3D Reconstruction from a Single RGB Image

Project Report

Project X Mentorship Programme

at

Community of Coders, Veermata Jijabai Technological Institute

September 2023

ACKNOWLEDGEMENT

Firstly, we would like to thank our mentor, **Soham Mulye** for his commendable role as our mentor throughout this project. His timely guidance, expertise in the domain and insightful feedback were crucial in shaping our project.

We are also grateful to VJTI's COC for continuing the tradition of their seniors and providing this golden opportunity that made our project possible.

Labhansh Naik
labhanshvnaik2104@gmail.com

Mrudul Pawar
mvpawar2005@gmail.com

Param Parekh
parammparekh13@gmail.com

CONTENTS

1. OVERVIEW

3D reconstruction from a single RGB image, also known as monocular 3D reconstruction, is a challenging and exciting field within computer vision. It aims to recover the three-dimensional structure of objects and scenes using only a 2D image as input. While multi-view stereo reconstruction, which utilizes multiple images or camera views, is more robust and accurate, the single-image 3D reconstruction presents a unique set of challenges and opportunities.

Monocular 3D reconstruction is a fundamental problem with a wide range of applications in various domains, including robotics, augmented reality, autonomous vehicles, virtual reality, and more. The ability to estimate the 3D structure from a single image has the potential to revolutionize how we interact with the digital and physical worlds.

Aim of our project is to reconstruct a 3D scene from a single image. The input is going to be a single image which would contain real-life objects and output will be a 3D scene containing a mesh of every object present in the scene.

2.3D-Terminology

1. RGB-D Image

For each pixel in an image, a depth map provides the distance from the camera to the corresponding point in space. This gives a single-channel image of the same size as the input image, with the corresponding depth value,z at (x,y) position. The absolute depth values are sometimes mapped to the range [0, 255] and, together with a normal RGB image, the depth map is given as the fourth channel of the so-called RGB-D images with points closer to the camera appearing darker and the points further away appearing brighter.

As depth maps are created from a single viewpoint, they represent a very sparse 3D geometry of the scene containing only points directly in the line of sight of the camera. They say nothing about the occluded planes, nor do they say anything about the 3D orientation of different faces of an object in the scene. For this reason, RGB-D images are sometimes called 2.5D because they cannot represent a complete 3D topology on their own. They are a partial surface model, with only the shape of the front face of the surface represented.

2. Point Cloud

A point cloud is a set of 3D points in space. Theoretically, they are able to exactly represent a complete 3D scene by storing the position of every point in space. Depending on how many and which points are present in the point cloud, it can be both a solid and a surface model of the scene.

However, due to limited computational memory, it is often necessary to downsample them to reduce the size of the dataset. This can be done by removing the points which are very close to each other or which are not needed to understand the visible shape of the 3D surfaces.

Point clouds can be extremely useful for representing 3D shapes, but they can also be difficult to work with. Sometimes it is necessary to convert them to a mesh in order to get a more accurate representation of the object.

3. Mesh

A mesh is a collection of 3D points (or vertices) that have been connected together with edges to form the surfaces (or faces) of 3D objects. Vertices are connected in a way that the faces are made up of many polygons adjacent to each other. Usually, these polygons are triangles, and the meshes are called “triangulated meshes”. Meshes can be used to represent the surface models of a 3D scene as “wireframes”.

4. Voxels

A voxel is a 3D equivalent of a pixel. Voxel-based models represent objects as a collection of cubes (or voxels) stacked in space like a 3D grid. They represent a discretized, solid model of the scene. Accuracy of the 3D model depends on the size of the voxels making up the objects. The bigger the voxels, the more “pixelated” the surfaces of the objects appear.

3. Open3D Library

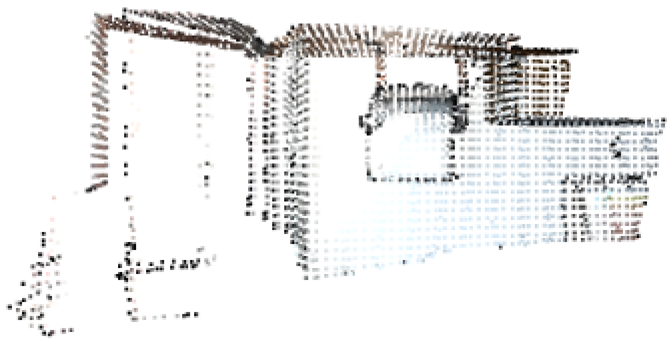
Open3D is an open-source, cross-platform, and easy-to-use library designed to work with 3D data, from point clouds and meshes to various 3D geometry representations. In this essay, we will delve into the key features and applications of Open3D and discuss how it has contributed to the advancement of 3D data processing.

Importance of Open3D in a 3D environment and processing is analogous to the importance of OpenCV in 2D image processing.

Open3D is a versatile library with a wide array of features and capabilities. Some of the key features that make it a valuable tool for researchers and developers include:

1. 3D Data Processing: Open3D provides essential functionalities for working with 3D data, such as loading, saving, and visualizing point clouds and 3D models.

2. Point Cloud and Mesh Processing: The library supports point cloud and mesh manipulation, allowing users to perform operations like downsampling, filtering, alignment, and reconstruction with ease.



3. 3D Visualizations: Open3D offers interactive 3D visualization tools, making it convenient for users to inspect and analyze their 3D data.

