# 3D CHARACTER RECONSTRUCTION FROM 2D ORTHOGONAL IMAGES

BY

KOK YUNG THOW

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2025

## COPYRIGHT STATEMENT



This Final Year Project proposal is submitted in partial fulfilment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project proposal represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project proposal may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

## ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Ts Dr Saw Seow Hui who has given me this bright opportunity to engage in an 3D reconstruction research project. It is my first step to establish a career in Computer Science field. A million thanks to you.

To a very special person in my life, Tan Pei Shi, for her patience, unconditional support, and love, and for standing by my side during hard times. Finally, I must say thanks to my family for their love, support, and continuous encouragement throughout the course

## ABSTRACT

The demand for 3D content in games and films has driven the need for more efficient character modelling workflows. Traditionally, 3D character creation from concept art requires time-consuming manual modelling and sculpting, followed by retopology, texturing, and rigging. These processes may need weeks to complete for a single character. While recent AI-based 3D generation models offer faster alternatives, but lack of fine control over the output and are often requires extensive post processing to ensure the quality of the mesh.

This research is motivated by the need to support artists in traditional 3D character modelling workflow that refine the details of mesh from primitive shapes. Instead of fully automating the modelling process, this project aims to introduce an algorithm that reconstruct primitive cubes of stylized 3D character from front, left, back and right orthogonal views images by integrating interactive image segmentation and blender scripting method. This enables a rapid starting point for geometry detail adjustment using box modelling or digital sculpting techniques.

The project objectives are to: (1) review the existing state-of-arts for solving 3D generation and reconstruction, (2) design an algorithm capable of reconstructing primitive shapes of 3D characters from four orthogonal images, optimized for limited computational resources.

This research contributes a flexible and artist-friendly reconstruction algorithm that capable of generating part-level primitive shapes, which is efficient for generating high-fidelity mesh using box modelling and digital sculpting techniques. The algorithm can generate primitive shape from at least two orthogonal view images. The proposed algorithm is organized into modular stages, allowing artists and developers to further improving the pipeline by integrating computer vision, AI models and 3D modelling and scripting techniques. Finally, the model's performance will be evaluated based on topology quality, editability, and usability. This work not only improves the character modelling pipeline, but also supports future studies in 3D reconstruction and digital content creation,



Area of Study: 3D Reconstruction, 3D Digital Modelling
Keywords: Multiview 3D Reconstruction, Artificial Intelligent, Blender Scripting, Orthogonal View Input, Computer Vision

## TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF SYMBOLS

| | |
|---|---|
| $C$ | Character |
| $m$ | Mesh |
| $x$ | x axis point |
| $y$ | y axis point |
| $z$ | Z axis point |
| $H_{cart}$ | *horizontal component in Cartesian coordinate system* |
| $V_{cart}$ | *vertical component in Cartesian coordinate system* |
| $W$ | *width or height or a square image* |
| $f$ | front view image |
| $l$ | left view image |
| $b$ | back view image |
| $r$ | right view image |

## LIST OF ABBREVIATIONS

| | |
|---|---|
| *3D* | Three dimensional |
| *2D* | Two dimensional |
| *AI* | Artificial Intelligent |
| *CNN* | Convolutional Neural Network |
| *OV-CV-BPY* | Orthogonal View – Computer Vision – Blender Python |
| *API* | Application Programming Interface |
| *AABB* | Axis-Aligned Bounding Box |
| *NDC* | Normalized Device Coordinates |
| *SAM* | Segment Anything Model |
| *min* | Minimum |
| *max* | Maximum |
| *pixel* | Image Coordinates |
| *Cart* | Cartesian Coordinates |

# Chapter 1: Introduction

This chapter provides project's problem statement and motivation, objectives, scope, contributions and the organization of the report.

## 1.1 Problem Statement and Motivation

This section presents the core challenges addressed in the project and the motivation behind them:

- Traditional 3D modeling techniques remains essential for character modeling, especially in real time rendering and animation workflows, which requires manual detail refinement on primitive shapes is often more efficient than working with high polygon meshes generated by AI models.
- The existing 3D shape generation models can reconstruct 3D characters from few images but requires intensive post adjustment manually on the topology from professionals to ensure the further usability of the output.
- The existing generative AI models can solve 3D object reconstruction on static for 3D printing objects well but lack focus on characters for 3D animation which requires clean topology and UV texture mapping.

## 1.2 Objectives

The aim of this project is to provide a review on current 3D shape generation methods and explore the integration of computer vision and script-based digital modeling techniques to contribute on interactive 3D reconstruction approach that improve manual 3D character modeling workflows.

The specific objectives are:

- To review the existing state-of-arts for solving 3D generation and reconstruction.
- To design an algorithm capable of reconstructing primitive shapes of 3D characters from four orthogonal images, optimized for limited computational resources.

## 1.3 Project Scope

- The term "3D character" mentioned in this project refers to stylish characters, not photorealistic human characters.

- This thesis covers a review of related work on solving 3D object reconstruction or generation from synthesis data like images and text. While the focus is on stylized characters and general 3D shapes.
- An interactive OV-CV-BPY algorithm is introduced to reconstructs part-level primitive cubes of stylized characters from front, left, back and right 2D orthogonal view images without relying on pre-trained 3D models.

## 1.4 Contributions

Automation in 3D digital modeling has long been a goal for developers and artists. Current advances in 3D diffusion transformer AI models can generate high detailed shapes and textures from minimal input, such as single or descriptive text prompts. These models inherit the 2D generative AI models that can learn 3D representations from point clouds, meshes, and other explicit formats by encoding them into implicit 3D structures. This research presents a review of these state-of-arts methods.

When a diffusion model received a prompt, it iteratively denoises and synthesizes novel object views, decoding them into a final 3D shape. However, this process is computationally intensive and often produce outputs that require post-processing like topology cleanup and polygon counts reduction before they are usable in animation process.

This research demonstrates the feasibility of assisting 2D and 3D artiest in converting orthogonal-view concept art into low-fidelity primitive 3D shapes using an interactive segmentation-based approach. These reconstructions serve as quick starting point for manual refinement, improving the traditional stylized character modeling process by adjust the scale of character's parts in the modeling process. This project contributes a lightweight, performance-balanced method for 3D reconstruction tasks.

## 1.5 Report Organization

Chapter 2 provides reviews on current state-of-arts in 3D object reconstruction and generation, and review on related work when exploring the possibility of lightweight and performance trade off solution. Chapter 3 introduces the designed algorithm that reconstruct primitive shapes from orthogonal view images. Chapter 4 reports the high-level design of algorithm

implementation. Chapter 5 reports the experimental results, and the validity and generality justification on the designed solution.

## Chapter 2: Literature Review

This section introduces the review of technologies in solving 3D shape generation and reconstruction and the review of related work on designing OV-CV-BPY algorithm.

### 2.1 Manual Digital Modelling

There are two main handcraft approaches in 3D character modeling using software in current 3A game's industry standard, including polygonal or box modeling, and 3D sculpting.

#### Polygonal Modelling

Polygonal Modelling refers to a method of modeling a character or object by manually adjusting the vertices, edges, faces of the geometry shape or polygons using 3D software like Blender and ZBrush. This method provides full control of creating 3D character surface, defined by mesh, a collection of vertices, edges, and faces. It enables artists to create good topology mesh, and the format of the mesh such as glb. can be supported across 3D software like 3Ds Max, and Maya, and it is flexible for rigging and animation in producing animation films and 3D games. However, the speed of modeling depends on an artist's skills. It is time-consuming to model a complex shape such as realistic human characters or any irregular organic shapes.

#### 3D Sculpting Modelling [3]

Sculpting modeling is a method that is similar to sculpting real clay. 3D artists use brushes to push, pull, and smooth the digital mesh instead of working with polygon faces. This method is a common way to model 3D characters in 3A games and films. It allows 3D artists to create high-detail shapes and irregular organic shapes with a natural workflow. However, since sculpting modeling deals with high-resolution mesh, retopology needs to be performed by artists before importing them into real-time applications to ensure rendering optimization. Sculpting modeling requires highly skilled 3D artists, and it is time-consuming when modeling very high-resolution characters, such as monsters, and non-human characters.

### 2.2 3D Reconstruction

3D reconstruction is an automatic task that reconstructs structures of an object or a scene from real-world captured data. Traditionally, this task mainly solving reconstructing objects from

images captured by cameras or sensors. As generative AI become popular recently, 3D reconstruction techniques like structure from motion is adapted in innovative trained systems that reconstruct or generate 3D objects from fewer images. Most of the 3D reconstruction systems are usually used to reconstruct complex objects such as statues, buildings, or a large-scale environment from calibrated images, which are commonly used in 3A video games and movies.

**Photogrammetry [11, 12, 14]**

Photogrammetry refers to an image based contactless 3D reconstruction method that reconstructs 3D models by analyzing multiple 2D photographs or images taken from different angles and then reconstructing the depth and the surface structure. Photogrammetry can reconstruct 3D models with very high details and realistic textures. Additionally, compared to polygonal modeling and sculpting modeling, it is more accessible since only standard cameras or smartphones are needed. However, photogrammetry is computationally expensive in terms of CPU computation and processing time and requires many good coverage images to reconstruct high-accuracy models. Lastly, it is sensitive to lighting, such that the capturing environment needs to be controlled when reconstructing models from real humans, else there will be many noisy results that may need to be cleaned in 3D software. There are some common applications available for photogrammetry are Polycam, Meshroom, COLMAP [11, 12], RealityCapture and 3D scanning devices.

**Implicit Representations**

Nerf [4, 5, 6, 14] and 3D Gaussian Splatting [7, 8, 14] are new methods that solve the limitations of photogrammetry, such as time-consuming processing time and the need for intensive images as input, and lighting sensitivity problems. Both of the methods can reconstruct 3D models with few images, and the result is very realistic very fast, and can reconstruct class, water, and reflections which photogrammetry cannot. This can be achieved by combining the implicit representations of 3D objects with deep learning and generative AI. However, Nerf and 3D Gaussian Splatting need to be trained before they can reconstruct objects from images.

Nerf is the first 3D reconstruction that adapted deep learning in reconstructing objects from a few images after being trained with multi-view images of an object. It reconstructs the object as a neural field, not in the form of point clouds or meshes, thus artists cannot edit the shape of

the reconstructed object, and due to its limitation in real-time rendering, 3D Gaussian Splatting overcomes this issue.

In the 3D Gaussian Splatting reconstruction pipeline, the point clouds reconstructed are represented in 3D Gaussian, and cannot be edited directly, the visual output is rendered using rasterization, which is much faster than Nerf that using ray tracing rendering. In conclusion, 3D Gaussian splatting can support real-time rendering. However, these new representation-based methods do not support the post-refinement of the reconstructed object.

Although there are methods to convert this implicit representation into glb. mesh format, the post-manual refinement of the model by the artist using sculpting, and retopology is inevitable.

### 2.3 Generative AI

**Automatic 3D Shape Generation Model**

The rise of diffusion models, transformers and GANs accelerated the development of 3D shape generation tasks, including creating 3D characters or objects. For example, open-sourced 3D shape generation models such as Unique3D [18] and Hunyuan3D [19] can generate 3D mesh from a single image, and the quality is very high with less time required compared to 3D reconstruction methods. These models often utilize a multi-view synthesis diffusion model from a single image. Recently, [hunyuan3D studio] presents a platform that provides end-to-end pipelines in 3D assets generation such as automates part-level 3D shape generation, mesh polygon generation and semantic UV that is ready for production.

However, the final result cannot be controlled, and the quality will drop dramatically when reconstructing complex objects such as full-body characters. This may be due to the limited trained dataset. Besides, the size of a full-body character will be reduced when the mesh is reconstructed. The reconstructed mesh often has poor details in terms of soft edges and hard edges when reconstructing complex details character. This motivated the proposed method, which integrates segmentation when reconstructing the character, such that the artist can perform preprocessing in controlling the final output of the 3D model.

The figures (Figure 2.1 to 2.4) below show the review of closed model of hunyuan3D [31], which are Hunyuan3D 2.5 and Hunyuan 3D 3.0.

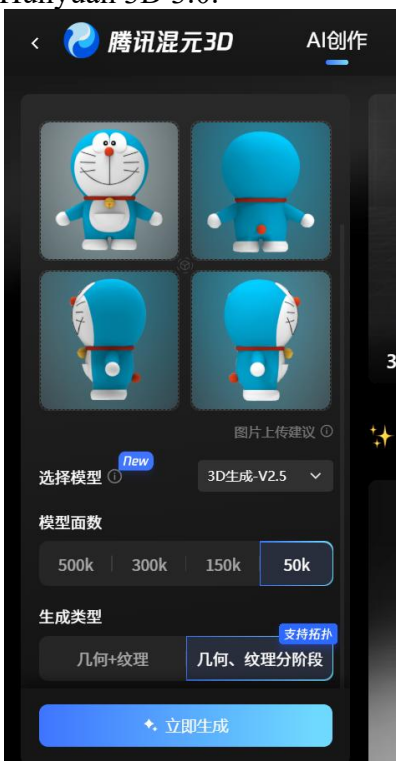



Figure 2.1: Showing the option of testing huyuan3D 2.5

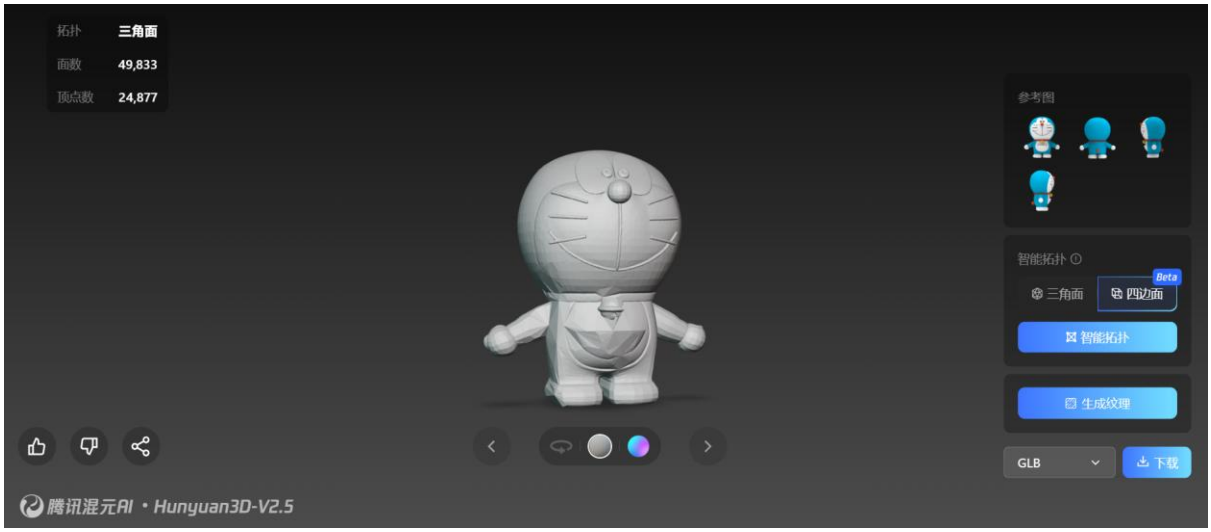



Figure 2.2: Showing the result of huyuan3D 2.5 Shape before retopology

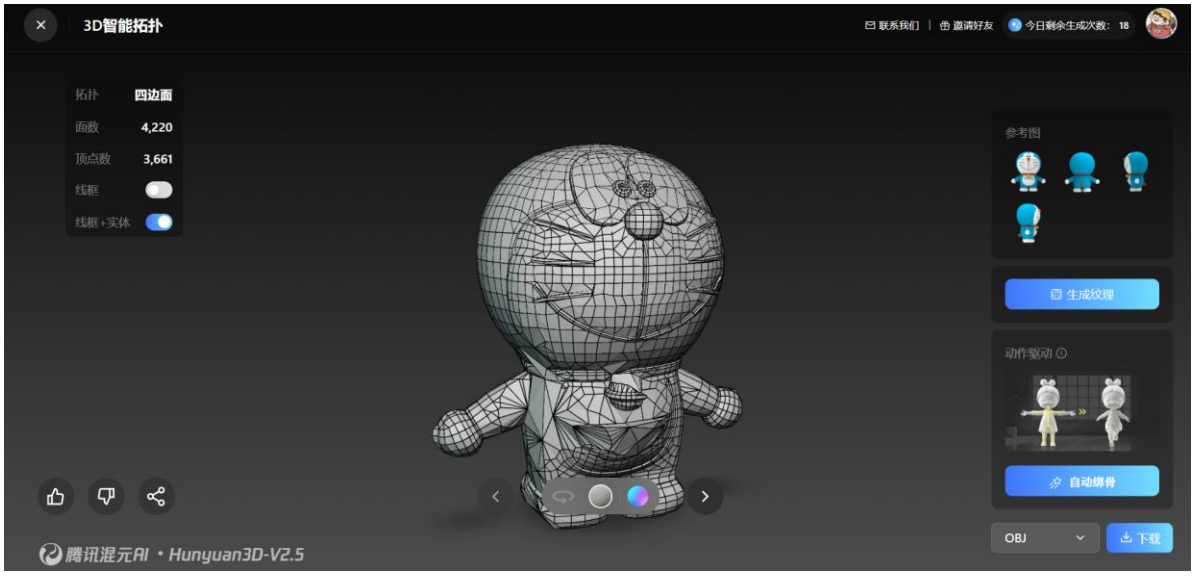



Figure 2.3: Showing the option of testing huyuan3D 2.5 Shape after quad retopology


Figure 2.4: Showing the option of testing huyuan3D 3.0 Shape after quad retopology from 500k faces

## 2.4 Other approaches

### Interactive system in 3D modeling

[14] propose a method that allows artists to manually annotate the orthogonal view image when modeling the characters with generative shapes. This can improve the workflow of modeling characters.

