g-RIPS Sendai 2024

THE MITSUBISHI-A TEAM

Final Report

Authors:
Kean Fallon[1]
Xiwen Jiang[2]
Melchior N'Bouke[3]
Ryu Ueno[4]
Changyu Zhou[5]

Mentors:
Dr. Shunsuke Kano[+]
Dr. Masaki Ogawa[+]
Dr. Masashi Yamazaki[*]
Mr. Akinobu Sasada[*]

Institute
[1] Iowa State University
[2] University of California, Irvine
[3] AIMS South Africa
[4] Hokkaido University
[5] Tohoku University
[+] Academic Mentor, Tohoku University, MathCCS
[*] Industrial Mentor, Mitsubishi Electric

August 8, 2024

Contents

1 Introduction

Point clouds are a collection of finitely many points in three-dimensional space, where each point represents an objects position. Point clouds are typically generated by 3D scanning devices like LiDAR (Light Detection and Ranging) scanners, depth cameras, or photogrammetry techniques. They can be converted into the familiar formats of mesh models, CAD models, or NURBS surface models through a process known as surface reconstruction. As the output of 3D scanning processes, point clouds offer a rich source of 3D spatial information that is vital for object detection across various domains.

One crucial step involved in object detection application is processing point clouds to extract feature vectors. These feature vectors contain information that a model then uses to classify, segment, or perform object detection on the point cloud with.

In the state of research today, point clouds are processed by either hand-crafted or deep learning methods. For the former, mathematical rules are designed and then applied to extract feature vectors which are then fed into an appropriate classifying, segmenting, or object detection model. For the latter, we can either directly process point clouds or convert point clouds into mesh, voxel or multi-view data and process the converted data. Both have strengths and weaknesses: hand-crafted methods do not need as much data to work well, yet do not perform as well as deep learning methods, while deep learning methods perform well but require large, and notably expensive, data sets to train on. However, the large and expensive data sets required for deep learning methods to train is cost-prohibitive to many who wish to use them. In this report we introduce a novel hand-crafted feature vector extraction method called Multi-scale Geometry and Topology (**MsGT**). Since hand-crafted methods don't require large data sets to train on, we circumvent the associated data costs. Moreover, while not as accurate as deep learning methods, hand-crafted methods are still effective enough to be used for a variety of nontrivial tasks. In the following section, we provide all of the necessary background information, making explicit much of what we have considered to this point. If the reader is familiar with point cloud processing and the state of existing research, they may pass over this section with little consequence.

2 Background

2.1 Preliminaries

Definition 2.1. A *point cloud* is a set of finitely many points in three-dimensional Euclidean space.

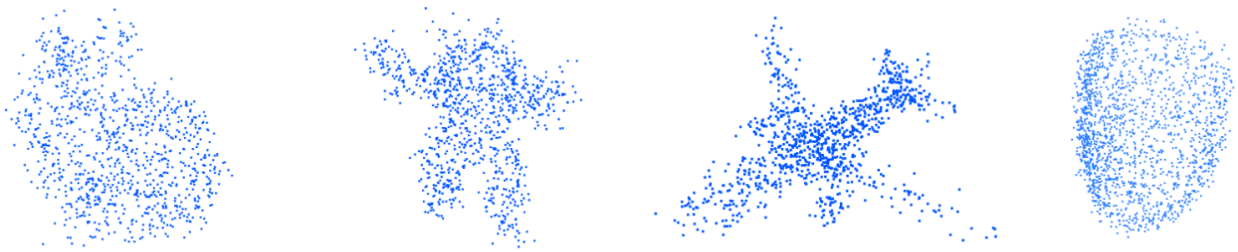

Figure 1: Examples of point clouds

Intuitively, we just consider a point cloud as a collection of points in $\mathbb{R}^3$, identifying them with vectors when necessary for computing things such as inner products. If desired, one could include m different attributes, such as color or intensity, by taking the collection of points to be in $\mathbb{R}^{3+m}$. However, we consider point clouds to be defined exactly as in definition 3.1 and that convention will remain for the rest of this document.

Encountering a *neural network* is inevitable when working with 3D data classification, hence is the other essential architecture that must be explained.

Definition 2.2. A *neural network* is a directed graph with vertices called "nodes" through which an input is passed, resulting in an output.

Neural networks are modeled on the human brain, hence the *neural* in neural network. Each node generally consists of an activation function composed with another function. To achieve the desired result, a loss function for the output is created and the parameters that define the node functions are tuned by various methods, such as gradient descent with respect to the loss function. Rather than attempt a more rigorous definition then the one above, we provide an example.

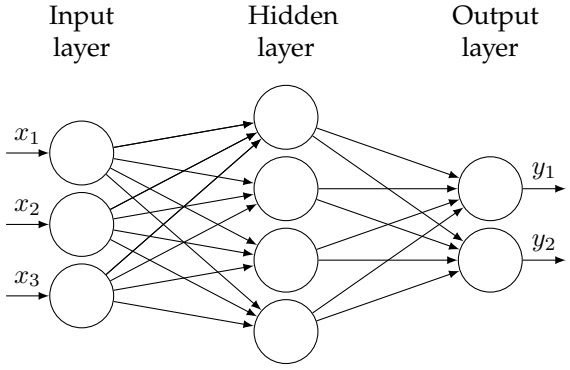

Figure 2: A basic single-layer neural network.

Example 2.3. Figure 2 shows a basic single-layer neural network with three inputs, four nodes, and two outputs. The layers that contain the nodes (or "neurons") are called *hidden layers* and a network with many hidden layers is referred to as *deep*. In this example, each input is passed to each node. The ith node receives each of the inputs and constructs a linear combination

$$\varphi_i(x_1, x_2, x_3) = a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 + b_i$$

where $a_{i1}, a_{i2},$ and a_{i3} are the *weights* and b_i is the *bias*. Afterwards, the linear combination is fed into an activation function R such as a sigmoidal function or a rectified linear unit (ReLU). The result $R(\varphi_i(x_1, x_2, x_3))$ is then sent to each of the subsequent nodes where a similar process happens, and so on. At the end, the loss between expected output and actual output is calculated and the parameters (i.e. the collective weights and biases) are adjusted to improve the model. The process then repeats.

Neural networks are at the forefront of data science. There is a large selection of literature available on the practicality of neural networks and the theory behind them, including a host of universal approximation theorems. Notably, they are often applied to the 3D data processing applications that we are interested in: classification, segmentation, and 3D object detection.

Object classification is the most straightforward of the three applications above, as its only task is to provide a category (or classification) of a point cloud, which the method selects from a list that it has been trained on. Segmentation is the process of classifying multiple objects within a point-cloud, and therefore is more difficult. Finally, 3D object detection is the task of constructing bounding boxes around separate objects in a point cloud. Object detection is the most interesting and arguably useful of these applications, and so unsurprisingly the most difficult. The information required to perform any of the above three applications is contained in what is called a feature vector.

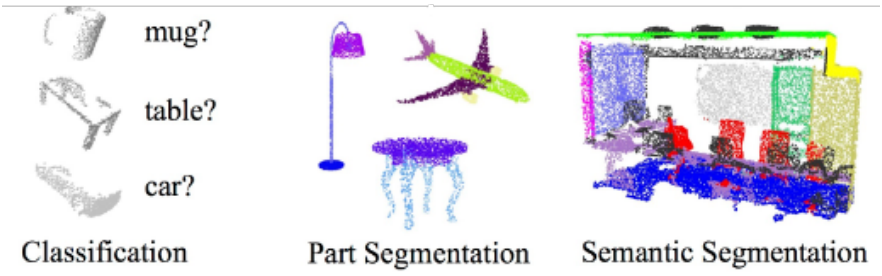

Figure 3: Point cloud classification and segmentation.

Definition 2.4. A *feature vector* of a point cloud is a vector $v \in \mathbb{R}^n$ whose n components represent different numerical data corresponding to the point cloud.

For example, a feature vector v of a colorized point cloud could include 6 data entries: the first three its (x, y, z) coordinates relative to some reference, and the second three it's RGB values (r, g, b). The concatenated vector $v = (x, y, z, r, g, b)$ lives in $\mathbb{R}^6$ and provides a simple example of a feature vector.

As previously discussed in the problem statement, the objects of interest in this document are feature vectors. Our goal in this project is to design a *hand-crafted feature vector extractor*. In other words, we want to design a mathematical algorithm that takes in point cloud information, applies mathematics in a unique way, and provides a feature vector for input into an existing model. We select an example of such a process which can be found in [62].

Example 2.5. Given a point cloud, fix a radius r and select a point p. In the sphere of radius r and center p, compute the normals n_i (with respect to the sphere) of each point p_i contained therein, then reorient them in a consistent manner. Next, find the source point p_s and target point p_t by the following:

$$\text{if } \arccos\langle n_i, p_j - p_i\rangle \leq \arccos\langle n_j, p_i - p_j\rangle \text{ then set } p_s = p_i \text{ and } p_t = p_j, \text{ else } p_s = p_j \text{ and } p_t = p_i.$$

Set $u = n_s$, $v = (p_t - p_s) \times u$, and $w = u \times v$ (this collection is called a Darboux frame). The four geometric features collected are:

$$\begin{aligned} f_1 &= v \cdot n_t \\ f_2 &= \|p_t - p_s\| \\ f_3 &= u \cdot (p_t - p_s)/f_2 \\ f_4 &= \arctan(w \cdot n_t, u \cdot n_t) \end{aligned}$$

While there is more to the process in [62], this provides a short and sufficient example of hand-crafted feature extraction.

A notable property of point clouds is that they contain the same geometric information regardless of the perspective you view them from. This property also creates an obstacle that must be overcome to effectively process the data in point clouds and extract a feature vector.

Definition 2.6. The *Euclidean group* of $\mathbb{R}^3$, denoted $E(3)$, is the collection of all rotations, reflections, and translations of $\mathbb{R}^3$. In other words

$$E(3) \cong O(3) \rtimes T(3) \quad \longleftrightarrow \quad E(3)/T(3) \cong O(3)$$

where $O(3)$ is the orthogonal group and $T(3)$ is the translational group. To put it another way, the Euclidean group $E(3)$ is the collection of all transformations of $\mathbb{R}^3$ that preserve distances between points.

The motivation for including this definition is the following: in processing point clouds, it is required that the processing does not change after point clouds are rotated, reflected, or the point of view is shifted. In other words, point cloud processing needs to be invariant under applications of the elements of $E(3)$ on the point cloud. It should be readily seen that this is a necessary requirement for a functioning point cloud processor.

