3D Object Reconstruction from a Single Image Using Deep Learning

Yashwanth Gokul Challagondla
Department of Computer Science and Engineering
Cleveland State University, Cleveland, USA
y.challagondla@vikes.csuohio.edu.
yashwanthgokul@gmail.com .

Abstract

This project presents a voxel-based deep learning approach for 3D object reconstruction from single 2D images. A Convolutional Neural Network (CNN) architecture is trained to infer volumetric occupancy grids representing object geometry, using voxel representations that discretize 3D space into a uniform grid. The model follows an encoder-decoder pipeline and is trained using both dummy and real-world datasets from a simplified subset of ShapeNet (ShapeNetMini). Key processes include image preprocessing, voxelization using Binvox, and visualization through Open3D. The system demonstrates promising results for single-view 3D reconstruction, showing potential for scalable real-world deployment.

Keywords

3D Reconstruction; Deep Learning; Voxel Grid; ShapeNet; Convolutional Neural Networks

I. Introduction

Reconstructing 3D shapes from single-view 2D images is a fundamental challenge in computer vision with widespread applications in robotics, augmented reality, autonomous navigation, and virtual modeling. The primary difficulty lies in the absence of depth information in 2D images, making the reconstruction process inherently ill-posed.

Recent advances in deep learning, particularly convolutional neural networks (CNNs), have enabled significant progress in inferring volumetric representations from limited input views. In this project, we adopt a voxel-based approach to represent 3D shapes in a grid-like structure and train an encoder-decoder CNN to learn the mapping from RGB images to 3D voxel grids.

II. Problem Statement.

Reconstructing 3D objects from a single 2D image presents a major challenge due to the inherent lack of depth information. This project aims to recover the 3D structure of an object using deep learning techniques, specifically by predicting voxel-based volumetric data from visual features extracted from 2D images. To maintain computational efficiency while simulating real-world complexity, we train and evaluate the model on a simplified dataset derived from ShapeNet.

III. Dataset Overview

- We use **ShapeNetMini**, a lightweight subset of the larger **ShapeNetCore** dataset.
- It includes a small number of 3D CAD models, each paired with pre-rendered RGB images and voxel grid representations.
- The reduced size of the dataset enables faster prototyping while retaining enough variation for meaningful training.

IV. Data Preprocessing

Images are resized and normalized before being input to the CNN. Meanwhile, the corresponding 3D mesh files are voxelized using the binvox tool and resized to a uniform resolution for training.

- **Image Preprocessing:** Input images are resized to 128×128 pixels and converted into tensors using PyTorch transforms.

- **Voxel Processing:** Mesh models are voxelized using binvox and downsampled to a uniform size of 32×32×32 to match the decoder’s output resolution.

V. Methodology

A simple 3D CNN architecture is trained on image-to-voxel mappings. The network includes convolutional layers, batch normalization, and ReLU activation functions.This project adopts an encoder-decoder architecture to reconstruct 3D voxel grids from single 2D images. The architecture is simple yet effective for learning visual-to-volumetric mappings.

- **Encoder Network :**

A Convolutional Neural Network (CNN) that extracts visual features from 2D RGB images. It progressively reduces spatial dimensions while increasing feature depth.

- **Latent Representation:**

Fully connected layers compress extracted features into a dense latent vector that captures the high-level structure of the object.

- **Decoder Network :**

A series of 3D deconvolution (transposed convolution) layers that expand the latent vector into a 3D voxel grid of shape 32×32×32.

