ENS-CVXNET: CONVEX DECOMPOSITION OF COMPLEX SCENES

BY

SEEMANDHAR JAIN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2024

Urbana, Illinois

Adviser:

Professor David Forsyth

ABSTRACT

Generating accurate convex decompositions of indoor scenes from single RGB images is a challenging task with numerous applications in computer graphics. Current state-of-the-art methods employ encoder-decoder neural networks to convert RGB images into a fixed number of simple primitives (e.g., parallelepipeds). However, these approaches have limitations in capturing long-range dependencies and transferring global information, which can impact their accuracy and generalization capability across diverse indoor environments. To address these limitations, we propose ENS-CVXNet, an ensemble approach that leverages the strengths of various convex decomposition techniques and incorporates additional geometric information as summaries. Our core analysis explores well-established algorithms such as VHACD, COACD, and BSP-Net, as well as the integration of global features extracted by PointNet from the input point cloud. By combining multiple models trained with different summaries and configurations, ENS-CVXNet selects the best-performing model for each input image based on its ability to generate accurate depth predictions, evaluated against ground truth depth maps or predictions from state-of-the-art depth predictors. Through extensive experiments and evaluations on the NYUv2 dataset, we demonstrate that ENS-CVXNet outperforms the baseline method, achieving a 20% decrease in AbsRel from 0.093 to 0.0744, and improving the overall precision and quality of convex decomposition for indoor scenes. Our ensemble approach effectively combines the strengths of various techniques, leveraging geometric summaries and adapting to diverse scene characteristics, results in more accurate and robust 3D geometric reconstruction from single RGB images.

To my parents (Alka and Yogesh Jain) and siblings (Nihali and Prarthi Jain), whose unwavering love and support has been the bedrock of my journey.

To my advisor, Dr. David Forsyth, whose guidance and wisdom have been instrumental in shaping this work.

To my friends, who have been a constant source of encouragement and motivation, making this endeavor a truly enriching experience.

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my advisor, Dr. David Forsyth, for their invaluable guidance, unwavering support, and insightful feedback throughout this research endeavor. Their expertise, patience, and encouragement have been instrumental in shaping this work and fostering my growth as a researcher.

I would also like to extend my heartfelt appreciation to my colleagues and collaborators, who have contributed their knowledge, ideas, and valuable discussions, enriching this research and making it a truly collaborative effort.

Finally, I would like to acknowledge the unwavering love and support of my family and friends, whose constant encouragement and understanding have been a constant source of motivation throughout this journey.

This research would not have been possible without the support provided by CS department, and I am grateful for their generosity and belief in the significance of this work.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Generating accurate convex decompositions of 3D scenes from single RGB images is a complex problem. Current state-of-the-art methods, such as the work by Vavilala et al. [1], employ learned regression techniques to convert RGB images to a fixed number of simple primitives. While this baseline approach has shown promising results, it is primarily based on an encoder-decoder neural network architecture, which may have limitations in capturing long-range dependencies and transferring information on a global scale.

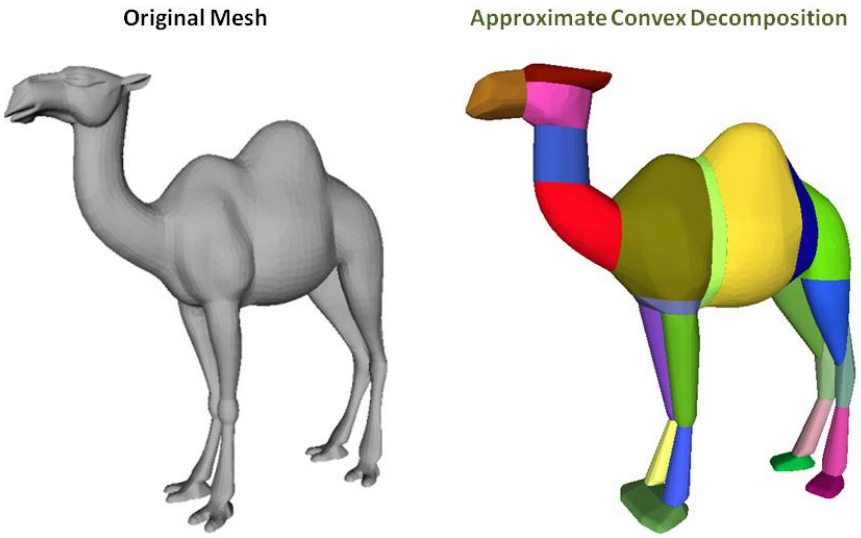


Figure 1.1: An example of convex decomposition applied to a mesh, breaking it down into simpler convex components. This figure is taken from here.

Figure 1.1 shows an example of convex decomposition applied to a 3D mesh of an object, breaking it down into simpler convex components. While this example illustrates convex decomposition on a single object mesh, our work focuses on generating convex decompositions of entire indoor scenes from single RGB images, as demonstrated in Chapter 4.

To address these limitations and improve the accuracy of convex decomposition, we propose an ensemble approach that leverages the strengths of various convex decomposition techniques and incorporates additional geometric information as summaries. Our core analysis is grounded in the examination of well-established algorithms, including Volumetric Hierarchical Approximate Convex Decomposition (VHACD) [2] and Approximate Convex

Decomposition for 3D Meshes with Collision-Aware Concavity and Tree Search (COACD) [3].

While VHACD and COACD can accurately represent the locations of objects in the scene, their generated convex primitives often have complex and arbitrary shapes, unlike the simple parallelepipeds produced by the baseline method. To capitalize on the strengths of these algorithms while preserving the simplicity of the baseline output, we propose using the outputs of VHACD and COACD as geometric summaries, providing additional information to the baseline encoder-decoder network.

Specifically, we explore various approaches to incorporate the geometric summaries, such as using the means or medians of the convex primitives generated by COACD, or integrating global features extracted by PointNet [4] from the input point cloud. These summaries are stacked with the output of the encoder and fed into the decoder, allowing the network to leverage the geometric information during the learning process.

By combining multiple models trained with different summaries and configurations, our proposed ensemble approach, ENS-CVXNet, selects the best-performing model for each input image based on its ability to generate accurate depth predictions. This selection process is facilitated by comparing the depth predictions of each model against the ground truth depth map or predictions from state-of-the-art depth predictors like MiDaS [5].

Through extensive experiments and evaluations as shown in Chapter 4, we demonstrate that the ENS-CVXNet ensemble approach, which integrates the strengths of various convex decomposition techniques and leverages geometric summaries, outperforms the baseline model and improves the overall precision and quality of simple convex decomposition for indoor scenes.

The rest of the thesis is organized as follows:

Chapter 2 provides a literature review, discussing relevant work in the areas of primitive decomposition. Chapter 3 gives background information on the baseline approach for convex decomposition of indoor scenes from Vavilala et al. [1], including details on their encoder-decoder architecture, loss functions, and evaluation metrics. Chapter 4 presents our proposed ENS-CVXNet ensemble approach, detailing the various convex decomposition algorithms explored, the use of geometric summaries. Chapter 5 concludes the thesis, summarizing the key contributions of our work and discussing potential future research directions. Finally, Appendix A contains supplementary materials, including additional tables.

CHAPTER 2: LITERATURE REVIEW

The concept of representing 3D scenes or objects using a set of primitive shapes has been an active area of research in computer vision for decades [6, 7, 8]. Such primitive-based representations offer several advantages, including parsimonious abstraction [9], natural segmentation [6, 10], and simplified geometric reasoning [11, 12, 13]. The primary challenge lies in selecting primitives that can be easily inferred from image data while allowing for effective shape approximation.

Early work on convex decomposition focused on individual objects rather than entire scenes. Tulsiani et al. [14] demonstrated a learned procedure that can parse 3D shapes into cuboids without requiring ground truth segmentations during training. Deng et al. [9] introduced CVXNet, a method that recovers 3D representations of objects as unions of convex polytopes from point cloud and image data, again without relying on ground truth segmentations. More recently, an advanced version of CVXNet has been developed [15] to decompose entire scenes into a comprehensive set of primitives, forming the foundation for the work presented in this work.

Researchers have explored various approaches to decompose and understand indoor scenes from visual data. Hedau et al. [16, 17, 18] proposed methods for recovering room layouts as cuboids, identifying furniture, and estimating free space within rooms. Hoiem et al. [19, 20] focused on parsing outdoor scenes into vertical and horizontal surfaces, while Gupta et al. [21] demonstrated a parse into blocks. For indoor scenes, techniques have been developed to recover room layouts [22, 23], estimate plane layouts [23, 24], and impute patch-like primitives from data [25]. Jiang [26] demonstrated parsing RGBD images into primitives by solving a 0-1 quadratic program, while Kluger et al. [27] proposed a RANSAC-like greedy algorithm for identifying cuboids sequentially.

Traditional approaches to primitive fitting often involved minimizing a cost function by choosing an appropriate set of primitives [28]. However, recent advancements in deep learning have enabled predicting suitable primitives directly from data, potentially avoiding local minima issues faced by descent-based methods [9, 14, 24, 29]. Various primitives have been explored for 3D representation, including cuboids [30, 31, 32, 33], superquadrics [34], planes [24, 35], and generalized cylinders [29, 36]. Despite these efforts, existing methods still face challenges in producing varying numbers of primitives per scene [37] and accounting for different scene characteristics.

Our work aims to address these limitations by proposing an ensemble approach, ENS-CVXNet, which combines the strengths of various convex decomposition techniques and incorporates additional geometric information to improve the accuracy and quality of convex

decomposition for indoor scenes from single RGB images, on top of the recent work [15, 38] (baseline).

CHAPTER 3: BACKGROUND

Current state-of-art convex decomposition for indoor scenes uses a learned regression procedure to convert a single RGB image to simple primitives [1]. They build an encoder-decoder architecture that takes RGB and segmentation maps as input. After some polishing, their network outputs a fixed number of primitives (14 in their paper). Since it is based on encoder-decoder based neural network, its likely to have difficulty in Capturing Long-Range Dependencies of the image, like the exact global features, and possibly different objects in an image. There are many ways to improve that, and we have proposed an ensemble approach in section 4, that beats this approach in terms of accuracy. Since the proposed approach is based on [1], let's first understand the working of their method.

3.1 HOW IT WORKS

In their work, the authors proposed an inference framework designed to process RGBD images alongside segmentation maps. They proposed an encoder-decoder architecture, the training of the network is supervised by a variety of loss functions that essentially guide the network to differentiate between inside and outside labeled samples.

The network outputs a fixed number of primitives which is trained with certain losses that we have discussed later. After that, the polishing step is applied to improve the decomposition based on descent and greedy strategies.

Their architecture is based on CVXNet [39], and using their predefined losses and hyperparameter, fine-tuning and adding some more losses to generate parallelopiped convexes. In their work, they introduce a series of loss functions in the framework to train the network for optimal primitive decomposition of indoor scenes. This framework comprises several components, each targeting a specific aspect of the decomposition process to ensure that the

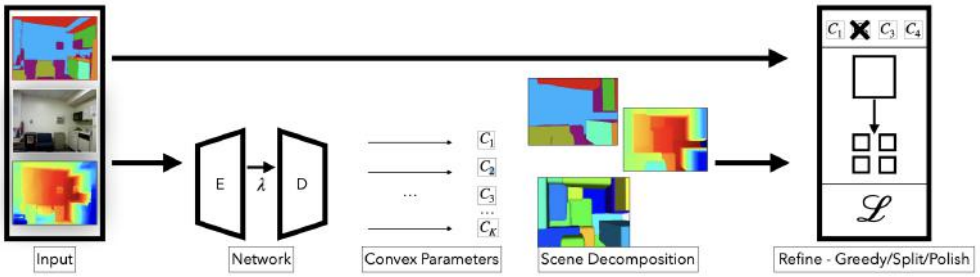


Figure 3.1: This figure shows an encoder-decoder architecture that takes RGB and segmentation maps as input and outputs a fixed number of primitives, taken from the baseline [1].

Cfg.	Prune	Refine	Depth $_{GT}$	Seg $_{GT}$	n_{init}	n_{used}	AbsRel ↓	RMSE ↓	Mean ↓	Median ↓	11.25° ↑	22.5° ↑	30° ↑	Seg Acc ↑
noInit	×	✓	✓	✓	24	24.0	0.447	1.604	81.286	80.566	0.021	0.061	0.096	0.359
noSeg	×	×	×	×	24	24.0	0.179	0.664	41.687	38.224	0.115	0.291	0.391	**0.626**
withSeg	×	×	×	×	24	24.0	0.310	1.231	52.305	48.846	0.088	0.231	0.317	0.528
A	×	✓	×	×	24	24.0	0.163	0.679	40.692	35.697	0.124	0.313	0.424	0.623
B	×	✓	×	×	24	24.0	0.166	0.696	41.019	35.964	0.122	0.310	0.421	0.623
C	✓	✓	×	×	24	14.4	**0.144**	**0.603**	**38.235**	**33.621**	**0.133**	**0.335**	**0.451**	0.615
D	✓	✓	✓	×	24	14.0	0.098	0.513	37.361	32.402	0.144	0.353	0.469	0.619
E	✓	✓	✓	✓	24	13.9	0.098	0.514	37.355	32.395	0.144	0.353	0.469	0.618
ref	-	-	✓	✓	-	-	0.110	0.357	14.9	7.5	0.622	0.793	0.852	0.719

Table 3.1: Table 3.1 presents the final evaluation metrics reported in the baseline paper [1] for their convex decomposition method. The authors conducted an ablation study with different configurations on the 654 test images from the NYUv2 dataset, evaluating against ground truth depth, normals, and segmentation. The table shows the performance of various setups, including without initialization (noInit), without segmentation loss (noSeg), with segmentation loss (withSeg), and different refinement strategies (A, B, C, D, E). The metrics reported are Absolute Relative error (AbsRel), Root Mean Squared Error (RMSE), normal error metrics, and pixel-wise segmentation accuracy. The best non-oracle configuration (C) achieved an AbsRel of 0.144, RMSE of 0.603, and a segmentation accuracy of 0.615, outperforming the random initialization (noInit) and demonstrating the effectiveness of their approach.

network outputs in a certain way.