A final piece of point cloud information that must be respected is invariance under a relabeling. In other words, the feature extraction method which processes a point cloud must not depend on the order in which it reads the points. Again, it does not take much thought to see why this is an important feature.

To summarize, any feature extraction method (hand-crafted or not) M must adhere to the following two rules:

1. M is invariant under $E(3)$
2. M is does not depend on the order of its inputs.

These principles have guided existing research and consequently must guide ours as well.

2.2 State of Current Research

Point cloud feature extraction can be separated into two primary methods: feature extraction based on deep learning and hand-crafted feature extraction. Deep learning methods are the most recent and currently most successful methods. This is not surprising due to the power of deep learning models in many modern day applications. Hand-crafted methods are nonetheless still useful, even while being outperformed by deep learning models, due to their ability to work on small data sets while still performing effectively.

2.2.1 Deep Learning Methods

While not the focus of our statement, much of current research focuses on processing point clouds directly via deep learning. Therefore, we will (briefly) acknowledge and summarize deep learning methods here.

In 2017, Qi et al. introduce PointNet in their landmark paper [51]. PointNet is a deep learning method that applies a transformer network module to point clouds before aggregating the point features to a single global feature vector with a symmetric max pooling function.

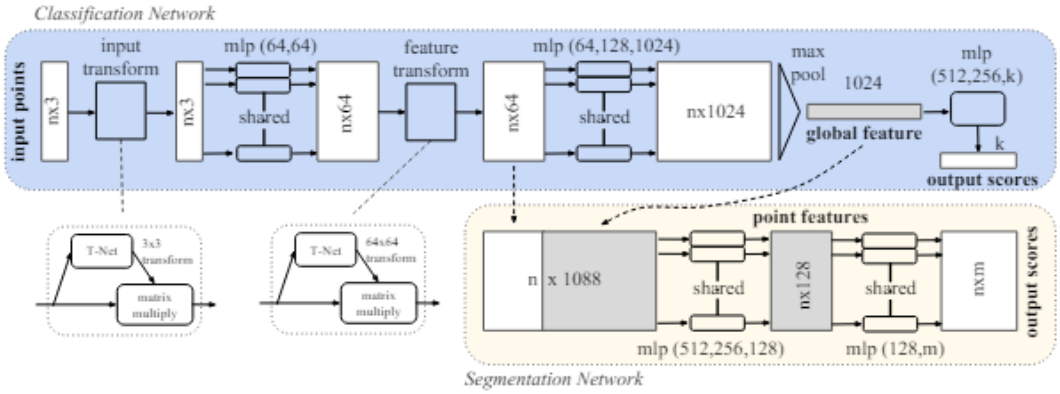

Figure 4: The PointNet architecture.

The key features of PointNet are its T-Net module and symmetric aggregation function, which provide solutions to the $E(3)$ invariant and unordered properties of point clouds, respectively. Before PointNet, deep learning models had not been applied to point clouds due to the difficulties in reconciling these

properties. PointNet, its successors PointNet++ [52] and PointNeXt [53], and their contemporaries, such as in [3, 80, 78, 81, 72, 45, 71, 44, 25] have all successfully handled these obstacles and opened the door to applying deep learning on point clouds with significant success.

However, the training necessary for deep learning methods requires datasets that are expensive to generate. In particular, point cloud datasets must be manually collected and annotated by hand. Given the large dataset required, this process requires substantial resources which may not be readily available to many who wish to use it. For this reason, hand-crafted methods still have relevance as a way of circumventing resource restrictions, even with the greater success of deep learning methods.

2.2.2 Hand-Crafted Methods

Hand-crafted feature extraction has a longer history in 3D data collection, with results dating back at least as early as 1998 [32]. However, the success of PointNet in 2017 has in turn commanded the attention of researchers towards deep learning models. As a result, work in hand-crafted feature extraction has slowed significantly.

As opposed to directly processing via a neural network, hand-crafted feature extraction processes point clouds by detecting geometric structures mathematically, such as distances between points, normal vectors, and angles, among others. It then reports them in a feature vector, sometimes after a classification process such as binning. Existing results in hand-crafted feature extraction can be classified into three different categories: local feature extraction, global feature extraction, and a combination of the two (hybrid feature extraction). Here we provide a brief summary of the results in each, including some of their strengths and weaknesses.

Local Feature Extraction. Local feature extractors are designed to pull local information from points, such as surface curvature. These local feature extractors can be further divided into two categories: those which use a *local reference frame* (LRF) and those which do not.

The former establishes a LRF for a given point, partitions the neighborhood around the point with respect to the LRF, and then processes the point cloud by observing spatial distribution, geometric properties, or other relationships. It is common in spatial distribution methods to perform eigenanalysis on the spherical support of a point and spectral decomposition is used frequently. Also ubiquitous is the identification of the covariance matrix (weighted or unweighted) of a point, i.e. the symmetric, positive semi-definite matrix indexed by the points in a neighborhood whose ij-th entry corresponds to the covariance between points p_i and p_j. Examples of these methods are [68, 65, 29, 6, 70]. LRF-based methods that apply geometric properties are less common but still exist: in fact example 3.5 is an example of such a method.

Non-LRF methods do not use reference frames unique to the image itself. In particular, they pull information from points outside of a given point neighborhood to provide more context, which in turn gives a more rigorous description of the features. One type of non-LRF method, called *signature-based*, involves this description being compressed into a signature which is then compared to signatures of other patches/neighborhoods. The other types are statistical methods, e.g. histogram descriptors.

Histogram descriptors are a type of feature representation commonly used in computer vision and pattern recognition tasks. These descriptors capture the distribution of certain properties or characteristics of data points within a specified neighborhood or region. In the context of 3D laser range data classification, histogram descriptors are utilized to encode information about the surrounding points or normals relative to a query point.

Histogram descriptors maintain a histogram that quantifies the occurrence or distribution of specific properties of neighboring points. By binning the values of these properties into a histogram, the descriptor provides a compact and informative representation of the local geometry or shape features around a point. This allows for effective discrimination between different classes or categories of objects based on their spatial characteristics.

Examples of non-LRF methods can be found in [32, 43, 63, 36, 42, 67, 50, 56, 57, 61, 62, 39, 30, 19, 8, 12] with statistical methods found in [21, 20, 69, 79, 83].

Global and Hybrid Feature Extraction. Global feature extraction methods return a feature vector for an entire point cloud, which can be useful for tasks such as object classification. Similarly to local feature extraction, global feature extractors tend to extract either spatial distribution information or geometric attributes. Although different than the local feature extractors, methods that fall into this category are similar in spirit. These include methods such as Fourier descriptors, Hu moments, Bag-of-Visual-Words, contour-based methods, and more (see [58, 46, 41, 40, 38, 4, 76, 17]).

As one may expect, hybrid feature extraction combines properties of global and local feature extractors to aggregate local and global information for data processing tasks. Therefore, they would be useful in tasks (e.g. segmentation) where both global and local information is desired. Hybrid feature extractors can be found in [73, 27, 24, 48].

3 Our approach

MsGT extracts features using two core concepts: Multi-scaled Geometry, and Topology. In particular, **MsGT** provides new, novel geometric features that show promise in the field of pointcloud classification. After collecting local geometric features at n different scales and then topological features globally, our method concatenates them, resulting in something of the form:

$$\text{feature vector} = \left[\ \text{global} \ \middle|\ \text{local}_1 \ \middle|\ \cdots \ \middle|\ \text{local}_n \ \right]^T$$

Afterwards, the vector is normalized and then individual segments are weighted to improve accuracy. For more on our explicit constructions, see section 4.

3.1 Geometry

Our novel geometric features are inspired by geometry and the study of smooth surfaces. In the theory of smooth surfaces, many geometric quantities are available to us, such as Guassian and mean curvature. These two curvatures alone are enough to determine how the surface is shaped locally around a point. These geometric informations can also be obtained on pointclouds, but the methods of obtaining them are not unique. We were able to discover new methods of obtaining these curvatures. In particular, we use these methods to find local geometric features, that is, features that are determined on each point of the point cloud. Local features have advantages of being fast and simple while also detecting the important details in the point cloud.

We need normal vectors on each point of the surface in order to obtain the Gaussian and mean curvatures. Just like obtaining curvature, the methods used to obtain normal vectors are not unique [60, 31, 5]. To obtain the normal vectors on each point in the point cloud, we use a method based on the principal component analysis [28].

Definition 3.1. Let p be a point in a point cloud P.
(1) The **center c** of P is the mean of all the points in P:

$$\mathbf{c} = \frac{1}{|P|}\sum_{q\in P} q.$$

(2) The **weighted covariance matrix** C of p obtained by the neighborhood $\{q_1, ..., q_N\}$ around p is given by the following:

$$C = \frac{1}{N}\sum_{i=1}^{N} w_i(q_i - m)(q_i - m)^T. \tag{3.1}$$

Here, $w_i = \|q_i - p\|^{-1}$ is a weight defined on each neighboring points, and m is the center of the point cloud $\{q_1, ..., q_N\}$. If we let $X_j = \{(q_1)_j, ..., (q_N)_j\}$ for $j \in \{1, 2, 3\}$, the elements of the covariance matrix are

$$C_{ij} = \mathrm{Cov}_w(X_i, X_j), \quad i, j \in \{1, 2, 3\},$$

where Cov_w is the covariance weighted by $w = \{w_1, ..., w_N\}$.
(3) We have non-negative eigenvalues $0 \leq \lambda_0 \leq \lambda_1 \leq \lambda_2$ and corresponding eigenvectors v_0, v_1, v_2 of the covariance matrix. The eigenvectors v_1, v_2 generate a tangent plane on the point p. The **normal vector** $\mathbf{n}(p)$ of p is given by following equation:

$$\mathbf{n}(p) = \begin{cases} v_0 & if \quad \langle v_0, p - \mathbf{c} \rangle \geq 0, \\ -v_0 & if \quad \langle v_0, p - \mathbf{c} \rangle < 0. \end{cases}$$

Here, $\langle \cdot, \cdot \rangle$ is the Euclidean inner product.