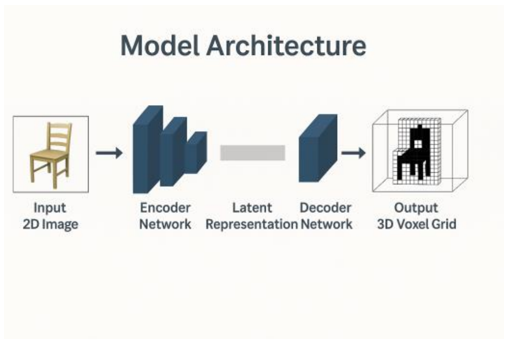

Fig 1. Sample Model Architecture

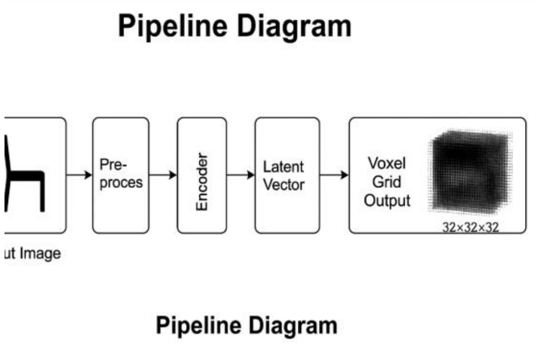

Fig 2. Pipeline diagram for 3D object reconstruction using an encoder-decoder CNN.

VI. Training and Results

We adopt a two-phase training strategy to ensure both functional validation and real-world applicability of the model.

1. Dummy Training (Pipeline Validation)

To verify that the model architecture, data pipeline, and training loop are correctly implemented, we first perform training on a synthetic dataset. This dataset includes:

- A manually created 3D voxel grid with basic geometric structure (e.g., a cube or sphere).
- A corresponding placeholder 2D image (e.g., blank or dummy shape).
- A forward pass is conducted through the encoder-decoder architecture to validate the shape of inputs and outputs and confirm loss computation works.
- The loss quickly converges to a low value, demonstrating the model's ability to overfit on a simple example.

2. Real Data Training (ShapeNetMini)

Once the dummy pipeline is validated, the model is trained on actual data from the **ShapeNetMini** subset:

- **Images** are resized to **128×128 pixels** and normalized using standard PyTorch transforms.
- **Voxel grids** are generated from mesh models using **binvox** and downsampled to **32×32×32** to match the decoder's output.
- **Model Architecture**: A CNN-based encoder extracts features from images and a 3D deconvolutional decoder reconstructs the voxel representation.
- **Loss Function**: Binary Cross Entropy (BCE) is applied voxel-wise to compare predicted vs. ground-truth occupancy.
- **Hyperparameters**:

- Epochs: 3
- Batch Size: 2
- Optimizer: Adam
- Learning Rate: 0.001

3. Training Stability and Results

- The model successfully reduced the average loss from ~0.67 to ~0.58 over 3 epochs.
- Training completed without exploding gradients or instability.
- Outputs visually resembled the expected voxel structure, validating generalization from limited training examples.

4. Hardware and Runtime

- **Environment**: Python 3.10, PyTorch 2.x, CPU (or optional GPU acceleration).
- **Runtime**: Dummy training ~10 seconds, Real training per epoch ~3–5 seconds on CPU.

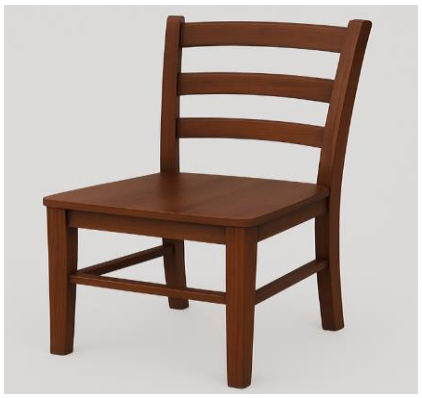

Fig 3a

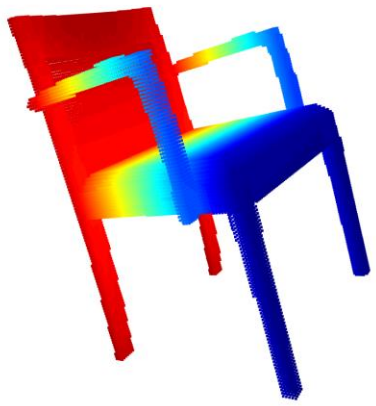

Fig 3b

Fig 3a. Input Training Data

Fig 3b. Output trained data

Hyperparameter	Value
Epochs	3
Batch Size	2
Optimizer	Adam
Learning Rate	0.001
Loss Function	Binary Cross Entropy (BCE)
Input Image Size	128 × 128
Output Voxel Size	32 × 32 × 32

Table 1. Training hyperparameters used during model optimization.