### Automated single view reconstruction with computer vision approach

[15] proposed a method to reconstruct a character from a single view image into the textured mesh. It adapts skeletonization [17] to reconstruct the skeleton of the character from the image input outline, and mesh is generated based on the skeleton, finally using a texturing algorithm to generate the final output. This method skips the need for a training model, but it performs badly when reconstructing characters with poses, covered by clothes, and with equipment.

## 2.5 Mind map of 3D Reconstruction Technologies

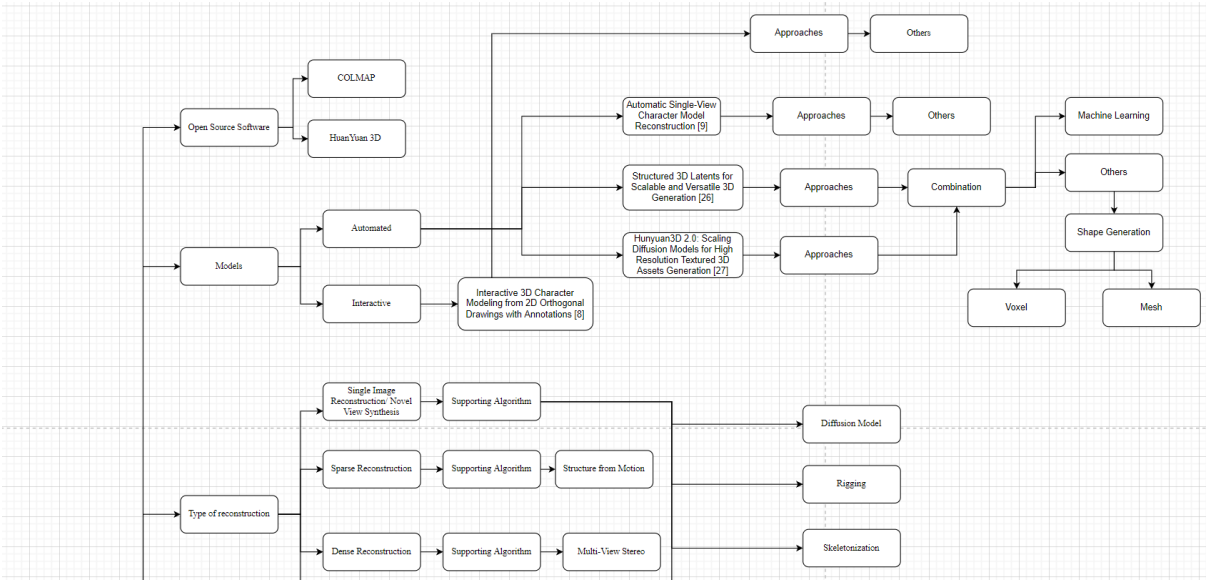



Figure 2.5: Research Mind Map part 1

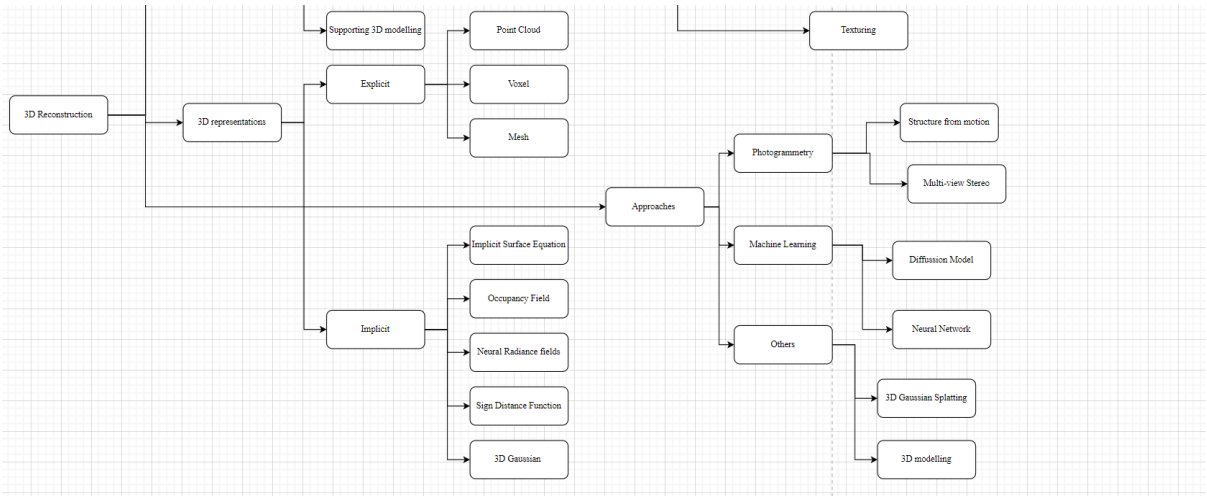



Figure 2.6: Research Mind Map part 2

Figures 2.5 and 2.6 show the 3D reconstruction techniques mind map.

## 2.6 Review of Related Work for Designing Algorithm OV-CV-BPY

This section presents a review of relevant literature that informed the design of the OV-CV-BPY algorithm, which addresses the challenge of 3D character reconstruction from orthogonal-view images. A detailed overview of the algorithm is provided in Chapter 3.1.

### 2.6.1 Segmentation

SAM 2 [25] is employed for interactive image segmentation. This pre-trained model plays a key role in the segmentation phase of the OV-CV-BPY pipeline, enabling precise region extraction from orthogonal views with minimal user input.

### 2.6.2 Blender Scripting

Blender's scripting capabilities [26] allow for the procedural generation of 3D shapes with clean topology using its Python API [27]. Built-in operations and modifiers support detailed geometry refinement, which is essential for skilled artists working on stylized character models. The Python API also empowers developers to contribute to the Blender ecosystem by creating add-ons and automation tools, helping the community evolve through continuous updates and enhancements.

## Chapter 3: Algorithm

This chapter discuss the problem formulation of the research question/problem. Next, OV-CV-BPY algorithm is introduced to reconstruct character's primitive shapes from four orthogonal view images.

### *3.1 Overview*

This section introduces how the 3D character reconstruction from orthogonal images problem is formulated.

1. *Let Character c is composition of "mesh" shapes: m1, m2, …, mn.*
2. *Hence,* $mesh = \{(x_n, y_n, z_n) \mid n = 1, 2, \ldots, N\}$ *where* $N$ *is total number of vertices*

The Figure 3.1 below is the illustration of a Doraemon character composited by meshes and the highlighted head mesh with naming m8.

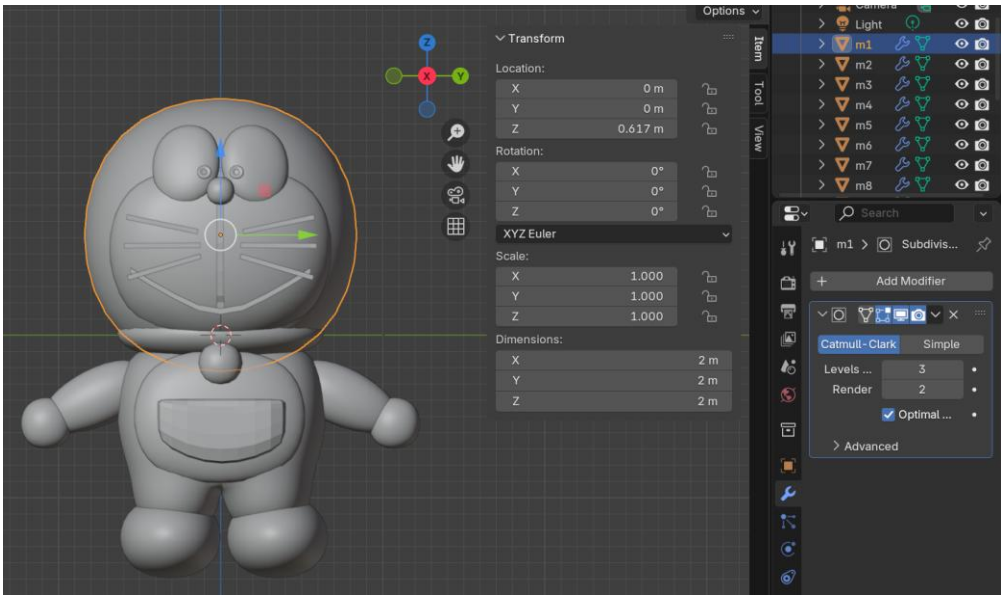



Figure 3.1: Highlighted Head Mesh of Doraemon

### 3.1.1 OV-CV-BPY

This section provides an overview of the OV-CV-BPY algorithm, which is designed to reconstruct part-level 3D primitive cubes from four orthogonal-view images: front, left, back, and right. These input images are assumed to have a 1:1 width-to-height ratio.

The algorithm targets the reconstruction of stylized character shapes using primitive geometry, serving as a foundation for box modeling workflows commonly used in character design. It is composed of four main stages: **Image Preprocessing, 2D Mesh Polygon Mask Segmentation, 3D Feature Reconstruction, BPY 3D Mesh Generation**.

The input to the algorithm is a set of four orthogonal-view images. The final output is a Blender file containing the reconstructed character meshes, ready for further refinement and detailing.

The Figure 3.2 below shows the block diagram of the OV-CV-BPY algorithm.

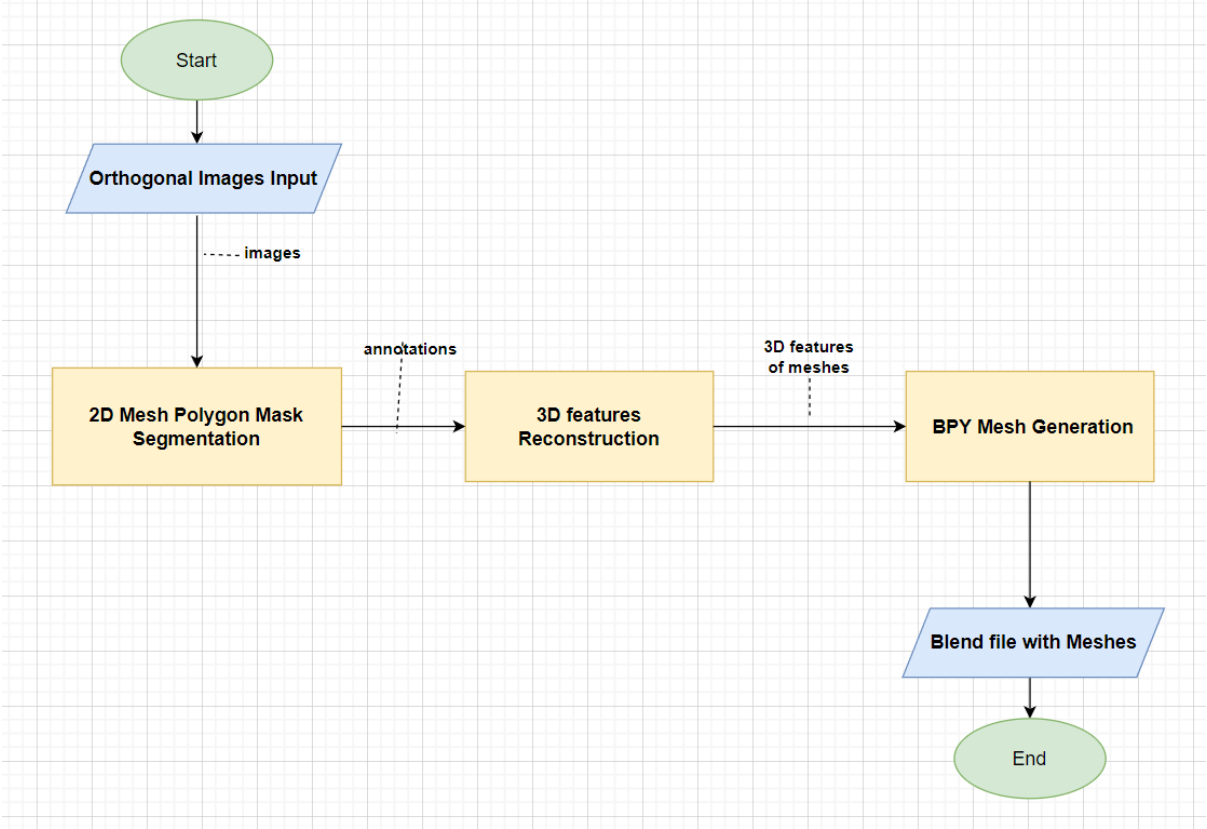



Figure 3.2: 3D Reconstruction System with MV-CV-BPY Algorithm Overview Block Diagram

### 3.1.2 2D Mesh Polygon Mask Segmentation

This section describes how the mesh 2D polygon mask segmentation step of the OV-CV-BPY algorithm being implemented via CVAT [28] tool to solve the question. The Figure 3.3 below shows the required input of the step and the output of the step implementation.

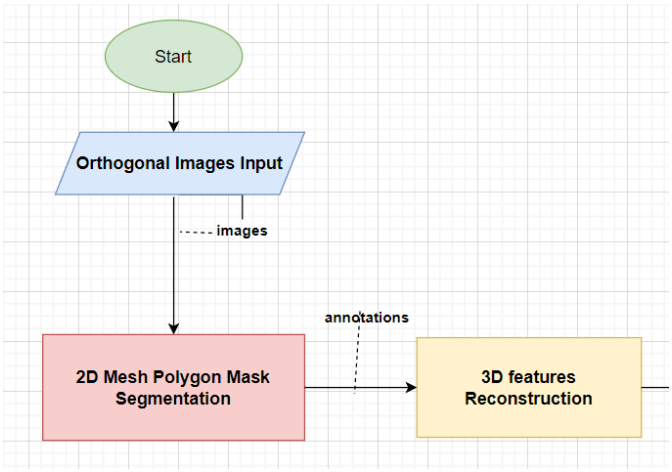



Figure 3.3: Illustration of Input, Process and Output for Polygon Mask Segmentation

After the input images are resized, the next step in the pipeline is segmentation. There are three common approaches to segmentation: automatic, manual, and semi-automatic. This project adopts a semi-automatic, interactive method known as Interactive Shape Segmentation, which leverages the AI-powered segmentation tool in CVAT using the SAM 2 model [25].

The goal of this step is to segment and annotate the individual parts of the character from each orthogonal view. The output is a COCO-format annotation JSON file containing structured data for each character part. These parts—referred to as “objects” in segmentation terminology—are labeled with the class name mesh, as introduced in Section 3.1. This labeling enables consistent mapping of mesh objects across orthogonal views, as discussed in Section 3.2.3. The implementation details are provided in Section 4.2.

### 3.1.3 3D Feature Reconstruction

This section introduces the 3D Feature Reconstruction block, which comprises a set of procedures that transform mesh object annotations—obtained from the 2D Mesh Polygon Mask Segmentation step (Section 3.2.2)—into 3D axis-aligned bounding boxes (AABBs). Each AABB encodes the spatial dimensions and position of a character's part mesh, serving as a geometric approximation of its volume.