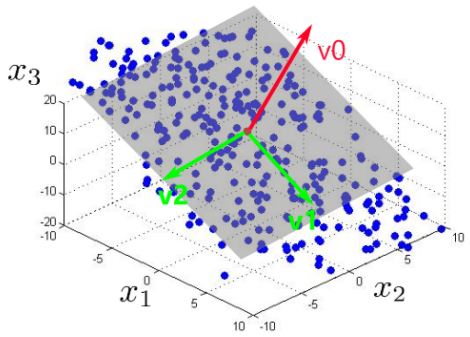

Figure 5: Tangent plane and its normal vector

Remark 3.2. The normal vector $\mathbf{n}$ is a $\mathbb{R}^3$ valued function on the point cloud. Rotate a point cloud P to P_A by $A \in O(3)$, that is, $P_A = \{Ap \mid p \in P\}$. Let $\mathbf{n}_{PA}$ be the normal vector of P_A. Then we have $\mathbf{n}_{PA}(Ap) = A\mathbf{n}(p)$ for $p \in P$.

Gaussian curvature and mean curvature are obtained by using the principle curvatures. On a smooth manifold M, the principle directions on p are the unit vectors $u \in T_pM$ such that the directional derivative $D_u\mathbf{n}$ of $\mathbf{n}$ along u, that is,

$$D_u\mathbf{n} = \lim_{t \to 0} \frac{\mathbf{n}(\gamma_u(t)) - \mathbf{n}(\gamma_u(0))}{t}$$

is linearly dependent to u. Here, $\gamma_u(t)$ is some curve on M such that $\gamma_u(0) = p$ and $\dot{\gamma_u}(0) = v$. The principle curvatures k_1, k_2 on p are determined by two linearly independent principle directions $u_1, u_2 \in T_pM$ by

$$k_i = \begin{cases} \|D_{u_i}\mathbf{n}\| & if \quad \langle u_i, D_{u_i}\mathbf{n} \rangle \geq 0, \\ -\|D_{u_i}\mathbf{n}\| & if \quad \langle u_i, D_{u_i}\mathbf{n} \rangle < 0, \end{cases} \quad i \in \{1, 2\}.$$

The Gaussian curvature K_p and mean curvature H_p on p are defined as follows:

$$K_p = k_1 k_2, \quad H_p = \frac{1}{2}(k_1 + k_2).$$

Let p be a point in a point cloud, and $\{q_1, ..., q_N\}$ a neighborhood of p. It is difficult to obtain the principle directions or the principle curvatures on point clouds. Instead, we directly define the Gaussian and mean curvatures. For a point q_i in the neighborhood, we define the curvature $k(p, q_i)$ along $q_i - p$ of

by

$$k(p, q_i) = \frac{\|\mathbf{n}(q_i) - \mathbf{n}(p)\|}{\|q_i - p\|}.$$

Here, $\|\cdot\|$ is the norm defined by the Euclidean inner product. The absolute value of Gaussian curvature is defined by

$$K_p = \left(\prod_{i=1}^{N} k(p, q_i)\right)^{\frac{1}{N}}.$$

On the other hand, when we add the curvature along $q_i - p$, we also want to consider the angle between $\mathbf{n}(q_i)$ and $q_i - p$. Therefore, the mean curvature on p is obtained by the following:

$$H_p = \frac{1}{N}\sum_{i=1}^{N}\left(k(p, q_i) \cdot \frac{\langle \mathbf{n}(q_i), (p - q_i)\rangle}{\|p - q_i\|}\right)$$

It should be clear from inspection and remark 3.2 that both K_p and H_p are invariant under distance-preserving maps. Any rotation will not affect an inner product (hence also a norm), and translation does not change our results. This latter fact is because $T(p) - T(q) = p - q$ for any translation T and points p, q. To improve our results as much as possible, we include existing local features that are defined using the eigenvalues of covariance matrices on points [74]. Let p be a point in a point cloud, and $0 \leq \lambda_0 \leq \lambda_1 \leq \lambda_2$ the eigenvalues of the covariance matrix on p determined by some neighborhood. The local features we use are determined by the following equations:

$$\frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{1}$$

$$(\lambda_0 \cdot \lambda_1 \cdot \lambda_2)^{\frac{1}{3}} \tag{2}$$

number	feature	description
(1)	surface variation	how the points around are arranged close to a plane
(2)	omnivariance	how points are spread out around

In addition, we compute the centroid of each neighborhood as a way of collecting elementary information about local geometry.

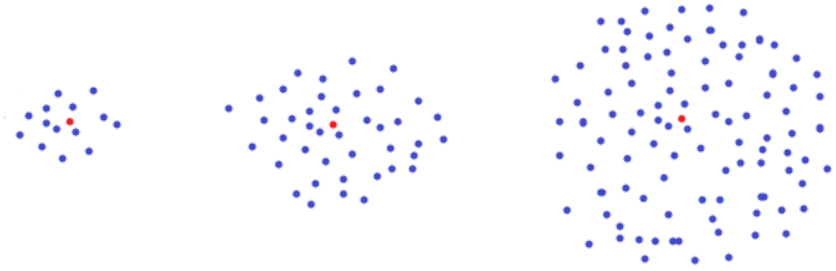

Figure 6: Extract local features at different scales

3.2 Multi-scale

While fine details may be clearer at a higher resolution for some geometric features, some features are more visible at a lower resolution. By using a multi-scale approach, we can extract features at different

scales to capture a comprehensive view of point clouds features. By multi-scale, we mean extracting features along discretely varying sizes of neighborhood (see Figure 6).

Extracting features at different scales avoids the problem of determining a one-size-fits-all choice of neighborhood at the expense of some additional computing time. As the metric we evaluate our success on is accuracy, this was deemed a worthwhile tradeoff.

3.3 Topology

Although there has been substantial progress on extracting and encoding discriminative geometric information, research looking into the topological structure of data has only been examined recently with the rise of topological data analysis (TDA). In TDA, a branch of mathematics and computational science that applies concepts from algebraic topology to analyze and extract information from complex datasets, persistent homology is used to analyze the topological features of a dataset across multiple scales or resolutions. It captures how long topological features (such as connected components, loops, and voids) persist as we vary a parameter, often the radius of a ball used to probe the space.

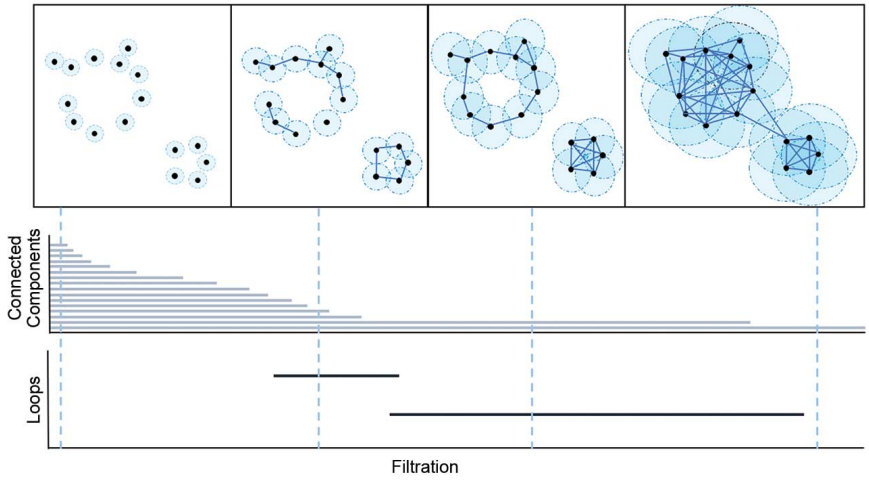

Figure 7: Filtration and corresponding plot (See [37])

In practice, the first step of the TDA Analysis pipeline is to define a filtration of simplicial complexes for some data. A simplicial complex, which is a set of simplices, can be seen as a higher dimensional generalization of graphs. The complexes are used to approximate shapes such as non-discrete sets and continuous mathematical shapes like curves, surfaces and more generally, manifolds. A filtration, an increasing sequence of sub-complexes of a simplicial complex, can be seen as ordering the simplices included in the complex. Simpicial complexes, including Vietoris-Rips complexes, Cech complexes and alpha complexes often come with a specific order. In persistent homology, filtration helps track how topological features such as connected components, loops, and voids appear and disappear as the parameter changes.

Persistence diagrams are a key visualization tool used in persistent homology. It shows how long topological features, such as connected components (0-cycles), loops or holes (1-cycles), and higher-dimensional features (2-cycles, 3-cycles, etc.), persist across different scales or resolutions. As shown in Figure 9, as the radius parameters varies, connected components and 1-dim loop get assigned a birth and death.

Persistent diagrams are difficult for machine learning to handle since they represents data as multisets in a half-plane, which complicates their direct application in many models. To overcome this difficulty, several vectorization techniques have been developed to transform persistence diagrams into formats more adaptable to machine learning algorithms. Notable methods include persistence landscapes [9] and persistence values.

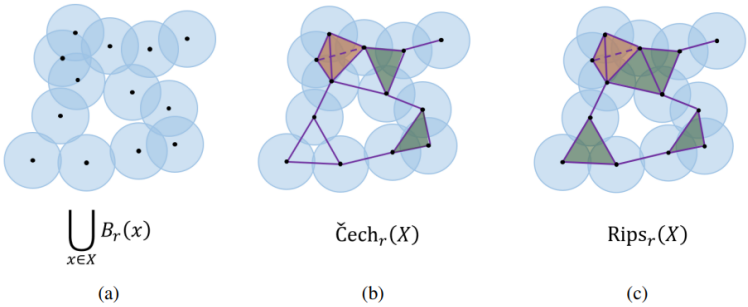

Figure 8: Different filtration methods (See [37])

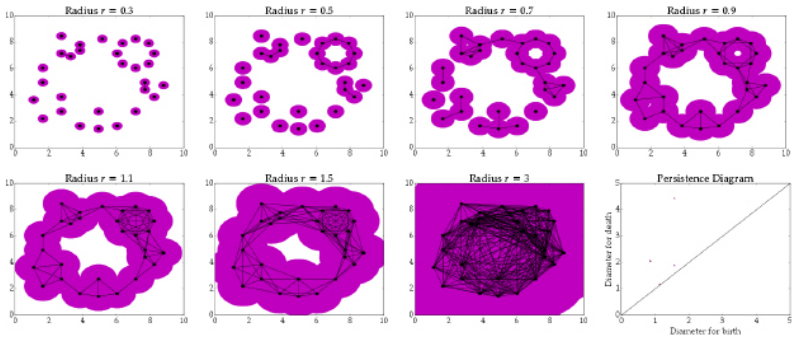

Figure 9: Persistent homology is used with point clouds by constructing Rips complexes (See [47])

We get persistence values from the persistence diagram. For each feature α with birth b_α and death d_α, we define by $v_\alpha = d_\alpha - b_\alpha$ the persistence value of the feature α. For the MsGT feature extracion, we compute the mean, median, standard deviation and skewness of the features and add them to our final feature vector.