4. **Datasets**

In order to train the model, we require special datasets. The datasets which not only contain the image but also their 3D meshes. There are 2 main datasets which are being used in this project and those are:

4.1. Shapenet Dataset

ShapeNet is a massive library of 3D models of things like furniture, cars, animals, and everyday objects. These models come in different file formats and also include information like what category they belong to and pictures of them from different angles.

People in the computer vision and machine learning fields use ShapeNet to help their computers understand and work with 3D shapes better. It has over 12000 CAD Models with 270 categories and it is densely annotated.

Researchers often leverage ShapeNet to advance the state of the art in tasks related to 3D object understanding and manipulation, as well as to create realistic 3D content for applications like virtual and augmented reality. This dataset has spurred the development of numerous deep learning models and has contributed significantly to advancing the field of 3D computer vision and graphics.

4.2. Pix3D Dataset

Pix3D is a specialized dataset designed for the field of computer vision. It's unique because it focuses on objects often found in indoor scenes, like home furniture and decor. Unlike other datasets, Pix3D provides not only 3D models of objects but also real images of those objects captured in various everyday settings, like living rooms and bedrooms.

This combination of 3D models and real photos makes it an invaluable resource for training and testing computer vision algorithms that aim to understand and manipulate indoor objects, which is crucial for applications such as augmented reality and interior design.

5. Approaches to accomplish the Project

5.1. Open3D Approach

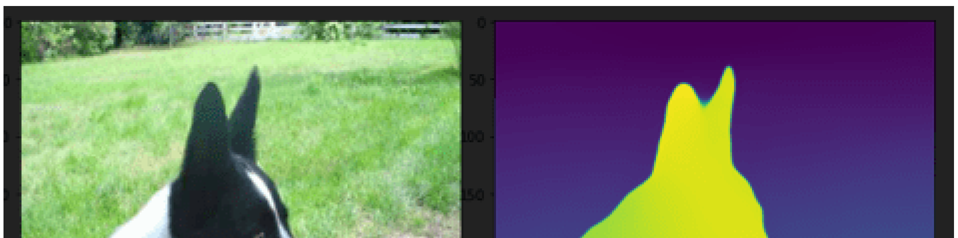
In the Open3D Approach we're going to be using the built in function of the Open3D Library in order to generate the point cloud of the single input image. The issue we run into, is that the function to create the point cloud requires 2 inputs. Firstly being the simple RGB Image but the second one being the 2D Depth Map image of the Image.

In order to obtain the depth map from a monocular view we use the state-of-the-art machine learning model for the task, namely MiDaS, which is implemented in PyTorch.

5.1.1. MiDaS Depth Map

MiDaS, which stands for Monocular Depth Estimation in Real-time with Deep Learning, is a deep learning model that is designed to estimate depth maps from single 2D images. Developed by NVIDIA, MiDaS has gained attention for its impressive ability to infer depth information accurately and in real-time. Here's a brief overview of MiDaS depth map generation.

The main aim of us using the MiDaS image is to get a depth Map. The MiDaS being a state-of-the-art for depth map creation using monocular view will give us the perfect depth map which is going to be used as a input for a function in Open3D which is going to yield the point cloud of the image



But we observe that especially in multi-object images, the library has a hard time to figure out the point cloud due to increased complexity.

In order to resolve this issue we came up with the approach of masking each object separately and creating point clouds of each object separately and concatenating them together. We expected it to give better outputs as the input will have less complexity.

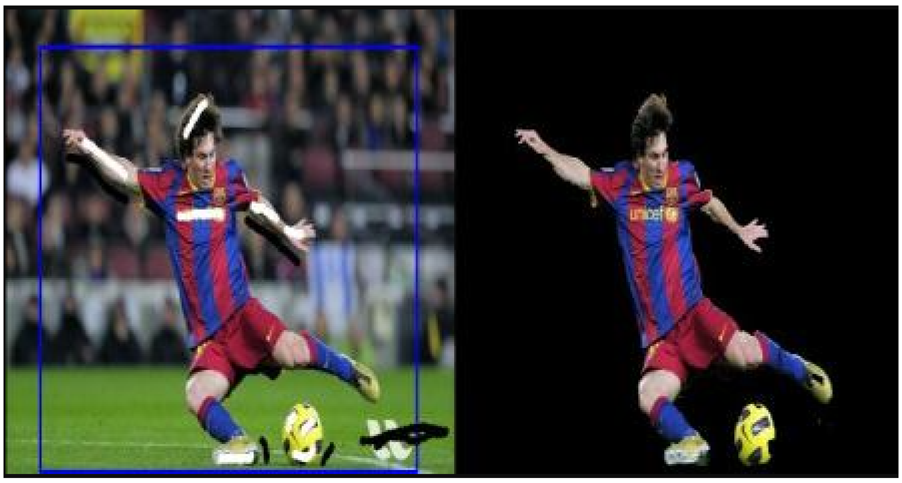
5.1.2. Masking for better outputs

There are mainly 3 ways we tried to mask out the objects present in the scene. Those are:

5.1.2.1. Grabcut Algorithm

GrabCut is a semi-automatic image segmentation technique that allows users to extract the foreground from an image by defining an initial bounding box or region. It employs graph cuts to iteratively refine the segmentation by estimating the foreground and background regions based on color and texture cues.

