Decomposition of 3D Objects into Geometric Primitives

Sakshi Cholli, Shrusti Girmath and Anilkumar Kulkarni
School of Electrical and Electronics, KLE Technological University, Hubballi, Karnataka, India

Keywords: 3D Object Decomposition, Geometric Primitives, Point Clouds, Convolutional Neural Networks (CNNs), Deep Learning, Shape Analysis, 3D Geometry, Segmentation, Feature Extraction, Surface Approximation, Computational Geometry, Machine Learning for 3D Data, Voxelization, Mesh Processing, Point Cloud Processing, Primitive Fitting, 3D Shape Recognition, Object Reconstruction, Computer Vision, Artificial Intelligence

Abstract: For applications in robotics, CAD systems, and scene comprehension, 3D objects must be broken down into their component geometric primitives. Current methods frequently depend on manually created features or regularised transformations, such as voxelization, which result in quantisation artifacts and inefficiencies. This study, which draws inspiration from PointNet, suggests a unified neural network architecture that breaks down 3D objects into basic geometric forms like spheres, cylinders, and planes by directly processing raw point clouds. Our model incorporates extra modules to learn local geometric characteristics for accurate decomposition, while utilizing PointNet's capability to handle unordered point sets, guaranteeing permutation invariance.

1 INTRODUCTION

The decomposition of 3D objects into geometric primitives is a fundamental problem in computer vision and computational geometry, with applications spanning robotics, augmented reality, computer-aided design (CAD), and autonomous systems. This process involves breaking down complex 3D shapes into simpler components such as planes, spheres, cylinders, and other primitive geometries. Such decomposition enables a more interpretable and efficient representation of 3D objects, facilitating downstream tasks like simulation, manipulation, and rendering.

Traditional methods for 3D object decomposition often rely on handcrafted algorithms and feature engineering, which may struggle to generalize across diverse object geometries and noisy datasets. With the rise of deep learning, data-driven approaches have shown remarkable promise in addressing these limitations. PointNet, a pioneering deep learning architecture, has emerged as a robust solution for processing raw point cloud data. By directly operating on unordered sets of 3D points, PointNet preserves permutation invariance and learns global and local geometric features effectively.

This research aims to leverage the PointNet framework to address the challenge of decomposing 3D objects into their constituent geometric primitives. The study explores how PointNet's feature extraction capabilities can be adapted to identify and segment primitives within complex point cloud representations. By focusing on a data-driven approach, this work seeks to overcome challenges associated with noise, incomplete data, and diverse object geometries, contributing to a scalable and generalized solution for 3D object decomposition.3D object decomposition into geometric primitives simplifies complex object representations, enabling efficient storage, manipulation, and analysis. Traditional methods rely heavily on data transformation into regular grids or handcrafted geometric approximations, which are computationally intensive and fail to generalize across varied datasets.

PointNet, a deep learning framework designed for point cloud data, has demonstrated success in 3D classification and segmentation tasks by directly processing unordered sets of points. This work extends PointNet to address the specific challenge of geometric decomposition. By identifying local and global point features, our approach achieves robust decomposition of objects into primitive shapes, providing a scalable and interpretable solution.

Cholli, S., Girmath, S. and Kulkarni, A.
Decomposition of 3D Objects into Geometric Primitives.
DOI: 10.5220/0013622800004664
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 3rd International Conference on Futuristic Technology (INCOFT 2025) - Volume 3*, pages 496-502
ISBN: 978-989-758-763-4
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

2 RELATED WORK

The related work section provides an overview of existing methods and research relevant to the decomposition of 3D objects into geometric primitives. For the given problem, the section would include the following topics:

1.Traditional Methods for Geometric Decomposition Traditional approaches rely on geometric algorithms and handcrafted rules to identify primitives in 3D models: RANSAC Based Plane Detection: Techniques such as Random Sample Consensus (RANSAC) are commonly used for detecting planes in point clouds by fitting geometric models iteratively. While effective for simple shapes, these methods struggle with noise and complex structures. Model Fitting and Optimization: Optimization-based techniques aim to fit geometric primitives like spheres, cylinders, and cones by minimizing error metrics. However, these are computationally expensive and sensitive to parameter tuning. Region Growing Methods: Algorithms that group points into regions based on geometric similarity have been used but often fail with incomplete or noisy data.