3.2 LOSS FUNCTION

The overall loss function for training the convex decomposition network can be written as:

$$\mathcal{L}\text{total} = \mathcal{L}\text{sample} + \lambda_1\mathcal{L}\text{unique} + \lambda_2\mathcal{L}\text{ortho} + \lambda_3\mathcal{L}\text{align} + \lambda_4\mathcal{L}\text{vol} + \lambda_5\mathcal{L}_{\text{entropy}} + \lambda_6 \quad (3.1)$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5$ are weight hyperparameters for the respective losses, and λ_6 is remaining losses. The individual losses are defined as follows:

- $\mathcal{L}_{\text{sample}}$: The primary sample loss that encourages correct classification of points as inside or outside the convex primitives.
- $\mathcal{L}_{\text{unique}}$: The unique parameterization loss, modified from [39] to penalize small offsets in the halfplane parameters:

$$\mathcal{L}_{\text{unique}} = \frac{1}{H}\sum_h \|d_h\|^2 + \frac{1}{H}\sum_h \left\|\frac{1}{d_h}\right\|^2 \quad (3.2)$$

  where $H = 6$ is the number of halfplanes per convex, and d_h is the offset parameter

for halfplane h.

- $\mathcal{L}_{\text{ortho}}$: The orthonormality loss for the predicted Manhattan world parameters $\mathcal{M} \in \mathbb{R}^{3\times3}$:

$$\mathcal{L}_{\text{ortho}} = \frac{1}{9}\sum(\mathcal{I} - \mathcal{M}^\top\mathcal{M})^2 \tag{3.3}$$

  where $\mathcal{I}$ is the identity matrix.

- $\mathcal{L}_{\text{align}}$: The alignment loss that encourages convex faces to follow the Manhattan world directions:

$$\mathcal{L}_{\text{align}} = 1 - \frac{1}{6}\sum\mathcal{N}\cdot\mathcal{W} \tag{3.4}$$

  where $\mathcal{N}$ is a 3×6 matrix of predicted convex normals, and $\mathcal{W} = [\mathcal{M}; -\mathcal{M}]$ is a 3×6 matrix of Manhattan world basis vectors.

- $\mathcal{L}_{\text{vol}}$: The volume loss that encourages convexes to be predicted in decreasing order of volume:

$$\mathcal{L}_{\text{vol}} = \frac{1}{K}\sum\text{ReLU}(\mathcal{V}[1:] - \mathcal{V}[0:-1]) \tag{3.5}$$

  where $\mathcal{V} \in \mathbb{R}^K$ is a vector of convex volumes, and K is the total number of convexes.

- $\mathcal{L}_{\text{entropy}}$: The segmentation loss that encourages convexes and their faces to respect segmentation boundaries:

$$\mathcal{L}_{\text{entropy}} = \frac{1}{6K}\sum\text{entropy}(\mathcal{C}_f\mathcal{L}) \tag{3.6}$$

  where $\mathcal{C}_f \in \mathbb{R}^{6K\times N}$ is a matrix representing the indicator function response for each face, $\mathcal{L} \in \mathbb{R}^{N\times41}$ is a one-hot encoding of segmentation labels, and K is the number of convexes.

The overall loss $\mathcal{L}_{\text{total}}$ is minimized during training to obtain a set of convex primitives that accurately represents the input 3D scene while satisfying the desired properties encoded by the individual losses.

All of these losses are effectively used to train the model on NYUv2 dataset. They used Absolute Relative difference (AbsRel)[40], defined as

$$AbsRel = \frac{1}{|N|}\sum_{y\in N}\frac{|y-\hat{y}|}{\hat{y}} \tag{3.7}$$

where N is the set of available pixels in the manually annotated ground-truth. They got an AbsRel of 0.098 on test data with ground truth depth maps used during refinement instead

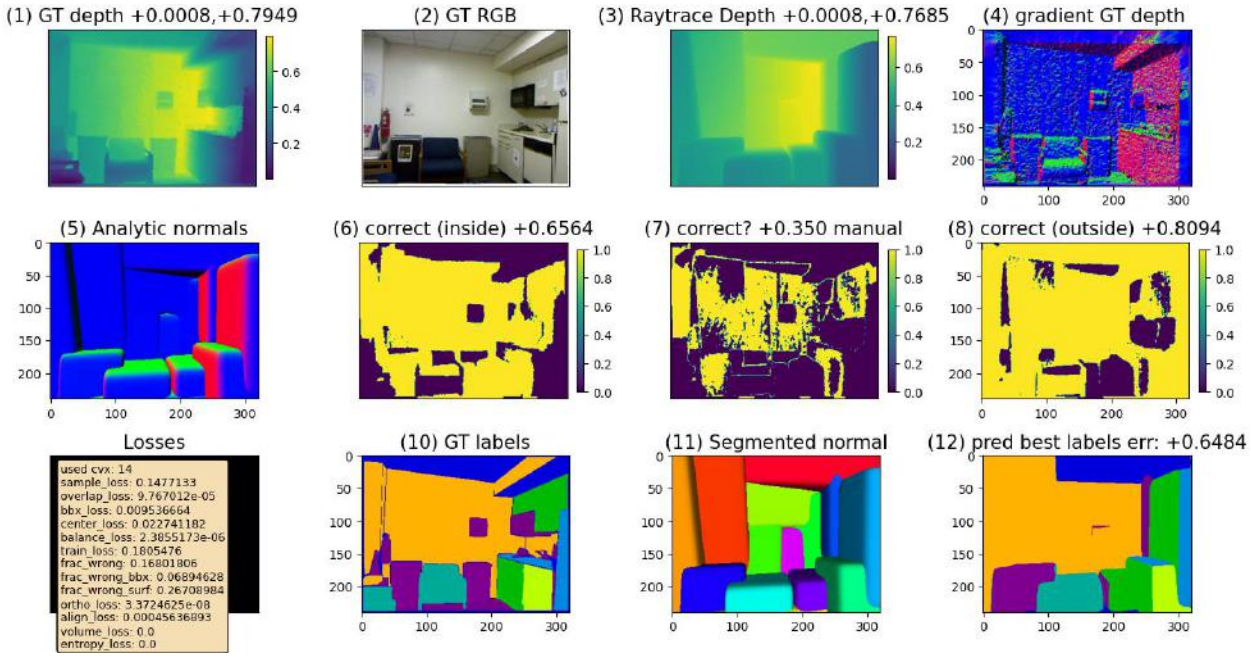


Figure 3.2: This figure presents a qualitative study of their method on a test image never seen by the network during testing. The first row shows: 1) the ground truth depth, 2) the ground truth RGB image, 3) the raytraced depth predicted from the network's output convexes, and 4) the gradient of the ground truth depth (normal). The second row displays: 5) the analytic normal predicted from the raytraced depth, 6) points correctly identified as inside the surface, 7) points correctly identified on the surface, and 8) points correctly identified as outside the surface. The third row contains: 9) the values of the different loss terms used during training, 10) the ground truth segmentation labels, 11) the predicted convex decomposition, and 12) the predicted segmentation from the network. This highlights the difficulty of the optimization problem and shows results from just the decoder in the network. We will use the same Tableau in the rest of the thesis.

of inferred. Their evaluation results are shown in table 3.1.

Figure 3.2 shows the work of their approach on one of the images on the NYUv2 dataset (index=0 of the image), and we use the same image to compare various approaches in the rest of the thesis. In the figure 3.2, we are trying to predict the best convexes, see (11) Segmented Normal, by comparing the Raytrace Depth (3) with the GT depth (1) with the help of AbsRel.

3.3 IMPROVEMENTS

The baseline convolution network has limitations in capturing long-range dependencies and transferring information on a global scale, which leaves room for improvement. In the next chapter, we will explore various methods to address these limitations and enhance the

performance of the convex decomposition model.

In the subsequent chapters, we will look into the details of our proposed ensemble method. We will present the architecture of the individual models, the training process, and the ensemble method. We will also conduct extensive experiments to evaluate the performance of our approach and compare it against the baseline and other state-of-the-art methods.

CHAPTER 4: METHODS

4.1 OVERVIEW

The baseline convolution network has limitations in capturing long-range dependencies and transferring information on a global scale. We hypothesize that by providing the decoder with additional information about the depth map and global features, we can improve the model's performance. There are several approaches to supply this supplementary information, which we refer to as the summary.

Given the diverse nature of indoor scenes, with variations in wall sizes, floor areas, and object densities, it is challenging for a single model to accurately decompose the image into simple primitives with good AbsRel in all scenarios. Instead of relying on a single model, we propose an ensemble approach where multiple models are trained, each potentially specializing in different aspects of the scene. During evaluation, we select the best-performing model based on its ability to generate accurate depth predictions for the given input image.

By utilizing ground truth depth maps or predictions from state-of-the-art depth predictors like MiDaS [5] during the evaluation phase, we can compare the depth predictions of our various models against the reference and choose the most accurate one for each input image.

The core of our analysis is grounded in the examination of several well-established Convex Decomposition Algorithms. We will see Volumetric Hierarchical Approximate Convex Decomposition (VHACD), known for its hierarchical decomposition capabilities [2]. Additionally, we will explore the Convex Objects Approximation and Clustering Decomposition (COACD) [3], and Binary Space Partitioning Networks (BSP-Net) [41], use of PointNet [4] as summary, each offering unique perspectives and solutions for different types of scenes.

Following the use cases and results of these algorithms, we will see some Experimental Configurations for Convex Decomposition. This section will discuss which configurations yield the most promising results, enhancing our understanding of the optimal settings for each method. The discussion will be presented in Section 4.3.

Building on the information gained from these experiments, we will introduce ENS-CVXNet, a Convex Decomposition Network, in Section 4.6. ENS-CVXNet is an ensemble of different networks built on top of the learned-summary baseline method, methods represent our attempt to mix the strengths of various decomposition techniques into a unified, robust framework for Convex Decomposition. Finally, we discover that the ensemble approach, which combines the strengths of various convex decomposition techniques and leverages different summaries, works better than using any individual summary in single model alone.

4.2 DATASET

The NYUv2 dataset [42] is used in our experimentss. The NYUv2 dataset consists of 1449 RGBD images, which we divide into the standard 795-654 train-test split. While 795 images may seem like a relatively small number for training neural networks, we demonstrate that the convex decomposition predicted by the network serves as a good starting point for subsequent refinement, leading to very accurate results.

For our experiments, we often use a small subset of the testing images to find the best hyperparameters, as evaluating on the entire set of 654 test images can be computationally expensive and time-consuming. We will specify the subset size for each experiment where applicable.

Our method is implemented using TensorFlow, and we train our models on an NVIDIA A40 GPU. We utilize the Adam optimizer with a learning rate of 0.0001. Given the parameters of a collection of convex components for a test image, we can perform ray marching from the original viewpoint to obtain a depth map, part segmentation, and normals. Our method involves ray marching and interval halving at the first intersection point. For evaluation, we compare the depth associated with the predicted convex decomposition against the input depth map, using the standard depth error metrics on the 654 NYUv2 test images from [43].

Our training procedure requires depth maps with known camera calibration parameters. Following previous scene parsing work, we focus on the NYUv2 dataset [42], as it provides the necessary depth information and calibration parameters.

4.3 CONVEX DECOMPOSITION ALGORITHMS

4.3.1 VHACD

Volumetric Hierarchical Approximate Convex Decomposition (VHACD) was applied directly on the mesh of the input image to decompose it into an initial set of primitives. These primitives were then used as a summary input to the baseline model. Different hyperparameters in VHACD were experimented with to decompose the mesh into convex primitives, keeping the number of primitives smaller than 14. The reason for using 14 primitives is to maintain consistency with the baseline model, which uses 14 primitives in their final results. This allows for a fair comparison between the proposed method and the baseline. Interestingly, using VHACD alone, results close to the baseline model were achieved but with complex primitives, suggesting that the output of VHACD can used as a summary to guide the baseline model in finding better convex decompositions.

Figure 4.1 demonstrates an example of applying VHACD to the mesh of the original image. The results obtained using different hyperparameters are listed in Appendix A.2. Although the convexes generated by VHACD have complex and arbitrary shapes, they can sometimes accurately represent the locations of objects in the scene.

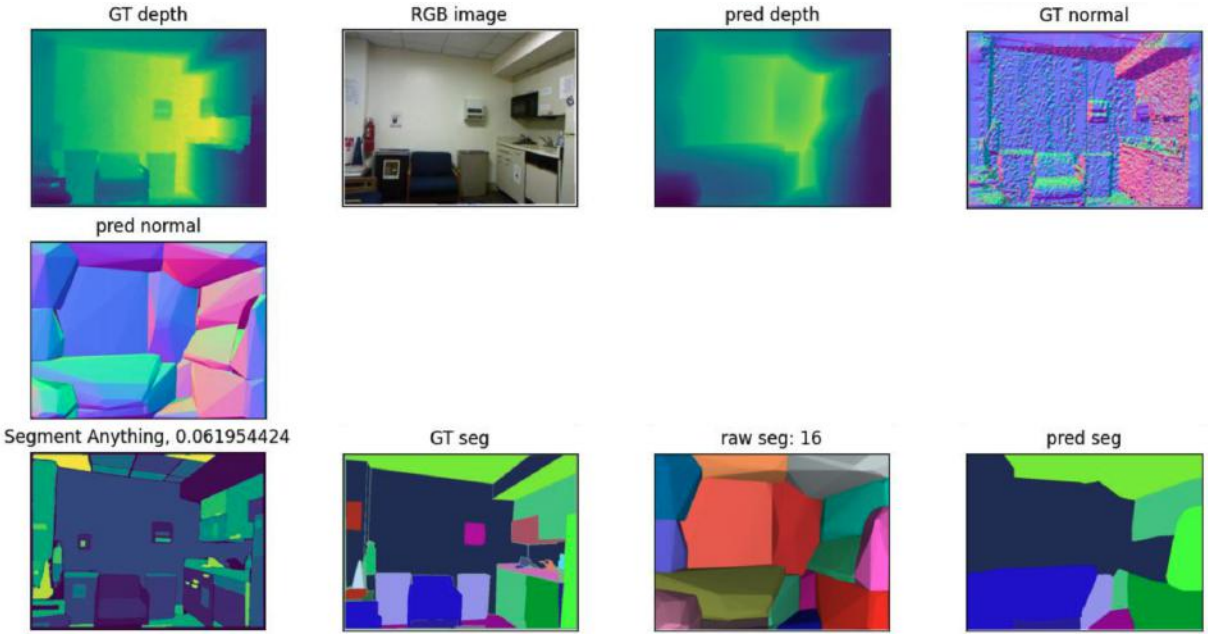


Figure 4.1: This figure shows the results of applying VHACD directly on the mesh of the input image, generating a convex decomposition consisting of 16 primitives. When compared to the baseline model in Figure 3.2, it is evident that the convexes generated by VHACD in the segmented normals have more complex and arbitrary shapes. However, these VHACD-generated convexes occasionally align well with the actual locations of objects in the scene. The AbsRel achieved by VHACD in this example is 0.061, which is comparable to the baseline model shown in Figure 3.2. It is important to highlight that VHACD generates complex primitives with only multiple faces, unlike baseline.