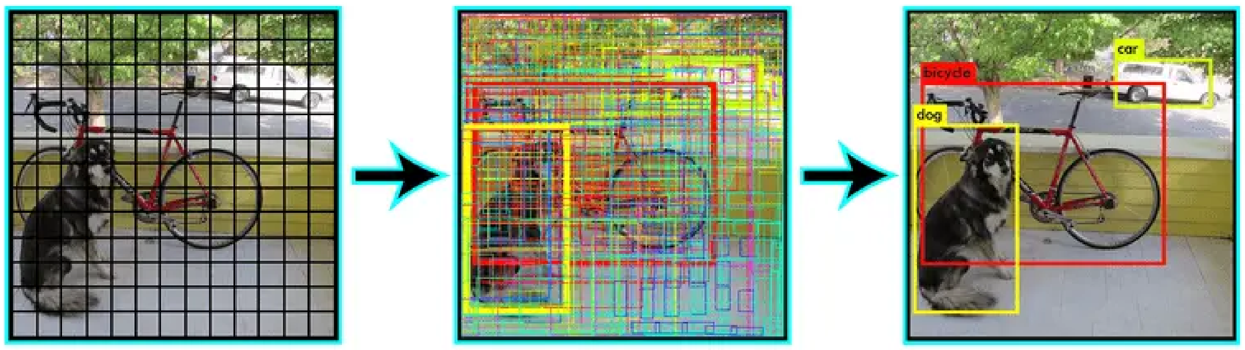
With its ability to produce accurate and fine-grained object masks, GrabCut is widely used in applications like image editing, object recognition, and image matting, enabling efficient and precise isolation of objects from their backgrounds.



5.1.2.2. YOLO Algorithm

YOLO, which stands for "You Only Look Once," is a popular and efficient object detection algorithm used in computer vision and deep learning. What sets YOLO apart is its ability to rapidly identify and locate multiple objects within an image or video frame in real-time. YOLO operates by dividing the image into a grid and then predicting bounding boxes and class probabilities for each grid cell. This means it doesn't need to scan the image multiple times, making it much faster than traditional object detection methods.

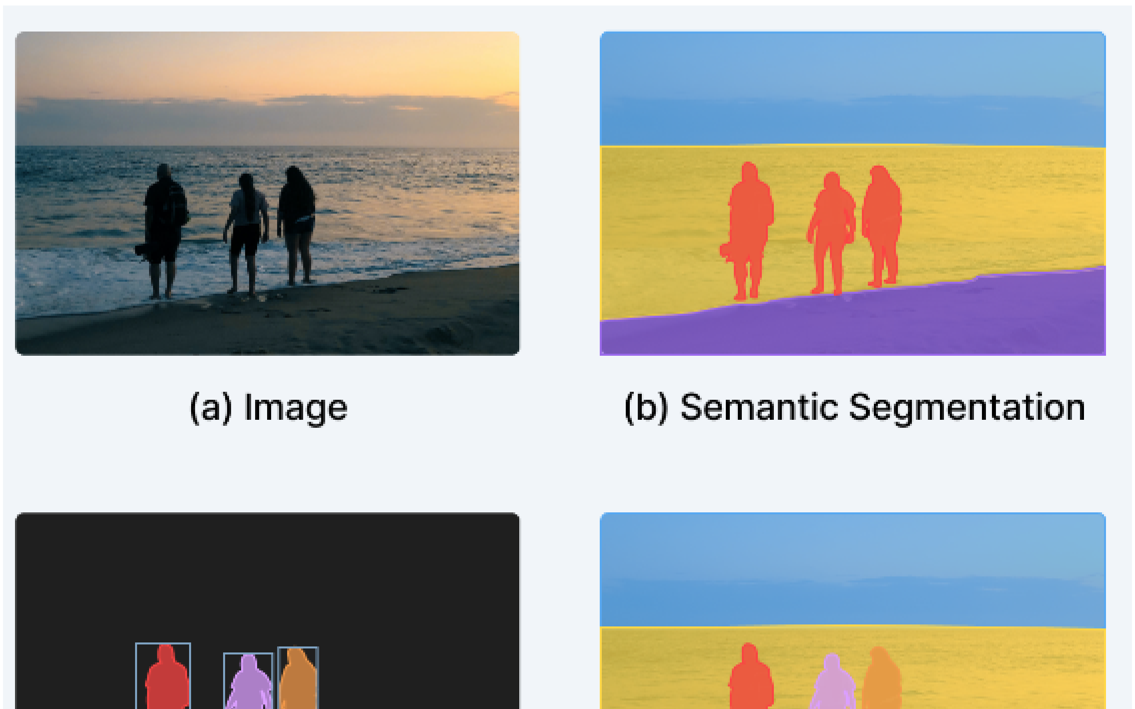
YOLO is widely used in applications like autonomous driving, surveillance, and object tracking, where real-time object detection is critical. It balances speed and accuracy, making it a practical choice for many real-world scenarios. Its simplicity and effectiveness have led to several iterations and variations of the YOLO algorithm, with each version aiming to improve accuracy and speed while maintaining its real-time capabilities.



5.1.2.3. Panoptic Segmentation

Panoptic segmentation is an advanced computer vision task that unifies instance segmentation (identifying individual object instances) and semantic segmentation (labeling each pixel with a category). It aims to provide a comprehensive understanding of an image by segmenting both objects and stuff (e.g., roads, sky) in a single process.

