Equine back-surface modeling and shape classification using ML-techniques

Abilkaiyr Abikesh,

Bachelor of Science in Mechanical and Aerospace Engineering

Submitted in fulfillment of the requirements
for the degree of Master of Science
in Mechanical & Aerospace Engineering

NAZARBAYEV UNIVERSITY

School of Engineering and Digital Sciences
Mechanical & Aerospace Engineering Department
Nazarbayev University

53 Kabanbay Batyr Avenue,
Astana, Kazakhstan, 010000

Supervisor: Associate Professor Konstantinos Kostas

Co-Supervisor: Associate Professor Desmond Adair

2025, April

DECLARATION

I hereby, declare that this manuscript, entitled "*Equine back-surface modeling and shape classification using ML-techniques*", is the result of my own work except for quotations and citations, which have been duly acknowledged.

I also declare that, to the best of my knowledge and belief, it has not been previously or concurrently submitted, in whole or in part, for any other degree or diploma at Nazarbayev University or any other national or intentional institution.

_ _ _ _ _Abilkaiyr_ _ _ _ _ _

Name: Abilkaiyr Abikesh

Date: 11.04.2025

ABSTRACT

The current state of saddle manufacturing faces several challenges. The main concern is the lack of a universally accepted horse-back measurement system that makes the whole manufacturing process time consuming and labor intensive. The aim of this research is to combine parametric modeling with Machine Learning techniques for shape classification to develop a software tool which will make saddle design and its manufacturing process more efficient. To develop an appropriate neural network, the raw dataset containing horse-back measurements is required to firstly develop parametric models with appropriate design variables and ultimately corresponding saddle surfaces. Using geometric properties of the surfaces, a number of features will be extracted in order to develop and train the corresponding classification algorithm. Several categories will be identified based on surface shape characteristics and representative surfaces for each category will be produced. The results obtained during the course of this research will be used to create manufacturable saddle models that fit performance standards and maintain the well-being of horses.

Key words: *Parametric Modeling, Feature extraction, Saddle fitting, Shape classification, Machine Learning.*

Acknowledgements

I would like to express my deep appreciation and gratitude to all those who have contributed to the successful completion of this thesis. First and foremost, I would like to thank my thesis supervisor Professor Konstantinos Kostas and co-supervisor Professor Desmond Adair whose guidance and expertise have been instrumental in completing this thesis.

I want to acknowledge the help I received from Mr. Aktan Akhmetov. As an expert in saddle manufacturing, he gave me access to measuring templates as well as databases of previous measurements. He also provided help with measuring horses and guidance with regards to the requirements of the equine community and of course I must mention the financial support that he offered during the course of this thesis.

I also want to extend my sincere thanks to Nazarbayev University. I am grateful for all the opportunities this great institution provided to finish this thesis, as well as the necessary resources and facilities it gave access to carry out the research. I am grateful for the world-class education and supportive community the university shaped. The guidance and mentorship provided by our professors, instructors, and support staff have been instrumental in shaping our academic journey.

Table of content

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS & SYMBOLS

ML – Machine Learning

NURBS – Non-Uniform Rational B-Splines

GMM – Gaussian Mixture Models

PCA – Principal Component Analysis

MDS – Multidimensional Scaling

Isomap – Isometric Feature Mapping

LLE – Locally Linear Embedding

CHAPTER 1 - INTRODUCTION

1.1 Research Background

It is a well-known fact that different horses have different back morphology. Horse's overall movement patterns, its biomechanics and athletic performance are related to the shape of its back [1]. Oftentimes the improper understanding of these issues leads to severe back pains of the horse, can reduce its performance and cause health issues; sometimes horses will be unsuitable for riding.

The main reason for these health issues is unfit saddles. Failure to account for the correct horse-back surface shape leads to ill-fitting saddles that can cause pressure points, muscle atrophy, and behavioral problems [2]; see also Fig. 1. Saddle fitting problems arise due to shortcomings of traditional saddle fitting methods, which heavily rely on subjective assessment.

The analysis of horse-back shapes can be used in veterinary applications. It can aid in detection of spinal issues, monitoring treatment progress, and understanding age related changes in equine backs.

Sport horses are expected to perform competitively for 10+ years in dressage, show jumping and cross country. People engaged in those sports generally recognize that an ill-fitting saddle may ruin a horse in a fairly short period of time. Therefore, saddle fitting is usually given due attention as an essential factor of horse health, longevity and performance. Time and effort are invested in the horse and the rider.