4.3.2 COACD

Secondly, Convex Objects Approximation and Clustering Decomposition (COACD) was employed to decompose a given mesh into a set of nearly convex components. COACD offers an advantage over VHACD by capturing both the global structure and intricate details of input shapes. This addresses a common issue in conventional approaches that may overlook certain features. COACD introduces a collision-aware concavity metric that robustly assesses approximation errors by considering the distance between a shape and its convex hull.

Extensive testing and experimentation were conducted with different hyperparameters in COACD, eliminating the need for network interference. The results demonstrated that

using COACD alone achieves performance comparable to the baseline model and surpasses VHACD. This finding suggests that COACD results can be leveraged as references to enhance the baseline model's ability to identify more efficient convex representations. Figure 4.2 illustrates an example of applying COACD to the mesh of the original image. The results of different hyperparameters are listed in Appendix A.1.

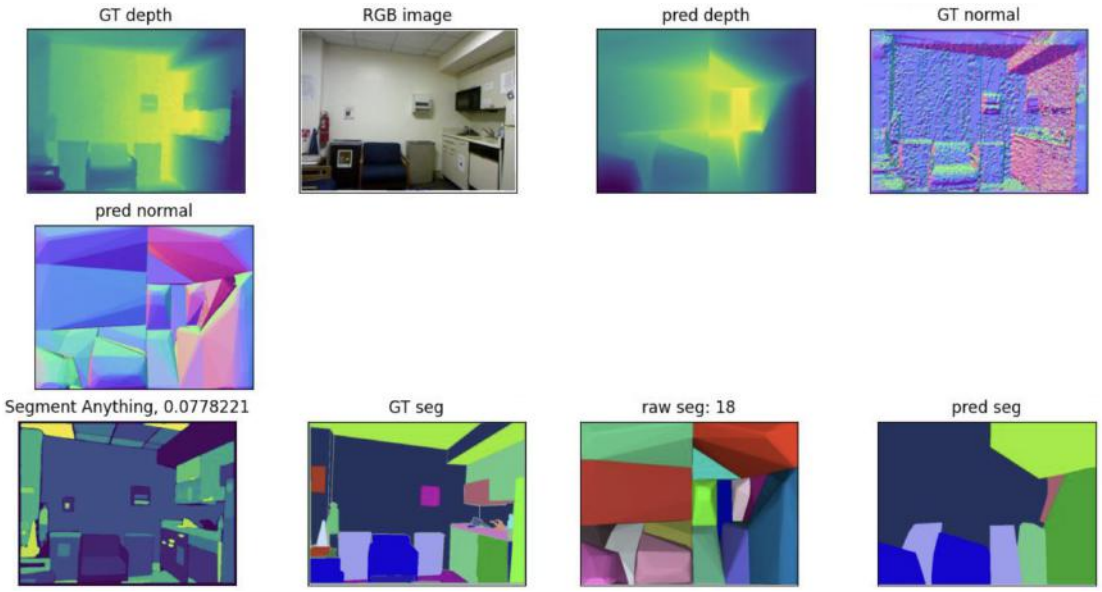


Figure 4.2: This figure shows the results of applying COACD directly on the mesh of the input image, generating a convex decomposition with 18 primitives. When compared to the baseline model's output in figure 3.2, the convexes produced by COACD in the segmented normals exhibit more complex and arbitrary shapes. It also increases the number of faces of a convex (in hundreds) when compared to the baseline (Parallelepiped has 6 faces). However, these COACD-generated convexes often accurately represent the locations of objects in the scene. Moreover, the average AbsRel obtained by COACD across all images in the test dataset surpasses that of VHACD.

Figure 4.2 shows an example of how we apply VHACD to the mesh of the original image, and the results of different hyperparameters are listed in appendix A.1.

As seen in the figure, COACD can accurately detect the chair and dustbin, unlike VHACD in Figure 4.1. Comparing both tables reveals that COACD performs better than VHACD. Based on the results in Appendix A.1, the best-performing hyperparameters have been fixed for subsequent experiments.

4.4 POST-PROCESSING AFTER DECOMPOSITION

After initially decomposing images into convex parts using the COACD method, each

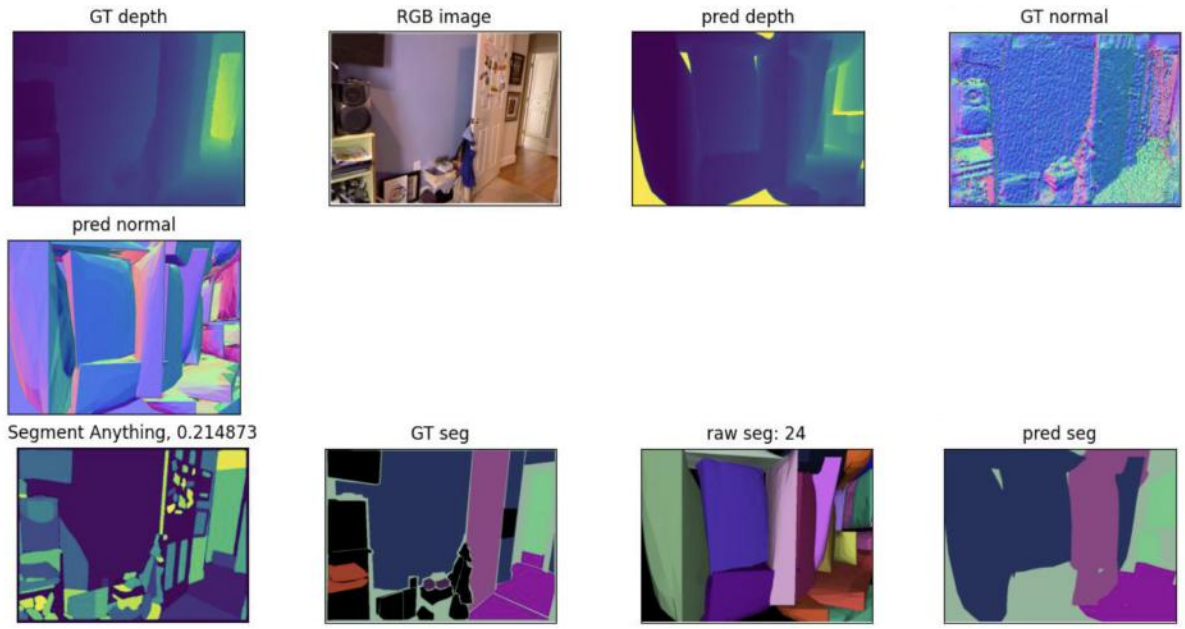


Figure 4.3: This figure shows the results of applying Blender API smoothing as a polishing method to the COACD decomposition shown in Figure 4.2. While smoothing can sometimes improve the AbsRel, it may also introduce holes or gaps in the depth map, and the number of faces of a convex (in hundreds), as evident in this example.

resulting convex part went through further processing using different experimental configurations. These configurations, such as Axis-Aligned Bounding Box, Non-Axis Aligned Bounding Box, Parallelepiped Bounding, and others, were applied to individually refine and analyze the convex components.

4.4.1 Smoothing using Blender

After decomposition, each convex part was smoothed using the Blender software's API in Python. This smoothing step aimed to refine the surface quality of the convex parts by removing any sharp edges or irregularities introduced during the decomposition process. Smoothing contributed to increase the functional quality of the convex parts.

An example of Blender smoothing is shown in figure 4.3. For some curved surfaces, it works better, but majorly the application of blender API as a postprocessing doesn't help much in improving the AbsRel.

4.4.2 Axis-Aligned Bounding Box

Each convex part was enclosed within an Axis-Aligned Bounding Box (AABB). This

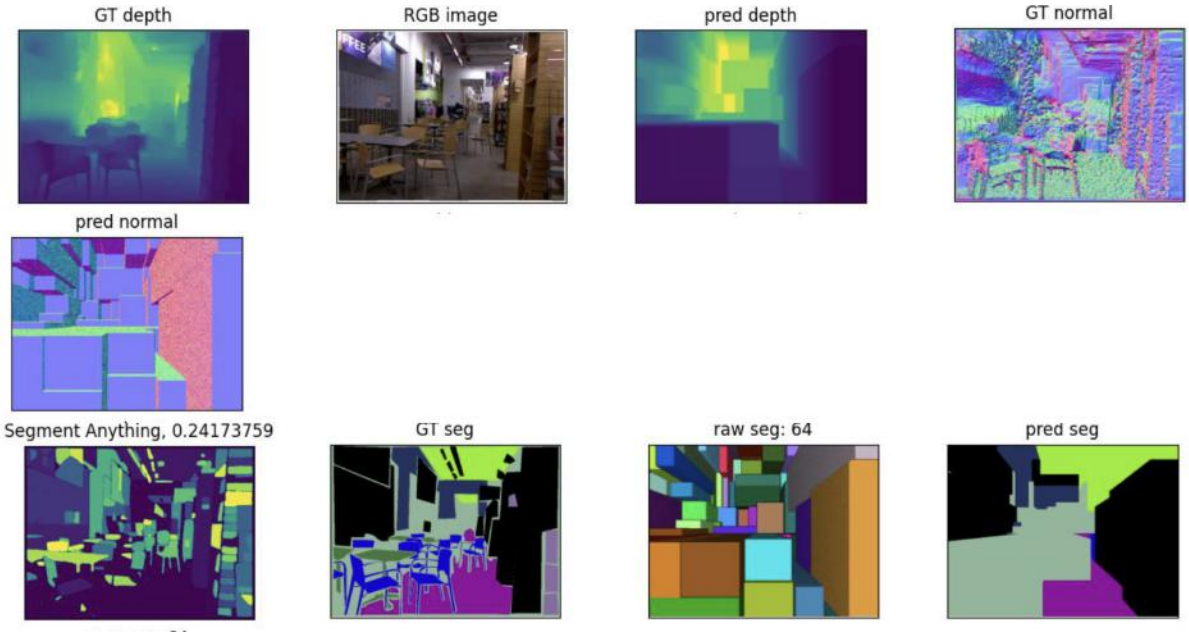


Figure 4.4: This figure shows the Axis-Aligned Bounding Box on the decomposition from COACD as shown in fig 4.2. To compare it with the baseline, we tried to compress the convex into a simple smallest volume Axis-Aligned Bounding Box. As you can see in this image, it generates a lot of small cuboids to maintain the AbsRel, increasing the count of simple primitives, unlike what we wanted to get the convex decomposition with the least number of Parallelepiped.

approach facilitated the spatial organization and efficient collision detection for the convex parts. By aligning the bounding boxes with the coordinate axes, computational simplicity was maintained, like baseline.

To find the AABB, given the centroid of each convex from the Open3D library, the code was written to determine the axis-aligned coordinates. The AABB method creates a bounding box that is aligned with the coordinate axes. It starts by finding the minimum and maximum coordinates of the mesh vertices along each axis (x, y, z). These minimum and maximum coordinates define the extents of the bounding box. Then it defines eight vertices of the bounding cube using these minimum and maximum coordinates. The vertices represent the corners of the bounding cube

Figure 4.4 illustrates an AABBs to the convex parts obtained from the decomposition process. It has been observed that the AABB approach tends to perform better when the number of final convexes is relatively small. To further enhance the results, various greedy approaches were explored to merge different Axis-Aligned Boxes. Additionally, postprocessing techniques were applied, which occasionally led to improvements in the AbsRel, particularly in cases with a limited number of boxes.

4.4.3 Non-Axis Aligned Bounding Box

Alternatively, Non-Axis Aligned Bounding Boxes (NAABBs) were applied to each convex part, offering a tighter and more customized fit than AABBs. This configuration was especially useful for parts with complex geometries, as the bounding box could adapt to the part's orientation and shape, potentially reducing the empty space within the bounding box and increasing the precision of spatial queries and collision detection.

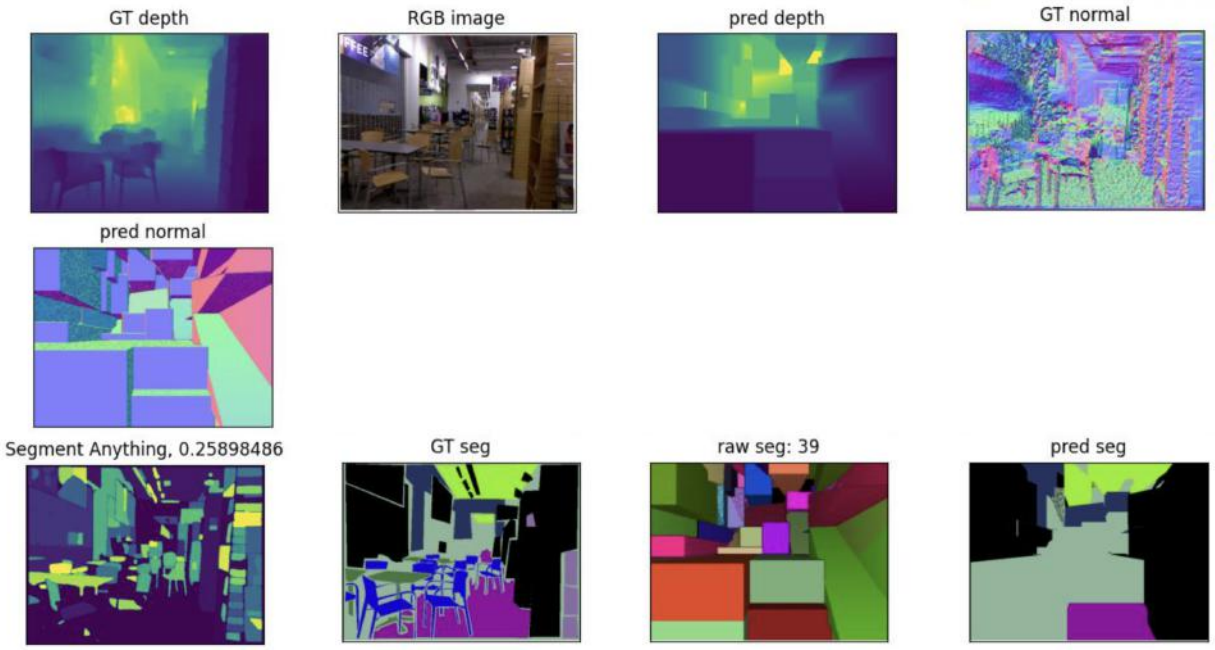


Figure 4.5: This figure shows the use of Non-Axis-Aligned Bounding Box on the decomposition from COACD as shown in fig 4.2, in order to compare it with the baseline. As you can see in this image, it works better than AABB 4.4, in terms of number of primitives while maintaining AbsRel, but still generates a lot of small parallelepipeds to maintain the AbsRel, increasing the count of simple primitives, unlike what we wanted to get the convex decomposition with the least number of Parallelepipeds.