Panoptic segmentation is the combination of semantic segmentation and instance segmentation combined which not only gives us information about the objects to us but also of the surroundings.



5.1.3. Drawbacks of this approach

There are quite a few drawbacks to this approach. First of all the point cloud which is created is just a concave map of the objects which are present in the scene even after implementing the state-of-the-art resources and techniques.

Also, the mesh which will be created by this approach is not ideal at all and can't be used in any application.

Due to these drawbacks we decided to explore another approach. Which is a Machine Learning Model using the architecture of Pixel2Mesh.

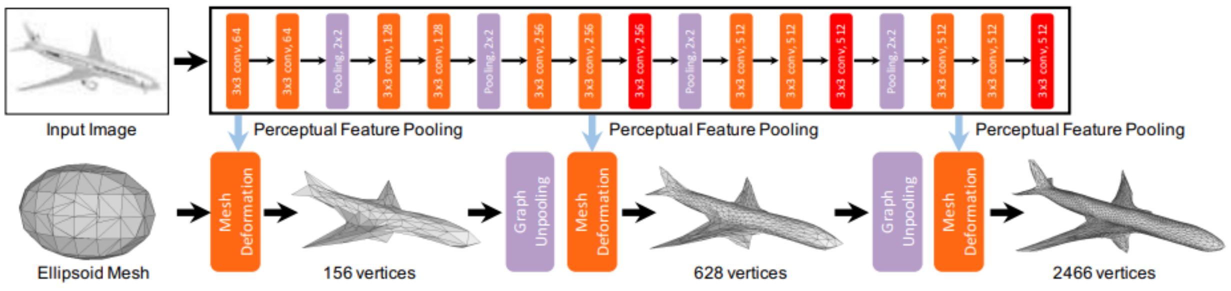
5.2. Pixel2Mesh Approach

Pixel2Mesh is a computer vision approach that focuses on converting 2D images, typically photographs, into 3D mesh models. The goal is to create a 3D representation of an object or scene from its 2D image, allowing for a more immersive understanding and manipulation of the visual content. Pixel2Mesh leverages deep learning techniques to achieve this.

The pixel2Mesh process typically involves a two-step approach. First, a neural network takes the 2D image as input and produces a set of 3D point cloud data that approximates the shape of the objects in the image. This initial point cloud captures the basic structure but lacks fine-grained details. In the second step, another neural network refines this point cloud into a more detailed 3D mesh model, complete with surfaces and texture information.

Architecture and Data Modalities:

The architecture of Pixel2Mesh is as follows:



We can see that 2 data modalities have been combined in order to accomplish the task of reconstructing a 3D object. Those 2 Modalities are:

<u>5.2.1. 2D Data Modality</u>

First let us look at 2D Convolutional Neural Network which is responsible to extract information and details from single RGB image. Convolutional Neural Network we'll be using a VGG-16 architecture. We store the output after every major layer and feed it to the 3D Data Modality.

This is necessary in order to keep the mesh aligned and in the context of the image rather than becoming something completely different.

<u>5.2.2. 3D Data Modality</u>

3D data modality is actually the modality that creates the mesh. We'll be using Mesh deformation layers and Graph unpooling Layers in order to deform the 3D mesh.

This is the block which is responsible for deforming the original ellipsoid and the new meshes which will be created with the knowledge from the image. The vertices (C) are used to extract information from the image with the help of perceptual feature pooling layer.

Perceptual pooling layer is used for taking 3D coordinates of vertices from the current mesh and map its 2D representation on the image.

Graph unpooling layer is used to increase the number of vertices in the mesh which will be fed into the GCNN (Graph Convolution Neural Network) in order to get more detailed mesh over the layers, this follows a coarse-to-fine method.

Pixel2Mesh is a standard approach which is used for 3D Reconstruction from a single RGB image. But there are still a lot of drawbacks for this model, which doesn’t make this the most optimal solution for our project.

5.2.3. Drawbacks

Pixel2Mesh is trained on the ShapeNet dataset, which does not contain any real life instances but only 3D environment rendered models and images.

And this itself heavily diminishes its ability to infer real-life images and instances. Pixel2Mesh is only able to give valid mesh if the image is masked out(which we accomplished by using the methods in the Open3D Approach), and place it in a transparent background.

We also need to resize the image to a certain size which causes shearing and squeezing giving the CNN a hard time to extract features and gives us garbage mesh.

Real life objects don’t come in one ideal size but consist of a huge variety of shapes and sizes and again this approach isn’t compatible with the real-life instances.