In contrast to those "olympic" equestrian disciplines, the issues of ill-fitted saddles, horse health and longevity are overlooked or neglected in some parts of the world and in some parts of equestrian community regardless of geographical location. A horse is viewed as a low-cost unit of transport, a cheap tool for amateur sporting activities, leisure riding, or livestock herding that can be replaced with a new horse at a minimal cost while an injured horse is withdrawn to continue its life on pasture or is sold for meat. A horse is considered a temporary object that is not worth investing in beyond the commonly accepted minimum. That can be said to a certain extent of horse racing and traditional ethnic equestrian sports like kokpar (buzkashi, kokboru) where horses are subjected to substantial injury-prone physical pressure.

Horses with back-related issues require very expensive treatment or they face the risk of early retirement, thus forcing the owners and trainers to spend an additional sum on training new horses. Better understanding of horse's back shape and resolving the saddle fitting problem is also economically beneficial. It increases the performance of horses, reduces the welfare problems that usually arise with unfit saddles, all of which will save huge amounts of money for the entire equestrian industry [3].

Figure 1 Examples of horseback injuries related to improper saddle fitting

Current methods of saddle fitting employ visually assessing the saddle without any kind of deliberate techniques to increase fitting accuracy, these methods are very subjective and often prone to human errors. To measure the contact area of the saddle, manufacturers put pressure mats on horse-back. While this approach has its benefits, it does not take into account the dynamic shift of back muscles of the horse. The information on how saddles interact with horse-back surface during dynamic motion is rarely taken into account during the saddle design process.

1.2 Problem Statement

On top of previously mentioned challenges, the equestrian community in general and saddle manufacturers in particular face problems related to the horse measurement system. To be more precise, there are no universally accepted saddle and equine back measurement systems in the world, nothing close compared to human footwear and clothes size charts [4]. Eventually, this leads to the complexity of saddle making process, which results in unfit, incompatible and inappropriate saddle designs. Many horses suffer from spinal pain and deformation. Consequently, this contributes to the poor health and dangerous behavior of the horse [5]. As mentioned above, there is no universal classification system regarding horse-back surfaces. Although a system for specific cases with horse-back disorders exists, a survey conducted by Lesimple et al. revealed that equine professionals consider existing terminology to be too vague and loose [6].

1.3 Research Motivation

This research is driven by the potential to significantly improve both manufacturing efficiency and equine welfare through the application of modern computational methods to a traditional craft. The current approach involving manual and time-intensive process of saddle-fitting can be improved through the development of automated measurement and classification system. Introduction of advanced machine learning techniques can address the gap in the equestrian community concerning saddle fitting. By developing accurate and appropriate horse-back surface categories, this research could help reduce health issues related to poor fitted saddles.

1.4 Aim and objectives

This research aims to develop a software tool capable of collecting data describing equine back surfaces and employing machine learning techniques to classify these shapes into an appropriate number of categories.

The specific objectives of the research are to:

1. Collect and preprocess the data from the measurements
2. Create a parametric model that accurately represents the horse-back shape
3. Identify key morphological features that can help us distinguish the surfaces into different types
4. Develop machine learning techniques to classify equine backs into relevant, distinct categories

By achieving all of the above objectives, the research project makes a valuable contribution to address challenges in saddle fitting.

1.5 Research questions

During the course of this research, we tried to answer several relevant questions:

- Which parameters can better describe the morphological variation of horse-back surfaces?
- Can we classify these surfaces into a finite number of distinct categories?
- What process parameters and defects detected?
- How accurately can machine learning algorithms classify equine back shapes?

1.6 Research Scope

We started with a reliable pool of measurements for nearly 60 horses in Kazakhstan. These measurements were obtained using a manual approach, alongside 13 3D scans of a selected set of their horse backs that we were able to procure using LiDAR scanning technology available in smartphones. On top of that we also have information about cross-sectional profiles of 250 horses coming from Canada. Overall, the pool of horses contains a diverse set of horse breeds, such as Thoroughbreds, Trotters, Arabians, Akhal-Teke, etc. We aim in first generating an accurate NURBS (Non-Uniform Rational B-Splines) representation [7] of back surfaces in Rhino3D CAD application [8] using horizontal and vertical positions

of cross-sectional templates for our horses. This approach provides us with a flexible and precise way to create appropriate surfaces, while also providing us with opportunities to ensure accuracy of this approach.

From experts in the field and industry we already know that the withers and croup points, angles from the lowest point on the spine of the horse to said withers and croup, as well as integral geometric characteristics, like surface and area moments, can act as relevant features for classification.