2. Deep Learning for 3D Data Processing The rise of deep learning has led to novel methods for understanding and processing 3D data: Volumetric Representations: Early works converted point clouds into 3D voxel grids and applied 3D convolutional neural networks (e.g., VoxNet, 3DShapeNets). These methods suffer from high memory and computational costs due to voxelization.(Ding et al., 2023)(Liu et al., 2019)(Ioannidou et al., 2017). Multiview CNNs: Techniques like Multi-View CNNs render 3D shapes into 2D projections and process them using 2D convolutional networks. While effective for classification, this approach loses detailed geometric information, making it unsuitable for precise primitive decomposition. Spectral CNNs: Applied on mesh data, these methods process shapes using graph-based representations. However, they are limited to manifold meshes and struggle with generalizing to point clouds.

3. PointNet and Point-Based Networks

PointNet introduced a breakthrough in processing raw point clouds directly: It demonstrated how symmetric functions, like max pooling, ensure permutation invariance for unordered point sets. Extensions of PointNet, such as PointNet++ and PointCNN, focused on capturing local point features and hierarchical structures, enabling finer segmentation and part identification.(Liu and Tian, 2024) However, these methods primarily target classification and segmentation tasks, without specific adaptations for decomposing 3D objects into primitives.

4. Primitive Decomposition with Deep Learning

Recent works have explored the application of deep learning for geometric decomposition: Learning-Based Fitting: Some models directly predict primitive parameters (e.g., plane equations, cylinder radii) using neural networks. While promising, these require large labeled datasets with annotated primitives, which are scarce.(Fu et al., 2023)(Huang et al., 2018) Hybrid Methods: Approaches combining traditional RANSAC with learned features from deep networks have shown improvements in robustness and efficiency. Self-Supervised Learning: Techniques leveraging self-supervision to identify geometric primitives without labeled data are emerging but remain experimental.

5. Limitations of Existing Methods Scalability: Most methods struggle to handle large-scale, dense point clouds due to computational inefficiency. Robustness: Traditional and some learning-based methods are sensitive to noise, occlusions, and incomplete data. Generalization: Models trained on specific datasets or object categories often fail to generalize to unseen shapes and environments.

3 PROBLEM STATEMENT

The framework is designed to process 3D objects represented as ordered point sets to decompose them into geometric primitives. Each 3D object is defined as a set of n points $\{P_i \mid i = 1, \ldots, n\}$, where each point P_i is characterized by its (x,y,z) coordinates. Additional feature channels, such as normals or colors, may also be incorporated when available; however, the focus remains primarily on the spatial (x,y,z) representation.

For the decomposition task, the input point cloud is uniformly sampled from a 3D object while preserving its inherent order. The proposed deep learning framework identifies and segments points into groups corresponding to specific geometric primitives, including planes, cylinders, and spheres. The network outputs $n \times p$ scores, where n is the number of points in the object, and p is the number of primitive categories. Each score represents the likelihood that a point belongs to a specific primitive. By leveraging the ordered structure of the data set and integrating deep learning with geometric post-processing techniques, the framework achieves effective decomposition of 3D objects into their constituent geometric primitives.

3.1 Pointnet

PointNet, introduced by a Stanford University researcher in 2016, aims to classify and segment 3D image representations. The approach leverages a data structure known as a point cloud, which consists of a collection of points representing the geometry of a 3D object or shape. However, due to its irregular structure, point clouds are only applicable to specific use cases.Traditionally, many researchers transformed point clouds into alternative representations, such as voxels (3D volumetric pixels), before processing them through deep neural networks. This transformation, however, often results in overly large datasets and introduces quantization errors, potentially altering the natural characteristics of the 3D structure. In their work, the authors present a novel technique that directly processes point clouds, enabling efficient classification and segmentation without requiring intermediate transformations.

3.2 Architecture

The input for the suggested architecture is Point Sets that are extracted from a Point Cloud. With each point represented by its coordinates (x_i, y_i, z_i), a Point Cloud is a collection of 3D points P_i.

Either a direct sampling of the item's form or an extraction from a segmented scene Point Cloud is used for the object classification job. For semantic segmentation, the input may be a smaller area of a 3D scene obtained via object region segmentation, or it could be a single object for part segmentation.