An example of NAABBs is shown in figure 4.5. It works better than AABB in terms of overall AbsRel. The NAABBs method creates a bounding box that is oriented based on the principal axes of the mesh. We implement the code, it begins by calculating the center of the mesh vertices. Then, it centers the vertices by subtracting the center coordinates from each vertex. This step translates the mesh to the origin. Next, the covariance matrix of the centered vertices is computed. The covariance matrix provides information about the spread and orientation of the vertices. Eigenvalues and eigenvectors of the covariance matrix are calculated. The eigenvectors represent the principal axes of the mesh, and the eigenvalues indicate the variance along each axis. The eigenvectors are sorted based on the

eigenvalues to align the bounding box with the principal axes. The mesh is then rotated to align with the eigenvectors using a rotation matrix. The minimum and maximum points of the aligned vertices are calculated to determine the extents of the oriented bounding box (OBB). Eight corners of the OBB are computed in the aligned coordinates. These corners are then transformed back to the original coordinate system.

4.4.4 With Parallelepiped Bounding

In some cases, parallelepiped bounding was applied to certain convex parts, providing a three-dimensional bounding geometry that could be oriented in any direction. This flexibility allowed for an even closer fit to the convex parts than NAABBs, which could be particularly advantageous for optimizing spatial efficiency and collision detection accuracy in densely populated 3D environments.

We implements the Parallelepiped method to find the minimum bounding box of a set of 3D points. The code takes the convex hull points and an angle increment as input and iteratively rotates the point set around the x-axis and z-axis using the specified angle increment. At each rotation, a rotation matrix is constructed, and the convex hull points are transformed by applying the rotation matrix. The minimum and maximum values of the rotated points along each axis (x, y, z) are calculated to determine the width, height, and depth of the bounding box. The volume of the bounding box is computed as the product of these dimensions and stored for each rotation. After exploring all possible rotations, the code reconstructs the rotation matrix associated with the minimum bounding box and projects the convex hull points onto the rotated frame. The center point and corner points of the bounding box are calculated based on the minimum and maximum coordinates of the projected points. Finally, it returns the optimal rotation angles, and the projected corners.

4.4.5 Original Convex Decomposition

For comparison purposes, the convex primitives were left in their original state after decomposition, without any further bounding or smoothing. This approach served as a control to assess the impact and necessity of additional post-processing steps on the utility and performance of the convex parts in practical applications.

Appendix A.3 shows that after obtaining convex primitives from COACD, a post-processing step converts them into parallelepipeds for comparison with the baseline model's output. The last column denotes different configurations tested on the same hyperparameter, to assess which one works the best. Although this combination doesn't outperform the baseline yet,

it suggests using COACD primitives as a summary input to the baseline could improve its AbsRel metric.

4.5 ENHANCING THE BASELINE WITH GEOMETRIC SUMMARY

From the last section, we have fixed COACD as the method for summary estimation. This section will compare the baseline with the summary-infused baseline on different types of summaries.

Figure 4.6 illustrates the architecture of how the summary is passed through the encoder-decoder network. The output of COACD is stacked with the output of the decoder and then passed to the encoder. This allows the network to incorporate the geometric information from the COACD (summary) into the learning process.

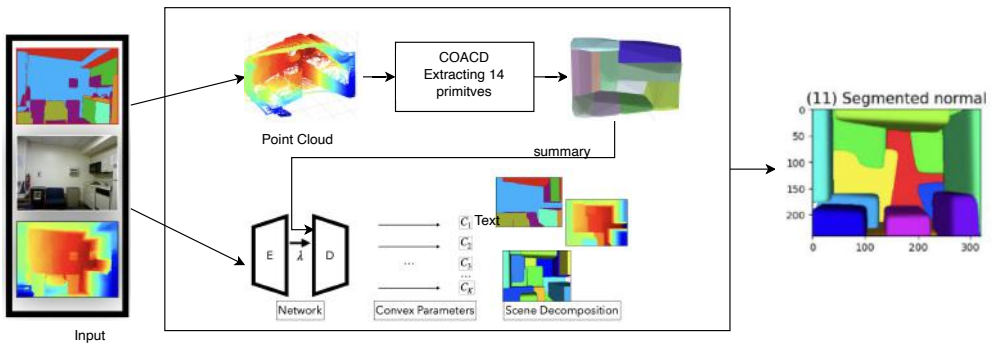


Figure 4.6: Architecture of passing the COACD summary through the encoder-decoder network. The output of COACD is stacked with the output of the decoder and then passed to the encoder, allowing the network to incorporate the geometric information from the summary.

In the following subsections, we will explore various approaches that can lead to different summaries being passed to the network. These approaches aim to provide the network with additional geometric information to enhance its performance in convex decomposition tasks.

4.5.1 Using the means of each convex from COACD as summary

We saw that the convex decomposition from COACD will give somewhat exact locations of different objects, using the center of this convex we pass this to the network as an addition input stacked with the encoder output to the decoder of the architecture. Figure 4.7 represents how it looks like after final convex decomposition.

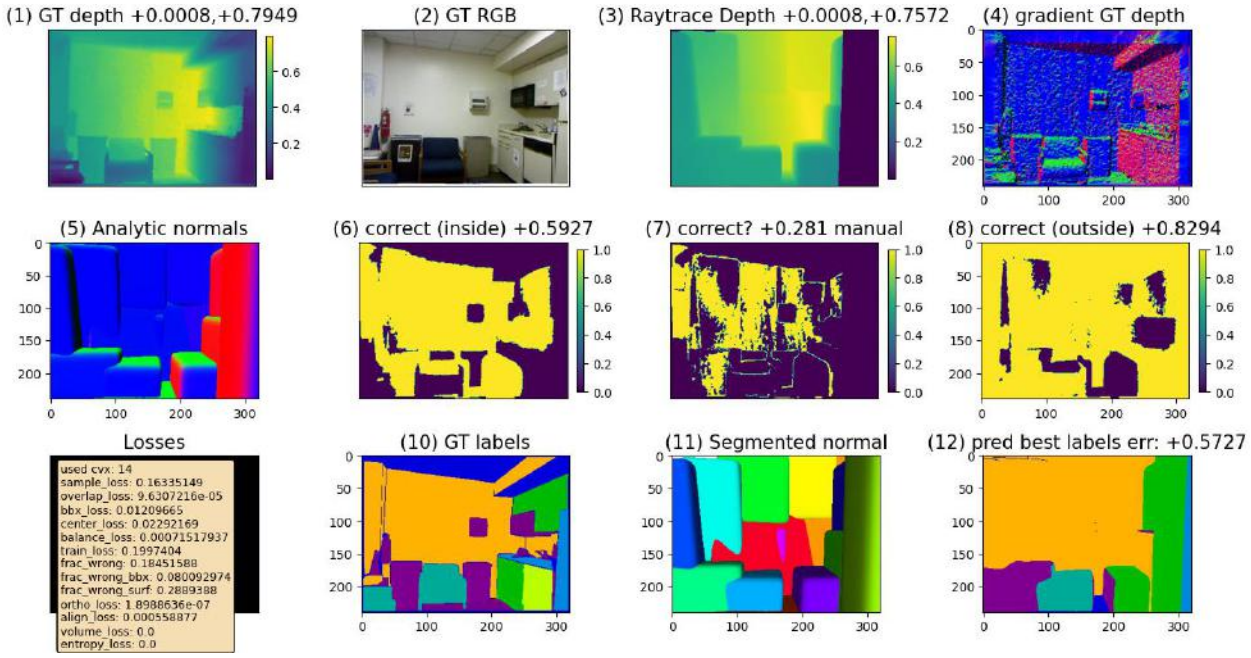


Figure 4.7: This figure shows the final convex decomposition after using the mean coordinates of each convex primitive from COACD as summary to the baseline network. The image shows the input RBG image (1) and the predicted convex components (11), where the centroids of the convex components are used as summary, and might be aligned with the mean locations obtained from COACD predicted from the network

4.5.2 Using the median of each convex from COACD as summary

Similar to using the means of the convex primitives from COACD, we can utilize the medians of these convex components as summary for the baseline model. The convex decomposition from COACD provides approximate locations of different objects in the scene. By calculating the median point of each convex, we can pass these median coordinates as additional input to the network, stacked with the encoder output before feeding into the decoder of the baseline architecture. Figure 4.8 illustrates how the final convex decomposition looks after incorporating the medians of the convex components as summary.

This approach, using the medians instead of the means, could potentially provide a different perspective or representation of the convex components' locations, which yield some improvements when compared to the mean, leading to a 0.0057 reduction in AbsRel (from 0.1144 to 0.1087) and a 0.0231 reduction in RMSE (from 0.5334 to 0.5103) when evaluated on a subset of test images, as shown in Table 4.1 that shows the AbsRel and RMSE of a subset of test images to compare these three different approaches. The integration of the summary to the baseline does not improve the AbsRel by much, but in some cases, it improves a lot.

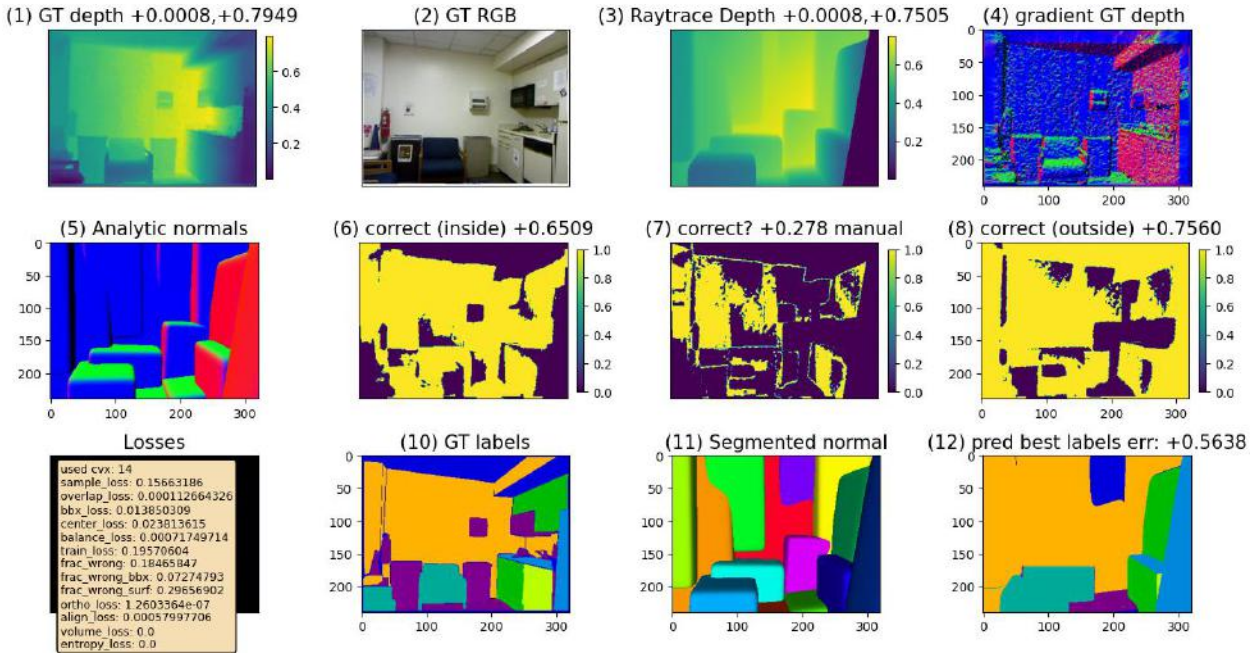


Figure 4.8: This figure shows the final convex decomposition after using the median coordinates of each convex primitive from COACD as summary to the baseline network. Compared to using the mean coordinates as summary, the median approach better captures the representative location of each convex component.

Model	AbsRel	RMSE
Baseline	0.0634	0.3020
Baseline with COACD Mean	0.1144	0.5334
Baseline with COACD Median	0.1087	0.5103

Table 4.1: Comparison of models based on AbsRel and RMSE metrics. Note that this comparison of models is on a subset of the test data for computational efficiency.

4.5.3 Using PointNet as summary

PointNet is a proven neural network architecture [44], designed to process point cloud directly, making it highly versatile and applicable to various tasks involving 3D point clouds. In our approach, we use the capabilities of PointNet by extracting the global features from the input point cloud, as shown in Figure 4.9 from the PointNet paper [4]. These global features effectively encode the geometric information and shape characteristics of the point cloud.

a) Directly using PointNet network's global features as summary

Initially, we passed 32 output features from PointNet to the decoder without training this with the baseline. On the test set, we saw an AbsRel of 0.101 and RMSE of 0.475.

b) Integrating PointNet Architecture to the network decoder, with end-to-end Training

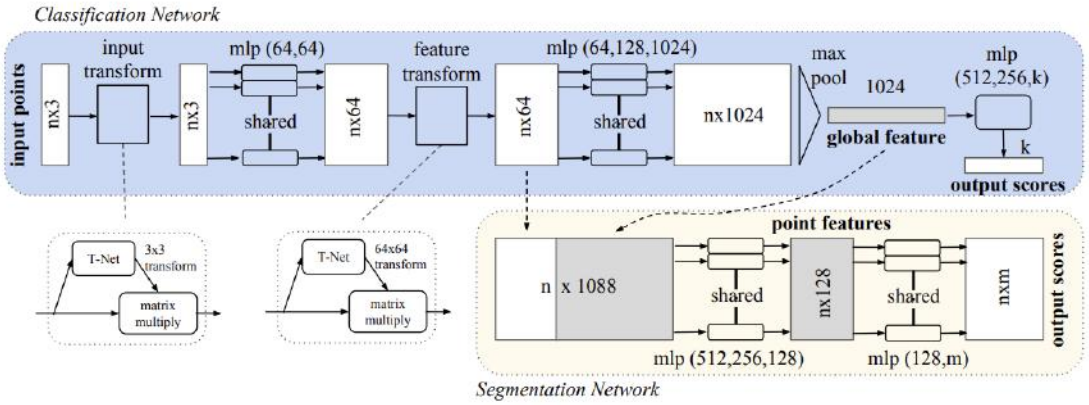


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. "mlp" stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

Figure 4.9: The architecture of PointNet, illustrating the process of global feature extraction from point clouds. This figure is taken from [4].