Definition 3.3. The population mean is the sum of all the population values divided by the total number of population values.

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n}, \tag{3.2}$$

where

- μ is the population mean,
- n is the total number of observations,
- x_i are a particular value and

Definition 3.4. The *median* m is the middle data point when the dataset is arranged in order from smallest to largest. If there are two middle values then we take the average of the two values.

Definition 3.5. The *standard deviation* σ measures how spread out the data is. A simple formula for calculating the standard deviation is:

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu)^2} \tag{3.3}$$

Definition 3.6. The *skewness* $\tilde{\mu}$ is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. The formula for calculating the standard deviation is:

$$\tilde{\mu} = \frac{\sum_{i=1}^{n} (x_i - \mu)^3}{\sigma^3 (n-1)}. \tag{3.4}$$

Persistence landscape converts persistence diagrams into a series of functions over a real line, providing a scalar summary that captures the essential features of the diagrams.

- Construction of persistence landscape
  - **Landscape function:** created from the persistence diagram using weighted sums of piecewise linear functions,
  - **Evaluation:** landscapes are evaluated at specific points to generate feature vectors.
- Advantages
  - **Dimensionality reduction:** simplifies the persistent diagram while preserving essential topological information,
  - **Computational efficiency:** facilitates easier and faster analysis compared to raw persistent diagrams.

Our MsGT model computes persistence landscapes with the *Ripser* python package and includes them in our feature vector, forming the other part of our topological set.

4 Simulation

In this section we will explain the dataset we used in this study, the explicit construction of our feature extraction method, and discuss the results and figures from our simulation. In particular, we analyze the effectiveness of our model against other benchmarks, the importance of the inclusion of a multi-scale approach and topological global data, and the effectiveness of our novel geometric features compared to standard existing geometric features. All classification was performed using XGBoost, a gradient boosting method, with parameters:

Multiclass classification was also performed with other classifiers and overall accuracy was recorded: see table 3 in the appendix.

4.1 Data and Data Pre-processing

The dataset employed for this study is ModelNet10, a well-established benchmark in the domain of 3D object recognition and classification. ModelNet10 comprises 10 distinct categories, each containing a variety of 3D CAD models, including everyday items like chairs, tables, airplanes, and cars. The training and testing data were directly downloaded from the https://modelnet.cs.princeton.edu/ ModelNet website.

To prepare the data for network training and evaluation, we performed several preprocessing steps. Initially, we sampled each 3D model to obtain a uniform set of 1024 points. This fixed sampling rate ensured a consistent input size, optimizing the network's processing efficiency. Subsequently, we normalized the point coordinates, scaling them relative to each model's bounding box. This normalization step ensured that all points fell within a standardized range, enhancing the robustness and convergence speed during training.

To further augment the dataset and improve the model's generalization ability, random noise was introduced to the point cloud data. This addition of noise created slight variations in point positions, simulating real-world conditions. For reference later on, the classes are numbered as follows:

4.2 Construction

The explicit construction of our feature vector goes as follows:

- For each point p in the pointcloud, we record the following features in a local neighborhood:
  - Gaussian curvature
  - Mean curvature
  - Omnivariance
  - Surface Variation
  - Mean position of neighbors

  The selection of the supplementary geometric features (omnivariance and surface variation) was decided by both testing and a difference in feature highlighting (see appendix, figures 16 through 19). It should be noted that mean position is given in (x, y, z) coordinates, hence this feature has three scalar entries as opposed to the others, which have one. Thus there are seven scalar entries for each point in each neighborhood.

- These measurements are taken at five different sizes of neighborhood: 16, 32, 64, 128, and 256. The neighborhood size was decided based on limited testing and with the philosophy that exponentially increasing size is necessary to detect changes in large neighborhoods. These measurements are concatenated for each point, then added to the feature vector for a given image. Nearest neighbors search is done by the KDTree method.

- The geometric portion of our extraction method thus yields

$$\text{(\# of nbhds)(\# of scalar entries)(\# of points)} = (5)(7)(1,024) = 35,840 \text{ scalar entries}$$

- For global feature extraction, we first recorded statistical information from persistence values. Specifically, mean, median, standard deviation, and skewness. Afterwards, we included persistence landscapes computed for the persistence diagrams of each point cloud, with the number of landscapes set to 7 and number of steps at 200, yielding $1,400$ scalar entries. Again, this was chosen by limited testing. All persistence diagrams were computed using the *Ripser* python package. In summation, for each pointcloud, these topological features contributed 1404 points. Thus our feature vectors contained $37,244$ entries.

4.3 Benchmark Testing

The first section of our testing compares the success of our method against a deep-learning benchmark in PointNet and a hand-crafted benchmark in Fast Point Feature Histogram (FPFH) [56]. Both were chosen due to their popularity and also the availability of existing code, which made implementation more time-efficient. Figure 10 gives the overall accuracy of MsGT when performing multiclass classification compared to both benchmarks, while figure 11 gives the confusion matrix, providing more detail as to which classes MsGT performs well on (confusion matrices for PointNet and FPFH can be found in the appendix).

One can see from the first figure that while MsGT does not reach the level of PointNet in terms of accuracy, it far exceeds FPFH. We consider PointNet to be a bar that we strive for despite the fact that it may be unattainable, and FPFH the bar we need to beat for meaningful results. In this view, MsGT has thoroughly succeeded in providing a new, accurate classifier from the perspective of hand-crafted feature extractors.

From the second figure, it becomes clear that the success of MsGT varies greatly from class to class. The authors suspect this may be because of both the shape of the object and the contents of the data. For

instance, many pointcloud files in the training set had less than 1024 points and had to be upsampled to be the right size. Some classes, such as class 2 (chairs), had a higher proportion of large (greater than 1024 points) point clouds than other classes, and this was a class that MsGT performed well on.

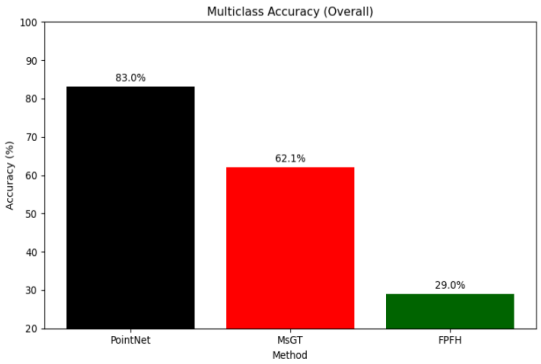

Figure 10: Overall accuracy (Multiclass)

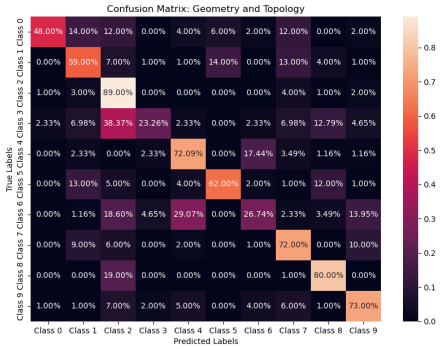

Figure 11: MsGT (Multiclass)

While MsGT did not perform well on some classes in multiclass classification, it performed very well on others. As an interesting remark, the two classes which saw the highest rate of success were classes that PointNet struggled with (classes 2 and 8, chairs and tables). This may suggest that hand-crafted methods that rely on mathematics, such as MsGT, can fill in the gaps left by deep learning methods, a topic we will discuss again in the next section. In any case, the success we saw with MsGT in classes 2, 7, 8, and 9 led us to measure the accuracy of our method in binary classification with these classes.

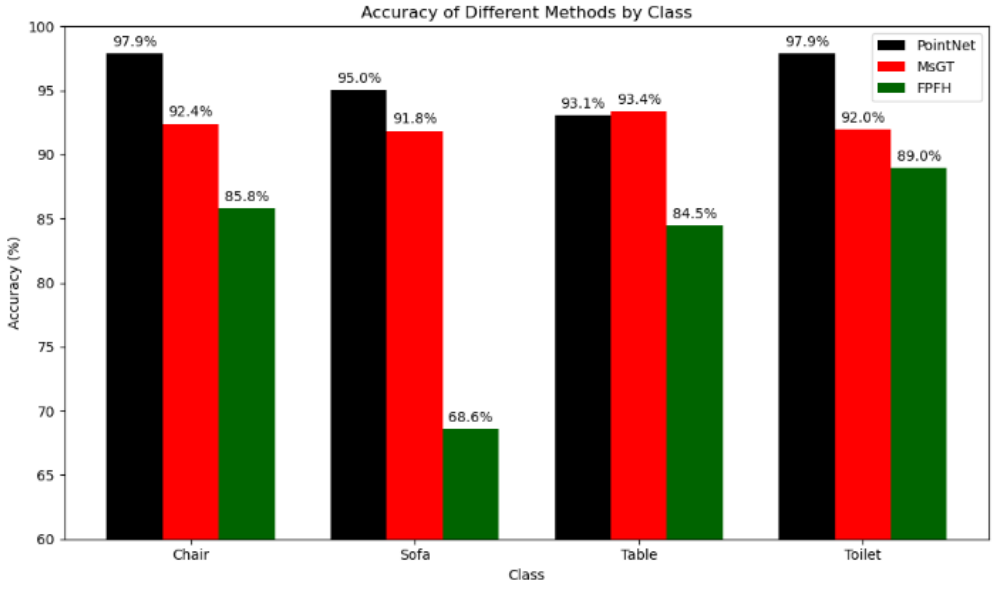

Figure 12: Accuracy (Binary Classification)

As expected and as apparent from figure 12, MsGT does quite well in binary classification tasks with these four classes, exceeding 91% accuracy in all of them and far outperforming FPFH. Of particular note is class 8 (tables) where MsGT provides higher accuracy ratings than PointNet. We believe that this is evidence that our method is well-suited to binary classification tasks in a real-world setting. Table 1 summarizes the results of this subsection.

Classifier	Multi-class	Binary			
		Chair	Sofa	Table	Toilet
MsGT	62.115%	92.401%	91.850%	93.392%	91.960%
PointNet	83.040%	97.907%	95.044%	93.062%	97.907%
FPFH	29.000%	85.793%	68.612%	84.471%	88.987%

Table 1: Classification Results for Different Classes and Feature Extraction Models

4.4 Topology and Multi-scale Testing

As was outlined in section 3, our method has two key components: multi-scale geometry and global topology. To determine the relevance of these components, we test multi-scale and binary classification on two stripped-down versions of our method: one with only geometric features extracted at a fixed size of neighborhood (no topology or multiscale), and then one with topological global features added (no multiscale). We then compare the results with MsGT.