Key Characteristics of Point Sets

1. **Permutation Invariance:** Point Clouds lack inherent structure, and a collection of N points can have $N!$ different permutations. Any processing method must ensure that the output remains consistent regardless of the order of points.
2. **Transformation Invariance:** Outputs for classification and segmentation tasks should not be affected by geometric transformations such as rotations or translations applied to the input points.
3. **Point Interactions:** Important contextual information is frequently carried by nearby sites. As a result, points shouldn't be discussed separately. Because segmentation tasks yield more pertinent information than classification tasks, these interactions are particularly crucial.

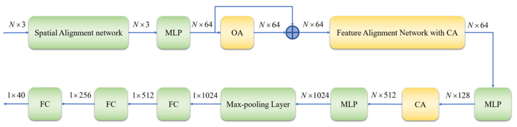

Figure 1: the fusion of pointnet shown in green and 3 transformer modules shown in yellow represents our point-based classification model with fc standing for interconnected layer and $\oplus$ representing matrix addition

4 POINTNET ARCHITECTURE

The PointNet architecture is designed with simplicity and effectiveness in mind. The classification network begins by applying a shared multi-layer perceptron (MLP) to transform each of the n input points from 3 dimensions to 64 dimensions. Notably, the same MLP is applied to all n points to ensure consistency. In the subsequent layer, each of these n points is further transformed from 64 dimensions to 1024 dimensions. A max-pooling operation is then used to aggregate these features into a single global feature vector in $\mathbb{R}^{1024}$. Finally, a three-layer fully connected network (FCN) maps this global feature vector to k output classification scores.

Pointnet is built with a focus on simplicity and efficiency it utilizes a shared multi-layer perceptron mlp to map each of the n points from a 3d input space into a 64-dimensional feature representation a critical feature of the design is the consistent application of the same mlp across all n points subsequently these features are further transformed into a 1024-dimensional space for each point a max-pooling layer then aggregates these features forming a comprehensive global feature vector this global representation is finally processed through a three-layer fully connected network fcn to produce k classification scores.

5 METHODOLOGY

5.1 Dataset Preparation

- The data set is fetched from the Kaggle repository using the Kaggle API.
- The ModelNet10 dataset, which contains 3D object files in `.off` format, is downloaded, extracted, and loaded into the working directory for processing.
- The data set structure is printed to verify the presence of files and directories.

5.2 Point Cloud Preprocessing

5.2.1 File Loading

- If the input file is a mesh (e.g., . off), it is converted to a point cloud using Poisson disk sampling to generate a uniform set of points.
- If the input file is already a point cloud, it is directly loaded.
- A validation step ensures that the loaded data contains valid points. If the point cloud is empty, an error is raised.

5.2.2 Input Representation

- Each 3D object is represented as a point cloud, where each point contains its (x, y, z) coordinates. Additional features such as normals or colors may also be included if available.

5.3 Deep Learning Framework for Feature Extraction

A custom deep learning framework is implemented to extract features from the point cloud:

- **Convolutional Layers (`conv2d`):** Extract spatial features from the input point cloud with batch normalization and ReLU activation.
- **Max Pooling (`max_pool2d`):** Aggregate local features into a compact representation.
- **Fully Connected Layers (`fully_connected`):** Map features to higher dimensions and combine them for global understanding.
- **Transform Network (`get_transform`):** A spatial transformer ensures alignment of the input point cloud and feature space to a canonical coordinate system.
- **Dropout (`dropout`):** Regularization is applied to reduce overfitting during training.

5.4 Plane Segmentation Using RANSAC

5.4.1 Plane Detection

- The RANSAC (Random Sample Consensus) algorithm is used iteratively to detect planes in the point cloud.
- Key parameters, such as `distance_threshold`, `ransac_n`, and `num_iterations`, are optimized for precise plane detection.

Plane Detection Probability

The success probability of RANSAC is related to:

- The fraction of inliers (w) in the data.
- The number of iterations (N) required to ensure a good model with high probability (p).

The probability p of at least one sample being free of outliers is given by:

$$p = 1 - (1 - w^n)^N$$

Where:

- w: Fraction of inliers.
- n: For a plane, the minimum number of points needed to suit the model is $n = 3$..
- N: Number of iterations.

This formula is derived from binomial probability.