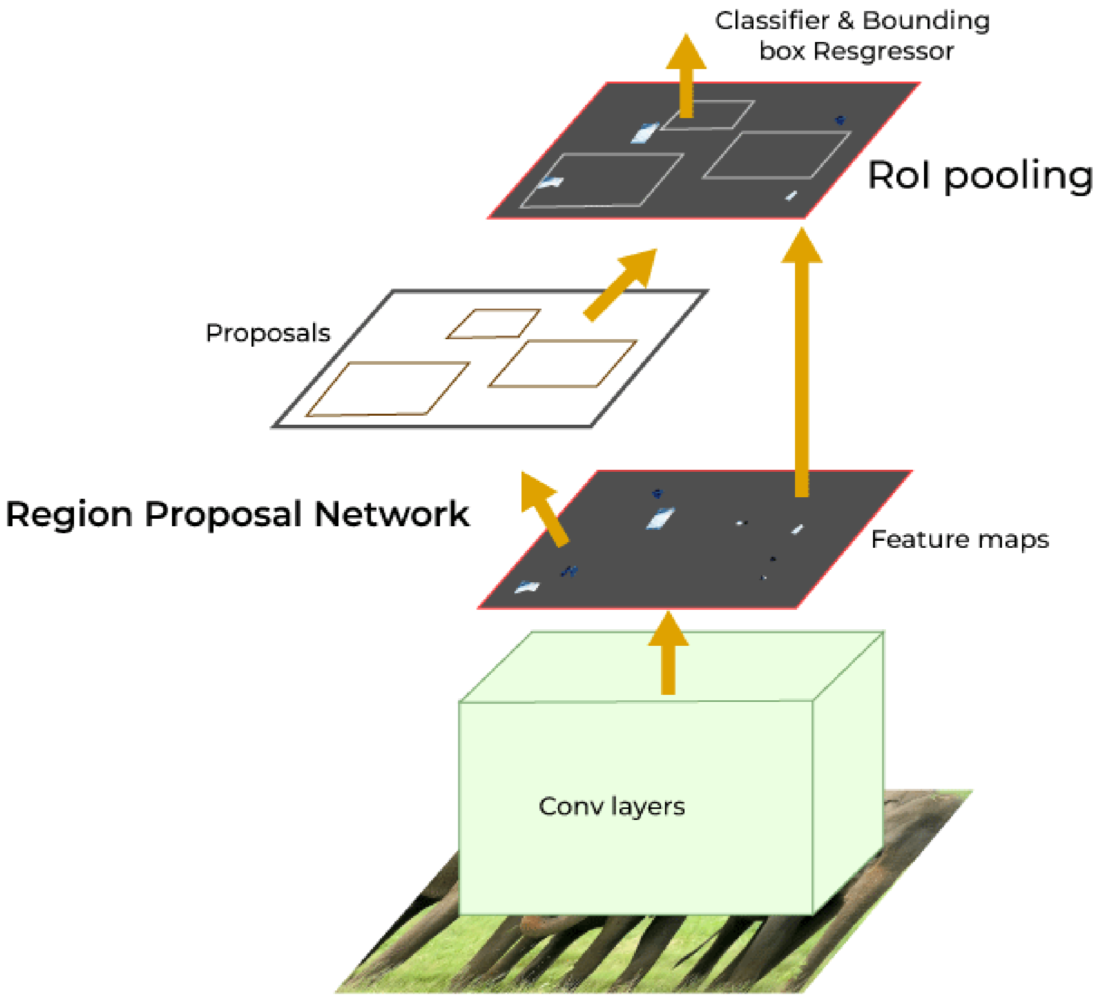
In order to create a valid output, we need to adapt to an approach which is robust to real-life instances and doesn't depend on the size of the objects present in the scene as well. And that's why we tried the next approach of Mesh-RCNN.

5.3. Mesh RCNN Approach

Mesh RCNN is the approach which aims at using state-of-the art techniques for both extracting information of the input image and to create the mesh out of it.

The extraction of information from the image in Mesh-RCNN is done by the method of Mask-RCNN. Mask RCNN is a layer added over Faster-RCNN.

5.3.1. Mask-RCNN



Faster-RCNN outperforms regular CNNs by having feature maps and other approaches integrated into it as well.

RPNs - Region Proposal Network(RPN) generates a set of bounding box proposals, which are potential object locations. These proposals are predicted by the RPN based on sliding windows over the feature maps. The RPN uses anchor boxes of various scales and aspect ratios to generate these proposals.

ROIs – Regions of Interests(ROIs) are the outputs of proposals of regions where objects can be present. These proposals are used to crop and align the corresponding regions from the feature maps.

After RoI pooling, Faster R-CNN uses two sibling fully connected networks. The first network performs object classification to determine the object category present in each proposal.

The second network performs bounding box regression to refine the coordinates of the bounding box around the object in each proposal. The bounding box regression improves the accuracy of the predicted object locations.

And at the end when the bounding box and object label is finalised a mask layer is applied to the objects present inside the bounding box of the objects present in the scene.

5.3.2. Mesh Creation and Deformation.

In the original Pixel2Mesh approach the mesh which would be formed will be topologically constrained this was due to the fact that the mesh being deformed is an ellipsoid and therefore the mesh will not be able to contain holes even if the real life object does. In order to conquer this problem we create our basic mesh by Voxel Branch

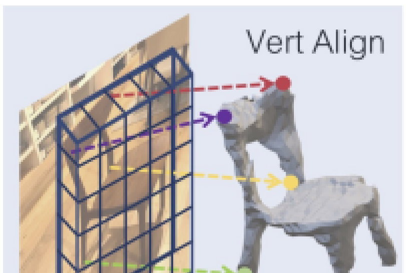
Voxel Branch - The voxel branch predicts a grid of voxel occupancy probabilities giving the course 3D shape of each detected object. It can be seen as a 3D analogue of Mask R-CNN's mask prediction branch: rather than predicting a M × M grid giving the object's shape in the image plane, we instead predict a G × G × G grid giving the object's full 3D shape.

Like Mask R-CNN, we maintain correspondence between input features and predicted voxels by applying a small fully-convolutional network to the input feature map resulting from RoIAlign. This network produces a feature map with G channels giving a column of voxel occupancy scores for each position in the input.

