Fast 3D Object Recognition using Surface Normal Distributions

Eduardo Di Santi

August 15, 2025

Abstract

This paper presents a novel and efficient method for 3D object recognition and similarity determination by comparing the distributions of their estimated surface normals. Our approach leverages the concept of representing a 3D object's surface as a multivariate normal distribution, where the statistical properties of the surface normals encapsulate the object's geometry and orientation. We propose using the Bhattacharyya distance, along with other distribution-based metrics, to quantify the similarity between objects. The low-dimensional nature of this statistical representation allows for extremely fast and scalable comparisons, making it ideal for large-scale databases and real-time applications. We benchmark our method against a deep learning-based approach using a pretrained model (PointGPT) and demonstrate its effectiveness. While the AI model shows slightly better performance, our statistical method provides a compelling trade-off, offering near real-time query speeds and minimal computational requirements, which are critical for applications such as quality control in manufacturing and efficient 3D model retrieval.

1 Introduction

Three-dimensional (3D) object comparison and similarity determination are fundamental tasks in fields like computer vision, robotics, and computer-aided design. Traditional methods often involve complex geometric feature extraction or rely on computationally expensive deep learning models **?**. While powerful, these deep learning approaches can require significant computational resources for training and inference, limiting their use in applications that demand rapid, on-the-fly comparisons.

This work proposes an alternative approach to address the need for fast and scalable 3D object recognition. Our method models the surface of a 3D object as a multivariate normal distribution of its estimated surface normals. This statistical representation effectively encodes the object's geometry into a low-dimensional "statistical embedding", enabling rapid comparison. We use the Bhattacharyya distance **?**, as our primary metric to quantify the similarity between these distributions and explore other metrics for a comprehensive

evaluation. The core contribution of this paper is to demonstrate the efficacy and efficiency of this statistical representation for fast 3D object recognition, benchmarking it against a state-of-the-art deep learning model to highlight the trade-offs between performance and computational speed.

2 Methodology

2.1 Object Representation via Surface Normal Distribution

Our method represents a 3D object not by its raw geometry, but by the statistical distribution of its surface normals. A point cloud is a fundamental representation of 3D geometry. First, we estimate the local surface normals for each point in a given point cloud. For this, various techniques can be employed, such as Principal Component Analysis (PCA) or least squares plane fitting on a local neighborhood of points **?**. Once the normal vectors are obtained, they are collectively treated as a set of data points to be described by a multivariate normal distribution.

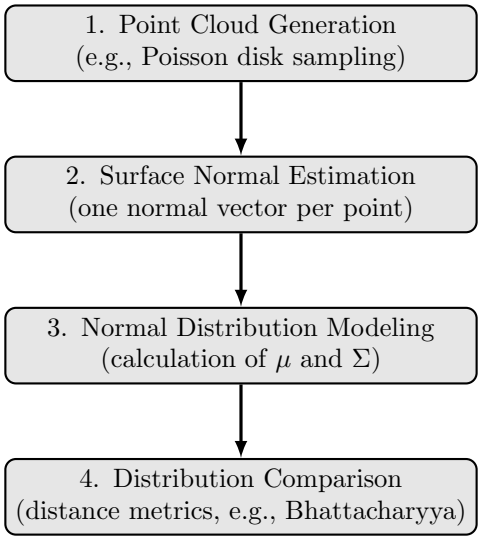

Figure 1: Diagram of the workflow for comparing 3D objects through normal distributions.

Statistical Embedding A multivariate normal distribution is characterized by a mean vector $\boldsymbol{\mu}$ and a covariance matrix Σ. The mean vector represents the central tendency of the surface normals, while the covariance matrix captures the spread and correlations between the normal components.

This process effectively converts a complex 3D object into a compact, low-dimensional statistical representation, which we refer to as a "statistical embed-

ding". This embedding, comprised of the mean vector and covariance matrix, serves as the basis for our similarity comparisons.