The result of this object mapping process is a structured collection of mesh data, which acts as the input for generating primitive subdivided cubes in the Blender scripting block. The Figure 3.4 below illustrates the object mapping workflow within the overall algorithm architecture

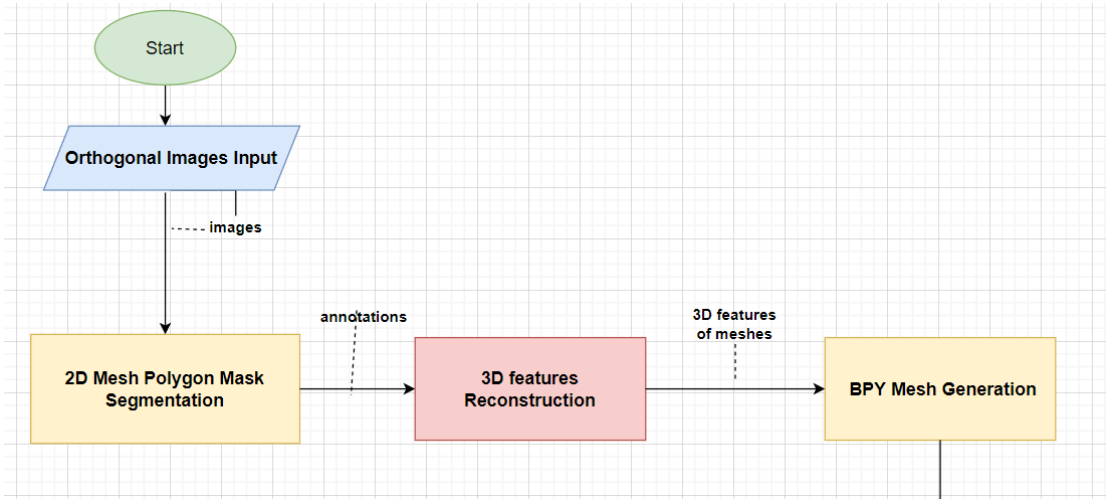



Figure 3.4: 3D Features Reconstruction Block

The output of this block is the mesh 3D features. It stores a pair of minimum 3D coordinate and maximum 3D coordinate. The data serves as the input of BPY 3D Mesh Generation (Sec. 3.2.4) block.

Numbered list below is the breakdown of 3D shape feature reconstruction block:

1. ***Preprocess and groups polygon mask annotations of mesh parts***
2. ***Normalize mesh parts coordinates***
3. ***Reconstructs 3D AABB***

### Polygon mask annotations preprocessing

This step organizes the polygon masks of character mesh parts by grouping annotations with the same name across different orthogonal views. For each mesh object, the segmentation coordinates from all available views are collected and stored. For example, if a mesh named "head" is segmented in the front, left, back, and right views, the system will associate four distinct segmentations with the "head" mesh object.

### Coordinates Normalization

Following annotation preprocessing, the next step is to normalize the image coordinates within each segmentation. This involves mapping pixel-based image coordinates to a normalized Cartesian coordinate system specific to each orthogonal view. Figures 3.5 to 3.8 below illustrates how pixel coordinates are transformed into orthogonal view-aligned Cartesian coordinates, enabling consistent spatial interpretation across views.

Front View:

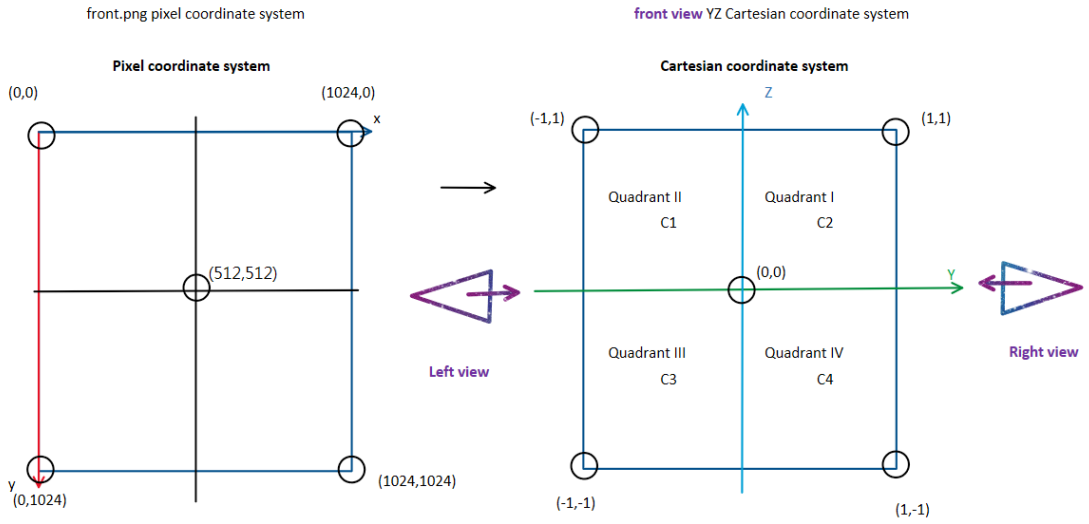



Figure 3.5: Illustration of Front View Image Coordinate Normalization

Left View:

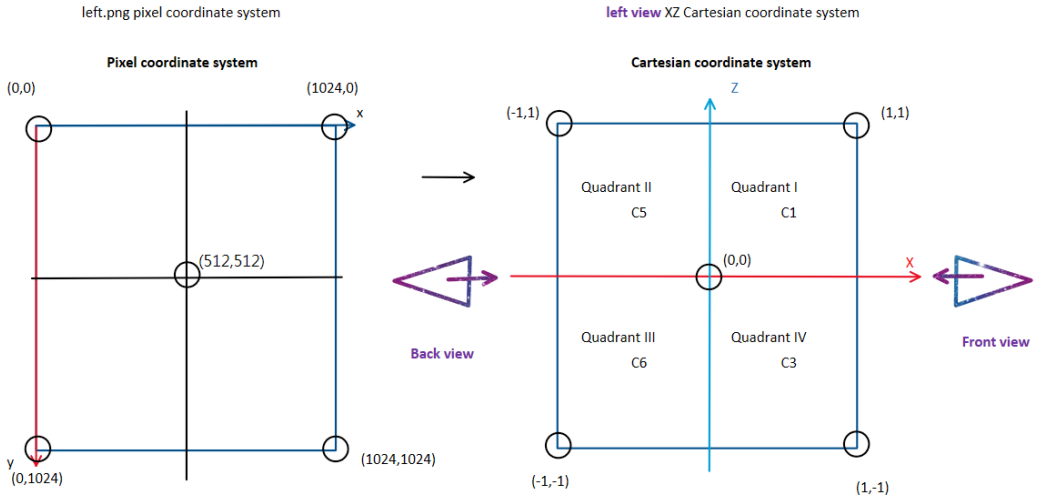



Figure 3.6: Illustration of Left View Image Coordinate Normalization

Back View

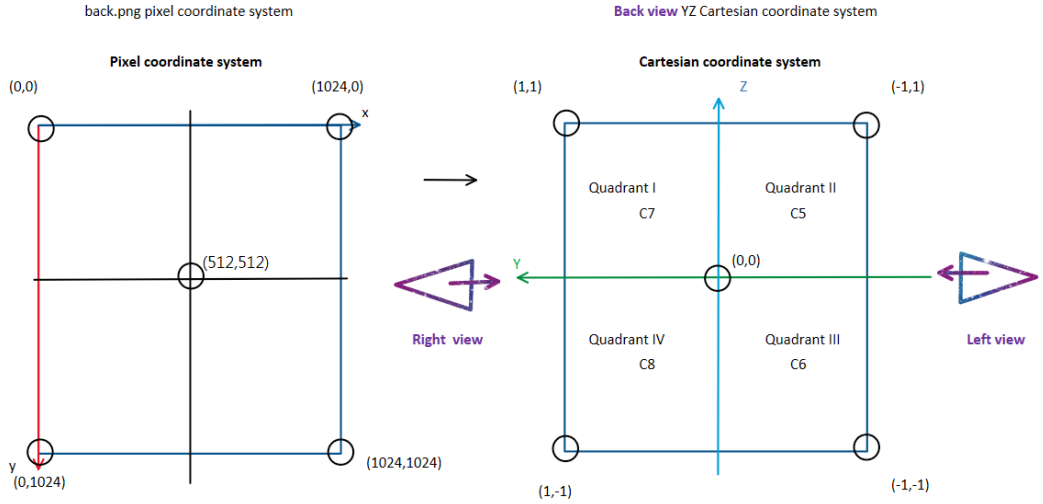



Figure 3.7: Illustration of Back View Image Coordinate Normalization

Right View

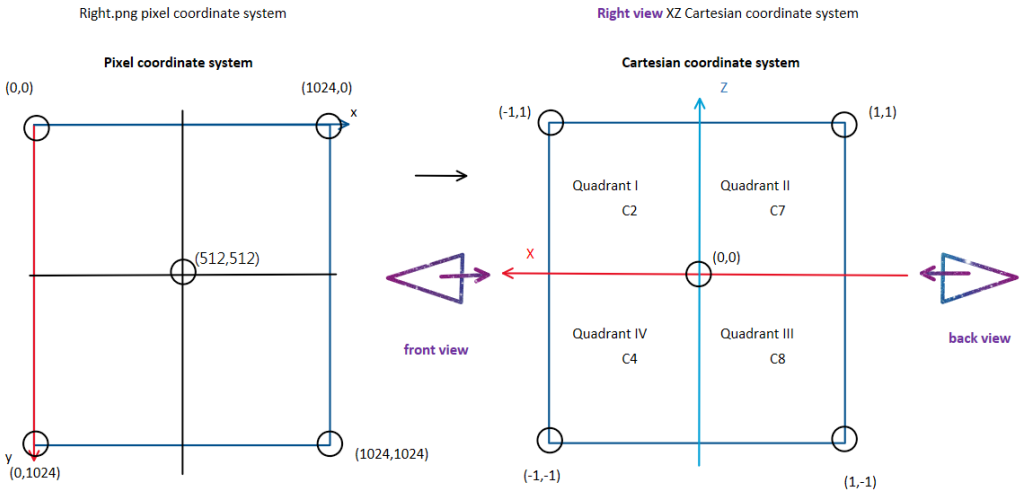



Figure 3.8: Illustration of Right View Image Coordinate Normalization

Formula 2 defines the method for normalizing image coordinates into Cartesian coordinates aligned with orthogonal views. This transformation converts the pixel-based segmentation coordinates of mesh parts into a consistent spatial representation across the front, left, back, and right views. The normalization assumes that all input images share equal width and height dimensions.

To achieve this, Formula 2 adapts the OpenGL viewport transformation (originally presented in Formula 1 [29]), which is commonly used to convert vertex positions from normalized device coordinates (NDC) to window space. By repurposing this transformation, the algorithm ensures that image coordinates are accurately mapped into orthogonal Cartesian space for further 3D reconstruction.

Formula 1:

$$x_w = \frac{width}{2} * x_{ndc} + x + \frac{width}{2} \quad \textit{Formula 1}$$

Where $x_w$ is $x$ in window space, $x_{ndc}$ is $x$ vertex in NDC space. Since the *width* and *height* of source images are in 1:1 ration in this algorithm (Sec. 3.1) The section below describes the adaption of Formula 1 in formulating equation 2 and 3.

Defining Formula 2 - 5:

**For front and back views**,

Let:

$x_{ndc} = y_{cart}$, *the first number of a coordinate of a mesh in yz Cartesian coordinate system.*

$x_w = y_{pixel}$, *the first value of an image coordinate of a mesh segmentation mask in image.*

$x_{ndc} = z_{cart}$, *the second number of a coordinate of a mesh in yz Cartesian coordinate system.*

$x_w = z_{pixel}$ *the second value of an image coordinate of a mesh segmentation mask in image.*

$pixel_{yz}$ = *Segmentation of a mesh in front and back view is a list of number with N size*

$cart_{yz}$ = *yz coordinates of a mesh in front and back view is a list of number with N size.*

Subject to:

$$pixel_{yz} = \{(y_{pixel\,i}, z_{pixel\,i}) \mid i = 1, 2, \ldots, N\} \quad \textit{Formula 2}$$

$$cart_{yz} = \{(y_{cart\,i}, z_{cart\,i}) \mid i = 1, 2, \ldots, N\} \quad \textit{Formula 3}$$

**For left and right view,**

Let:

$x_{ndc}$ = $x_{cart}$, *the first number of a coordinate of a mesh in* $xz$ *Cartesian coordinate system.*

$x_w$ = $x_{pixel}$, *the first value of an image coordinate of a mesh segmentation mask in image.*

$x_{ndc}$ = $z_{cart}$, *the second number of a coordinate of a mesh in* $xz$ *Cartesian coordinate system.*

$x_w$ = $z_{pixel}$ *the second value of an image coordinate of a mesh segmentation mask in image.*

$pixel_{xz}$ = *Segmentation of a mesh in front and back view is a list of number with N size*

$cart_{xz}$ = *xz coordinates of a mesh in front and back view is a list of number with N size.*

Subject to:

$$pixel_{xz} = \{(x_{pixel\,i}, z_{pixel\,i}) \mid i = 1, 2, \ldots, N\} \qquad \textit{Formula 4}$$

$$cart_{xz} = \{(x_{cart\,i}, z_{cart\,i}) \mid i = 1, 2, \ldots, N\} \qquad \textit{Formula 5}$$

Hence,

For Formula 6 and 7

Let:

$H_{cart}$ = *horizontal component in Cartesian coordinate system*

$V_{cart}$ = *vertical component in Cartesian coordinate system*

$W$ = *width or height or a square image*

$f$ = *front view image*

$l$ = *left view image*

$b$ = *back view image*

$r$ = *right view image*

Subject to:

$$H_{cart} = \begin{cases} \frac{2\,H_{pixel}}{W} - 1, \ when\ H_{pixel}\ is\ from\ f\ or\ l, with \begin{cases} H = y\ for\ f \\ H = x\ for\ l \end{cases} \\ \left(\frac{2\,H_{pixel}}{W} - 1\right) * -1, when\ H_{pixel}\ is\ from\ f\ or\ l, with \begin{cases} H = y\ for\ b \\ H = x\ for\ r \end{cases} \end{cases}$$

Formula 6

$$V_{cart} = 1 - \frac{2\,V_{pixel}}{W}\ ,\ when\ V_{pixel}\ is\ from\ f, l, b, r \qquad \text{Formula 7}$$

**3D AABB Reconstruction**

This step focuses on reconstructing the 3D axis-aligned bounding boxes (AABBs) of character mesh objects using their Cartesian coordinates extracted from orthogonal views. An AABB defines the spatial extent of a 3D mesh by encapsulating it within a box aligned to the coordinate axes. This geometric feature is essential for generating primitive cubes that represent individual character parts.

The reconstruction process relies on identifying a pair of 3D points: the minimum and maximum coordinates along each axis. These two points define the bounds of the mesh object. The equations for this reconstruction are presented in this section. The Figure 3.9 below illustrates the bounding box defined by these two points, highlighted with blue annotations.