Mesh Convolution Branch – The mesh from the voxel branch only provides a coarse 3D shape, and it cannot accurately model fine structures like chair legs. The mesh refinement branch processes this initial mesh (just voxels stacked together) , refining its vertex positions with a sequence of refinement stages.

Each refinement stage consists of three operations: vertex alignment, which extracts image features for vertices; graph convolution, which propagates information along mesh edges; and vertex refinement, which updates vertex positions.

Vertex Alignment yields an image-aligned feature vector for each mesh vertex1 . We use the camera's intrinsic matrix to project each vertex onto the image plane. Given a feature map, we compute a bilinearly interpolated image feature at each projected vertex position.



Concurrently, advances in 3D shape prediction have mostly focused on synthetic benchmarks and isolated objects. We unify advances in these two areas. We propose a system that detects objects in real-world images and produces a triangle mesh giving the full 3D shape of each detected object.

Our system, called Mesh R-CNN, augments Mask R-CNN with a mesh prediction branch that outputs meshes with varying topological structure by first predicting coarse voxel representations which are converted to meshes and refined with a graph convolutional network operating over the mesh's vertices and edges.

Graph Convolution propagates information along mesh edges. Given input vertex features {fi} it computes updated features.

$$f_i' = \mathrm{ReLU}\left(W_0 f_i + \textstyle\sum_{j \in \mathcal{N}(i)} W_1 f_j\right)$$

Here N (i) gives the i-th vertex's neighbors in the mesh, and W0 and W1 are learned weight matrices. Each stage of the mesh refinement branch uses several graph convolution layers to aggregate information over local mesh regions.

Vertex Refinement computes updated vertex positions

$$v_i' = v_i + \tanh(W_{vert}\,[f_i; v_i])$$

where Wvert is a learned weight matrix. This updates the mesh geometry, keeping its topology fixed. Each stage of the mesh refinement branch terminates with vertex refinement, producing an intermediate mesh output which is further refined by the next stage.

Loss Functions: There are majorly 3 losses which are being used, chamfer loss, normal loss and edge loss which are defined as follows.

Chamfer loss is the distance between the nearest points present in 2 point clouds P and Q and is given by.

$$\mathcal{L}_{\text{cham}}(P, Q) = |P|^{-1} \sum_{(p,q)\in\Lambda_{P,Q}} \|p - q\|^2 + |Q|^{-1} \sum_{(q,p)\in\Lambda_{Q,P}} \|q - p\|^2 \quad (1)$$

Normal loss corresponds to the absolute distance

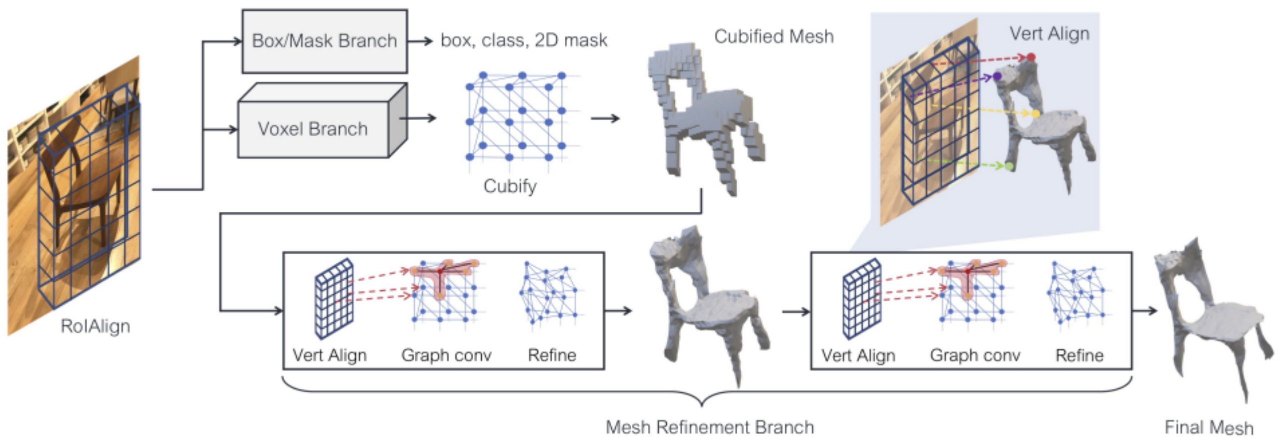
$$\mathcal{L}_{\text{norm}}(P, Q) = -|P|^{-1} \sum_{(p,q)\in\Lambda_{P,Q}} |u_p \cdot u_q| - |Q|^{-1} \sum_{(q,p)\in\Lambda_{Q,P}} |u_q \cdot u_p|.$$

The chamfer and normal distances penalize mismatched positions and normals between two point clouds, but minimizing these distances alone results in degenerate meshes.

High-quality mesh predictions require additional shape regularizer. And in this case we'll be using edge loss.

$$\mathcal{L}_{\text{edge}}(V, E) = \frac{1}{|E|} \sum_{(v,v')\in E} \|v - v'\|^2$$

Following is the complete architecture of the Mesh RCNN.



This model is trained on ShapeNet dataset but it is evaluated on a very tough and real-life instance based dataset of Pix3D which gives it a staggering advantage and gives ideal output for our project. But still there are some disadvantages that are required to be addressed.

5.3.3. Disadvantages

The masking output from the Mask-RCNN modality is still not up to the mark when compared with ideal standards and can be improved heavily.