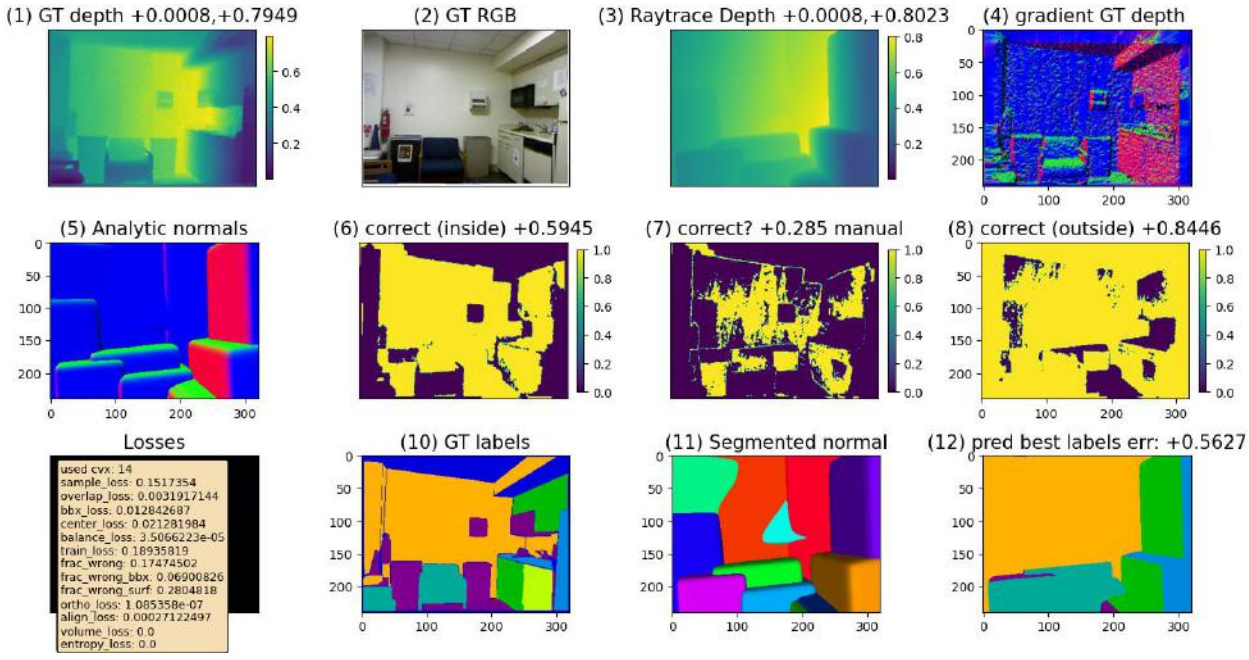


Figure 4.10: This figure shows the use of pre-trained PointNet network's 32 global output features as summary to the baseline model's decoder, without any training of the PointNet architecture. This approach achieved an AbsRel of 0.101 and an RMSE of 0.475 on the test set, indicating that while the PointNet features provide some useful information, their performance is limited without being fine-tuned jointly with the baseline model.

Subsequently, we enabled end-to-end training with the defined loss, maintaining the 32 output features. This approach slightly improved performance, yielding an AbsRel of 0.100 and an RMSE of 0.467. This can be seen in the following figure 4.11

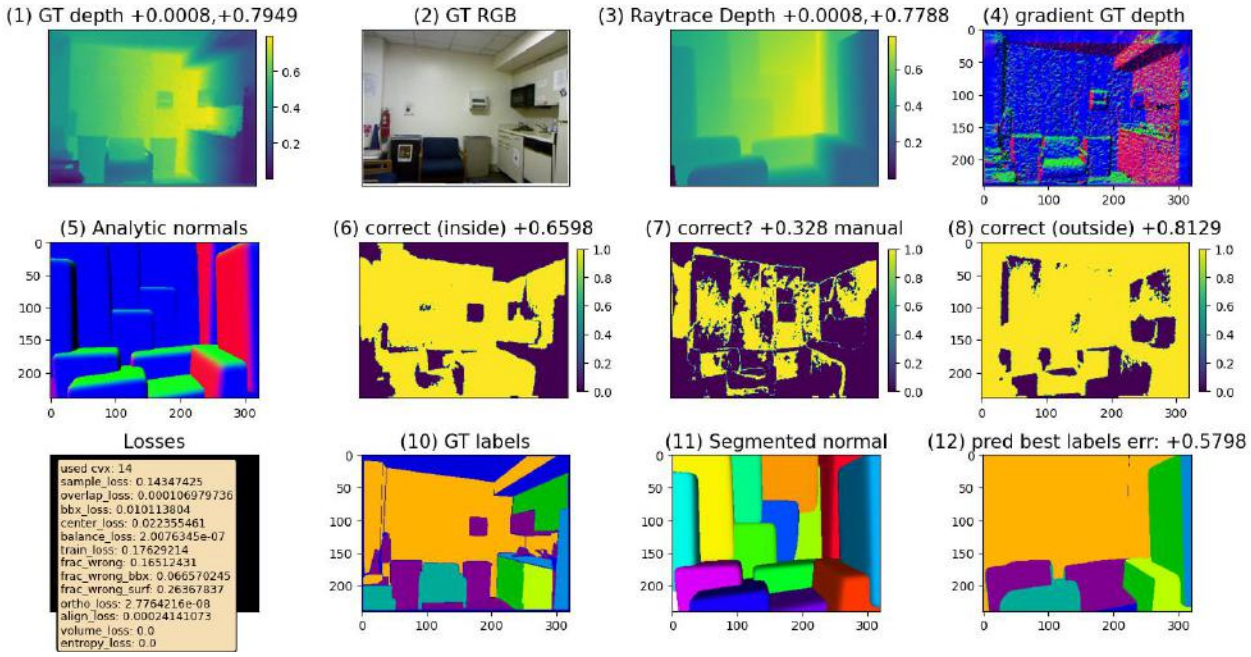


Figure 4.11: This figure shows the integration of the PointNet architecture into the baseline model's decoder and enabling end-to-end training with the defined loss function, while maintaining the 32 output features from PointNet. This approach slightly improved the performance, yielding an AbsRel of 0.100 and an RMSE of 0.467. The end-to-end training allowed the PointNet features to be fine-tuned jointly with the baseline model, leading to better performance compared to using the pre-trained PointNet features directly.

Expanding the output features from PointNet to 512 during end-to-end training led to a decrease in performance, with an AbsRel of 0.125 and an RMSE of 0.579, as can be seen in figure 4.12. This is due to the fact that the network is learning too many details neglecting the superficial global features.

c) Integrating PointNet with end-to-end Training and different Normalization

Introducing normalization like batch normalization to the 32 output features network during end-to-end training further improved the metrics, with an AbsRel of 0.095 and an RMSE of 0.456, as shown in the following figure.

Comparing different PointNet experiments that we tried, table 4.2 below summarizes the outcomes of all experiments, providing a clear comparison across different approaches. We see that an end-to-end with 32 output features and batch normalization improves the network for some specific cases where the depth is not too much.

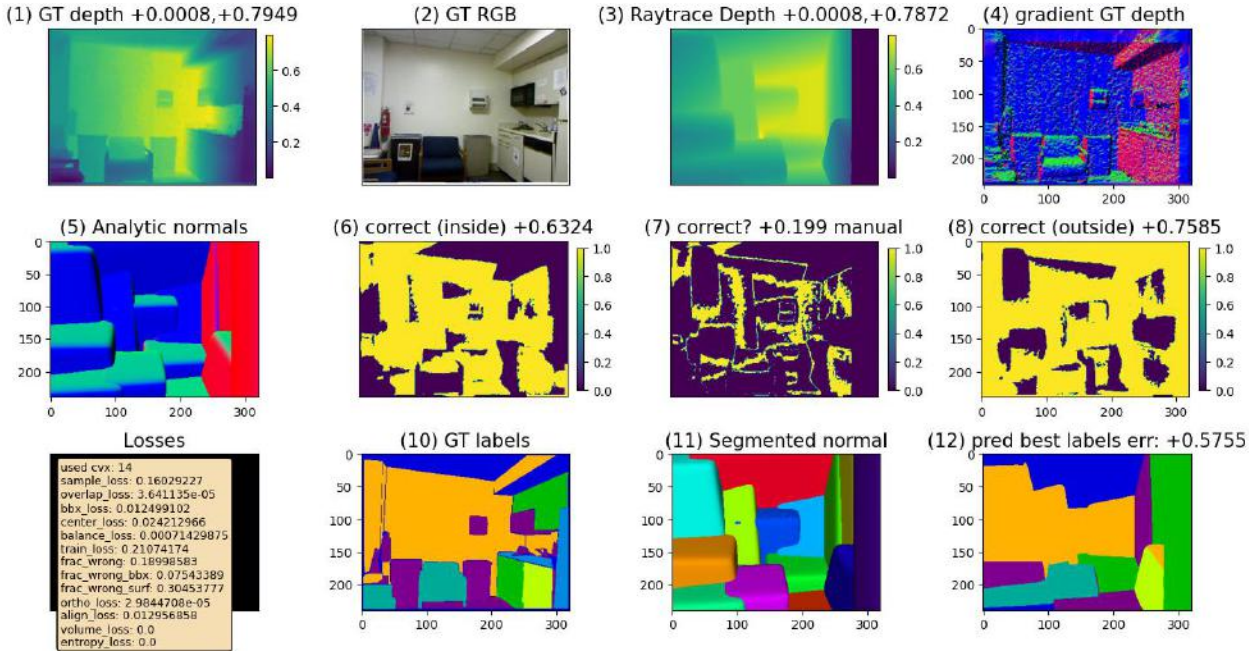


Figure 4.12: Visualization of the results obtained when expanding the output features from PointNet to 512 during end-to-end training with the baseline model. This approach led to a decrease in performance, with an AbsRel of 0.125 and an RMSE of 0.579. The increased number of output features allowed the network to learn too many details, neglecting the superficial global features, which negatively impacted the overall performance.

Experiment	AbsRel	RMSE
Non-Train with 32 Features	0.101	0.475
End-to-End with 32 Features	0.100	0.467
End-to-End with 32 and Batch Normalization	0.095	0.456
End-to-End with 512 Features	0.125	0.579
Forcing Zero Output	0.098	0.453
Baseline	0.093	0.452

Table 4.2: Summary of PointNet experiments. Note- These experiments are on the full test dataset.

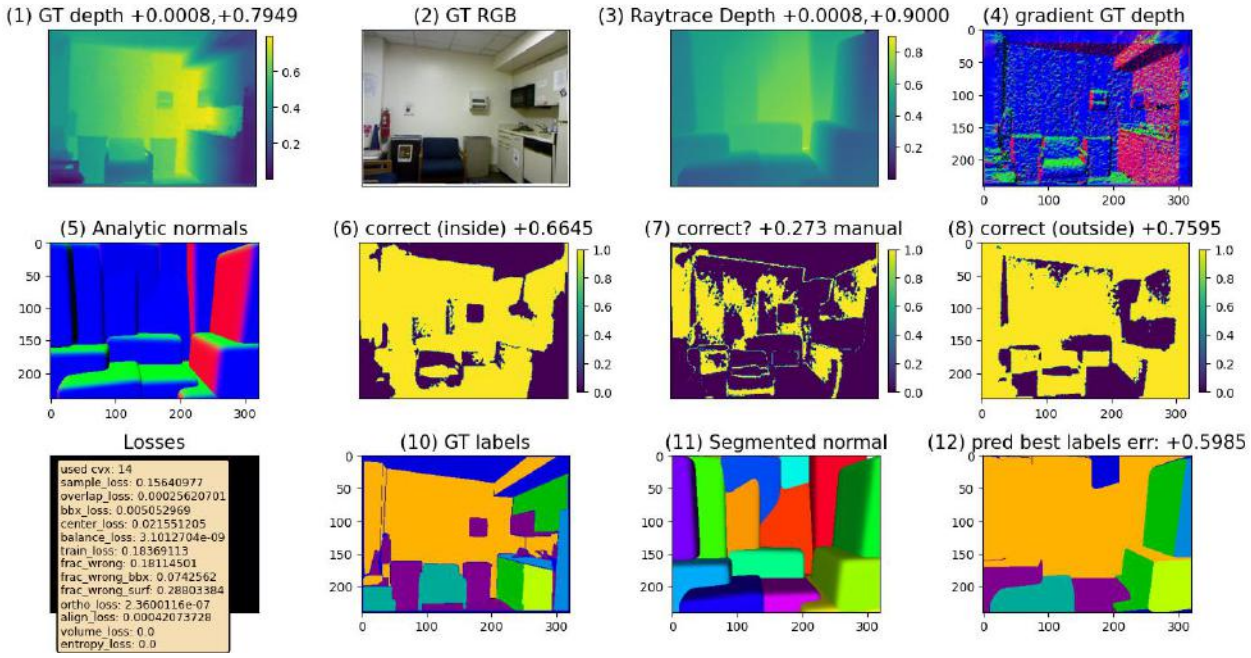


Figure 4.13: Results of introducing normalization, specifically batch normalization, to the 32 output features from PointNet during end-to-end training with the baseline model. This approach further improved the metrics, achieving an AbsRel of 0.095 and an RMSE of 0.456. The normalization step helped to stabilize the training process and allowed the network to better leverage the global features extracted by PointNet, leading to improved performance compared to the previous experiments.