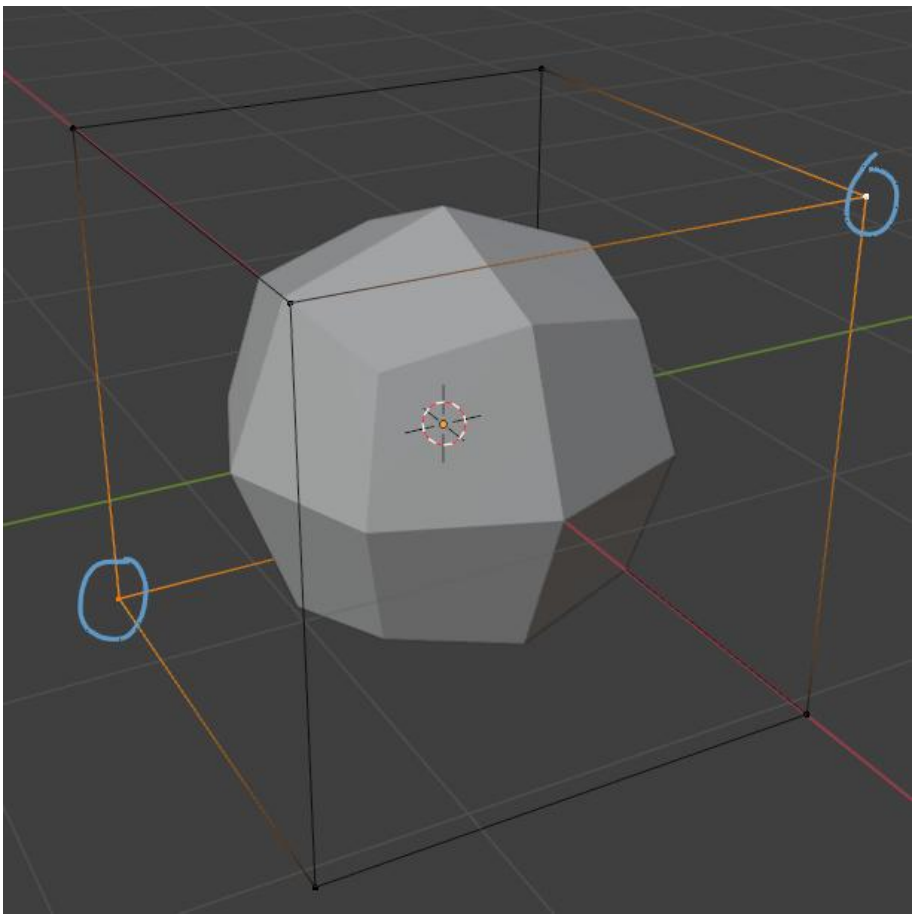


Figure 3.9: Illustration of minimum and maximum points

Formula 8 to 14,

Let:

$N$ = *total numbers of 2D orthogonal view cartesian coordinates of a mesh*

*mesh = a set of 3D coordinates or vertices of a character's part.*

$x_{min}$ = *smallest* $x_n$ *in set of N x coordinates*

$x_{max}$ = *largest* $x_n$ *in set of N x coordinates*

$y_{min}$ = *smallest* $y_n$ *in set of N y coordinates*

$y_{max}$ = *largest* $y_n$ *in set of N y coordinates*

$z_{min}$ = *smallest* $z_n$ *in set of N z coordinates*

$z_{max}$ = *largest* $z_n$ *in set of N z coordinates*

$AABB_{mesh}$ = *represented by a pair of minimum and maximum 3D coordinates of a mesh*

Subject to:

$$x_{min} = \min_{1 \le n \le N}(x_n) \quad \textit{Formula 8}$$

$$x_{max} = \max_{1 \le n \le N}(x_n) \quad \textit{Formula 9}$$

$$y_{min} = \min_{1 \le n \le N}(y_n) \quad \textit{Formula 10}$$

$$y_{max} = \max_{1 \le n \le N}(y_n) \quad \textit{Formula 11}$$

$$z_{min} = \min_{1 \le n \le N}(z_n) \quad \textit{Formula 12}$$

$$z_{max} = \max_{1 \le z \le N}(z_n) \quad \textit{Formula 13}$$

$$AABB_{mesh} = \{(x_{min}, y_{min}, z_{min}), (x_{max}, y_{max}, z_{max})\} \quad \textit{Formula 14}$$

### 3.1.4 BPY 3D Mesh Generation

This section outlines the formulation for generating primitive cubes for each mesh object, based on the minimum and maximum 3D coordinates obtained through reconstruction (see Formula 14). These coordinates define the axis-aligned bounding box (AABB) for each mesh part, which serves as the geometric basis for cube generation.

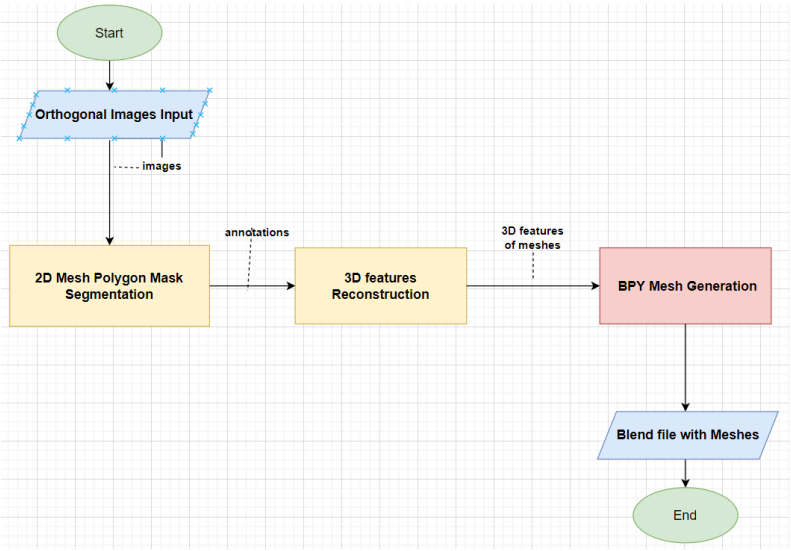



Figure 3.10: BPY Mesh Generation Block

Figure 3.10 illustrates the BPY 3D Mesh Generation block, which is the final stage of the OV-CV-BPY algorithm. In this stage, the reconstructed 3D features are translated into primitive cubes using Blender's Python scripting interface. The implementation details of this process are discussed in Chapter 4.

## Chapter 4: Experiment and Implementation

This section presents the implementation details of the algorithm described in Section 3.2, demonstrated through an example of reconstructing a 3D mesh of the character Doraemon using orthogonal-view images. The orthogonal concept art used in this demonstration was created by artist Ashutosh Kadam and sourced from his published artwork [25], as shown in Figure 4.1.

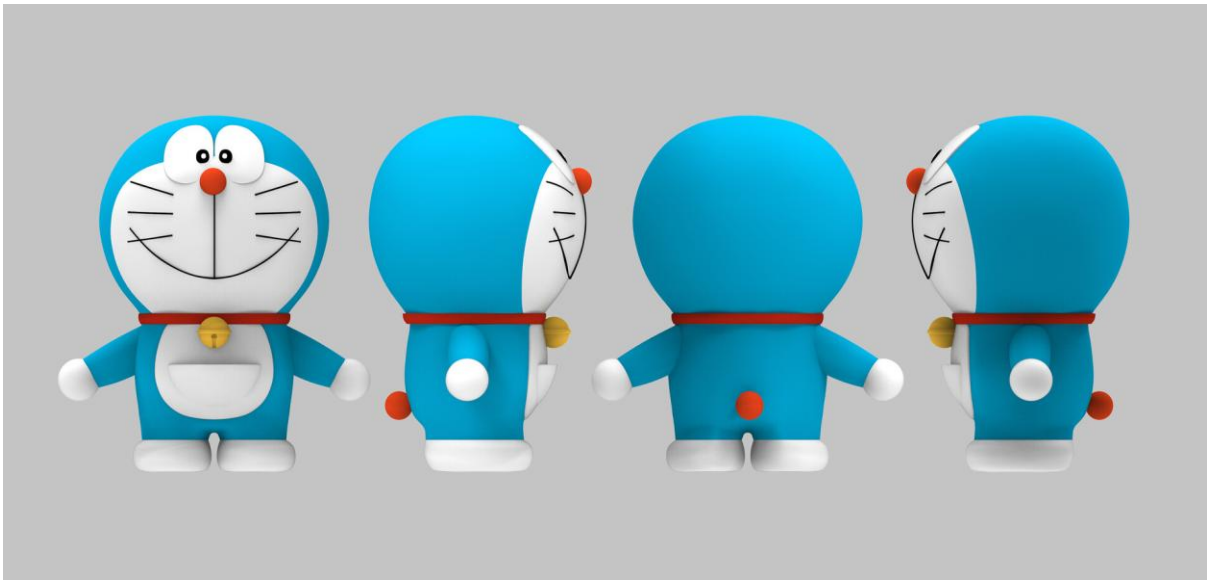


Figure 4.1: Doraemon Orthogonal View Image Reference by Ashutosh Kadam

The Figure 4.2 below illustrates the input and output of OV-CV-BPY modules to reconstructs Doraemon in pseudocode.

**Algorithm 1 3D_Reconstruction using OV_CV_BPY Overview**
Requires: Orthogonal images of target character

Step 1: Orthogonal Images Input Processing (Chapter 4.1)
- output: 608*608 size images

Step 2: 2D Mesh Polygon Mask Segmentation (Chapter 4.2)
- output: Json file contains annotations of Polygon Mask

Step 3: 3D feature Reconstruction (Chapter 4.3):
- output: Json file contains 3D AABB points of meshes

Step 4: BPY 3D Mesh Generation (Chapter 4.4):
- output: Blender file contains character primitive cubes of meshes

Return: Blender file contains target character meshes

Figure 4.2: OV-CV-BPY Algorithm Pseudocode

### *4.1 Input Preprocessing*

The orthogonal-view image of Doraemon was cropped into four separate images—one for each view (front, left, back, and right)—with the background removed. This preprocessing was performed using the online photo editor Photopea, which supports layer-based editing and transparent backgrounds. A sample screenshot of the editing process is shown in Figure 4.3.

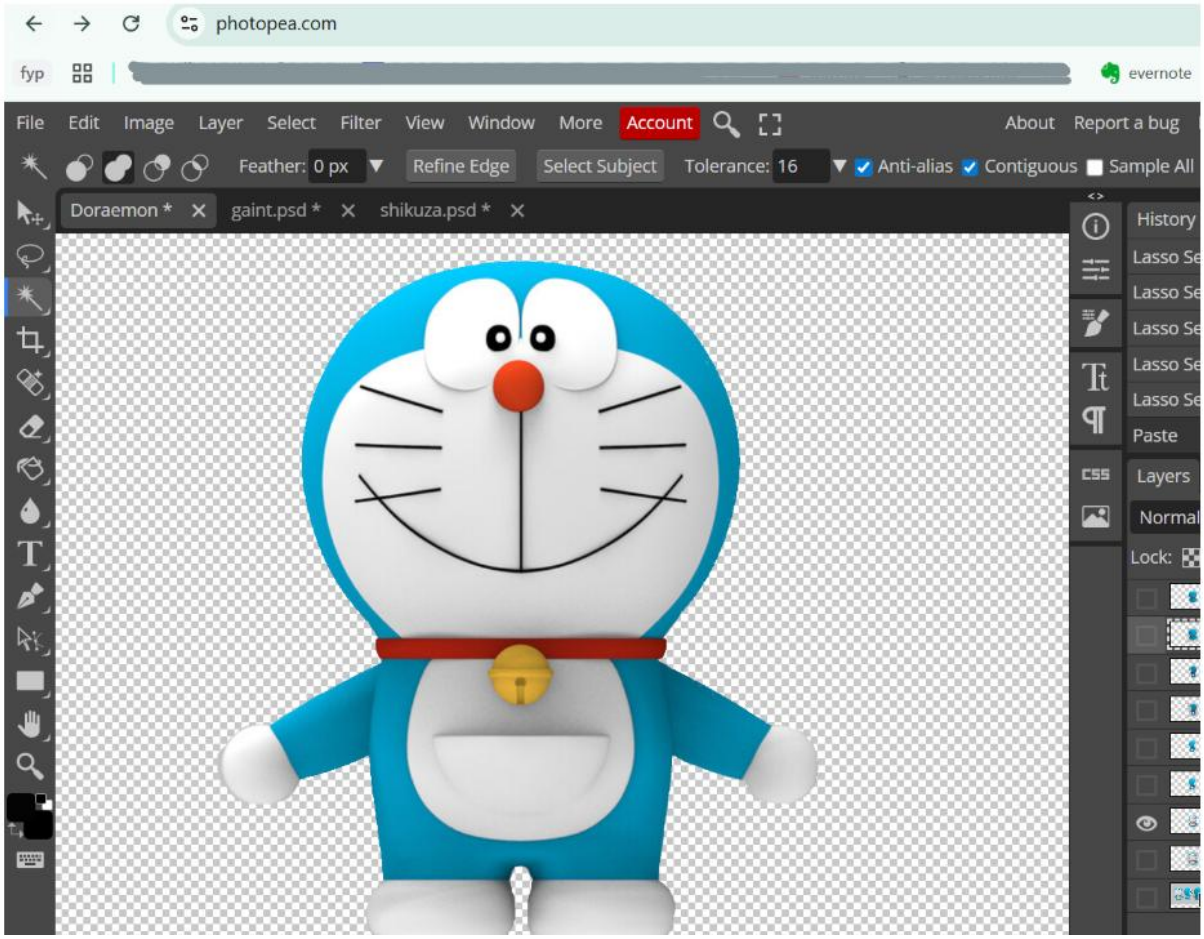



Figure 4.3: Screenshot of editing image

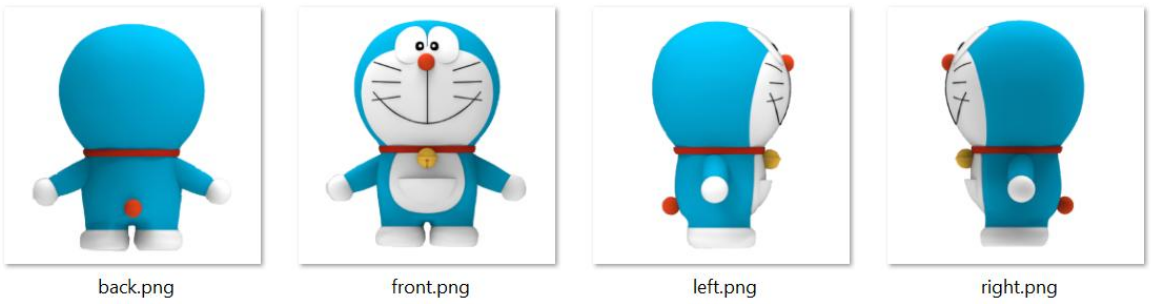



Figure 4.4: Four Orthogonal Views of Doraemon

A simple Streamlit application [30] was developed to provide a graphical user interface (GUI) for uploading and resizing orthogonal-view images. This tool streamlines the preprocessing step by allowing users to interactively adjust image dimensions before segmentation. Figure 4.4 displays a screenshot of the resized Doraemon images across the four orthogonal views. Figure 4.5 presents the resized front view image (front view.png) of Doraemon.

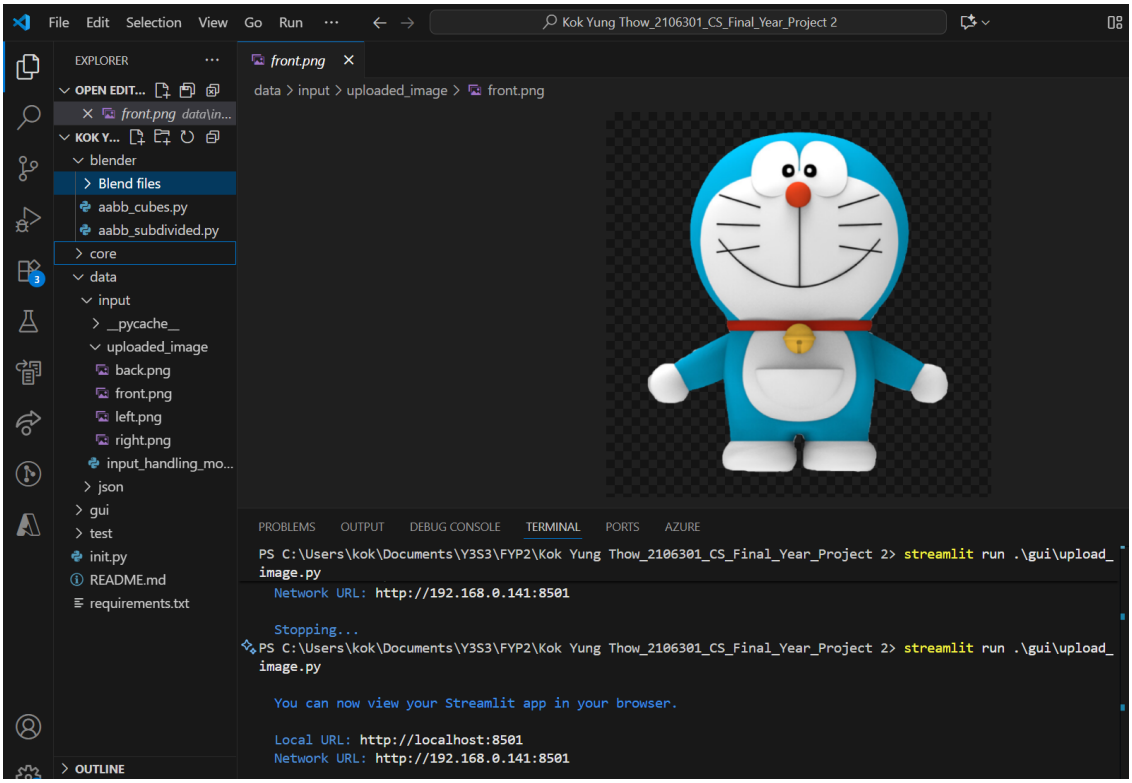



Figure 4.5: Illustration of Viewing Resized Image

### *4.2 Polygon Mask Segmentation*

This section demonstrates the process of segmenting parts of Doraemon using CVAT. Only one label class named “mesh” with attributes “name” and “view” used when implementing interactive 2D polygon masks segmentation with CVAT. Figure 4.6 shows a screenshot of the segmentation interface in the CVAT workspace, highlighting the annotated front view of Doraemon.