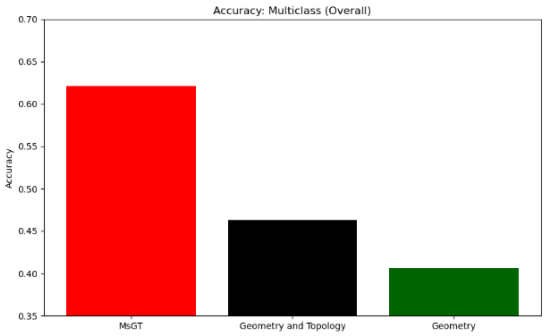

Figure 13: Overall accuracy (Multiclass)

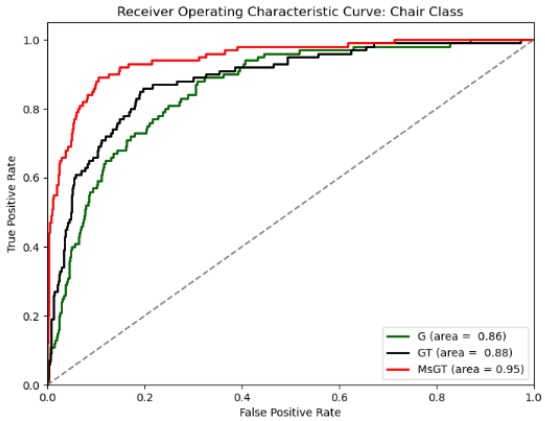

Figure 14: ROC Curve (Binary Class., chairs)

As visible in figure 13, the accuracy improves notably along each step: from Geometry to Geometry and Topology, and from Geometry and Topology to MsGT. Figure 14 gives the ROC curve for a select class in binary classification: more figures can be found in the appendix if desired, but each figure shows the same trend.

The key takeaway from these figures is that each of the components of MsGT work together to create something better than any of the individual parts. Geometry alone is not enough for an accurate model. Including topology improves the model, and once we use a multi-scale approach, it improves the model even further.

4.5 Geometric Feature Testing

Recall that our method uses two new geometric features K_p and H_p alongside some existing geometric features as a supplement. Our final set of testing shows that these new geometric features improve upon existing geometric features when it comes to classification tasks. In other words, our new geometric features are not just additional choices for features, they are the *better choices*.

Testing was done using only geometry with a multi-scale approach and the same neighborhood sizes as before. We tested H_p and K_p as a pair in multiclass classification and binary classification. For comparison,

testing was then done with all possible combinations of the following features (recall that λ_0, λ_1, and λ_2 are eigenvalues of the covariance matrix at a point):

Feature Number	Feature Name	Formula
0	Surface Variation	$\lambda_0(\lambda_0 + \lambda_1 + \lambda_2)^{-1}$
2	Linearity	$(\lambda_2 - \lambda_1)\lambda_2^{-1}$
3	Planarity	$(\lambda_1 - \lambda_0)\lambda_2^{-1}$
5	Omnivariance	$(\lambda_0 \cdot \lambda_1 \cdot \lambda_2)^{\frac{1}{3}}$
6	Anisotropy	$(\lambda_2 - \lambda_0)\lambda_2^{-1}$

Table 2: Features used in Geometry testing. Feature numbers appear in figure 15.

We recognize that this is not a comprehensive list, but due to the time restrictions of this project, we felt this was the most representative of other available geometric features and that these features had reasonable success in their own right.

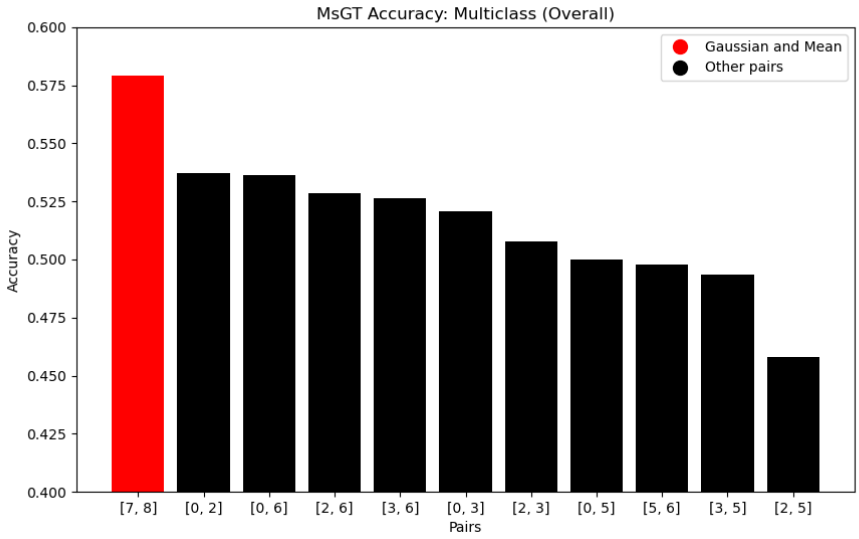

Figure 15: Accuracy (Multiclass)

We can see from figure 15 that our new features, when taken as a pair, far outperform any pair from table 2 in multiclass classification. We also performed tests with binary classification on the same classes chosen earlier, and similar trend occured. Our pair of features outperformed every pair of features from table 2 in binary classification of chairs, tables, and toilets, and was close to the best in classification of sofas. Bar graphs, ROC curves, and precision-recall curves for these tests can be found in the appendix.

5 Conclusion

In MsGT, we have constructed a novel hand-crafted feature extraction method that synthesizes topology and geometry on a hierarchical, or multi-scale, level. This method introduces novel and effective geometry to the field of point cloud processing. Moreover, the results of section 4 show that MsGT is a dependable, robust, and accurate hand-crafted feature extraction method. In particular, MsGT was able to outperform some existing hand-crafted methods and occasionally reach levels of accuracy achieved by deep learning models such as PointNet.

The MsGT method particularly excels in binary classification of certain objects such as chairs and tables (figure 12). Even after only preliminary testing, we can envision MsGT being used in real-world applications for these sorts of tasks. This is particularly useful for industry, as the goal that MsGT was meant to achieve was providing an accurate hand-crafted feature extraction method, thereby removing the cost constraints associated with deep learning methods. We believe that MsGT will meet this goal.

Finally, we recognize that MsGT has limitations. As seen in figure 11, MsGT does not perform to a satisfactory level on certain classes. Furthermore, we are not computer scientists, hence the coding of our algorithm is only partially optimized.

However, it is also true that MsGT is in it's preliminary stages, i.e. this version of MsGT is an alpha build. No tuning was done to this model and, as far as the goals we have for it, it is wholly incomplete. This implies that MsGT has much room to grow and improve even further, a topic we look to expand upon in the last section. Given the results so far and considerable room for tuning, we believe that MsGT has the potential to make a meaningful impact in the point cloud processing market.

6 Future Work

6.1 Weighting and Parameter Tuning

One remarkable aspect of MsGT is that it performs well in the absence of any parameter tuning or weighing of features. In other words, **MsGT was** (more or less) **not tuned for performance on ModelNet10**. In the first iterations of the method, we normalized the feature vector to take values in $[-1, 1]$. However, forgoing the normalization considerably improved results. Since the values that each feature can take vary greatly (e.g. persistence landscape entries can take values significantly greater than 100, while mean curvature tends to stay between -1 and 1), this implies that weighing the components of the feature vector should impact the accuracy of the model in a meaningful way. Moreover, the number of scalar entries dedicated to a particular feature also varies greatly. For instance, the statistical measurements of persistence values only account for four of the entries in the feature vector, or about 0.0001% of the feature vector, while persistence landscapes account for 1400, or about 4%.

$$\text{weighted feature vector (intuition)} = \left[\ \text{p. values} \ \middle|\ {}_{\text{landscapes}} \ \middle|\ \text{Gauss} \ \middle|\ \text{Mean} \ \middle|\ {}_{\text{Omnivariance}} \ \middle|\ \cdots \ \right]^T$$

Furthermore, weighing the components based on some sort of variable correlation was something that was always in our plans, but our project was time-limited. We believe this should be one of the first approaches for future work: it will likely be both easy to implement and make a considerable difference.

Finally, parameters for the *Ripser* package for computing persistence landscapes and the XGBoost classifier were not altered very much, if at all. The persistence landscape code underwent minor testing of a handful of different step and landscape combinations before we chose the best one, and this testing was done in the absence of multi-scale geometric features. Even further, XGBoost's parameters were not altered *at all*, and so we believe there is room to improve accuracy further here. Given the time limitations of our project, this was secondary to our other goals, as this should improve the accuracy of all methods and our

purpose was to compare our method with others. However, for a final product, a finely tuned classifier would present the clearest picture of MsGT.

6.2 Speed

As was noted earlier, the authors of this report are not computer scientists, rather we are mathematicians and statisticians. As such, most of the code written for the project could only be optimized to the limits of our knowledge and what we could learn in eight weeks. For instance, while we included some parallel computing and vectorization in our code, it is rather naive and not implemented everywhere.

In the future, we would like to focus on writing the package for our algorithm in the most efficient way. We are aware of packages out there that improve computation speed considerably, like cuda and numba, and would like to implement them. Moreover, it is possible that writing our algorithm for a different language, such as Julia, could improve our computation time considerably given the limitations of python. Ultimately, while we view our role in this project as the architects of a mathematical model, this model must also be tuned by the hands of computer scientists to make it relevant for industry.

6.3 Topological improvements

For future works, we can incorporate a few topological feature vectors. In our approach, we used persistent values and persistent landscapes to vectorize persistent diagrams. These are rudimentary features and limited in scope. For future work, we can incorporate other vectorizations of persistent diagrams, such as persistent entropy and persistent images.

Other than persistent diagrams, there are several other topological vectors that we can incorporate:

1. **Betti numbers** Betti numbers are fundamental invariants in algebraic topology that quantify the number of independent cycles of various dimensions in a topological space, essentially capturing the ”holes” in different dimensions.

2. **Euler characteristics** For a convex polyhedron, the Euler characteristic is given by:

$$\chi = V - E + F \tag{6.1}$$

   where V is the number of vertices, E is the number of edges, and F is the number of faces. The Euler characteristic is a topological invariant that provides a measure of a topological space's shape or structure. It is especially useful in the study of polyhedra, surfaces, and more general topological spaces.

3. **Reeb graphs** A Reeb graph is a tool used in topology and computational geometry to study the shape and structure of a space by analyzing the level sets of a function defined on it. It simplifies the complexity of the space into a graph that captures the essential features of the function's behavior. In essence, the Reeb graph offers a simplified view of a space's topology by focusing on how connected components change with respect to continuous function.