4.5.4 Using COACD median as summary and modified loss term

While using the COACD median as a summary input did not significantly improve the model's performance, we explored an alternative approach by incorporating the COACD median into the loss function. Instead of directly passing the COACD median to the decoder, we modified the loss term to account for the squared distance between the output convex median and the COACD median.

Specifically, we added an additional term to the overall loss function, as shown in Equation 4.1:

$$\mathcal{L}\text{total} = \mathcal{L}\text{original} + \lambda_6 \sum_{i=1}^{K} |\text{median}_i^{\text{output}} - \text{median}_i^{\text{COACD}}|^2 \tag{4.1}$$

By incorporating the COACD median information directly into the loss function, we aim to encourage the model to generate convex components whose medians are closer to the COACD medians. This approach leverages the summary knowledge from COACD without explicitly providing it as just input to the decoder, potentially allowing for more effective

integration and utilization of this information during the training process.

4.5.5 Fixing the convex centroid as COACD mean and using as summary to decoder network

In this approach, we fixed the centroid of each convex component predicted by the network to be the mean of the corresponding convex primitive obtained from COACD. We then used these fixed centroids as summary, providing them as additional input to the decoder network of the baseline model. By constraining the convex centroids to the COACD means, we aimed to leverage the accurate location information from COACD, while allowing the network to optimize the remaining parameters (e.g., dimensions, orientations) of the convex components during training.

Figure 4.14 illustrates how the COACD means are used as fixed centroids for the network to predict other parameters of the parallelepipeds. The big blobs represent the COACD mean locations, which are used as the fixed centroid positions for each convex component. The network is then responsible for predicting the remaining parameters, such as the dimensions and orientations of the parallelepipeds, while keeping the centroids fixed at the COACD mean locations. By incorporating this approach, we aim to take advantage of the accurate location information provided by COACD, while still allowing the network to optimize the other parameters of the convex components during the training process.

Table 4.3 shows the results introducing a loss term and fixing the center, on the whole 654 images of test set.

Experiment	λ_6	**AbsRel**	**RMSE**
Centroid Loss			
	0.0001	0.1237	0.5792
	0.00001	0.1652	0.6805
	0.001	0.2119	0.8806
Fixed Centroid			
	0.0001	0.1988	0.7936
	0.00001	0.1932	0.7805
	0.000001	0.1927	0.7601
ICCV Baseline		0.093	0.452

Table 4.3: Results with different Centroid Experiments, on the whole 654 images of test set.

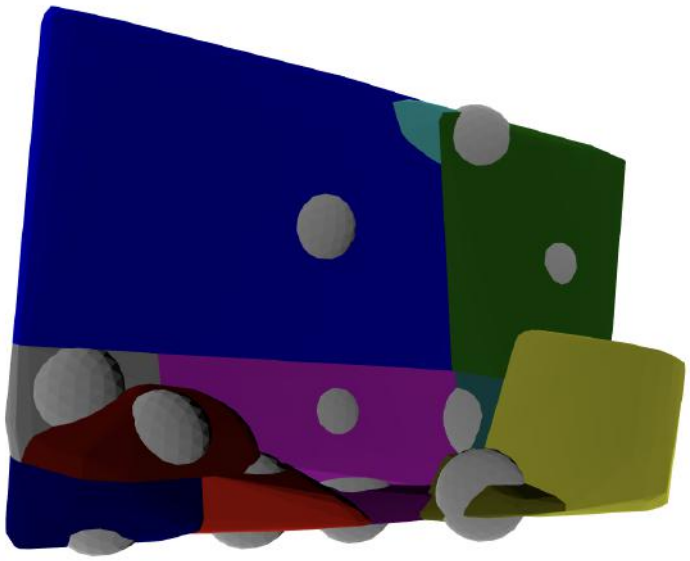


Figure 4.14: This figure shows the use of COACD means as fixed centroids for the network to predict other parameters of the parallelepipeds. The COACD means (big blobs) are used as the fixed centroid locations for each convex component. The network is then tasked with predicting the remaining parameters, such as the dimensions and orientations, while keeping the centroids fixed at the COACD mean locations.

4.6 OUR APPROACH: ENS-CVXNET

We saw that different trials lead to different results, some better for some suited environments. Building on the insights gained from the results of various convex decomposition algorithms and the subsequent enhancements through geometric summary and global feature integration, we propose an ensemble approach **ENS-CVXNet** as shown in figure 4.15.

Figure 4.16a and Figure 4.16b illustrate how ENS-CVXNet works for two different input images. The ensemble generates various outputs from different models, and the one with the best depth prediction is selected as the final output for each image. This selection process is based on comparing the depth predictions of each model with the ground truth depth map, using a reliable off-the-shelf depth predictor like MIDAS. In Figure 4.16a, the Baseline model is selected as the best output for the given input image, while in Figure 4.16b, the PointNet Summary (32 features and Batch Normalization) model is selected as the best output for a different input image. This demonstrates the adaptability of the ENS-CVXNet approach, as it can select different models based on their performance for each specific input image.

For the whole test set, ENS-CVXNet achieves an AbsRel of 0.0744, outperforming the baseline, on 654 test images. By leveraging the strengths of various convex decomposition techniques and selecting the best output based on depth accuracy, ENS-CVXNet improves

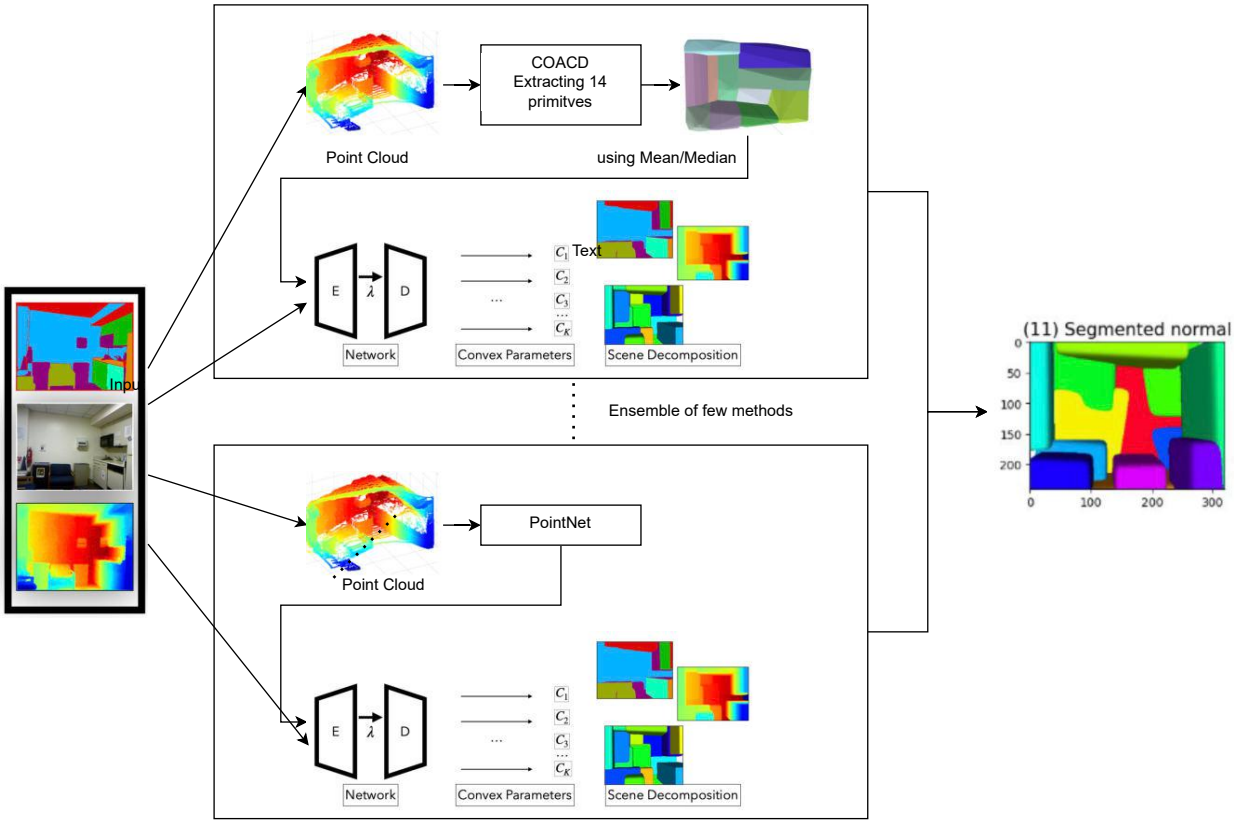
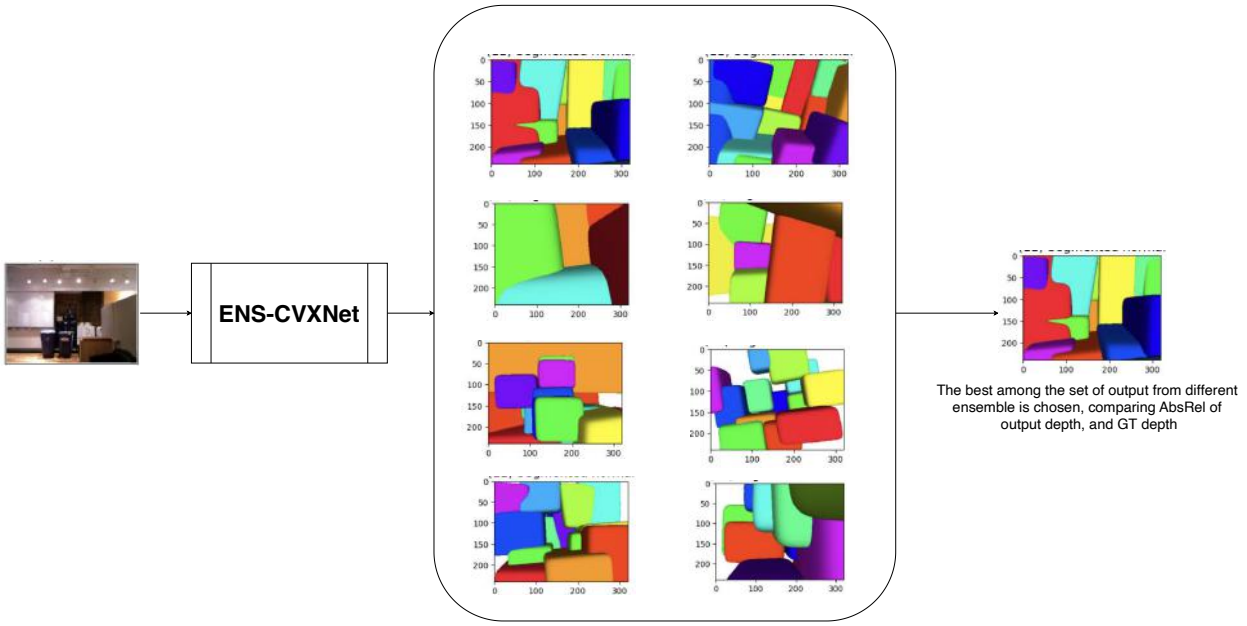
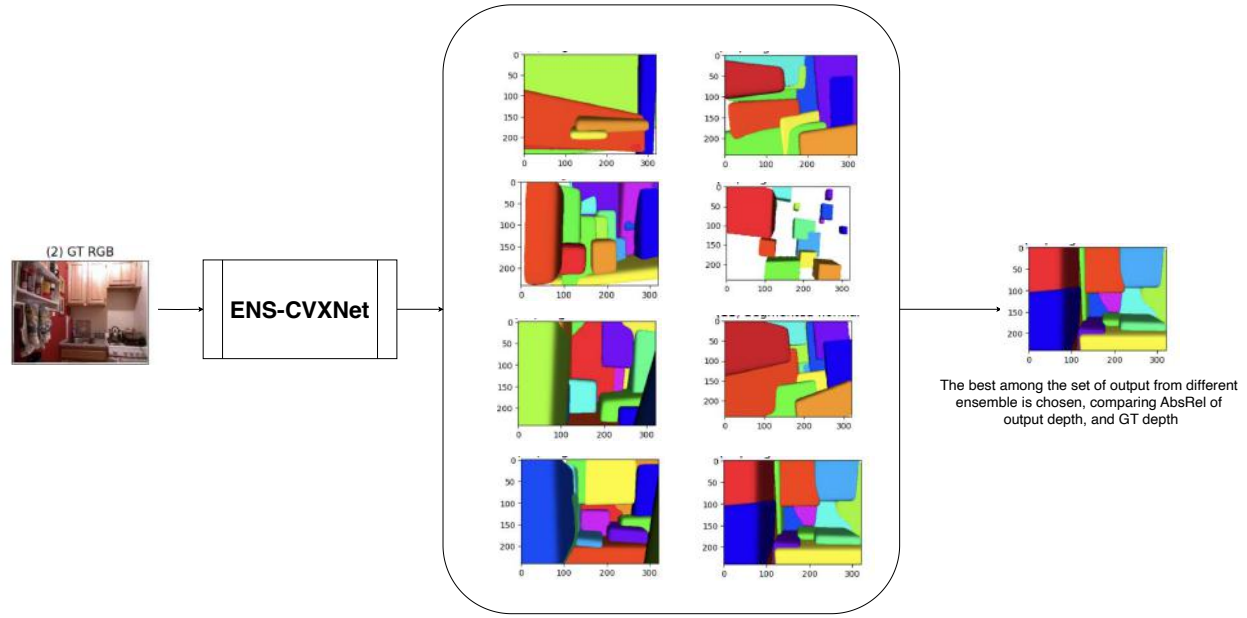


Figure 4.15: This schematic representation illustrates the proposed ENS-CVXNet ensemble approach for better convex decompistion. The input RGB image is processed by multiple convex decomposition models, including the baseline, COACD with mean summary, COACD with median summary, and PointNet, fixing the centers, and with varying configurations. Each model generates a convex decomposition of the input image, which is then evaluated against the ground truth depth map using an off-the-shelf depth predictor (e.g., MIDAS). The model with the lowest AbsRel is selected as the best output for the given input image. By integrating the best-performing methods from various convex decomposition techniques and leveraging depth as a selection criterion, ENS-CVXNet aims to improve the precision and quality of convex decomposition.