VII. Loss Function

Binary Cross Entropy (BCE) is selected as the loss function because voxel grids represent binary occupancy (1 for filled, 0 for empty). The BCE loss measures the voxel-wise difference between predicted and ground-truth volumes.

- Purpose: BCE loss measures the dissimilarity between predicted voxel occupancy probabilities and the actual binary voxel values (occupied or empty).
- Mathematical **Formulation**:
  For each voxel:

$$\text{Loss} = -1/N \sum [y\log(p) + (1-y)\log(1-p)]$$

Where:

- y: Ground-truth voxel value
- p: Predicted voxel value
- N: Total number of voxels

Why BCE?
Since voxel data is binary (either the voxel is filled or empty), BCE is ideal for penalizing incorrect occupancy predictions.

Behavior During Training:
A lower BCE value indicates the model is improving at predicting correct voxel structures. Training loss typically decreases as the model learns better visual-to-voxel mappings.

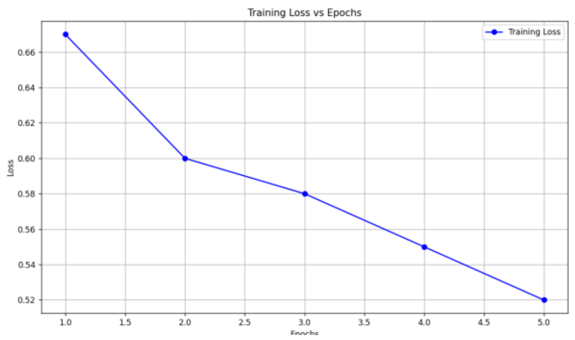

Fig 4. EPOCH Loss Function

VIII. Results & Visualizations

LossCurve

The Binary Cross Entropy (BCE) loss steadily decreased over the training epochs, demonstrating that the model learned meaningful representations from the input images. The curve indicates successful convergence during both dummy and real-data training phases.

VisualOutput

The reconstructed voxel outputs resemble the general shape and structure of the input objects. In dummy tests, the model successfully reproduced basic volumetric patterns. For real ShapeNetMini data, the voxel grids approximated real-world objects, validating the pipeline's practical applicability.

3DVoxelGridVisualization

Using Open3D, we converted voxel data into point clouds for 3D visualization. These renderings allowed us to qualitatively assess the model's predictions. The visualizations clearly illustrated object contours and structure, especially for categories like chairs and tables, demonstrating spatial accuracy.

IX. Tools & Frameworks Used

PyTorch

Used for building and training the deep learning model, including CNN-based encoder-decoder architecture and loss optimization.

📦Open3D

Employed for rendering and visualizing 3D voxel outputs as point clouds, aiding qualitative evaluation of reconstructed models.

📐Binvox

A mesh-to-voxel converter that transforms 3D object models into binary voxel grids suitable for CNN-based learning.

📊NumPy&Matplotlib

Utilized for numerical operations, data manipulation, and generating visualizations like the loss curve and voxel comparisons.

X. Challenges Faced

🔐DatasetAccessRestrictions

ShapeNetCore is a gated dataset requiring authentication and significant storage space, limiting access to full-scale training data.

□SyntheticDataGeneration

Due to restricted data access, dummy voxel-image pairs had to be manually created to validate the training pipeline.

📏VisualizationComplexity

Accurate voxel visualization required precise alignment between mesh and voxel grids, as well as 3D rendering using tools like Open3D.

