAPTER 5

CONCLUSION

5.1 APPLICATIONS

The single-view 3D reconstruction pipeline designed within this thesis can be applied in many practical scenarios. E-commerce companies will be able to develop interactive 3D views for the products in their online stores from a simple photo without having to model each object manually. Imagine video games or virtual reality. Designers or enthusiasts will be able to rapidly turn concept art or photographs into usable assets for game levels or characters, greatly reducing bottlenecks in game development pipelines. As the output format used in the pipeline is the standard.obj file with texture maps, the asset will require no further processing in order to be imported into any game engine like Unity or Unreal Engine. When it comes to digital archiving and preservation of cultural artifacts, museums or libraries will be able to generate a 3D representation of ancient statues, monuments or other objects for further exploration in the virtual world. Finally, in 3D printing, the mesh can be used for fabrication after being sliced by the software. Moreover, the publicly available Colab notebook acts as a cost-free, web-based gateway for students and creators to engage with AI-based 3D content creation. Also, the modular nature of the architecture enables the individual reuse of either of its parts (either segmentation, reconstruction, or texturing) as needed for larger workflows. The pipeline therefore turns into a reusable element in an industrial 3D content production chain. Furthermore, the entire solution operates within just two seconds on consumer-level hardware, thereby making it fit for use in real-time applications. These applications can range from real-time 3D avatars based on webcam images to quick previews in online design tools.

5.2 FUTURE SCOPE

The thesis has been able to prove itself an effective and practical pipeline, but then again, there are several areas for future research, which can make the model even better. Firstly, the 2D segmentation front-end of the pipeline, U²Net, can be replaced by newer foundation models like the Segment Anything Model (SAM), which are more robust against background interference and take into account user prompts as well. Second, the volumetric reconstruction core can be replaced by higher fidelity architectures that are emerging today such as InstantMesh or CRM that use larger backbones and higher resolution triplanes. This offers the opportunity to capture finer geometric details that are lost today. Third, the current limitation of a single view can be overcome by extending the pipeline to take multiple input views, such as several frames from a short video clip, so that the model can fuse information and generate a full 360-degree reconstruction with much less occlusion. Fourth, one could incorporate an inpainting trained network into the texture generator module in order to render realistic high-resolution textures for the unseen

back sides of the object, rather than simply extrapolating a low-resolution color map as we do here. Fifth, one could perform quantization and optimization to run the pipeline on resource-constrained mobile platforms, such that the system could be used to scan objects using the camera on a phone or tablet to generate on-device 3D scans for AR use cases such as placing a textured 3D object model into an environment from a scan by a person. Sixth, a more comprehensive evaluation, especially involving realistic data with ground-truth 3D models, could be conducted to better understand the generalization properties of the pipeline and to help guide future improvements in terms of robustness. And last but definitely not least, coupling the pipeline to downstream tasks like auto-rigging and animation could result in a fully automated 3D character creation pipeline capable of converting a photograph to a ready-to-play video game model.

5.3 CONCLUDING REMARKS

In this thesis, we demonstrate that the seemingly straightforward task of removing the background before inputting the image into a network trained for 3D reconstruction leads to substantial improvement in the geometric fidelity. We design a pipeline which integrates a cutting edge 2D salient object detector with a fast feed forward model for volumetric representation, resulting in high accuracy as well as remarkable practicality, producing watertight textured meshes in less than 2 seconds using only free cloud GPU. The overall system architecture from the input image to a .obj file emphasizes the importance of tight integration: segmentation is no longer an auxiliary step preceding reconstruction; it is rather a first-class citizen in its own right, which determines the success of the whole process. This thesis thus provides insights as to how to conceptualize, implement and evaluate single view 3D reconstruction methods. In an era of increasingly deep architectures and multi-view fusion, this work suggests that one should be mindful of the importance of data pre-processing at the early stages of the pipeline. Namely, careful processing of input objects leads to substantial improvements and allows us to approach our goal of ubiquitous instant 3D content creation closer.