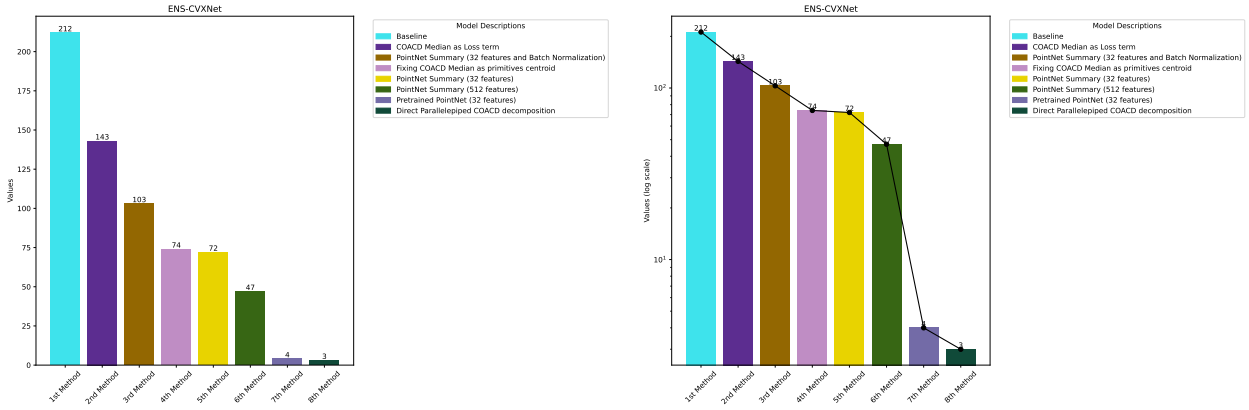


(a) Example 1: Baseline model selected as the best output



(b) Example 2: PointNet Summary (32 features and Batch Normalization) selected.

Figure 4.16: Visualization of the ENS-CVXNet ensemble approach for two different input images. The ensemble generates multiple convex decompositions using different models, with varying configurations. The depth predictions of each model are compared against the ground truth depth map using an off-the-shelf depth predictor (e.g., MIDAS). The model with the lowest AbsRel is selected as the best output for the given input image.



(a) Model selection count

(b) Model selection count (log scale)

Figure 4.17: Bar graphs illustrating the number of times each model is selected by the ENS-CVXNet ensemble approach on the test set. (a) shows the model selection count on a linear scale, while (b) presents the same data on a logarithmic scale for better visualization of less frequently selected models. Out of 50 different models, only 8 models are selected multiple times, with some good models being chosen more frequently. However, a small number of test images still benefit from the selection of different models, highlighting the importance of the ensemble approach in adapting to various scene characteristics. It doesn't hurt to remove some models which have been used fewer times while ensembling.

the overall precision and quality of simple convex decomposition.

By using the ensemble approach and selecting the best output based on GT depth or depth generated from MiDas, ENS-CVXNet achieves an impressive AbsRel of 0.074 on the test images, without any additional polishing steps. This performance surpasses the baseline model, as shown in Table 3.1, demonstrating the effectiveness of our proposed ensemble method in improving the overall accuracy of 3D geometric reconstruction.

Figure 4.17 presents two bar graphs illustrating the number of times each model is selected by the ENS-CVXNet ensemble approach on the full test set. The first graph (Figure 4.17a) shows the model selection count on a linear scale, while the second graph (Figure 4.17b) displays the same data on a logarithmic scale to better visualize the less frequently selected models. Out of the 50 different models in the ensemble, only 8 models are selected multiple times, with some good models being chosen more frequently. However, it is important to note that a small number of test images still benefit from the selection of different models, highlighting the importance of the ensemble approach in adapting to various scene characteristics and improving the overall accuracy of convex decomposition. Finally, we discover that the ensemble approach, which combines the strengths of various convex decomposition techniques and leverages different summaries, works better than using any individual summary in single model alone.

To evaluate the effectiveness of the ensemble approach, we conducted experiments with varying numbers of models included in the ensemble. Table 4.4 presents the AbsRel values achieved when using different ensemble sizes, ranging from a single model (the baseline) to an ensemble of 8 models. The results clearly demonstrate the benefits of the ensemble strategy, as the AbsRel metric consistently improves with the inclusion of more models.

Number of Models	AbsRel
1	0.093
2	0.088
3	0.084
4	0.080
5	0.077
6	0.075
7	0.074
8	0.074

Table 4.4: Impact of ensemble size on the AbsRel metric, demonstrating the consistent improvement in performance as more models are included in the ensemble.

Figure 4.18 shows the outputs of the 6 top-performing models, along with the ground truth, for 8 different input images. Each row corresponds to a single input image, with the ground truth shown in the first column. The remaining columns display the outputs from the 6 top models. The image with the highlighted blue border in each row represents the best output among the 6 models, as selected by the ENS-CVXNet ensemble approach for that particular input image.

This figure illustrates the diversity of outputs produced by the different models in the ensemble, as well as the ability of the ENS-CVXNet approach to select the most accurate convex decomposition for each input image, taking into account the strengths and weaknesses of the individual models in different scenarios.

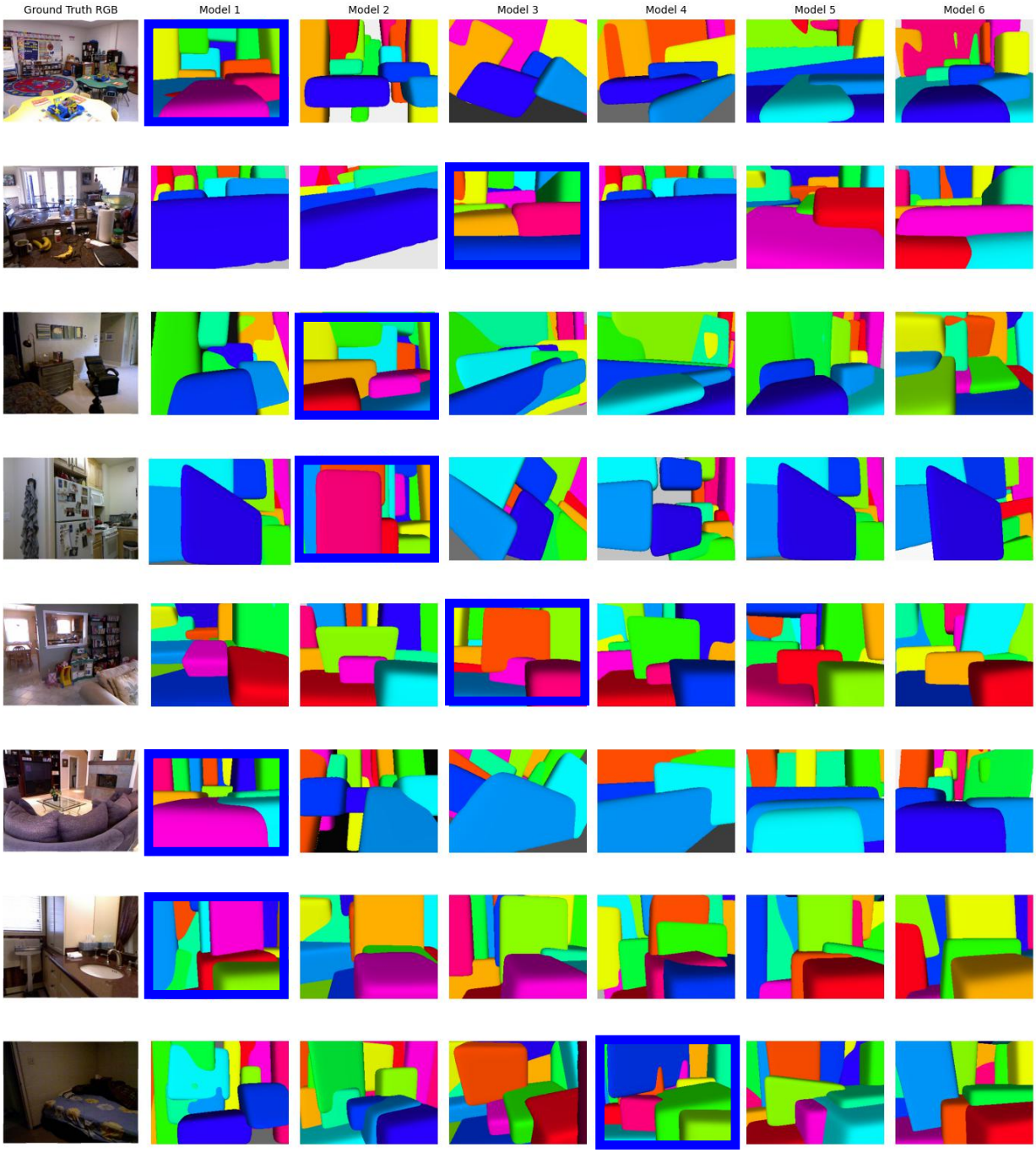


Figure 4.18: Outputs of the 6 top-performing models, along with the ground truth, for 8 different input images. Each row corresponds to a single input image, with the ground truth shown in the first column and the remaining columns displaying the outputs from the 6 top models. The image with the highlighted blue border in each row represents the best output among the 6 models, as selected by the ENS-CVXNet ensemble approach for that particular input image.

CHAPTER 5: CONCLUSION

In this work, we addressed the challenging task of generating accurate convex decompositions of indoor scenes from single RGB images. While current state-of-the-art methods uses encoder-decoder neural networks to convert RGB images into a fixed number of simple primitives, they have limitations in capturing long-range dependencies and transferring global information across diverse indoor environments.

To overcome these limitations, we proposed ENS-CVXNet, an ensemble approach that leverages the strengths of various convex decomposition techniques and incorporates additional geometric information as summaries. Our core analysis explored well-established algorithms such as VHACD, COACD, CVXNet, and BSP-Net, as well as the integration of global features extracted by PointNet from the input point cloud.

By combining multiple models trained with different summaries and configurations, ENS-CVXNet selects the best-performing model for each input image based on its ability to generate accurate depth predictions, evaluated against ground truth depth maps or predictions from state-of-the-art depth predictors. Extensive experiments and evaluations on the NYUv2 dataset demonstrated the effectiveness of our approach, with ENS-CVXNet achieving a 20% decrease in AbsRel from 0.093 to 0.0744, outperforming the baseline method.

The ensemble approach effectively combines the strengths of various models, leveraging geometric summaries and adapting to diverse scene characteristics, thereby improving the overall precision and quality of convex decomposition for indoor scenes. Moreover, the ability to select the most accurate convex decomposition for each input image based on depth predictions allows ENS-CVXNet to tailor its output to the specific characteristics of the scene, further enhancing its performance.

Looking ahead, further research could explore the integration of additional geometric cues, such as segmentation information, to further refine the convex decomposition process.

REFERENCES

[1] V. Vavilala and D. Forsyth, "Convex decomposition of indoor scenes," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 9176–9186.

[2] İ. Demir, D. G. Aliaga, and B. Benes, "Near-convex decomposition and layering for efficient 3d printing," *Additive manufacturing*, vol. 21, pp. 383–394, 2018.

[3] X. Wei, M. Liu, Z. Ling, and H. Su, "Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–18, 2022.

[4] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[5] R. Birkl, D. Wofk, and M. Müller, "Midas v3. 1–a model zoo for robust monocular relative depth estimation," *arXiv preprint arXiv:2307.14460*, 2023.

[6] I. Biederman, "Recognition by components: A theory of human image understanding," *Psychological Review*, 1987.

[7] T. Binford, "Visual perception by computer," *IEEE Conf. on Systems and Controls*, 1971.

[8] L. Roberts, "Machine perception of three-dimensional solids," Ph.D. dissertation, MIT, 1963.

[9] B. Deng, K. Genova, S. Yazdani, S. Bouaziz, G. Hinton, and A. Tagliasacchi, "Cvxnet: Learnable convex decomposition," 2020.

[10] A. van den Hengel, C. Russell, A. Dick, J. Bastian, D. Pooley, L. Fleming, and L. Agapito, "Part-based modelling of compound scenes from images," in *CVPR*, 2015.

[11] R. Nevatia and T. Binford, "Description and recognition of complex curved objects," *Artificial Intelligence*, 1977.

[12] J. Ponce and M. Hebert, "A new method for segmenting 3-d scenes into primitives," in *Proc. 6 ICPR*, 1982.

[13] S. Shafer and T. Kanade, "The theory of straight homogeneous generalized cylinders," Technical Report CS-083-105, Carnegie Mellon University, Tech. Rep., 1983.

[14] S. Tulsiani, H. Su, L. Guibas, A. Efros, and J. Malik, "Learning shape abstractions by assembling volumetric primitives," in *Computer Vision and Pattern Regognition (CVPR)*, 2017.

[15] V. Vavilala and D. Forsyth, "Convex decomposition of indoor scenes," 2023.

[16] V. Hedau, D. Hoiem, and D. Forsyth, "Recovering the spatial layout of cluttered rooms," in *Proc. ICCV*, 2009.

[17] V. Hedau, D. Hoiem, and D. Forsyth, "Thinking inside the box: Using appearance models and context based on room geometry," in *Proc. ECCV*, 2010.

[18] V. Hedau, D. Hoiem, and D. Forsyth, "Recovering free space of indoor scenes from a single image," in *Proc. CVPR*, 2012.

[19] D. Hoiem, A. Efros, and M. Hebert, "Automatic photo pop-up," *ACM Transactions on Graphics / SIGGRAPH*, vol. 24, no. 3, 2005.

[20] D. Hoiem, A. Efros, and M. Hebert, "Recovering surface layout from an image," *IJCV*, 2007.

[21] A. Gupta, A. Efros, and M. Hebert, "Blocks world revisited: Image understanding using qualitative geometry and mechanics," in *ECCV*, 2010.

[22] C. Zou, A. Colburn, Q. Shan, and D. Hoiem, "Layoutnet: Reconstructing the 3d room layout from a single rgb image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[23] S. Stekovic, S. Hampali, M. Rad, S. Deb Sarkar, F. Fraundorfer, and V. Lepetit, "General 3d room layout from a single view by render-and-compare," in *European Conference on Computer Vision*. Springer, 2020, pp. 187–201.

[24] C. Liu, J. Yang, D. Ceylan, E. Yumer, and Y. Furukawa, "Planenet: Piece-wise planar reconstruction from a single rgb image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2579–2588.

[25] D. Fouhey, A. Gupta, and M. Hebert, "Data-driven 3d primitives for single image understanding," in *ICCV*, 2013.

[26] H. Jiang, "Finding approximate convex shapes in rgbd images," in *European Conference on Computer Vision*. Springer, 2014, pp. 582–596.