RANSAC is a robust statistical algorithm designed to estimate the parameters of a mathematical model from a dataset that may contain a significant proportion of outliers.

i. *Random Sampling* Randomly select a minimal subset of the data points required to fit the desired model. For example, to estimate a plane with the equation:

$$ax + by + cz + d = 0,$$

a minimum of three points is required.

ii. **Model Fitting** Fit the model to the sampled points using standard techniques (e.g., solving linear equations or optimization).

iii. **Consensus Measurement** Compute the residuals for all data points to evaluate how well the model fits. The residual for a point (x_i, y_i, z_i) is calculated as:

$$d_i = \frac{|ax_i + by_i + cz_i + d|}{\sqrt{a^2 + b^2 + c^2}}.$$

Count the number of points (inliers) whose residuals are below a predefined threshold ε (e.g., $\varepsilon = 0.01$).

iv. **Statistical Concept** RANSAC iteratively performs the above steps (random sampling, model fitting, and consensus measurement). The quality of a model is evaluated based on the number of inliers it explains. After a predefined number of iterations, the model with the highest consensus (most inliers) is selected as the best estimate.

5.4.2 Primitive Extraction

- Points classified as inliers for a detected plane are grouped and visualized in a distinct color.
- Remaining points (outliers) are retained for further segmentation.

5.5 Thresholding

The algorithm employs a distance threshold (ε) to separate inliers from outliers:

Points with distances less than ε are classified as part of the plane.
Points with distances greater than ε are excluded (outliers).

This involves binary classification of points based on a fixed threshold, a common statistical decision boundary.

5.6 Residual Segmentation for Other Primitives

- After removing inlier points of detected planes, the residual point cloud is iteratively processed to detect other geometric primitives such as cylinders and spheres.
- Points not belonging to any primitive are classified as outliers and visualized in gray.

5.7 Visualization and Saving Results

5.7.1 Color-Coding

- Each detected geometric primitive is assigned a random color for easy visualization.

5.7.2 Output Generation

- The decomposed point cloud, including all primitives and outliers, is merged into a single point cloud.
- The result is saved in `.ply` format for further use or analysis.

5.8 Workflow

- An example file (`chair_0904.off`) is processed through the framework.
- The point cloud is decomposed into geometric primitives.
- The output is saved as `trial4.ply`.

6 RESULTS

The decomposition results highlight the exceptional capabilities of PointNet in accurately reconstructing and simplifying complex 3D objects into geometric primitives. The reconstructed primitives closely align with the original structures, preserving intricate details such as the ornate back design of the chair and the precise alignment of the table's legs. This demonstrates PointNet's ability to capture and retain fine-grained features from raw point cloud data, showcasing its strength in effectively representing detailed geometries. Furthermore, the use of geometric primitives allows for efficient approximation of the objects by reducing their complexity while maintaining their essential structural attributes. Such simplified representations are highly advantageous in computationally intensive applications like CAD modeling, robotics, and virtual reality, where reduced complexity leads to improved processing efficiency and faster computations. Moreover, the algorithm's robustness is evident in its ability to handle both simple flat surfaces and complex, intricate designs with ease, ensuring reliable performance across a wide range of geometries. This makes PointNet a powerful tool for tasks requiring precise, scalable, and efficient 3D object decomposition. The outcomes show how well the plane identification algorithm works with various 3d objects the algorithm makes use of open3ds arbitrary sample concurrence ransac technique accuracy was calculated using the formula below:

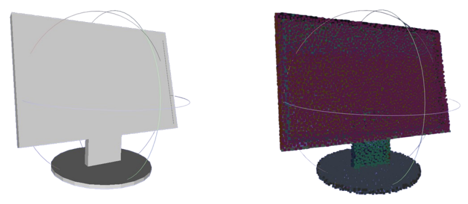
Figure 2: Monitor

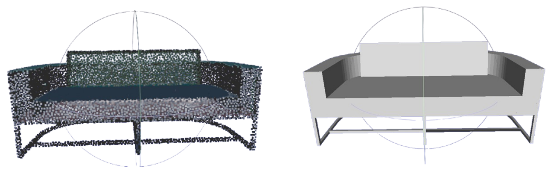
Figure 3: sofa

$$\text{Accuracy} = \min\left(\frac{\text{Detected Planes}}{\text{Expected Planes}}, 1.0\right) \times 100 \quad (1)$$