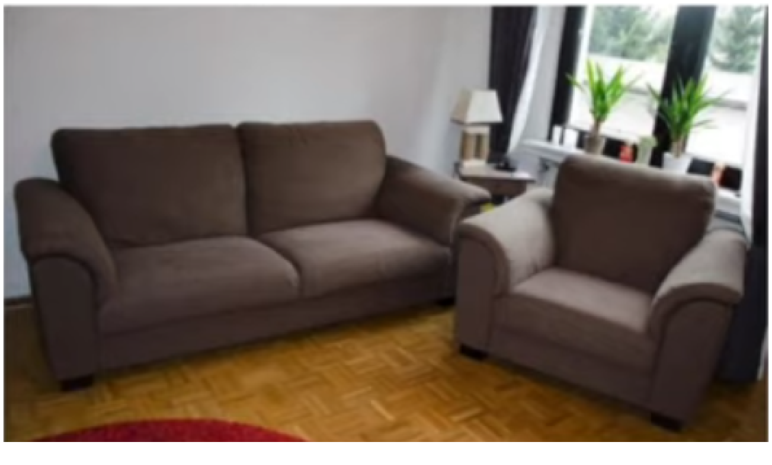
Another major disadvantage is that Mesh-RCNN only gives masks for the object with highest object detection probability and creates only their meshes. This in itself is a major problem when it comes to reconstructing an entire scene which has to include all the objects present in the scene. And this problem is solved in our final approach of the project.

5.4. Mask Integration with Mesh-RCNN

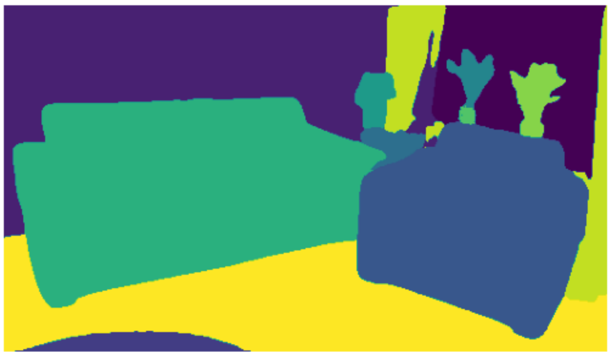
Earlier in the paper we saw 3 major segmentations which can be used for masking. Masking in this regard solves two major issues.

Firstly it aids the Mask-RCNN to give a better mask detection as the input itself is just a mask of that object. Secondly it enables us to pass different masks individually and obtain meshes for each object separately and then concatenate it together in order to reconstruct the final scene.

Lets take this image for example:



We first perform panoptic segmentation on the image. By this we get all the objects separated out and every instance of any object detected will have a different hue



This hue is then used as a reference to create masks of the instance which are required, i.e. important objects present in the image which are essential to reconstruct the 3D scene. In this case these are a couch and a sofa chair.

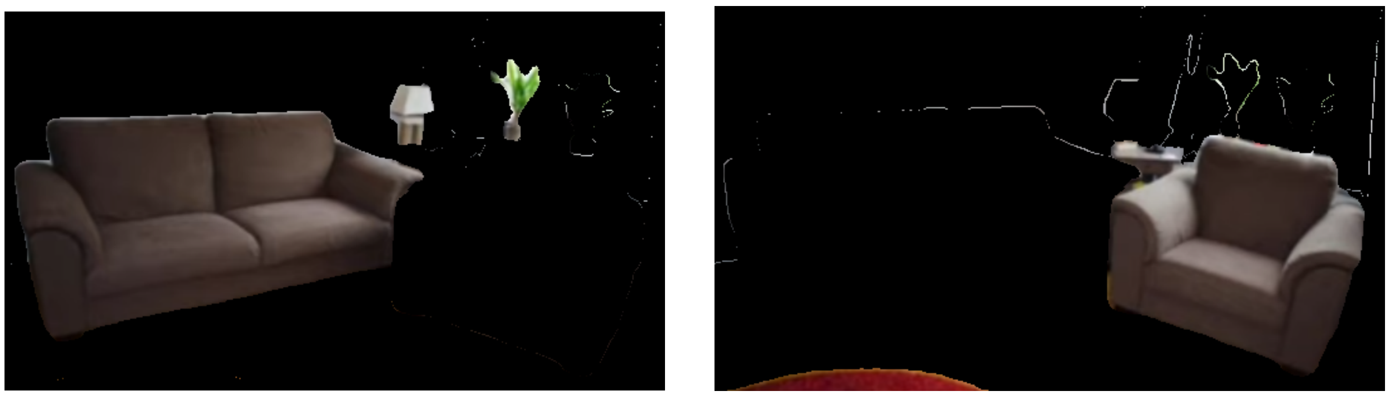


Fig. Separate meshes of couch and sofa chair

We then run the inference of the Mesh RCNN and pass each mask produced individually. By doing this we obtain 2 different meshes of couch and sofa as shown below.