[27] F. Kluger, H. Ackermann, E. Brachmann, M. Yang, and B. Rosenhahn, "Cuboids revisited: Learning robust 3d shape fitting to single rgb images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[28] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Comm. ACM.*, vol. 24, no. 6, pp. 381–395, 1981.

[29] C. Zou, E. Yumer, J. Yang, D. Ceylan, and D. Hoiem, "3d-prnn: Generating shape primitives with recurrent neural networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 900–909.

[30] M. Gadelha, G. Gori, D. Ceylan, R. Mech, N. Carr, T. Boubekeur, R. Wang, and S. Maji, "Learning generative models of shape handles," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 399–408.

[31] D. Roberts, A. Danielyan, H. Chu, M. Fard, and D. Forsyth, "Lsd-structurenet: Modeling levels of structural detail in 3d part hierarchies," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 5816–5825.

[32] D. Smirnov, M. Fisher, V. Kim, R. Zhang, and J. Solomon, "Deep parametric shape predictions using distance fields," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 558–567.

[33] C. Sun and Q. Zou, "Learning adaptive hierarchical cuboid abstractions of 3d shape collections," *ACM Transactions on Graphics (TOG)*, vol. 38, pp. 1–13, 2019.

[34] D. Paschalidou, A. Ulusoy, and A. Geiger, "Superquadrics revisited: Learning 3d shape parsing beyond cuboids," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10 336–10 345.

[35] Z. Chen, A. Tagliasacchi, and H. Zhang, "Bsp-net: Generating compact meshes via binary space partitioning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 42–51.

[36] L. Li, M. Sung, A. Dubrovina, L. Yi, and L. Guibas, "Supervised fitting of geometric primitives to 3d point clouds," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2647–2655.

[37] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *International Conference on Learning Representations*, 2017.

[38] V. Vavilala, S. Jain, R. Vasanth, A. Bhattad, and D. Forsyth, "Blocks2world: Controlling realistic scenes with editable primitives," 2023.

[39] B. Deng, K. Genova, S. Yazdani, S. Bouaziz, G. Hinton, and A. Tagliasacchi, "Cvxnet: Learnable convex decomposition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 31–44.

[40] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 12 179–12 188.

[41] Z. Chen, A. Tagliasacchi, and H. Zhang, "Bsp-net: Generating compact meshes via binary space partitioning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 45–54.

[42] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V 12.* Springer, 2012, pp. 746–760.

[43] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh, "From big to small: Multi-scale local planar guidance for monocular depth estimation," *arXiv preprint arXiv:1907.10326*, 2019.

[44] B. Zhang, S. Huang, W. Shen, and Z. Wei, "Explaining the pointnet: What has been learned inside the pointnet?" in *CVPR Workshops*, 2019, pp. 71–74.

APPENDIX A: EXPERIMENTAL RESULTS FOR CONVEX DECOMPOSITION ALGORITHMS

Table 2.1: Original COACD Convex Decomposition

ID	Error Threshold	MCTS Iterations	MCTS Max Depth	MCTS Nodes	Resolution	Thickness	AbsRL	Avg Number of Convex
1	0.20	100.00	3.00	20.00	2000.00	0.05	0.17	10.03
2	0.10	100.00	3.00	20.00	2000.00	0.05	0.12	39.47
3	0.10	100.00	3.00	20.00	1000.00	0.05	0.12	39.42
4	0.10	100.00	3.00	30.00	2000.00	0.05	0.12	39.17
5	0.10	100.00	3.00	20.00	5000.00	0.05	0.12	40.02
6	0.10	100.00	3.00	20.00	2000.00	0.10	0.12	41.80
7	0.10	100.00	5.00	20.00	2000.00	0.05	0.12	38.47
8	0.05	100.00	3.00	20.00	2000.00	0.05	0.10	124.52
9	0.15	1000.00	3.00	20.00	5000.00	0.05	0.14	18.17
10	0.40	100.00	3.00	20.00	2000.00	0.05	0.26	2.48
11	0.10	100.00	7.00	20.00	2000.00	0.05	0.12	38.28
12	0.30	1000.00	3.00	20.00	2000.00	0.05	0.21	4.38
13	0.20	1000.00	3.00	20.00	2000.00	0.05	0.17	10.13
14	0.10	500.00	3.00	20.00	2000.00	0.05	0.12	39.36
15	0.15	1000.00	3.00	20.00	1000.00	0.05	0.14	17.75
16	0.15	1000.00	3.00	20.00	2000.00	0.10	0.14	18.97
17	0.10	100.00	3.00	20.00	10000.00	0.05	0.12	40.50
18	0.10	100.00	3.00	40.00	2000.00	0.05	0.12	39.67
19	0.25	1000.00	3.00	20.00	2000.00	0.05	0.19	6.39
20	0.10	100.00	4.00	20.00	2000.00	0.05	0.12	40.01
21	0.10	200.00	3.00	20.00	2000.00	0.05	0.12	39.62
22	0.15	1000.00	3.00	20.00	2000.00	0.05	0.14	17.91
23	0.10	100.00	6.00	20.00	2000.00	0.05	0.12	38.76
24	0.15	1000.00	3.00	30.00	2000.00	0.05	0.14	17.66
25	0.10	1000.00	3.00	20.00	2000.00	0.05	0.12	39.09
26	0.15	1500.00	3.00	20.00	2000.00	0.05	0.14	17.93
27	0.15	1000.00	5.00	20.00	2000.00	0.05	0.14	17.33
28	0.15	1000.00	3.00	20.00	10000.00	0.05	0.14	18.27
29	0.15	1000.00	3.00	40.00	2000.00	0.05	0.14	17.84
30	0.15	1000.00	4.00	20.00	2000.00	0.05	0.14	18.38
31	0.15	2000.00	3.00	20.00	2000.00	0.05	0.14	17.91

Table A.1: Trying on different COACD Parameters

Res.	Depth	Concavity	Plane Ds.	α	β	γ	δ	Mode	Max Vert.	Absrel	Used Parts
100000	20	0.0025	4	0.05	0.05	0.01	0.05	0	16	0.1498	11.925
100000	20	0.0025	8	0.05	0.05	0.01	0.05	0	64	0.1550	11.933
100000	20	0.1	4	0.05	0.05	0.01	0.05	0	64	0.1917	6.375
100000	20	0.0025	4	0.75	0.05	0.01	0.05	0	64	0.1553	11.941
100000	20	0.0025	4	0.05	0.75	0.01	0.05	0	64	0.1550	12.475
100000	8	0.0025	4	0.05	0.05	0.01	0.05	0	64	0.1548	11.95
100000	20	0.0025	4	0.05	0.05	0.01	0.05	0	64	0.1549	11.925
100000	20	0.0025	4	0.05	0.05	0.01	0.05	0	128	0.1549	11.925
100000	20	0.005	4	0.05	0.05	0.01	0.05	0	64	0.1554	11.933
100000	20	0.0025	2	0.05	0.05	0.01	0.05	0	64	0.1548	11.917
100000	12	0.0025	4	0.05	0.05	0.01	0.05	0	64	0.1549	11.925
100000	20	0.0025	4	0.05	0.05	0.01	0.05	0	256	0.1549	11.925
100000	20	0.01	4	0.05	0.05	0.01	0.05	0	64	0.1562	11.692
100000	20	0.0025	4	0.01	0.05	0.01	0.05	0	64	0.1545	11.983
100000	20	0.0025	4	0.25	0.05	0.01	0.05	0	64	0.1553	11.817
100000	2	0.0025	4	0.05	0.05	0.01	0.05	0	64	0.2223	3.95
100000	20	0.0025	4	0.05	0.99	0.01	0.05	0	64	0.1570	12.167
100000	20	0.0025	4	0.1	0.05	0.01	0.05	0	64	0.1548	11.908
100000	20	0.0025	4	0.05	0.25	0.01	0.05	0	64	0.1535	12.067
100000	20	0.05	4	0.05	0.05	0.01	0.05	0	64	0.1673	9.633
100000	20	0.0025	4	0.99	0.05	0.01	0.05	0	64	0.1546	11.875
100000	4	0.0025	4	0.05	0.05	0.01	0.05	0	64	0.1647	9.883
100000	20	0.0025	4	0.05	0.05	0.01	0.05	0	8	0.2363	11.925
100000	20	0.0025	12	0.05	0.05	0.01	0.05	0	64	0.1565	11.8
100000	20	0.0025	4	0.05	0.1	0.01	0.05	0	64	0.1547	11.942
10000	20	0.0025	4	0.05	0.05	0.01	0.05	0	64	0.1694	11.075
100000	24	0.0025	4	0.05	0.05	0.01	0.05	0	64	0.1549	11.925
100000	20	0.0025	4	0.05	0.01	0.01	0.05	0	64	0.1549	11.833
100000	32	0.0025	4	0.05	0.05	0.01	0.05	0	64	0.1549	11.925
100000	20	0.0025	4	0.05	0.05	0.01	0.05	0	1024	0.1549	11.925
100000	20	0.5	4	0.05	0.05	0.01	0.05	0	64	0.2841	2.083
100000	20	0.0025	16	0.05	0.05	0.01	0.05	0	64	0.1565	11.825
100000	20	0.0025	4	0.05	0.05	0.01	0.05	1	64	0.1623	12.716
1000000	20	0.0025	4	0.05	0.05	0.01	0.05	0	64	0.1181	14.617
100000	20	0.0001	4	0.05	0.25	0.01	0.05	0	64	0.1535	12.092
100000	20	0.0005	4	0.05	0.25	0.01	0.05	0	64	0.1534	12.1
100000	20	0	4	0.05	0.25	0.01	0.05	0	64	0.1535	12.092
100000	20	0.0002	4	0.05	0.25	0.01	0.05	0	64	0.1534	12.133
1000000	20	0.0025	4	0.05	0.25	0.01	0.05	0	64	0.1198	14.383
10000000	20	0.0025	4	0.05	0.25	0.01	0.05	0	64	0.1123	14.875
16000000	20	0.0025	4	0.05	0.25	0.01	0.05	0	64	0.1116	14.833
10000000	20	0.0025	4	0.05	0.25	0.04	0.05	0	64	0.1645	5.483
10000000	20	0.0025	4	0.05	0.25	0.02	0.05	0	64	0.1340	9.067
10000000	20	0.0025	4	0.05	0.25	0.1	0.05	0	64	0.2180	2.817
10000000	20	0.0025	4	0.05	0.25	0.02	0.05	0	1024	0.1341	9.067
10000000	20	0.0025	4	0.05	0.25	0.02	0.05	0	8	0.2830	9.067
10000000	20	0.0025	4	0	0.25	0.02	0.05	0	64	0.1346	9.008
10000000	20	0.0025	4	0.05	0	0.02	0.05	0	64	0.1308	9.292
10000000	20	0.0025	4	0.05	0.1	0.02	0.05	0	64	0.1330	9.208
10000000	20	0.0025	4	0.8	0.25	0.02	0.05	0	64	0.1332	9.058
10000000	20	0.0025	4	1	0.25	0.02	0.05	0	64	0.1333	9.083
10000000	20	0.0025	4	0.6	0.25	0.02	0.05	0	64	0.1325	9.225
10000000	20	0.0025	4	0.9	0.25	0.02	0.05	0	64	0.1329	9.233

Table A.2: Trying on different VHACD Parameters

Resolution	Depth	Concavity	Plane Ds.	α	β	γ	δ	Mode	Max Vert.	Absrel	Used Parts	Algorithm	
1000000	20	0.0025	4	0.05	0.25	0.0015	0.05	0	64	0.0884	49.53	0	Original
1000000	20	0.0025	4	0.05	0.25	0.0015	0.05	0	64	0.1705	49.53	1	Axis Aligned
1000000	20	0.0025	4	0.05	0.25	0.0027	0.05	0	64	0.1883	34.63	1	Parallelepiped
1000000	20	0.0025	4	0.05	0.25	0.0028	0.05	0	64	0.1894	33.86	1	Parallelepiped
1000000	20	0.0025	4	0.05	0.25	0.0014	0.05	0	64	0.0878	51.43	0	Original
1000000	20	0.0025	4	0.05	0.25	0.0025	0.05	0	64	0.1852	36.43	1	Parallelepiped
1000000	20	0.0025	4	0.05	0.25	0.0011	0.05	0	64	0.0854	58.92	0	Original
1000000	20	0.0025	4	0.05	0.25	0.0028	0.05	0	64	0.0955	33.86	0	Original
1000000	20	0.0025	4	0.05	0.25	0.0016	0.05	0	64	0.0889	47.65	0	Original
1000000	20	0.0025	4	0.05	0.25	0.0025	0.05	0	64	0.0939	36.43	0	Original
1000000	20	0.0025	4	0.05	0.25	0.0016	0.05	0	64	0.1723	47.65	1	Parallelepiped
1000000	20	0.0025	4	0.05	0.25	0.0011	0.05	0	64	0.1619	58.92	1	Parallelepiped
1000000	20	0.0025	4	0.05	0.25	0.0027	0.05	0	64	0.0950	34.63	0	Original
1000000	20	0.0025	4	0.05	0.25	0.0014	0.05	0	64	0.1689	51.43	1	Parallelepiped
1000000	20	0.0025	4	0.05	0.25	0.0025	0.05	0	64	0.1626	36.43	2	Parallelepiped
1000000	20	0.0025	4	0.05	0.25	0.0037	0.05	0	64	0.0998	28.18	0	Original
1000000	20	0.0025	4	0.05	0.25	0.0038	0.05	0	64	0.1002	27.72	0	Original
1000000	20	0.0025	4	0.05	0.25	0.0041	0.05	0	64	0.1015	26.28	0	Original
1000000	20	0.0025	4	0.05	0.25	0.0040	0.05	0	64	0.1011	26.76	0	Original
1000000	20	0.0025	4	0.05	0.25	0.0033	0.05	0	64	0.0981	30.32	0	Original
1000000	20	0.0025	4	0.05	0.25	0.0032	0.05	0	64	0.0978	30.85	0	Original
1000000	20	0.0025	4	0.05	0.25	0.0030	0.05	0	64	0.0968	32.23	0	Original
1000000	20	0.0025	4	0.05	0.25	0.0035	0.05	0	64	0.0991	29.09	0	Original

Table A.3: COACD with Axis Aligned and Parallelepiped