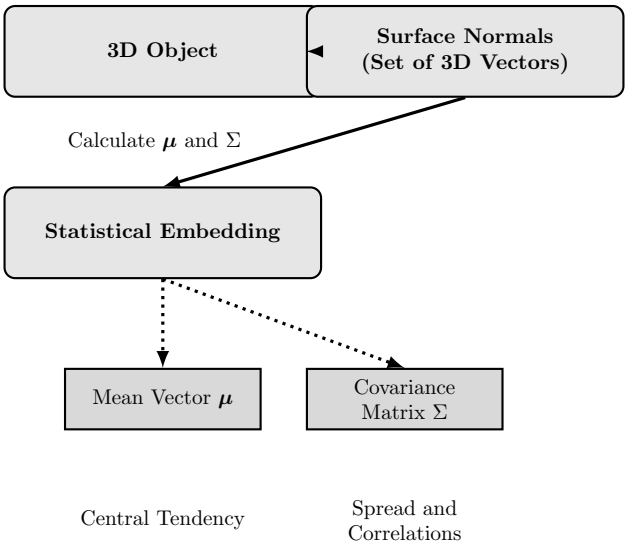

Figure 2: Diagram of the statistical embedding process. A 3D object is first converted into a set of surface normals, which are then summarized by a mean vector $\boldsymbol{\mu}$ and a covariance matrix Σ to form a compact statistical embedding.

The pseudocode for this estimation is as follows:

Algorithm 2 Code to describe a surface normal as a multivariate normal distribution

Require:
normals: A list or array of surface normal vectors for a 3D object
n: Number of normal vectors
dim: Dimension of the normal vectors (e.g., 3 for 3D)

Ensure:
mean: Mean vector
cov_matrix: Covariance matrix

function ESTIMATEMULTIVARIATENORMALDISTRIBUTION($normals, n, dim$)
  $mean \leftarrow$ array of zeros with dimensions $(dim,)$
  $covariance\_matrix \leftarrow$ 2D array of zeros with dimensions (dim, dim)
  for i from 1 to n **do**
    $mean \leftarrow mean + normals[i]$
  end for
  $mean \leftarrow mean/n$
  for i from 1 to n **do**
    $diff \leftarrow normals[i] - mean$
    $covariance\_matrix \leftarrow covariance\_matrix + (diff \cdot diff^T)$
  end for
  $covariance\_matrix \leftarrow covariance\_matrix/n$
  return $mean, covariance\_matrix$
end function

2.2 Similarity Metrics

To quantify the similarity between two 3D objects, we compare their respective multivariate normal distributions. We primarily use the **Bhattacharyya distance** **?** due to its effectiveness in measuring the overlap between two probability distributions. The Bhattacharyya distance $D_B(N_A, N_B)$ between two distributions N_A and N_B is computed as:

$$D_B(N_A, N_B) = \frac{1}{8}(\boldsymbol{\mu}_B - \boldsymbol{\mu}_A)^T \Sigma^{-1} (\boldsymbol{\mu}_B - \boldsymbol{\mu}_A) + \frac{1}{2} \ln \left(\frac{\det(\Sigma)}{\sqrt{\det(\Sigma_A)\det(\Sigma_B)}} \right)$$

where $\Sigma = (\Sigma_A + \Sigma_B)/2$. A lower Bhattacharyya distance indicates greater similarity.

Additionally, we evaluate other common distance metrics for comparison:

- **Mahalanobis Distance ?**: Measures the distance between a point and a distribution, or between two distributions, accounting for the data's variance.
- **Jensen-Shannon Divergence ?**: A symmetric and smoothed version of the Kullback-Leibler divergence, used to measure the difference between two probability distributions.

2.3 Deep Learning Comparison

To contextualize the performance of our method, we compare it against a deep learning-based approach. We use a pretrained model, specifically PointGPT-S **?**, a GPT-like model for point clouds. Instead of using the model for its original purpose of point recreation, we intercept its forward pass to extract a vector representation (embedding). These vector embeddings, typically of higher dimensionality (e.g., 384 dimensions), are then used to perform similarity queries via cosine similarity. This comparison allows us to evaluate the trade-offs between a feature-rich, data-driven representation and our computationally light, statistical approach.

3 Experimental Results

3.1 Benchmark Setup

The experiments were conducted using the ShapeNet dataset **?**. Our primary goal was to measure the effectiveness and computational efficiency of our statistical method against the deep learning model. The statistical embedding for our method was composed of 30 dimensions, based on statistical calculations of the surface normals for each dimension, including the covariance matrix, standard deviation, and percentiles.

Artifact	version
Python	3.9.16
Matplotlib	3.8.0
Open3d	0.14.1
IPython	8.14.0
Numpy	1.25.2
scipy	1.11.3

Table 1: Software requirements for reproducing the results

3.2 Performance Metrics

We evaluated both methods using standard information retrieval metrics: **recall@k** and **mean average precision (mAP@k)**. The goal was to check the retrieval of shapes with the same taxonomy as the query shape.