We anticipate that having a more robust topological tool set can improve the model. For instance, we think that the small improvement shown in 13 after including topology may be a result of underdeveloped topological architecture in MsGT, and not a sign of weakness of topological methods.

6.4 Improving PointNet

In our future research, we aim to enhance the performance of PointNet by integrating the handcrafted features extracted by MsGT. By leveraging the strengths of both methods, we believe that PointNet can achieve higher classification accuracy and robustness. Below are the key areas we will focus on to achieve this improvement:

1. **Feature Fusion**

   We plan to explore how to integrate handcrafted features extracted by MsGT with the automatic features learned by PointNet to enhance the model's classification performance. The MsGT method utilizes geometric techniques to extract local features and topological techniques to extract global features. The combination of local and global features can capture rich information in point cloud data, while PointNet excels at automatically learning high-level features from the data. By fusing these two types of features, we aim to leverage both the geometric and topological information from handcrafted features and the abstract representations from automatic features to improve the model's classification accuracy on complex point cloud data. Specifically, we will attempt to incorporate the local and global features extracted by MsGT into the feature extraction layers of PointNet and validate the effectiveness of this fusion strategy through experiments. Previous research has shown that integrating feature extraction methods into PointNet can significantly enhance classification performance. For example, in the study "PointNet meets Self-Attention Graph Pooling: A Synergistic Approach to Point Cloud Classification," the multi-class classification accuracy of PointNet was improved from 85% to 92% by incorporating their feature extraction method. We believe that MsGT can similarly enhance the performance of PointNet.

2. **Multi-Scale Feature Representation**

   One of the significant advantages of MsGT is its ability to extract multi-scale geometric features, which is crucial for handling point cloud data of varying scales. In our future work, we will explore how to utilize MsGT's multi-scale feature representation to enhance PointNet's performance. In point cloud data, different objects may have varying scales, and by incorporating MsGT's multi-scale features, we aim to enable PointNet to better handle these variations, improving the model's robustness and generalization capabilities. For instance, we can introduce MsGT's multi-scale feature representation into PointNet's input layer, allowing the model to learn simultaneously at different scales and thus enhancing its adaptability to diverse data.

3. **Improvement in Category Classification Performance**

   Analysis of the confusion matrices indicates that MsGT's classification accuracy for chair class (Class 2) is significantly higher than that of PointNet (89.00% compared to 62.00%). By integrating the features extracted by MsGT into PointNet, we can substantially improve PointNet's classification performance for the chair class. Furthermore, PointNet outperforms MsGT in some categories, and by analyzing the effective features in these categories, we can introduce them into MsGT to further optimize its feature extraction strategy. In our future work, we will focus on how to reduce misclassification between different categories in PointNet by combining MsGT's features to enhance overall classification accuracy.

4. **Binary Classification Performance Enhancement**

   In addition to multi-class classification, we have also conducted binary classification studies. Specifically, for the binary classification task of the table class (Class 7), MsGT's classification accuracy significantly surpasses that of PointNet. This indicates that MsGT has higher classification performance in handling binary classification tasks for specific categories. In our future work, we will further investigate how to apply the advantages of MsGT in binary classification tasks to PointNet, improving PointNet's performance in binary classification through feature fusion and optimization strategies.

References

[1] H. Adams, A. Atanasov, and G. Carlsson. Morse theory in topological data analysis. 12 2011.

[2] H. Adams, T. Emerson, M. Kirby, R. Neville, C. Peterson, P. Shipman, S. Chepushtanova, E. Hanson, F. Motta, and L. Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18(8):1–35, 2017.

[3] M. Afham, I. Dissanayake, D. Dissanayake, A. Dharmasiri, K. Thilakarathna, and R. Rodrigo. Crosspoint: Self-supervised cross-modal contrastive learning for 3d point cloud understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9902–9912, 2022.

[4] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, and G. Bradski. Cad-model recognition and 6dof pose estimation using 3d cues. In *2011 IEEE international conference on computer vision workshops (ICCV workshops)*, pages 585–592. IEEE, 2011.

[5] A. Asencio, S. Cardman, D. Harris, and E. Laderman. Relating expectations to automatically recovered design patterns. In *Ninth Working Conference on Reverse Engineering, 2002. Proceedings.*, pages 87–96, 2002.

[6] J. Behley, V. Steinhage, and A. B. Cremers. Performance of histogram descriptors for the classification of 3d laser range data in urban environments. In *2012 IEEE international conference on robotics and automation*, pages 4391–4398. IEEE, 2012.

[7] W. J. Beksi and N. Papanikolopoulos. Signature of topologically persistent points for 3d point cloud description. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3229–3234. IEEE, 2018.

[8] L. Bo, X. Ren, and D. Fox. Depth kernel descriptors for object recognition. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 821–826. IEEE, 2011.

[9] P. Bubenik et al. Statistical topological data analysis using persistence landscapes. *J. Mach. Learn. Res.*, 16(1):77–102, 2015.

[10] M. Carrière, F. Chazal, Y. Ike, T. Lacombe, M. Royer, and Y. Umeda. Perslay: A neural network layer for persistence diagrams and new graph topological signatures. In *International Conference on Artificial Intelligence and Statistics*, pages 2786–2796. PMLR, 2020.

[11] T. Chen, B. Dai, D. Liu, and J. Song. Performance of global descriptors for velodyne-based urban object recognition. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 667–673. IEEE, 2014.

[12] P. Cirujeda, X. Mateo, Y. Dicente, and X. Binefa. Mcov: a covariance descriptor for fusion of texture and shape features in 3d point clouds. In *2014 2nd International Conference on 3D Vision*, volume 1, pages 551–558. IEEE, 2014.

[13] K. Crane, F. de Goes, M. Desbrun, and P. Schröder. Digital geometry processing with discrete exterior calculus. 2013.

[14] O. Delgado-Friedrichs, V. Robins, and A. Sheppard. Morse theory and persistent homology for topological analysis of 3d images of complex materials. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 4872–4876. IEEE, 2014.

[15] J. P. S. do Monte Lima and V. Teichrieb. An efficient global point cloud descriptor for object recognition and pose estimation. In *2016 29th SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)*, pages 56–63. IEEE, 2016.

[16] O. Dovrat, I. Lang, and S. Avidan. Learning to sample. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2760–2769, 2019.

[17] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 998–1005. Ieee, 2010.

[18] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE transactions on image processing*, 6(9):1305–1315, 1997.

[19] D. Fehr, A. Cherian, R. Sivalingam, S. Nickolay, V. Morellas, and N. Papanikolopoulos. Compact covariance descriptors in 3d point clouds for object recognition. In *2012 IEEE international conference on robotics and automation*, pages 1793–1798. IEEE, 2012.

[20] A. Flint, A. Dick, and A. Van Den Hengel. Thrift: Local 3d structure recognition. In *9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications (DICTA 2007)*, pages 182–188. IEEE, 2007.

[21] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *Computer Vision-ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part III 8*, pages 224–237. Springer, 2004.

[22] X. Ge. Non-rigid registration of 3d point clouds under isometric deformation. *ISPRS journal of photogrammetry and remote sensing*, 121:192–202, 2016.

[23] S. Gumhold, X. Wang, R. S. MacLeod, et al. Feature extraction from point clouds. In *IMR*, pages 293–305, 2001.

[24] I. Hadji and G. N. DeSouza. Local-to-global signature descriptor for 3d object recognition. In *Computer Vision-ACCV 2014 Workshops: Singapore, Singapore, November 1-2, 2014, Revised Selected Papers, Part I 12*, pages 570–584. Springer, 2015.

[25] A. Hamdi, S. Giancola, and B. Ghanem. Mvtn: Multi-view transformation network for 3d shape recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1–11, 2021.

[26] X.-F. Han, S.-J. Sun, X.-Y. Song, and G.-Q. Xiao. 3d point cloud descriptors in hand-crafted and deep learning age: State-of-the-art. *arXiv preprint arXiv:1802.02297*, 2018.

[27] M. Himmelsbach, T. Luettel, and H.-J. Wuensche. Real-time object classification in 3d point clouds using point feature histograms. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 994–1000. IEEE, 2009.

[28] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *SIGGRAPH Comput. Graph.*, 26(2):71–78, jul 1992.

[29] J. Huang and S. You. Point cloud matching based on 3d self-similarity. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 41–48. IEEE, 2012.

[30] J. Huang and S. You. Detecting objects in scene point cloud: A combinational approach. In *2013 International Conference on 3D Vision-3DV 2013*, pages 175–182. IEEE, 2013.

[31] J. B. Jeans and D. Hong. Impass: Intelligent mobility platform with active spoke system. In *2009 IEEE International Conference on Robotics and Automation*, pages 1605–1606, 2009.

[32] A. E. Johnson and M. Hebert. Surface matching for object recognition in complex three-dimensional scenes. *Image and Vision Computing*, 16(9-10):635–651, 1998.

[33] S. H. Kasaei, A. M. Tomé, L. S. Lopes, and M. Oliveira. Good: A global orthographic object descriptor for 3d object recognition and manipulation. *Pattern Recognition Letters*, 83:312–320, 2016.

[34] S. Kobayashi and K. Nomizu. *Foundations of Differential Geometry, Volume 2*. Foundations of Differential Geometry [by] Shoshichi Kobayashi and Katsumi Nomizu. Wiley, 1963.

[35] A. Komarichev, Z. Zhong, and J. Hua. A-cnn: Annularly convolutional neural networks on point clouds. pages 7421–7430, 2019.

[36] J.-F. Lalonde, N. Vandapel, D. F. Huber, and M. Hebert. Natural terrain classification using three-dimensional ladar data for ground robot mobility. *Journal of field robotics*, 23(10):839–861, 2006.

[37] N. Lazar and H. Ryu. The shape of things: Topological data analysis. *Chance*, 34(2):59–64, 2021.

[38] B. Lin, F. Wang, F. Zhao, and Y. Sun. Scale invariant point feature (sipf) for 3d point clouds and 3d multi-scale object detection. *Neural Computing and Applications*, 29:1209–1224, 2018.

[39] K. B. Logoglu, S. Kalkan, and A. Temizel. Cospair: colored histograms of spatial concentric surflet-pairs for 3d object recognition. *Robotics and Autonomous Systems*, 75:558–570, 2016.

[40] M. Madry, C. H. Ek, R. Detry, K. Hang, and D. Kragic. Improving generalization for 3d object categorization with global structure histograms. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1379–1386. IEEE, 2012.