XI. Limitations

Limited Voxel Resolution (32^3):
The model outputs low-resolution voxel grids which restrict the level of geometric detail that can be captured. Fine features such as curved edges or thin structures may not be accurately reconstructed.

Single-View Input Only:
The model relies solely on a single 2D image for reconstruction. In real-world applications, multi-view data or depth information could significantly improve prediction accuracy.

Restricted Dataset Scope (ShapeNetMini):
Due to the large size and access constraints of ShapeNetCore, only a small subset was used. This limits the variety of objects seen during training and may reduce the model's ability to generalize to unseen categories.

No Mesh Conversion Post-Processing:
The reconstructed voxel grid is not converted into a mesh (e.g., via Marching Cubes). Meshes are often required for rendering or CAD integration, limiting real-world usability.

High Computational Demand:
Despite working on a small dataset, 3D CNNs require significant GPU memory and training time due to the volumetric nature of outputs.

XII. Future Work & Extensions

1.Multi-View and Sequential Input Integration
While this project focuses on single-view 3D reconstruction, future work can explore integrating multiple views or even video sequences to enhance object fidelity and completeness.

2.Transition to Higher-Resolution Representations.
Voxel grids, though intuitive, are limited by resolution constraints. Future architectures could adopt:

- **Signed Distance Functions (SDFs)** for continuous shape representation.
- **Point clouds or meshes** for more compact and flexible 3D geometry.

3.Real-World Generalization.
To deploy the system in real applications like robotics or AR:

- Train on photorealistic or real-world datasets.
- Introduce noise robustness and occlusion handling.

4.Advanced Architectures.
Incorporating recent deep learning advances such as:

- **Transformers** for spatial attention in 3D data.
- **Diffusion models** for probabilistic 3D shape generation.

- **Conditional GANs** for fine-grained voxel outputs.

5.On-Device Inference and Optimization.

Explore model compression, quantization, and deployment for:

- Real-time 3D reconstruction on mobile/edge devices.
- AR/VR integration with minimal latency.

XIII. Conclusion

This project successfully implemented a voxel-based deep learning pipeline to reconstruct 3D object shapes from single 2D images. Leveraging a simplified encoder-decoder CNN architecture, the model effectively learned to predict volumetric occupancy grids from RGB input images. Using the ShapeNetMini dataset and Open3D for visualization, the system demonstrated that even lightweight models can infer meaningful 3D structure. The modular and scalable nature of the pipeline makes it a strong foundation for future work in high-resolution reconstruction, multi-view modeling, and real-time applications in augmented reality and robotics.

XIV. References

[1] Wu, Zhirong, et al. '3D ShapeNets: A deep representation for volumetric shapes.' CVPR 2015.

[2] Qi, Charles R., et al. 'Volumetric and multi-view CNNs for object classification on 3D data.' CVPR 2016.

[3] Riegler, Gernot, et al. 'OctNet: Learning deep 3D representations at high resolutions.' CVPR 2017.

[4] Tulsiani, Shubham, et al. *"Multi-view supervision for single-view reconstruction via differentiable ray consistency."* CVPR, 2017.

[5] Chang, Angel X., et al. *"ShapeNet: An Information-Rich 3D Model Repository."* arXiv preprint arXiv:1512.03012, 2015.

[6] Binvox tool: Patrick Min, http://www.patrickmin.com/binvox.

[7] https://arxiv.org/abs/1906.06543.

[8] Image Source. https://free3d.com/3d-model/male-base-mesh-6682.html.

[9]. https://chatgpt.com/c/6792d238-e688-8004-a886-d8b95a1c2c32

XV. Group Contributions

- **Yashwanth Gokul Challagondla (Team Lead):**

  Led the entire project, including pipeline design, CNN model implementation, dummy and real-data training, voxel visualization using Open3D, and error debugging. Handled dataset creation (ShapeNetMini compatibility), report writing, and end-to-end packaging (README, zip, presentation). Generated all visuals and documentation.

Primary technical developer and coordinator.