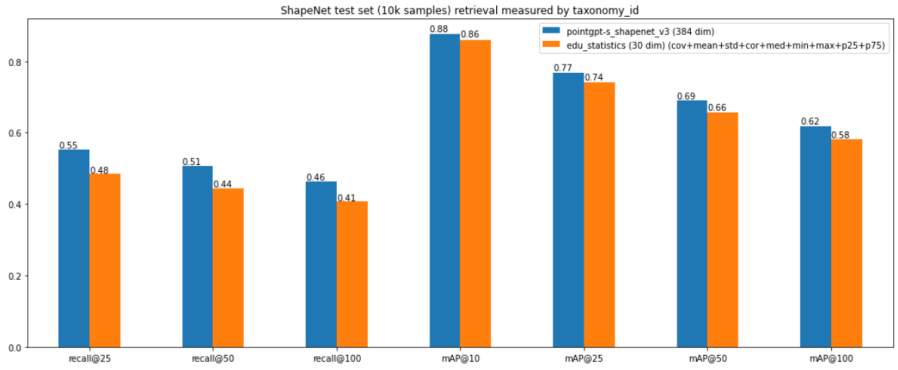

Figure 3: Recall and mAP metrics for both approaches

The results show that the AI method has a slightly better performance. This can be attributed to the fact that the AI model was trained on the ShapeNet training set, introducing a potential bias that contributes to its better performance on the test set.

3.3 Computational Efficiency

A key advantage of our statistical method is its computational efficiency. The 30-dimensional statistical embeddings are extremely fast to compute and require negligible storage. This allows for near real-time similarity searches, a task that would be significantly slower using the full forward pass of a complex deep learning model. Furthermore, our embeddings can be generated quickly on a CPU, eliminating the need for a dedicated GPU.

3.4 Empirical Example

Given the following three objects, our method provides a distance matrix for comparison:

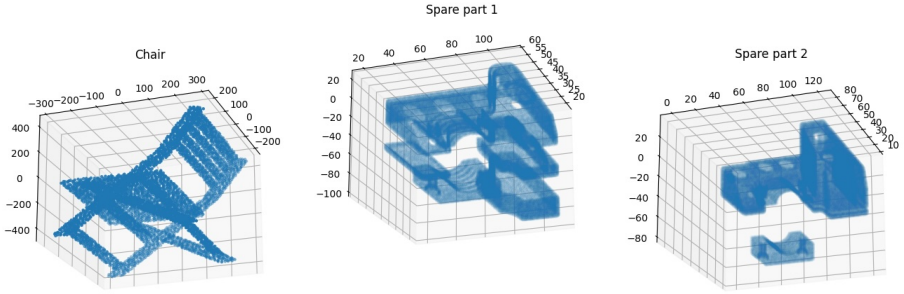

Figure 4: Algorithm input surfaces

	chair	spare part 1	spare part 2
chair	-	(30.74, 41.69)	(30.49, 45.78)
spare part 1	(30.74, 41.69)	-	(19.14, 0.00)
spare part 2	(30.49, 45.78)	(19.14, 0.00)	-

Table 2: Distances comparing normal surfaces, (not normalized, normalized)

4 Discussion and Future Work

Our findings indicate that a statistical representation based on surface normal distributions provides a highly efficient and effective alternative to deep learning for 3D object recognition. While a pretrained AI model may offer slightly higher accuracy, our method excels in computational speed and resource requirements, making it an ideal choice for applications where these factors are critical, such as quality assessment or IP field analysis.

For future work, we suggest exploring the integration of generative AI models with our statistical representations. This hybrid approach could address challenges like surface deformations by generating plausible variations of an object consistent with its learned statistical properties. By combining the efficiency of our method with the robust deformation handling of generative models, we can further enhance the accuracy and reliability of 3D object comparison.

5 Conclusion

We have demonstrated a method for comparing 3D surfaces using estimated normal distributions, providing a robust and fast approach for object recognition. Our statistical embedding, coupled with distance metrics like Bhattacharyya, offers a powerful tool for analyzing surface similarity with minimal computational overhead. This work paves the way for advancements in 3D object analysis and recognition, particularly in scenarios where rapid and scalable comparisons are essential.

References