[41] Z.-C. Marton, D. Pangercic, N. Blodow, and M. Beetz. Combined 2d–3d categorization and classification for multimodal perception systems. *The International Journal of Robotics Research*, 30(11):1378–1402, 2011.

[42] Z.-C. Marton, D. Pangercic, N. Blodow, J. Kleinehellefort, and M. Beetz. General 3d modelling of novel objects from a single view. In *2010 ieee/rsj international conference on intelligent robots and systems*, pages 3700–3705. IEEE, 2010.

[43] B. Matei, Y. Shan, H. S. Sawhney, Y. Tan, R. Kumar, D. Huber, and M. Hebert. Rapid object indexing using locality sensitive hashing and joint 3d-signature space estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1111–1126, 2006.

[44] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 922–928. IEEE, 2015.

[45] S. S. Mohammadi, Y. Wang, and A. Del Bue. Pointview-gcn: 3d shape classification with multi-view point clouds. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3103–3107. IEEE, 2021.

[46] M. Muja, R. B. Rusu, G. Bradski, and D. G. Lowe. Rein-a fast, robust, scalable recognition infrastructure. In *2011 IEEE international conference on robotics and automation*, pages 2939–2946. IEEE, 2011.

[47] E. Munch. A user's guide to topological data analysis. *Journal of Learning Analytics*, 4(2):47–61, 2017.

[48] A. Patterson, P. Mordohai, and K. Daniilidis. Object detection from large-scale 3d datasets using bottom-up and top-down descriptors. In *Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part IV 10*, pages 553–566. Springer, 2008.

[49] M. Pauly, R. Keiser, and M. Gross. Multi-scale feature extraction on point-sampled surfaces. In *Computer graphics forum*, volume 22, pages 281–289. Wiley Online Library, 2003.

[50] S. M. Prakhya, J. Lin, V. Chandrasekhar, W. Lin, and B. Liu. 3dhopd: A fast low-dimensional 3-d descriptor. *IEEE Robotics and Automation Letters*, 2(3):1472–1479, 2017.

[51] 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*, pages 652–660, 2017.

[52] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[53] G. Qian, Y. Li, H. Peng, J. Mai, H. Hammoud, M. Elhoseiny, and B. Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *Advances in Neural Information Processing Systems*, 35:23192–23204, 2022.

[54] S. Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. pages 486–493, 2004.

[55] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001.

[56] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE, 2009.

[57] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *2008 IEEE/RSJ international conference on intelligent robots and systems*, pages 3384–3391. IEEE, 2008.

[58] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *2010 IEEE/RSJ international conference on intelligent robots and systems*, pages 2155–2162. IEEE, 2010.

[59] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE international conference on robotics and automation*, pages 1–4. IEEE, 2011.

[60] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4, 2011.

[61] R. B. Rusu, A. Holzbach, M. Beetz, and G. Bradski. Detecting and segmenting objects for mobile manipulation. In *2009 IEEE 12th international conference on computer vision workshops, ICCV workshops*, pages 47–54. IEEE, 2009.

[62] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz. Persistent point feature histograms for 3d point clouds. In *Proc 10th Int Conf Intel Autonomous Syst (IAS-10), Baden-Baden, Germany*, pages 119–128, 2008.

[63] Y. Shan, H. S. Sawhney, B. Matei, and R. Kumar. Shapeme histogram projection and matching for partial object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):568–577, 2006.

[64] D. Smirnov and J. Solomon. Hodgenet: learning spectral geometry on triangle meshes. 40(4), jul 2021.

[65] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard. Narf: 3d range image features for object recognition. In *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, volume 44, page 2. Citeseer, 2010.

[66] P. Szutor and M. Zichar. Fast radius outlier filter variant for large point clouds. *Data*, 8(10):149, 2023.

[67] K. Tang, P. Song, and X. Chen. Signature of geometric centroids for 3d local shape description and partial shape matching. In *Computer Vision–ACCV 2016: 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part V 13*, pages 311–326. Springer, 2017.

[68] F. Tombari, S. Salti, and L. Di Stefano. Unique signatures of histograms for local surface description. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part III 11*, pages 356–369. Springer, 2010.

[69] R. Triebel, K. Kersting, and W. Burgard. Robust 3d scan point classification using associative markov networks. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2603–2608. IEEE, 2006.

[70] E. Wahl, U. Hillenbrand, and G. Hirzinger. Surflet-pair-relation histograms: a statistical 3d-shape representation for rapid classification. In *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, pages 474–481. IEEE, 2003.

[71] C. Wang, X. Ning, L. Sun, L. Zhang, W. Li, and X. Bai. Learning discriminative features by covering local geometric space for point cloud analysis. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–15, 2022.

[72] L. Wang, Y. Huang, Y. Hou, S. Zhang, and J. Shan. Graph attention convolution for point cloud semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10296–10305, 2019.

[73] Z. Wang, L. Zhang, T. Fang, P. T. Mathiopoulos, X. Tong, H. Qu, Z. Xiao, F. Li, and D. Chen. A multiscale and hierarchical feature extraction method for terrestrial laser scanning point cloud classification. *IEEE Transactions on Geoscience and Remote Sensing*, 53(5):2409–2425, 2014.

[74] M. Weinmann. Reconstruction and analysis of 3d scenes. In *Cambridge International Law Journal*, 2016.

[75] M. Weinmann, B. Jutzi, S. Hinz, and C. Mallet. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS Journal of Photogrammetry and Remote Sensing*, 105:286–304, 2015.

[76] W. Wohlkinger and M. Vincze. Ensemble of shape functions for 3d object classification. In *2011 IEEE international conference on robotics and biomimetics*, pages 2987–2992. IEEE, 2011.

[77] K. Wu, X. Li, R. Ranasinghe, G. Dissanayake, and Y. Liu. Risas: A novel rotation, illumination, scale invariant appearance and shape feature. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4008–4015. IEEE, 2017.

[78] T. Xiang, C. Zhang, Y. Song, J. Yu, and W. Cai. Walk in the cloud: Learning curves for point clouds shape analysis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 915–924, 2021.

[79] J. Yang, Z. Cao, and Q. Zhang. A fast and robust local descriptor for 3d point cloud registration. *Information Sciences*, 346:163–179, 2016.

[80] X. Ye, J. Li, H. Huang, L. Du, and X. Zhang. 3d recurrent neural networks with context fusion for point cloud semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 403–417, 2018.

[81] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19313–19322, 2022.

[82] H. Zhang, C. Wang, S. Tian, B. Lu, L. Zhang, X. Ning, and X. Bai. Deep learning-based 3d point cloud classification: A systematic survey and outlook. *Displays*, page 102456, 2023.

[83] G. Zhao, J. Yuan, and K. Dang. Height gradient histogram (high) for 3d scene labeling. In *2014 2nd International Conference on 3D Vision*, volume 1, pages 569–576. IEEE, 2014.

[84] Q.-Y. Zhou, J. Park, and V. Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018.

7 Appendix

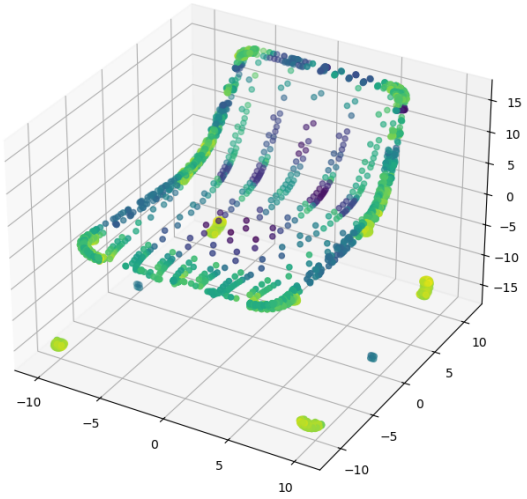

Figure 16: Gaussian

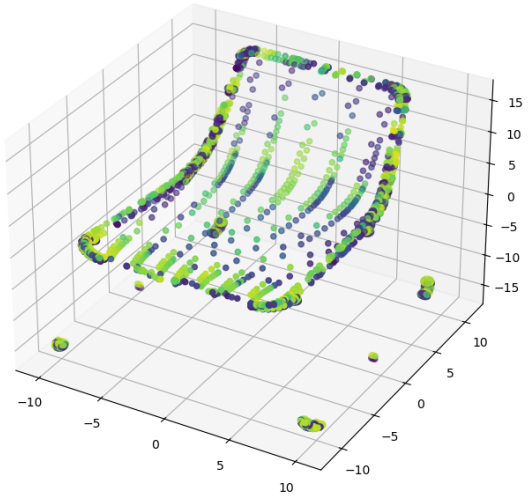

Figure 17: Mean Curvature

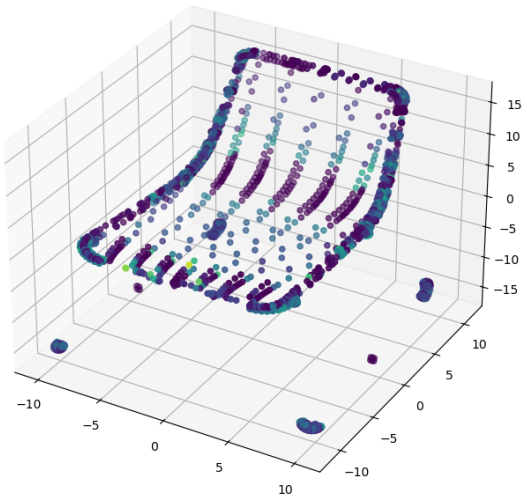

Figure 18: Surface variation

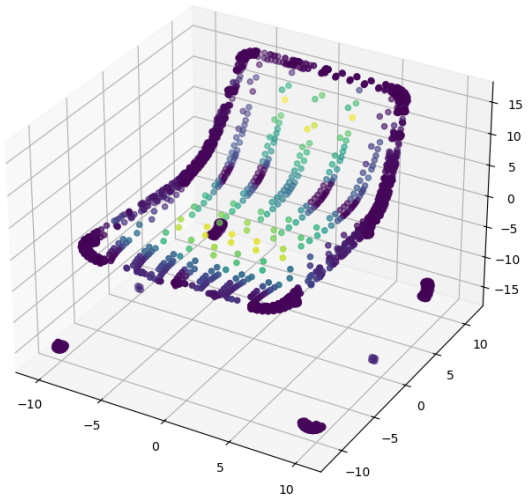

Figure 19: Omnivariance

Classifier	Accuracy	Recall	Prec.	F1
Logistic Regression	0.6441	0.6441	0.6391	0.6300
Random Forest	0.5743	0.5743	0.5516	0.5407
Ridge	0.4894	0.4894	0.4669	0.4664
Naive Bayes	0.4715	0.4715	0.5206	0.4582
Support Vector Machine	0.4110	0.4110	0.4934	0.3432

Table 3: Performance comparison of classification models on MsGT features.