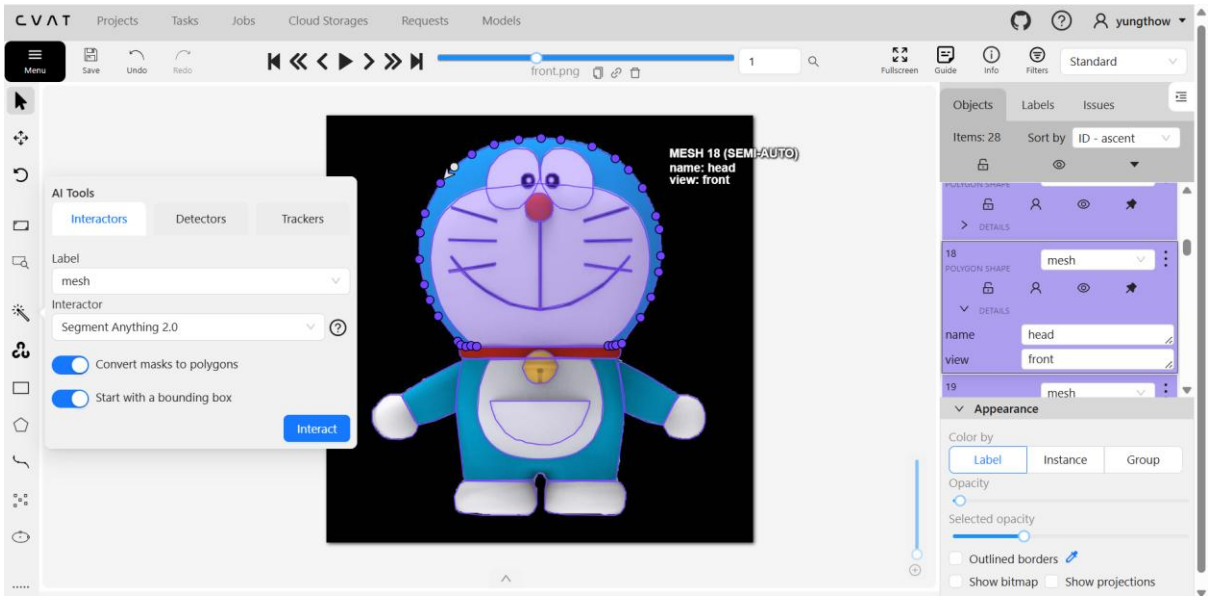



Figure 4.6: Screenshot of Segmenting Polygon Mask of Meshes

Figure 4.6 shows a highlighted polygon region—formally referred to as an object—assigned the label class **"**mesh" with an ID of 18. The associated attributes, **“**name” and “view” are used to track the shape of *Doraemon* across orthogonal perspectives. These attributes play a crucial role in constructing the final annotation JSON file, which serves as input to the **3D feature reconstruction** module (see Section 3.1.3).

In this example, Figure 4.6 illustrates the object labelled with **name = "head"** and **view = "front"**. Figure 4.7 also shows the CVAT workspace during the export process, where the annotated dataset is saved in **COCO 1.0 format** under the filename *demo_doraemon.zip*. After downloading and extracting the dataset, Figure 4.8 presents the file explorer view of the exported JSON file. Finally, Figure 4.10 demonstrate how an individual object from the JSON file in Figure 4.9 can be visualized by uploading it to ChatGPT for inspection

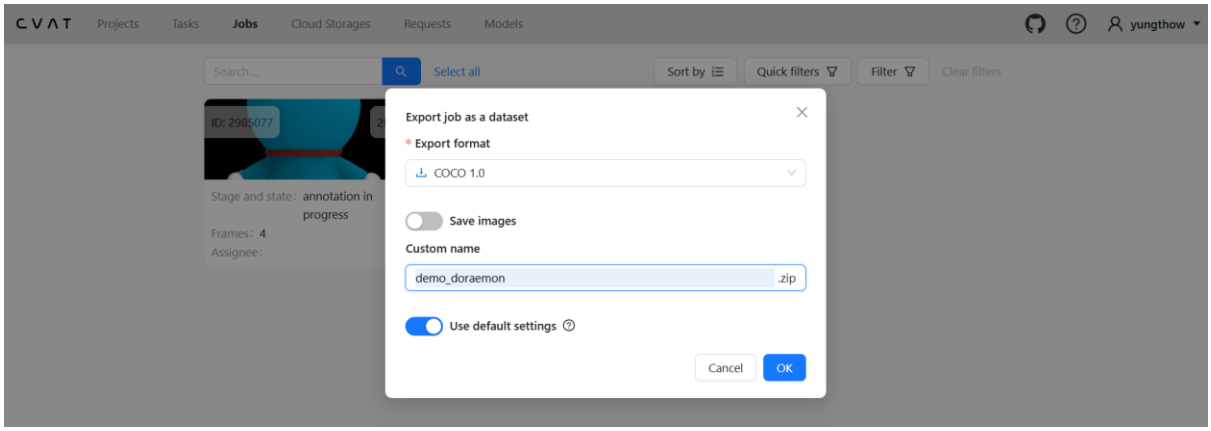



Figure 4.7: Screenshot of Exporting Annotations

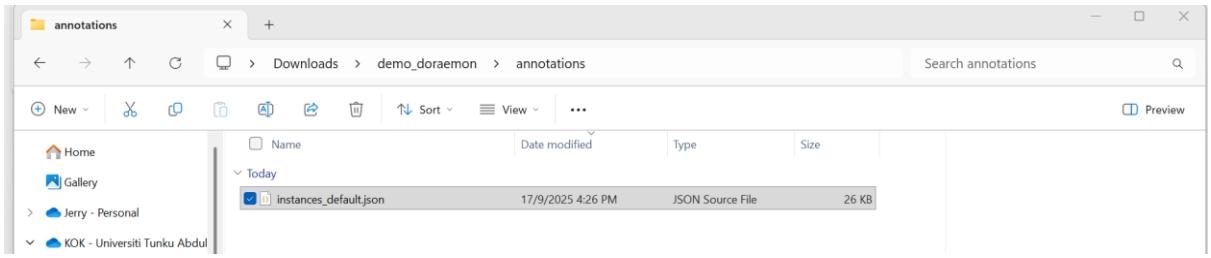



Figure 4.8: Screenshot of Viewing Annotations JSON file

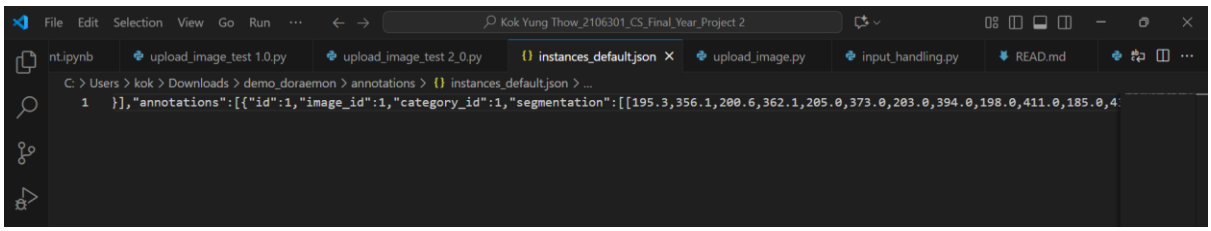



Figure 4.9: Screenshot of Annotations List in JSON file

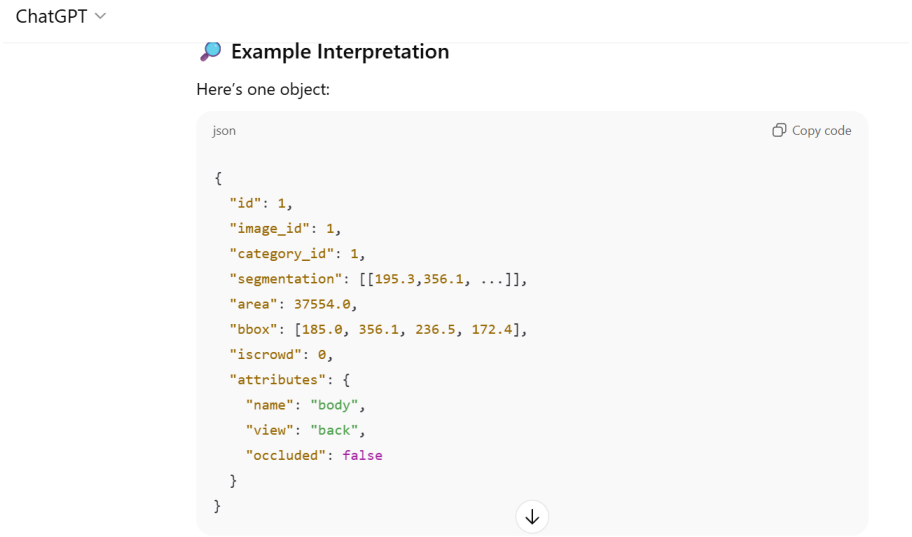



Figure 4.10: Illustration of Annotations List Format

### *4.3 3D Feature Reconstruction*

#### 4.3.1 Mask Annotations Preprocessing

The Figure 4.11 below shows the pseudocode for preprocess annotations to track unique mesh objects from views and store the corresponding segmentations:

```
Annotations Preprocessing
Requires: 2D polygon mask of mesh annotation json file
    Initialize a mesh dictionary such that for each mesh name, store:
      A set of unique views
      A dictionary of segmentations per view
      A placeholder for AABB (min/max coordinates)
    Load JSON data from the given path
    Extract list of annotations from the JSON
    For each annotation entry:
      Extract mesh name, view label, and segmentation data
      IF mesh name is missing: Skip this entry
      IF view already exists for this mesh: Skip to avoid duplicates
      ELSE:
        Add view to mesh_dictionary[name]["views"]
        Append segmentation data to mesh_dict[name]["segmentations"][view]
Return the populated mesh_dictionary
```

Figure 4.11: Pseudocode of Annotations Preprocessing

### 4.3.2 Coordinates Normalization

The Figure 4.12 below shows the pseudocode of preprocess segmentations' function: map pixel coordinates of segmentation to cartesian coordinates:

```
Preprocess Segmentations
Requires: mesh_dict, img_width:
  1. Define a mapping of view labels to transformation functions:
    - "front" and "left" → use front-left mapping function
    - "back" and "right" → use back-right mapping function
    - Each function receives img_width as a parameter
  2. FOR each mesh in mesh_dict:
    a. FOR each view and its segmentations:
      i. IF view has a corresponding transformation function:
        - Apply the transformation to the segmentations
        - Update mesh_dict with transformed segmentations
      ii. ELSE:
        - Print warning that no mapping function exists for this view
  Returns: updated mesh_dict
```

Figure 4.12: Pseudocode of Coordinates Normalization

### 4.3.3 3D Feature Reconstruction

Figure 4.13 below shows the pseudocode that reconstructs 3D feature of mesh from Cartesian coordinates of mesh.

**Reconstructs** $\boldsymbol{AABB_{mesh}}$

Requires: mesh dictionary

Strat:

For each mesh in mesh dictionary:

Identify the total number of orthogonal view coordinates set of a mesh

Group y components of first index of front and back views coordinates set

Group x components of first index of left and right views coordinates set

Group z components of second index of all views coordinate set

Compute $AABB_{mesh} = (x_{min},\ y_{min}, z_{min}), (x_{max},\ y_{max}, z_{max})$

Returns: $x_{min},\ y_{min}, z_{min}, x_{max},\ y_{max}, z_{max}$

Figure 4.13: Pseudocode of 3D Feature Reconstruction

## *4.4 Mesh Cubes Generation*

Figure 4.14 below shows the pseudocode that generate primitive cubes for mesh AABB data.

***Generates Mesh Cubes***

Requires: JSON file with AABB data of meshes

Start:

Load AABB data from JSON file

Clean all objects in Blender workspace

For each of the mesh:

Extract - $x_{min},\ y_{min}, z_{min}, x_{max},\ y_{max}, z_{max}$

Calculate center point of AABB

Calculate size of AABB

Add a cube at origin

Rename cube using mesh name

Scale cube to match AABB size

Move cube to AABB center

Return: Save blend file that contains the meshes

Figure 4.14: Pseudocode of Mesh Cubes Generation

## Chapter 5: Results and Discussion

### *5.1 Experimental Results*

#### 5.1.1 Coordinates Normalization

Figures 5.1 to 5.4 below show the Doraemon plots from each orthogonal view.

Front view:

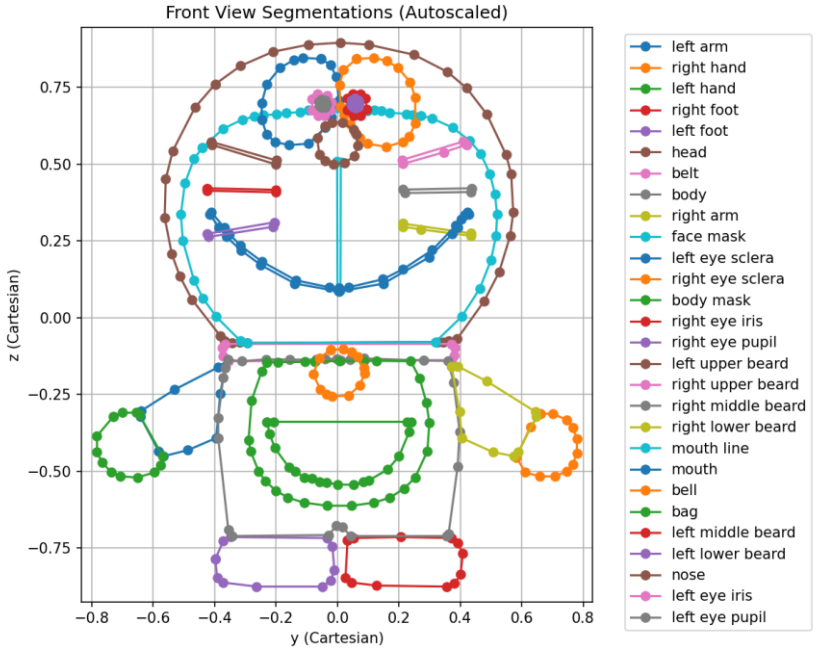



Figure 5.1: Front view of Doraemon plot from normalized pixel coordinates of segmentation

Left View:

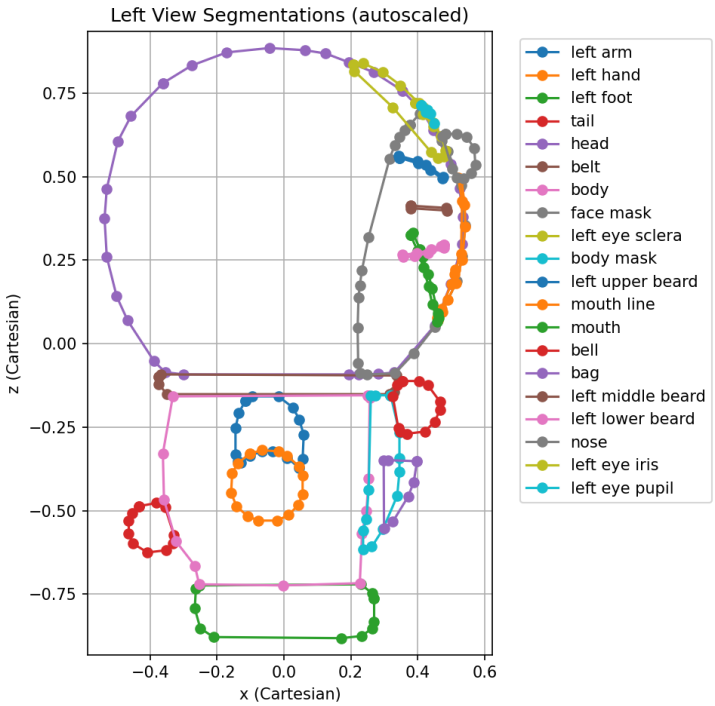



Figure 5.2: Left view of Doraemon plot from normalized pixel coordinates of segmentation

Back View:

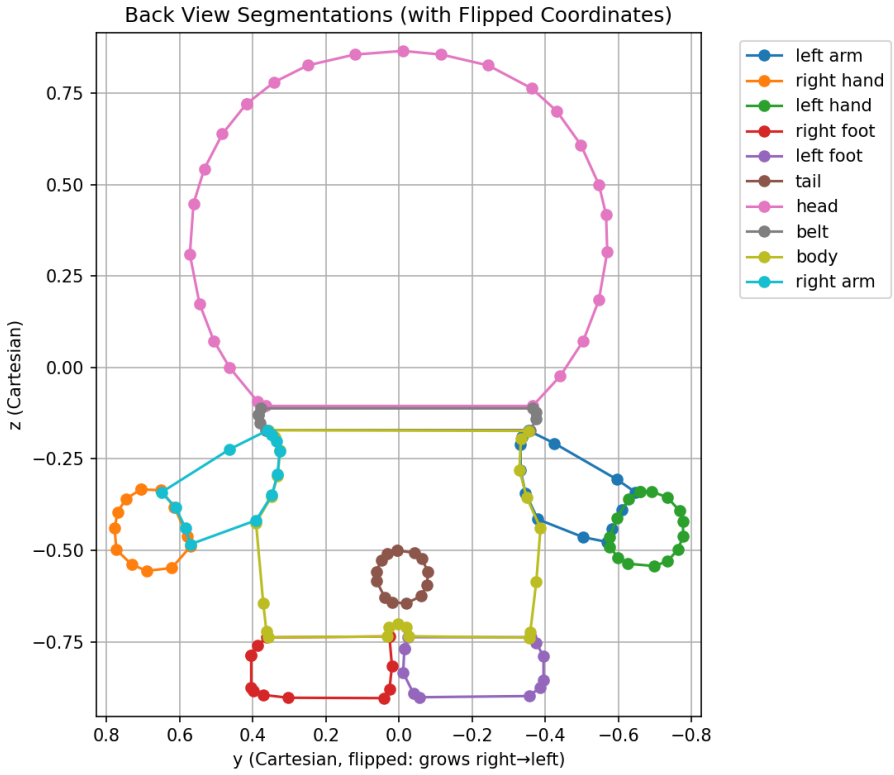



Figure 5.3 Back view of Doraemon plot from normalized pixel coordinates of segmentation

Right View

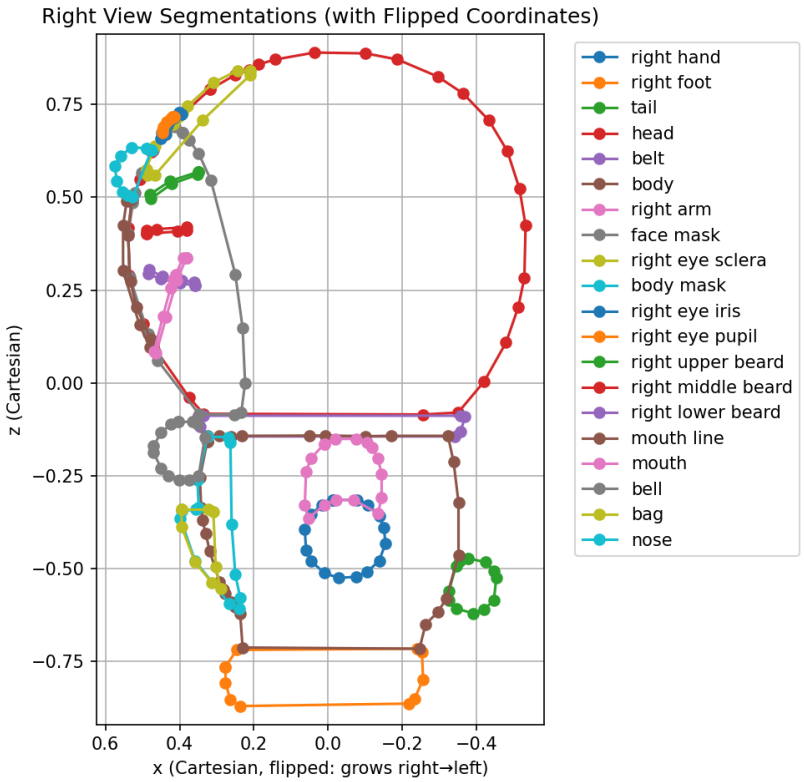



Figure 5.4: Right view of Doraemon plot from normalized pixel coordinates of segmentation

The plots shown in Figures 5.1 to 5.4 demonstrate the effectiveness of the coordinate normalization process, which successfully converts image-based segmentation coordinates into their corresponding Cartesian coordinates for each orthogonal view.

### 5.1.2 3D AABB Reconstruction

Figures 5.5 to 5.8 show the experimental results of reconstructing AABB of the meshes

Front view

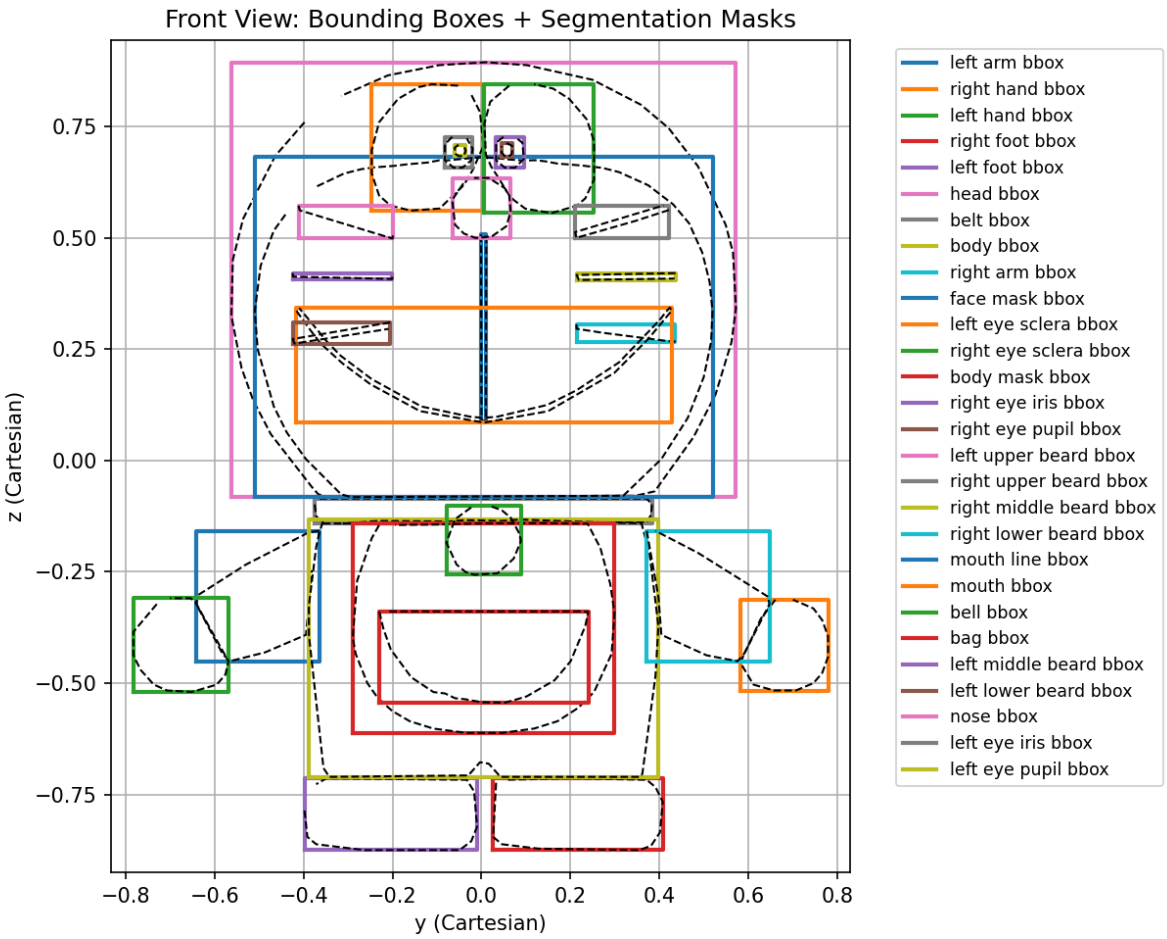



Figure 5.5: Front view of Doraemon Plot to Illustrate Bounding Boxes

Left view

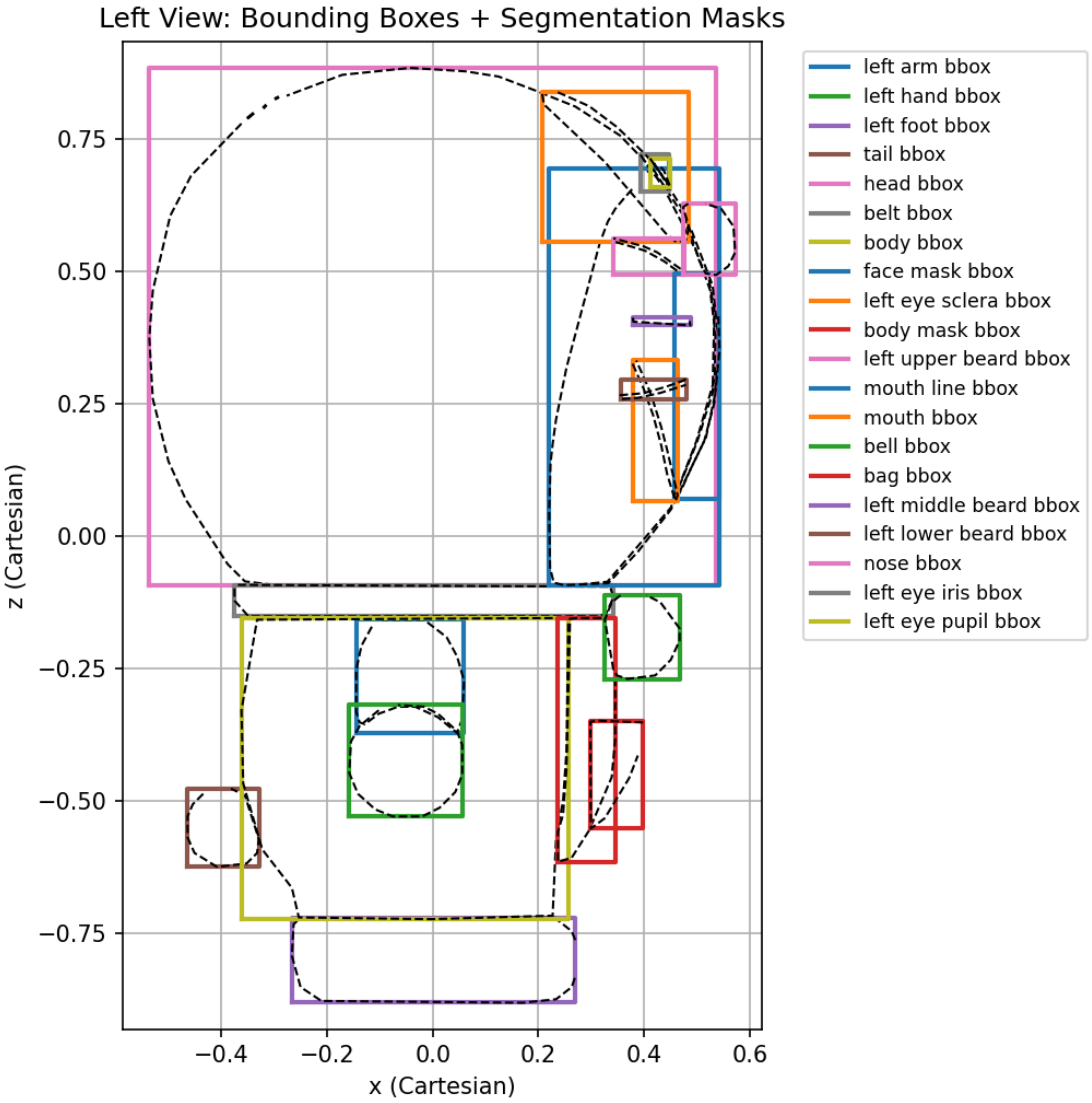



Figure 5.6: Left view of Doraemon Plot to Illustrate Bounding Boxes

Back view

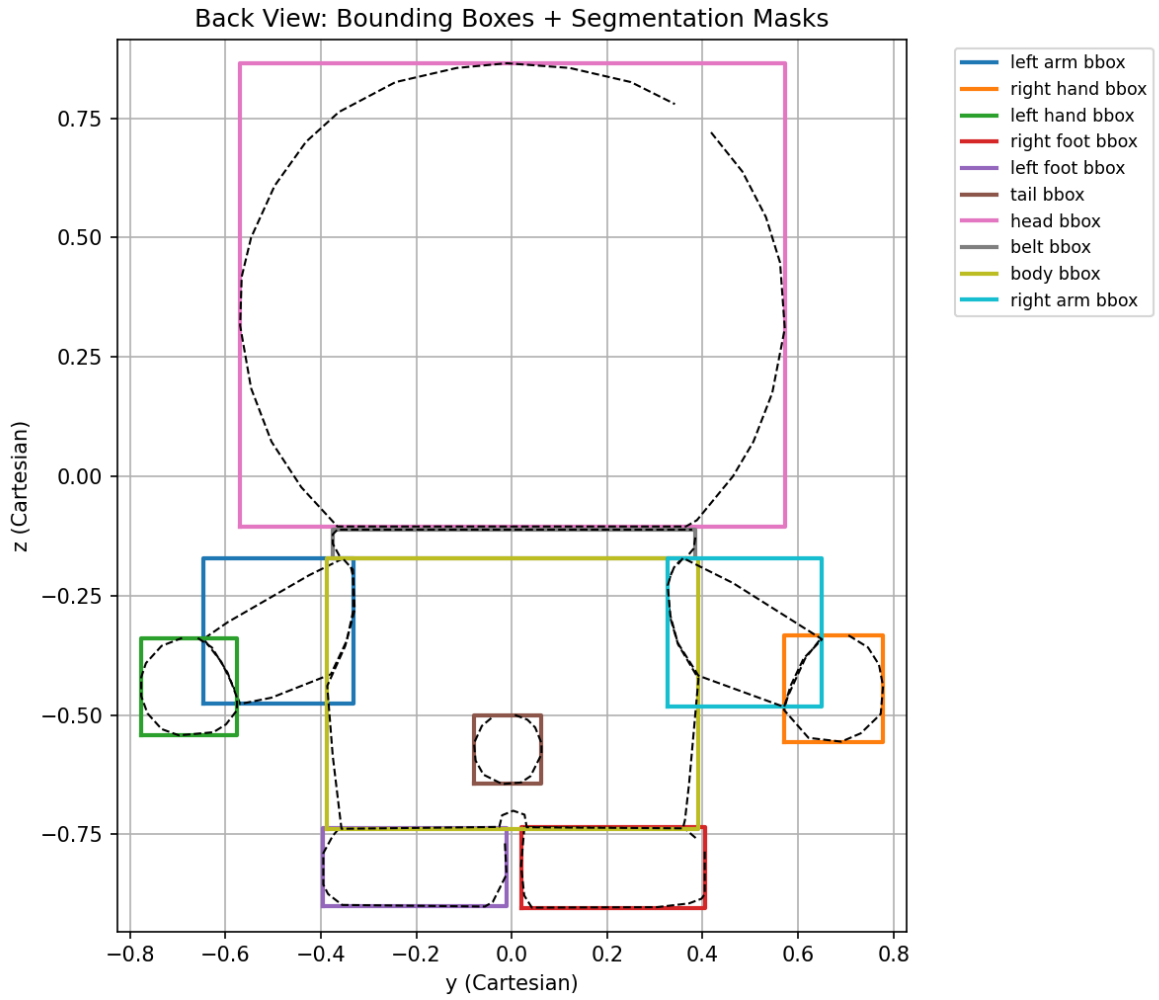



Figure 5.7: Back view of Doraemon Plot to Illustrate Bounding Boxes

Right View

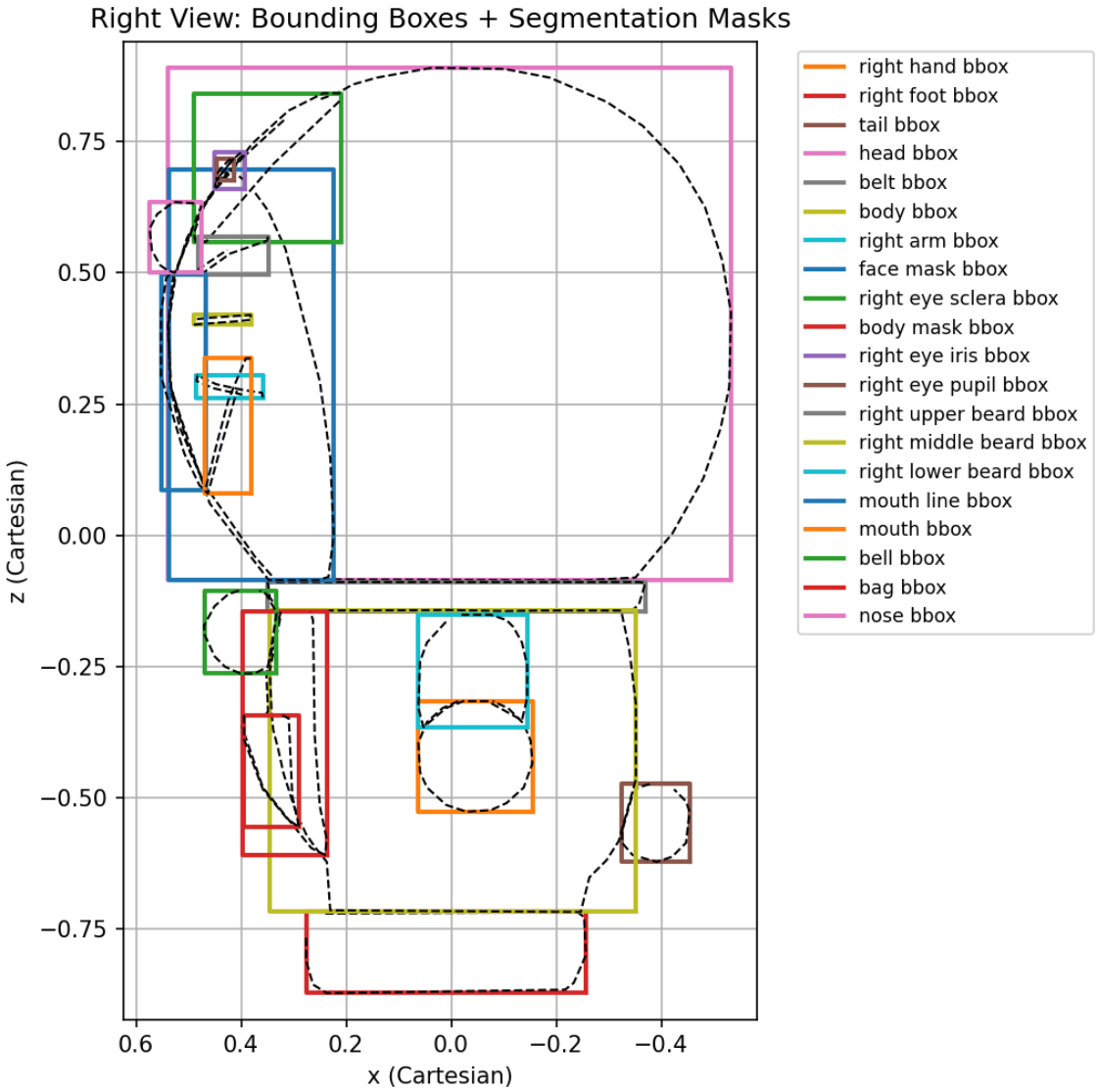



Figure 5.8: Right view of Doraemon Plot to Illustrate Bounding Boxes

These results demonstrate that the AABBs for Doraemon's mesh parts have been successfully reconstructed from the 2D coordinates extracted from orthogonal view coordinate systems.