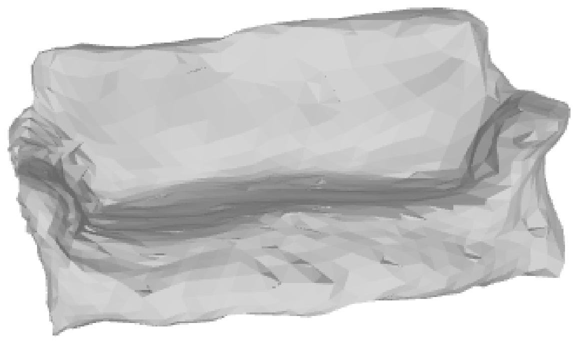


Fig. Mesh of sofa

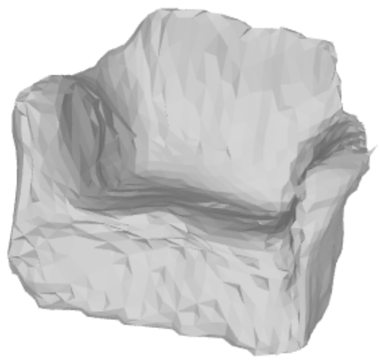
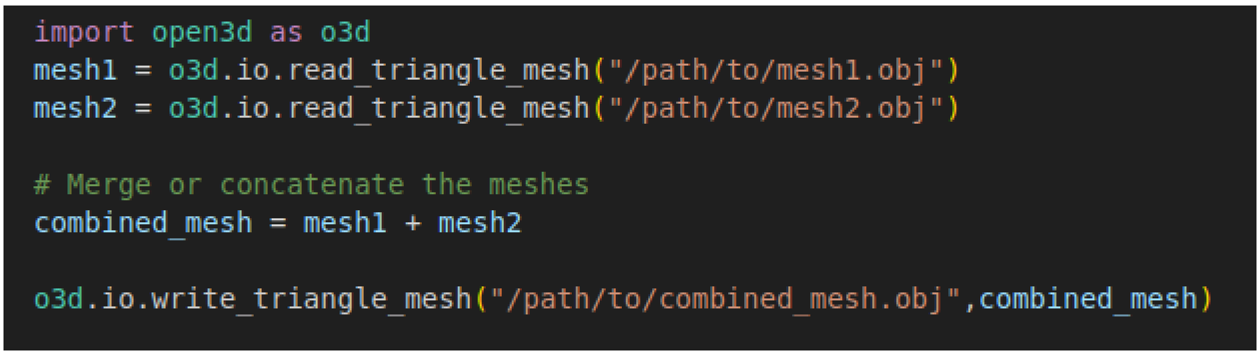


Fig. Mesh of Sofa

After creating the different masks for all the objects we use the mesh RCNN architecture in order to create meshes for all the objects separately, and then we concatenate the meshes created using Open3D functions.



```
import open3d as o3d
mesh1 = o3d.io.read_triangle_mesh("/path/to/mesh1.obj")
mesh2 = o3d.io.read_triangle_mesh("/path/to/mesh2.obj")

# Merge or concatenate the meshes
combined_mesh = mesh1 + mesh2

o3d.io.write_triangle_mesh("/path/to/combined_mesh.obj",combined_mesh)
```

And finally we get the output as the complete 3D scene.

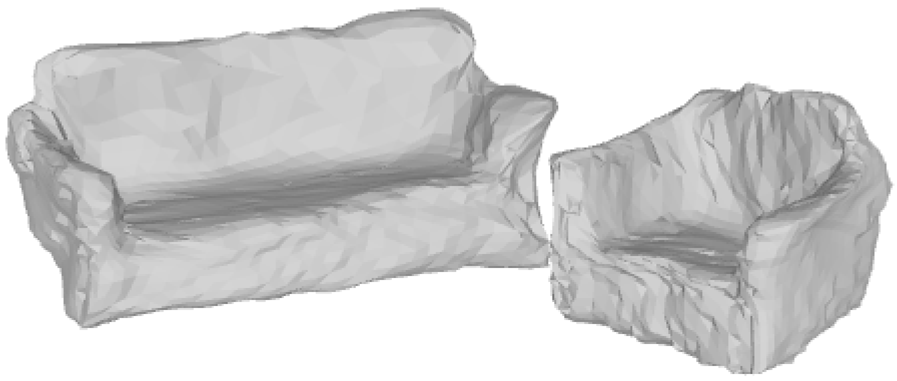


Fig. Final Mesh Output

6. Future Prospects

Till now we're able to create a combined mesh which aligns with the image. In future we aim at reconstruction of the wall and floor in order to create the entire scene present.

The model is restricted only to a defined number of interior objects such as bed, couch, chair because of the limited number of classes present in the Pix3D dataset. We aim at improving the dataset by either finding a more diverse dataset or adding additional categories to the existing dataset.

Due to GPU constraints we were unable to train the model to get an improved output. Therefore we plan to train on our new modified and diverse dataset in order to improve the diversity as well as the quality of mesh being produced.

7. References

1. 3 blue 1 brown Linear Algebra playlist

2. 3 blue 1 brown Neural Networks playlist

3. Deep Learning Specialization- Andrew Ng (Course 1, 2, 3 ,4)

4. Open 3D library documentation

5. Pixel2Mesh Paper by Nanyang Wang et al

6. For Image Segmentation Methods
    - https://huggingface.co/shi-labs/oneformer_coco_swin_large
    - https://huggingface.co/docs/transformers/main/en/model_doc/maskformer
    - https://huggingface.co/blog/mask2former

7. Pytorch implementation of Pixel2Mesh

8. Another pytorch implementation

9. Mesh R CNN by Justin Johnson et al

10. Pytorch3D documentation

11. Detectron2