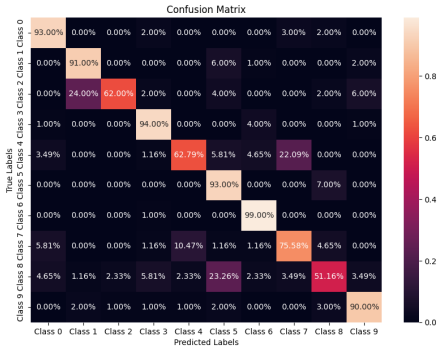

Figure 20: PointNet (Multiclass)

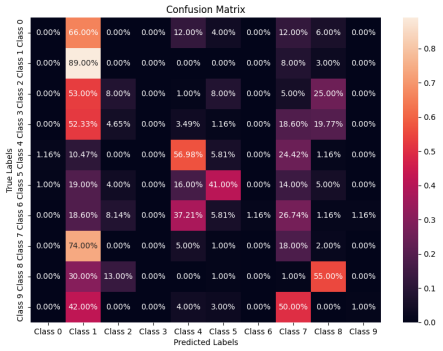

Figure 21: FPFH (Multiclass)

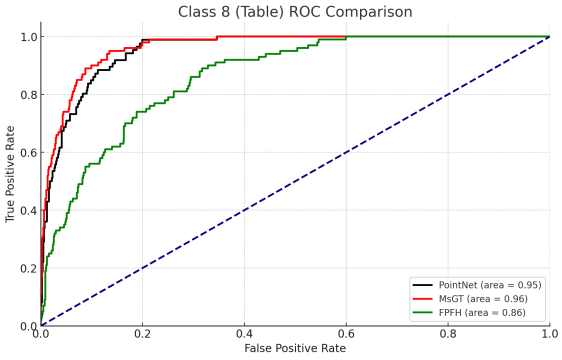

Figure 22: Binary Classification (Tables)

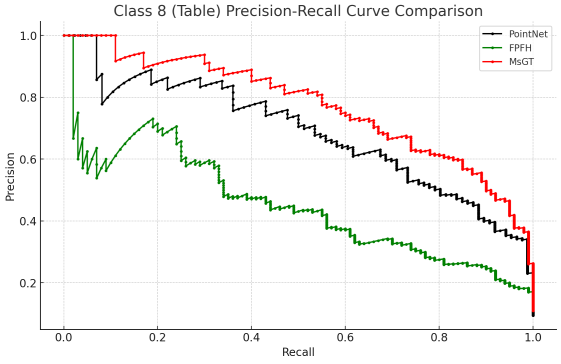

Figure 23: Binary Classification (Tables)

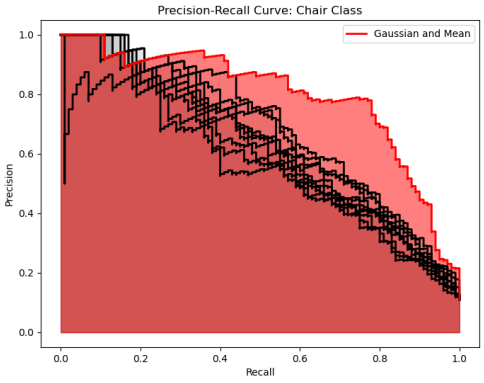

Figure 24: Geometry Testing (Binary)

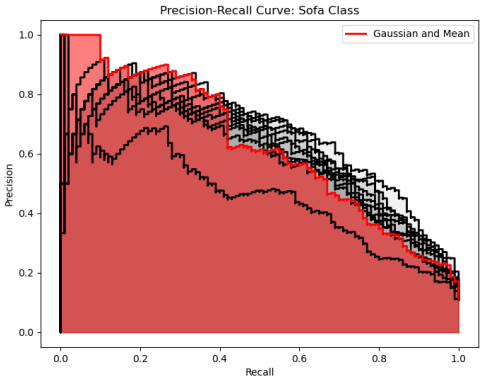

Figure 25: Geometry Testing (Binary)

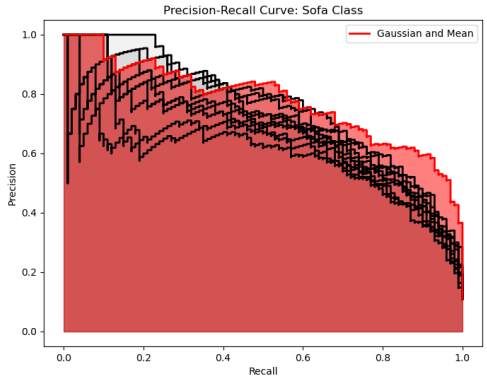

Figure 26: Geometry Testing (Binary)

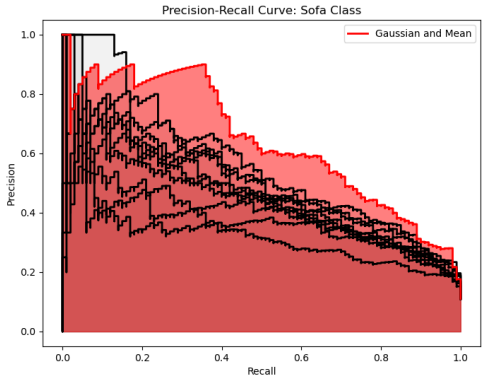

Figure 27: Geometry Testing (Binary)

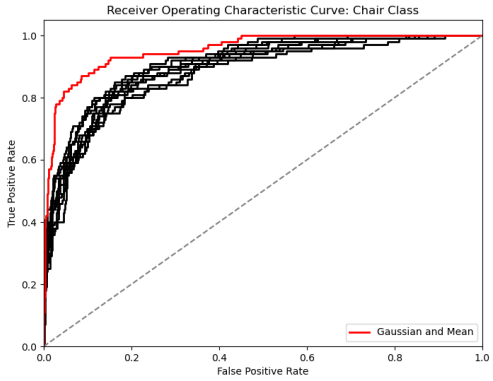

Figure 28: Geometry Testing (Binary)

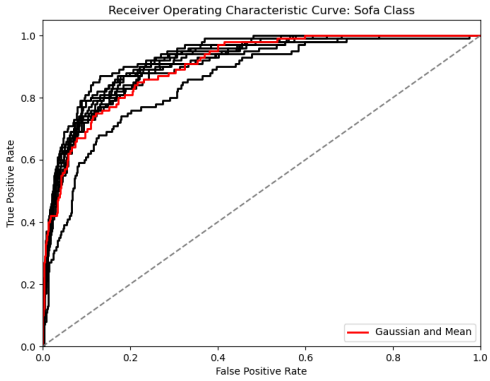

Figure 29: Geometry Testing (Binary)

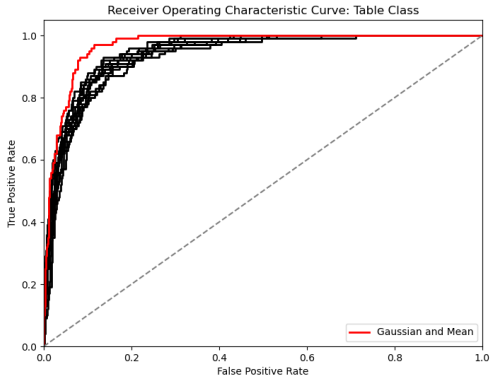

Figure 30: Geometry Testing (Binary)

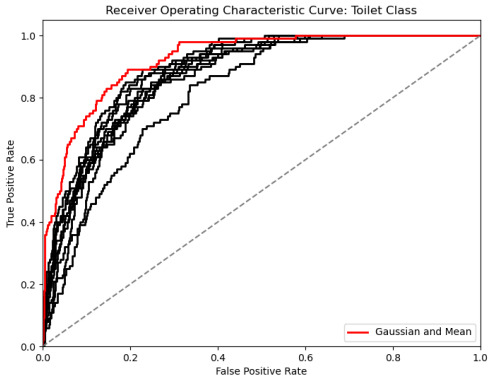

Figure 31: Geometry Testing (Binary)

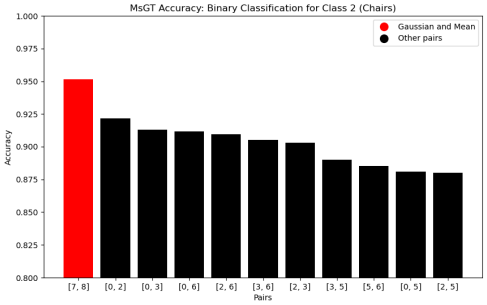

Figure 32: Geometry Testing (Binary)

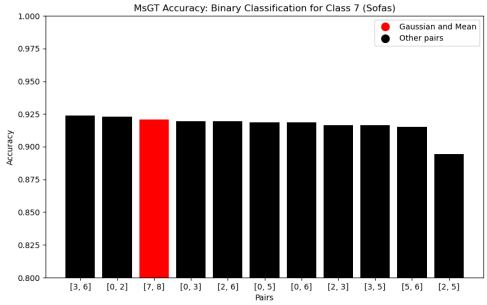

Figure 33: Geometry Testing (Binary)

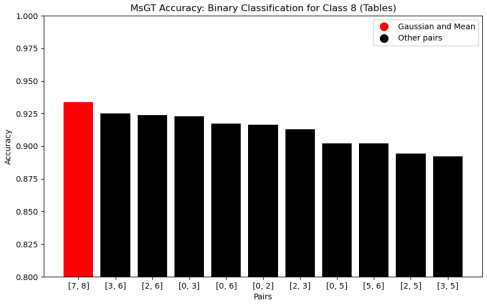

Figure 34: Geometry Testing (Binary)

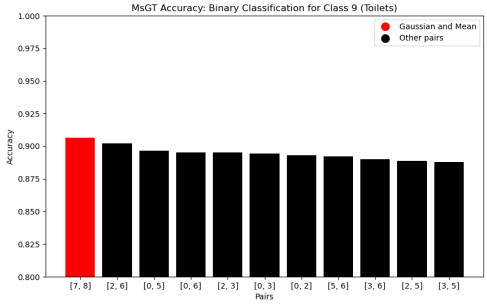

Figure 35: Geometry Testing (Binary)