### 5.1.3 3D Primitive Cubes Generation

Figures 5.9 to 5.12 show the primitive cubes had been successfully generated.

Front view:

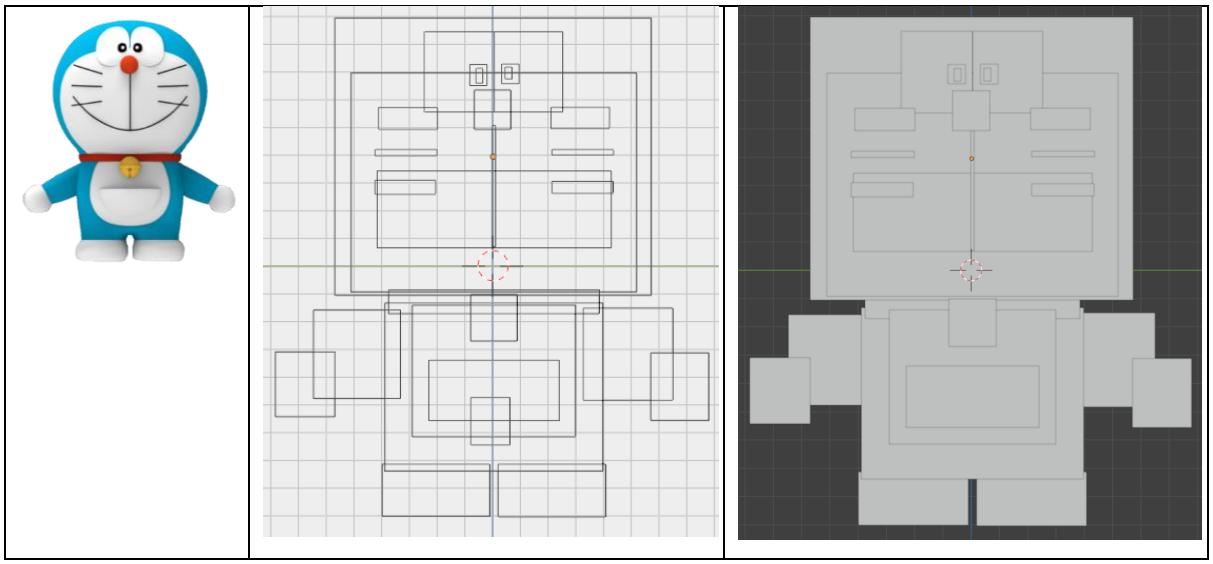


Figure 5.9: Comparing Front Original, Wireframe and Shading of Doraemon Meshes

Left view

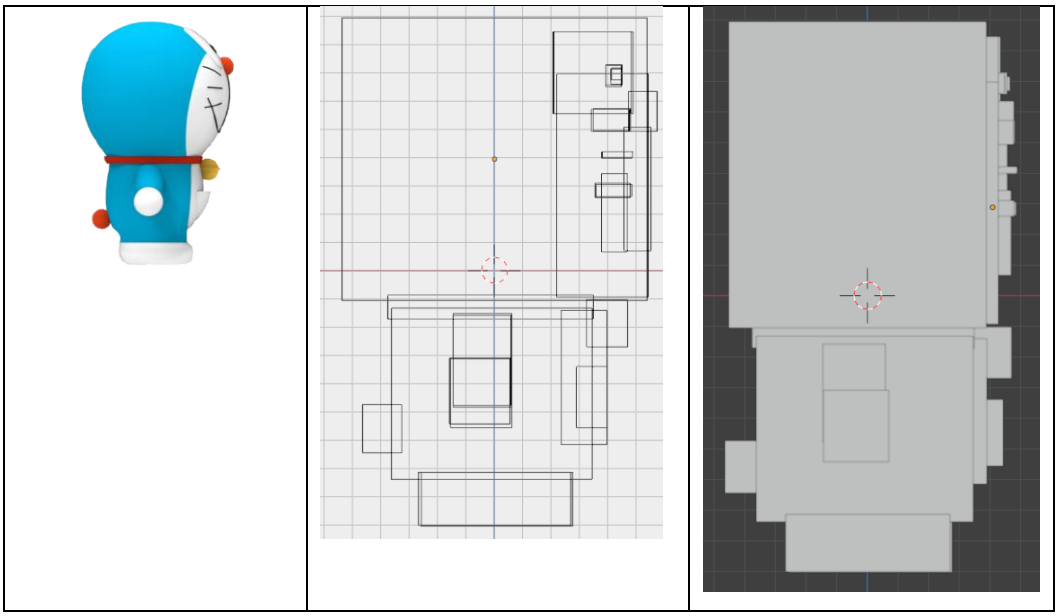


Figure 5.10: Comparing Left Original, Wireframe and Shading of Doraemon Meshes

Back view:

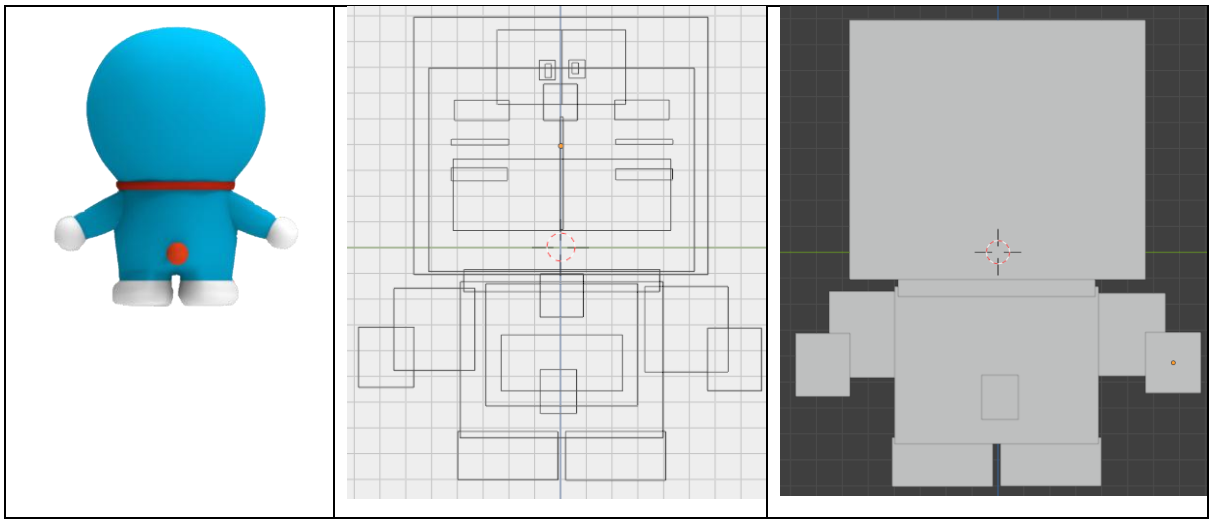


Figure 5.11: Comparing Back Original, Wireframe and Shading of Doraemon Meshes

Right view:

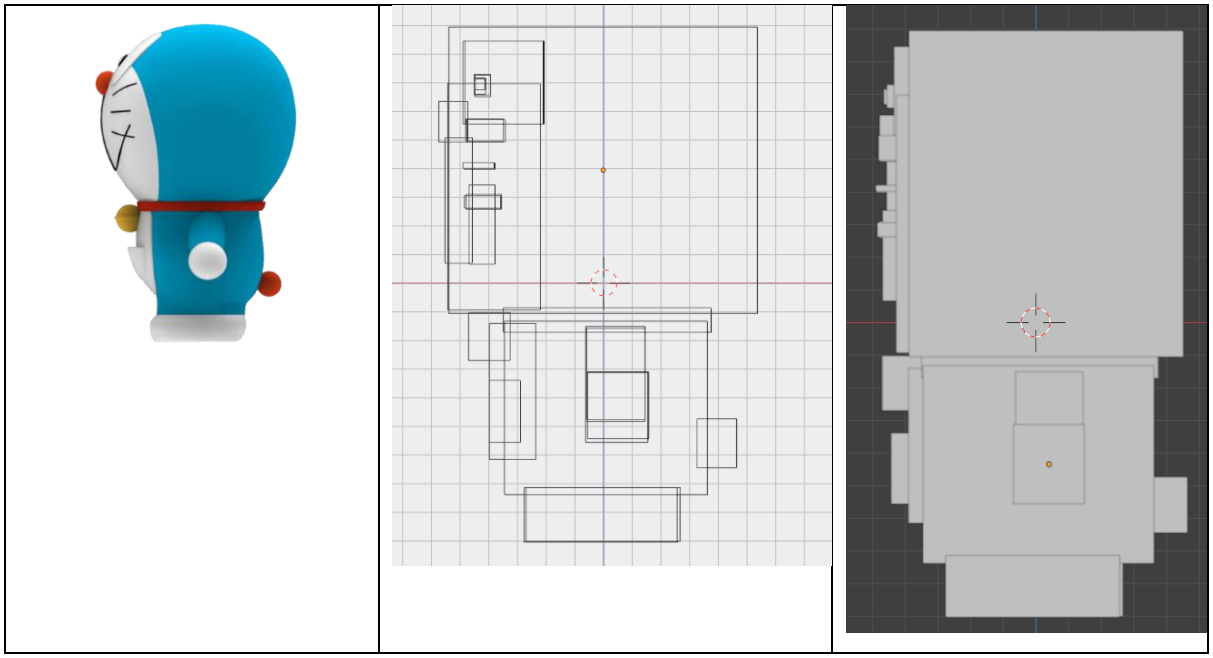


Figure 5.12: Comparing Right Original, Wireframe and Shading of Doraemon Meshes

Figures 5.9 to 5.12 above present the final output of the 3D Doraemon reconstruction generated from orthogonal-view images using the proposed algorithm. Each figure in this section is organized into three columns: the first column displays the original Doraemon image, the second column shows the reconstructed primitive cubes in Blender's wireframe viewport, and the third column presents the shading viewport after mesh positions have been adjusted for visibility.

From these results, we observe that AABB data alone is sufficient for generating basic primitive shapes such as cubes and cylinders. However, to capture more detailed 3D features and improve reconstruction fidelity, additional enhancements are recommended. These include incorporating top-view images, analyzing vertex distribution, generating supplementary views, and extracting semantic features from the input images. Such improvements would enable more accurate reconstruction of local object bounding boxes and finer mesh details

### *5.2 Comparative Results*

The Figures 5.13 and 5.14 below present a comparative analysis of 3D reconstruction outputs generated using three different methods: ChatGPT prompt Blender Script with sphere primitives (Fig. 5.13, a), OV-CV-BPY without manual post processing (Fig. 5.13.b) and Hunyuan3D 2.5 model with 500k faces Multiview option (Fig. 5.13.c) [31]. The Multiview option aligns to orthogonal view images.

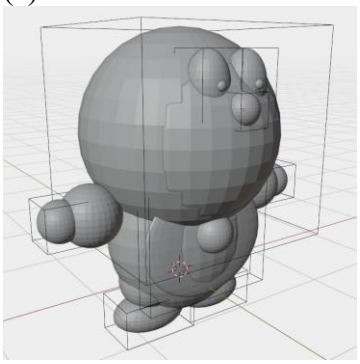
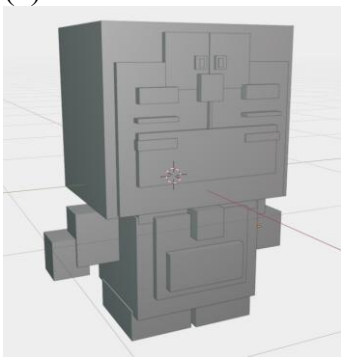
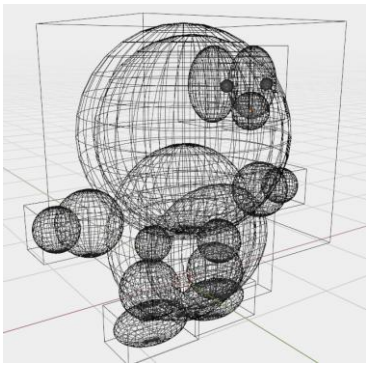
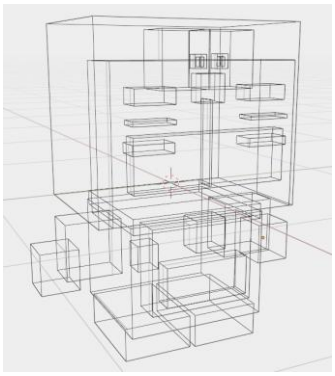
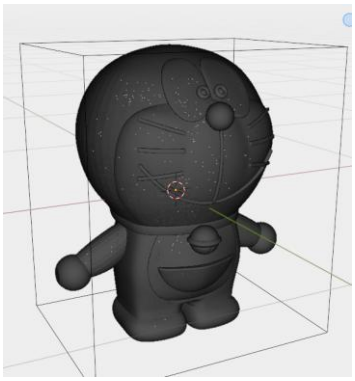
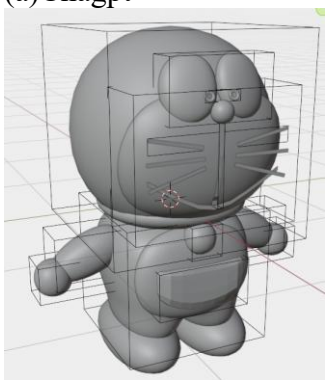
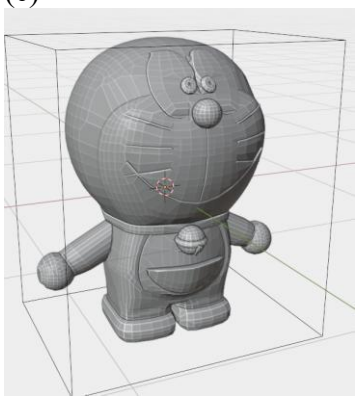
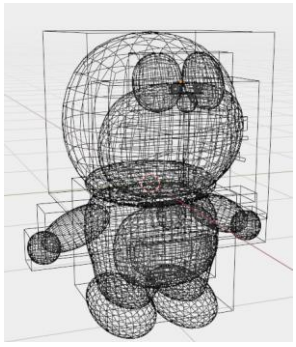
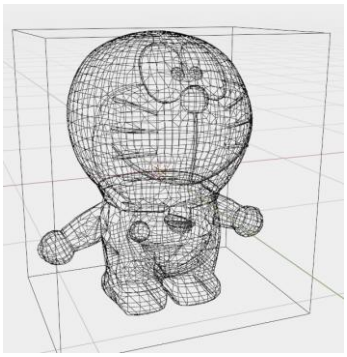
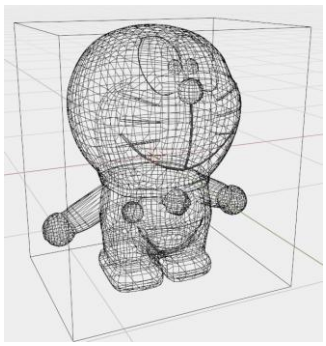
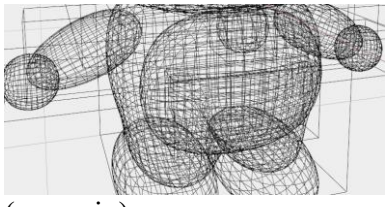
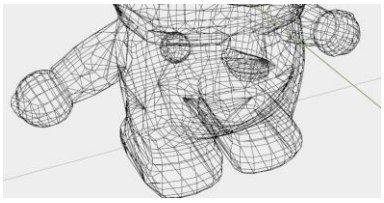
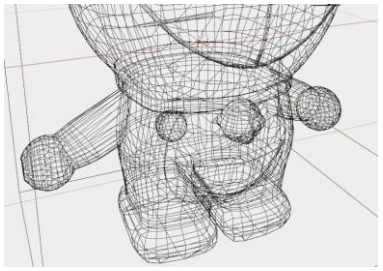
| (a) | (b) | (c) |
|---|---|---|
|  (shading viewport) |  (shading viewport) |  (shading viewport) |
|  (wireframe viewport) |  (wireframe viewport) |  (wireframe viewport) |
| Statistics | | |
| Objects 19 / 19<br>Vertices 9,158 / 9,158<br>Edges 18,848 / 18,848<br>Faces 9,728 / 9,728<br>Triangles 18,240 / 18,240 | Objects 29<br>Vertices 232<br>Edges 348<br>Faces 174<br>Triangles 348 | Objects 1 / 3<br>Vertices 290,399 / 290,399<br>Edges 789,234 / 789,234<br>Faces 499,994 / 499,994<br>Triangles 499,994 / 499,994 |
| Time required | | |
| Within 5 seconds with network latency | Less than 1 second | Within 2 minutes with network latency |