The algorithm identifies detected planes, while ex-

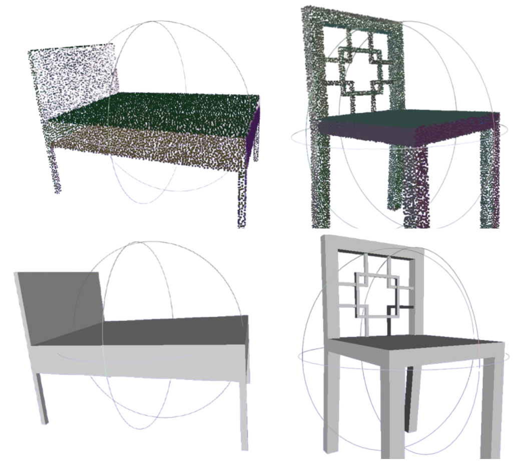
Figure 4: Montior;Bed

pected planes represent the known number of planes in the object. The elapsed time indicates the total time taken to process each object, including reading, detecting planes, and writing results.

Table 1: Comparison of Detected and Expected Planes for Various Objects

Object	Detected Planes	Expected Planes	Accuracy (%)	Elapsed Time (s)
Sofa	7	5	100.00	13.68
Bed	11	5	100.00	9.88
Chair	18	5	100.00	14.49
Table	4	5	80.00	32.16
Monitor	17	5	100.00	27.41

7 CONCLUSION

- PointNet demonstrates high efficiency and effectiveness in the decomposition of 3D objects into geometric primitives. Its use of symmetric functions like max-pooling enables efficient aggregation of global features from unordered point clouds, capturing key geometric structures such as planes, edges, and curves. This eliminates the need for manual feature extraction and parameter tuning, making PointNet superior to traditional methods.
- The architecture supports end-to-end training with standard back-propagation, optimizing performance without manual intervention. It directly processes raw point cloud data, avoiding computationally expensive conversions like voxelization and retaining finer details of the object. Compared to voxel-based methods like VoxNet and 3D-CNNs, Point-Net achieves better memory efficiency and faster inference by operating directly on unstructured data.

Pointnet's ability to learn semantic representations of geometric primitives enhances its robustness in handling complex designs, while its lightweight architecture ensures efficient training and scalability for large datasets. In contrast to point net, which improves local feature capture but increases numerical cost, point net is an effective tool for 3D object deconstruction because it balances accuracy and efficiency.

8 CONCLUSIONS

- PointNet demonstrates high efficiency and effectiveness in the decomposition of 3D objects into geometric primitives. Its use of symmetric functions like max-pooling enables efficient aggregation of global features from unordered point clouds, capturing key geometric structures such as planes, edges, and curves. This eliminates the need for manual feature extraction and parameter tuning, making PointNet superior to traditional methods.
- The architecture supports end-to-end training with standard backpropagation, optimizing performance without manual intervention. It directly processes raw point cloud data, avoiding computationally expensive conversions like voxelization and retaining finer details of the object. Compared to voxel-based methods like VoxNet and 3D-CNNs, PointNet achieves better memory efficiency and faster inference by operating directly on unstructured data.

ACKNOWLEDGEMENTS

The authors thank Sir Virupakshi Hosallimath, Asst. Prof. of Electrical and Electronics at KLE Technological University for their guidance and valuable feedback throughout this research. Special thanks to the other faculties, Asst. Prof. Renuka Ganiger and Asst. Prof. Pooja Chandargi for their assistance. Their support greatly enhanced the quality of this work.

REFERENCES

Ding, Z., Sun, Y., Xu, S., Pan, Y., Peng, Y., and Mao, Z. (2023). Recent advances and perspectives in deep learning techniques for 3d point cloud data processing. *Robotics*, 12(4).

Fu, R., Wen, C., Li, Q., Xiao, X., and Alliez, P. (2023). Bpnet: Bézier primitive segmentation on 3d point clouds. In *IJCAI International Joint Conference on Artificial Intelligence*.

Huang, J. et al. (2018). Deepprimitive: Image decomposition by layered primitive detection. *Computational Visual Media*, 4(4).

Ioannidou, A., Chatzilari, E., Nikolopoulos, S., and Kompatsiaris, I. (2017). Deep learning advances in computer vision with 3d data: A survey. *ACM Computing Surveys*, 50(2).

Liu, H. and Tian, S. (2024). Deep 3d point cloud classification and segmentation network based on gatenet. *Visual Computer*, 40(2).

Liu, Z., Han, Z., and Bu, S. (2019). Deep learning for 3d data processing. In *Deep Learning in Object Detection and Recognition*.