Figure 5.13: Comparing Mesh Generation Results Part 1

The Multiview option aligns to orthogonal view images. The figures below show the comparison of the results of output using OV-CV-BPY with manual post processing using subdivision modifier and grouped symmetry objects (Fig. 5.14.a), and Hunyuan3D 2.5 model with 50k faces Multiview option and quad retopology (Fig. 5.14.b) and Hunyuan 3D 3.0 500k Multiview option and quad retopology (Fig. 5.14.c) [31].

| (a)Chagpt | (b) | (c) |
|---|---|---|
|  (shading viewport) |  (shading viewport) |  (shading viewport) |
|  (wireframe viewport) |  (wireframe viewport) |  (wireframe viewport) |
|  (zoom in) |  (zoom in) |  (zoom in) |

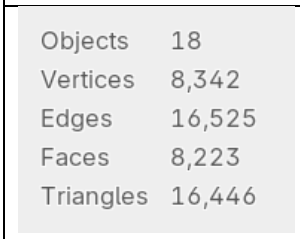
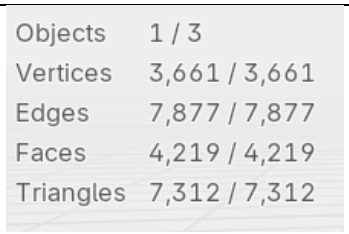
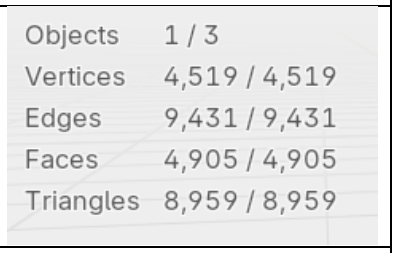
| Statistics | | |
|---|---|---|
|  |  |  |
| Time required | | |
| Within 10 minutes, subjected to changed for different skill levels | Within 5 minutes with network latency | Within 5 minutes with network latency |

Figure 5.14: Comparing Mesh Generation Results Part 2

The reconstructed Doraemon mesh aligns well with the problem statement and motivation regarding AI-generated meshes. While the geometric details are nearly perfectly reconstructed, the mesh lacks part-level segmentation. However, post-processing of the reconstructed mesh using the OV-CV-BPY algorithm demonstrates that geometric adjustments can be made with ease, indicating strong generalization capability of the algorithm.

As the part-level mesh generation feature offered by Hunyuan3D Studio [31] is not yet publicly available for testing, a direct comparison of part-level segmentation results cannot be conducted at this time.

Based on the experimental results presented in Sections 5.1.1 to 5.1.3, the primitive cubes representing Doraemon have been successfully generated, validating the effectiveness of the proposed algorithm. Furthermore, the manual post-processing results in Section 5.2 show that the algorithm significantly assists artists by automating the initial placement of primitive cubes—traditionally done manually at the world origin—allowing them to focus directly on refining geometric details through box modeling techniques.

# Chapter 6: Conclusion

## 6.1 Summary of Work

This thesis presents a comprehensive review of state-of-the-art 3D generative AI models and 3D reconstruction techniques, as discussed in Chapters 2 and 5. Chapters 3, 4, and 5 detail the problem formulation for 3D character reconstruction and introduce the OV-CV-BPY algorithm, along with its implementation and evaluation. The proposed algorithm can serve as a good starting in building high fidelity geometry from primitive shapes.

## 6.2 Challenges

The primary challenge encountered during this project was the limited GPU computational power available on the local machine. This constraint made it infeasible to fine-tune open-source 3D shape generation models without subscribing to cloud-based computing services, which introduces cost and accessibility barriers.

## 6.3 Future Work

Future extensions of the algorithm can benefit significantly from advanced 3D modeling expertise. As proficiency in digital modeling increases, users will find it easier to leverage scripting techniques to automate modeling tasks. This opens the door to developing personalized automation tools or community-driven add-ons that enhance productivity and streamline creative workflows

## REFERENCES

[1] L. Zhou, G. Wu, Y. Zuo, X. Chen, and H. Hu, “A Comprehensive Review of Vision-Based 3D Reconstruction Methods,” Apr. 01, 2024, Multidisciplinary Digital Publishing Institute (MDPI). doi: 10.3390/s24072314.

[2] Ikura_AI, 3D Reconstruction Technique (1): What Is Reconstruction? Why Do We Need It?. (Oct. 17, 2023). Accessed: Apr. 30, 2025. [Online Video].
https://youtu.be/LLSLSRgoIjY?si=zbQg0o5k4nzgreI1

[3] A. Durrant. “Translating a 2D concept into real-time 3D art.” Marmoset.co.
https://marmoset.co/posts/translating-a-2d-concept-into-real-time-3d-art/ (accessed

[4] Ikura_AI, 3D Reconstruction Technique (2): NeRF, AI Technology Innovation – “Memorising” The Scene Using Neural Network! (Oct. 26, 2023). Accessed: Apr. 30, 2025. [Online Video].
Available: https://youtu.be/KPO8BpzY0LA?si=S7GCLleVlwueUgsA

[5] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, R. Ng. (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. [Online]. Available: https://arxiv.org/pdf/2003.08934

[6] Mediastorm. Reconstructing a Mountain in Just 10 Minutes? How to Use the Amazing NeRF Technology!. (Feb. 1, 2024). Accessed: Apr. 29, 2025. [Online Video].
Available: https://youtu.be/Vm1W0HNEOWM?si=uXn1dJ6w8WnOfJmV

[7] Ikura_AI. 3D Reconstruction Technique (3): 3D Gaussian Splatting, let 3D Reconstruction Moved From Academic To Practical Application! Use Colorful Elliptical Balls To Represent Scenes!. (Nov. 9, 2023). Accessed: Apr. 29, 2025. [Online Video].
Available: https://youtu.be/UxP1ruyFOAQ?si=L-np7xTTAwk0-mzk

[8] B. Kerbl, G. Kopanas, T. Leimkuhler, G. Drettakis. “3D Gaussian Splatting for Real-Time Radiance Field Renderingm,” ACM Trans. Graph, vol.42, no.4, Aug, 2023.

[Online].https://repo-sam.inria.fr/fungraph/3d-gaussian splatting/3d_gaussian_splatting_low.pdf

[9] Ikura_AI. How does generative 3D affect the 3D industry? How does AI help with 3D creation?. (Jun. 4, 2023). Accessed: Apr. 30, 2025. [Online Video]. https://youtu.be/ilL9nZCXMI4?si=z7Gn-VG5Oz2DFLO6

[10] L. Cerkezi, P. Favaro. (2024). Sparse 3D Reconstruction via Object-Centric Ray Sampling. [Online]. Available: https://arxiv.org/pdf/2309.03008v2

[11] J. L. Schonberger, J. Frahm. (2016). Structure-from-Motion Revisited. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2016/papers/Schonberger_Structure-From-Motion_Revisited_CVPR_2016_paper.pdf

[12] J. L. Schonberger, E. Zheng, M. Pollefeys, J. Frahm. (2016). Pixelwise View Selection for Unstructured Multi-View Stereo.
[Online]. Available: https://demuc.de/papers/schoenberger2016mvs.pdf

[13] Fast Photogrammetry. FREE COLMAP A beginner tutorial, introduction to photogrammetry [fix GPU timeout]. (Nov. 9, 2019). Accessed: Apr. 29, 2025. [Online Video]. https://youtu.be/mUDzWCuopBo?si=3rFmGP6-JQB82qdV

[14] Mattthew Brennan. Photogrammetry / NeRF / Gaussian Splatting comparison. (Oct. 1, 2023). Accessed: Apr. 29, 2025. [Online Video]. https://youtu.be/KFOy354zf9E?si=AyEJAPAuiTbLZgxZ

[15] Z. Huang, H. Xie, T. Fukusato. (2022). Interactive 3D Character Modeling from 2D Orthogonal Drawings with Annotations.
[Online]. Available: https://arxiv.org/pdf/2201.11284

[16] P. Buchanan, R. Mukundan, M. Doggett. (2013). Automatic Single-View Character Model Reconstruction.
[Online].
Available:https://fileadmin.cs.lth.se/cs/Personal/Michael_Doggett/pubs/buchanan13-ACMR-SBIM.pdf

[17] C. G. Willcocks, F. W.B. Li. (2012). Feature-varying skeletonization.
[Online]. Available: https://frederickli.webspace.durham.ac.uk/wp-content/uploads/sites/103/2021/04/cgi12-FeatureVaryingSkeletonization.pdf

[18] K. Wu, F. Liu, Z. Cai, R. Yan, H. Wang, Y. Hu, et al. (2024). Unique3D: High-Quality and Efficient 3D Mesh Generation from a Single Image. [Online]. Available:
https://arxiv.org/pdf/2405.20343

[19] Z. Zhao, Z. Lai, Q. Lin, Y. Zhao, H. Liu, S. Yang, et al. (2025). Hunyuan3D 2.0: Scaling Diffusion Models for High Resolution Textured 3D Assets Generation. [Online]. Available:
https://arxiv.org/pdf/2501.12202

[20] AI Tool Designer-Jason. The most powerful AI-3D model is here! Tencent open source Hunyuan3D-2 review vs Microsoft Trellis!. (Jan. 22, 2025). Accessed: Apr. 30, 2025. [Online Video].
https://youtu.be/g-ThznwPxJ8?si=0jKvdNv3iDwJnOPV

[21] SVG PRO. How to Install Locally Hunyuan 3D-2 AI Image to 3D Object. (Jan. 29, 2025). Accessed: Apr. 30, 2025. [Online Video].
https://youtu.be/VTvS325hgV0?si=-CaeAKywoWtIF8Zl

[22] SVG PRO. Hunyuan 3D-2 Installing Build Tools & Model Security Patch. Accessed: Apr. 30, 2025. [Online Video].
https://youtu.be/8jOp6JZfGE8?si=EAnb5ZgmXyhWnkuT

[23] J. L. Schoenberger. “COLMAP 3.12.0.dev0 documentation.” Colmap.github.io
https://colmap.github.io/

[24] A. Kadam. “Doraemon toys.” ArtStation.
https://nopstack.artstation.com/projects/DxzPlR

[25] N. Ravi, V. Gabeur, Y. Hu, R. Hu, C. Ryali, T. Ma, et al. (2024) SAM 2: Segment Anything in Images and Videos.
[Online]. Available: https://arxiv.org/pdf/2408.00714

[26] Blender. Getting Started with Scripting in Python. (Oct. 28, 2023). Accessed: Sept. 20, 2025. [Online Video].
https://youtu.be/wWTAQP7-ZUQ?si=H0TzPRg0s3Coq9TZ

[27] Blender. “Blender Python API.” blender.org.
https://docs.blender.org/api/current/info_overview.html

[28] cvat. “Leading Data Annotation Platform.” cvat.ai.
https://www.cvat.ai/

[29] OpenGL. “Vertex Post-Processing.” khronos.org.
https://www.khronos.org/opengl/wiki/Vertex_Post-Processing
[30] Streamlit. “A faster way to build and share data apps.” streamlit.io.
https://streamlit.io/

[31] Tencent Hunyuan3D. Hunyuan3D Studio: End-to-End AI Pipeline for Game-Ready 3D Asset Generation.
[Online]. Available. https://arxiv.org/pdf/2509.12815

**POSTER**

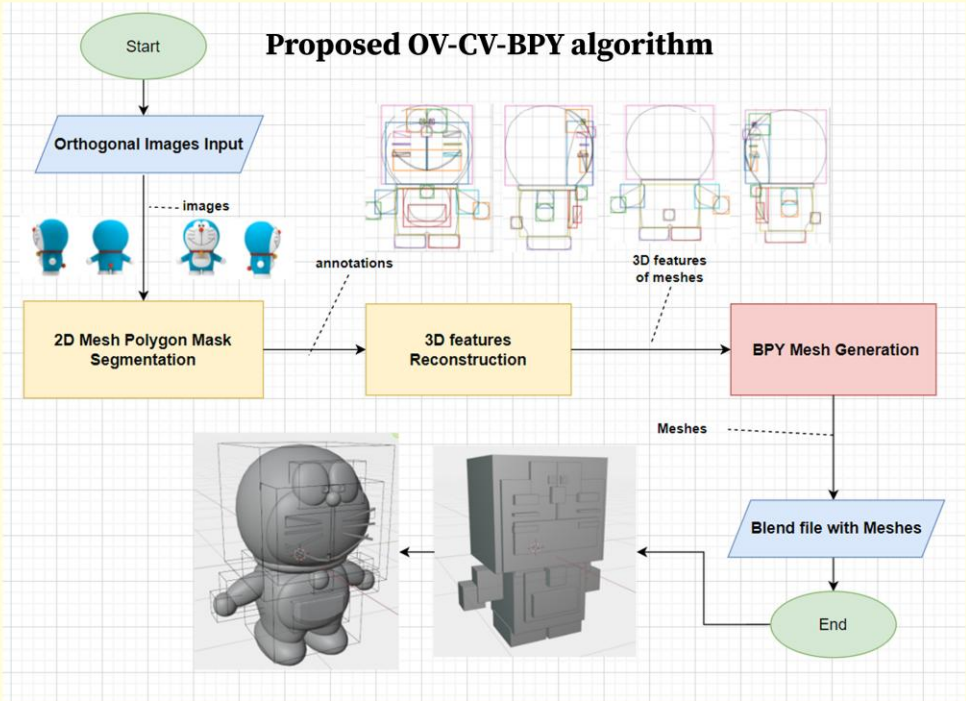
Faculty of Information and Communication Technology

# 3D Character Reconstruction from Orthogonal Images

Prepared by: Kok Yung Thow Supervisor: Ts Dr Saw Seow Hui

## Problem Statements &Motivation

- Traditional 3D character modeling techniques refine primitive shapes to high fidelity mesh from orthogonal concept arts
- An existing 3D shape generation model can easily generate a 3D character. However, most of the output 3D models offer limited control over the final result.




## Project Objectives

1. To review on 3D shape generation and reconstruction state-of-arts
2. To investigate the potential of hybrid and interactive 3D reconstruction approach that reconstructs 3D shape from orthogonal images.

## Contribution

1. This research can contribute to the advancement of future studies in 3D reconstruction and generation with digital modeling script.
2. proposed an algorithm that can assist artists in transforming orthogonal view concept art into primitive 3D meshes..