For the classification we employed various unsupervised machine learning algorithms, among them k-means, Spectral clustering, Agglomerative clustering and other clustering approaches [9], [10], [11], [12], [13]. For validations of results obtained from clustering we utilized and analyzed Silhouette score, Calinski-Harabasz index, Davies-Bouldin index [14], [15], [16].

While machine learning or parametric modeling have been widely used in many different areas of engineering, utilizing it specifically for equine applications will be an innovative approach for the entire industry. This work will help to implement these methods to veterinary practices, saddle fitting and performance evaluation of horses. Not only that, but it will be possible to use the results of this project in similar applications for different fields. It will also bridge the gap between biomechanics, computer science and equestrian practice. The research will lay foundation for future studies and works on this matter.

CHAPTER 2 - LITERATURE REVIEW

Parametric modeling of horse-back surfaces and classification of shapes of these surfaces is of vital importance in solving a number of issues related to saddle fitting, like the lack of standardized measurement system, which leads to complexity of saddle manufacturing. The review itself consists of studies about parametric modeling and Machine Learning techniques. As this work encompasses three major topics, the literature review is divided into three corresponding main sections.

2.1 Saddle Making Methods

The saddle making craft saw substantial changes across the centuries. It has moved from methods that heavily relied on artisanal techniques to more sophisticated and advanced methods. Traditional methods of saddle making were based on empirical knowledge that were passed down through generations of artisans. These processes hinged upon simple visual evaluations often neglecting the nuances of horse anatomy. With the evolution of manufacturing processes and invention of more sophisticated tools, contemporary saddle making has shifted towards a more individualized/customized approach. New techniques now emphasize not only comfort and performance of horses but also prevention of back-related issues.

Roots of saddle making sprung in antiquity, some archeological evidence suggest that it emerged as far back as 4000 BC [17]. But only in the medieval period it became a specialized trade, with established workshops where wooden saddle trees were created and covered in leather. These early saddles served primarily for their functional purposes and the rider's comfort; little thought was given to nuances of horse's unique anatomy.

Progression of saddle making can be divided into several stages. The pre-industrial period was dominated by handcrafted saddles. Then, industrial revolution began with automatization of certain aspects of saddle making, such as cutting and trimming of saddle trees, although, the saddle fitting process itself didn't see any revolutionary changes until the late 20th century. Only with the understanding that equine health and welfare directly correlates with how well the saddle fits on its back, the saddle making saw a rapid shift with saddle makers targeting horse's health too [18].

As previously mentioned, it is very important to fit saddles so that they match horses' unique back shape. Recent studies suggest that even minor mismatches can lead to significant health issues and muscle deformities [1]. Even custom-fitted saddles pose some serious challenges, like the ones concerning the precision of the measurements, and the problems related to the appropriateness of these measurements in the corresponding manufacturing process. Furthermore, these measurements are usually taken while the horse is still, which may not accurately reflect the dynamic nature of the horse and how the equine back surface performs under certain conditions and loads.

The problem also extends beyond the health and wellbeing of the horses, it also has economic implications. A lot of studies have mentioned that extensive overbearing pressure on the back muscles of the horses can result in muscle atrophy and altered movement patterns, which ultimately leads to reduced performance and early retirement [19], [20]. This motivates owners and trainers to spend substantial amounts of money and time on costly treatments, or training new horses.

Recent advances have already begun to enrich and improve traditional methods of saddle making processes. Certain saddle makers are starting to use digital tools to create appropriate, custom saddle designs for the horses, employ advanced 3D scanning and photogrammetry technologies to assess the back morphology [21], [22], [23]. However, these methods are not yet universally accepted among all saddle makers, largely due to complexity and the sheer cost of technology.

These challenges provide an ample reason to introduce more advanced technological solutions for saddle fitting problems. The integration of digital technologies and computational methods may offer promising avenues for addressing these longstanding issues [20].

2.2 Parametric modeling

Rapid evolution of Computer-Aided Design (CAD) software enables us to implement it in various ways and fields for modeling, analyzing and investigating all kinds of mechanisms or constructions. For example, it can be used for biomedical applications, in ocean engineering and many other fields [24], [25]. The opportunities that parametric modeling can provide for capturing and manipulating complex topological and morphological variations have relevance to our problem.

Parametric modeling involves describing a model's shape via a relative small set of determining parameters, and defining relationships between this set of parameters to create intricate designs that can be modified or changed by adjusting the value of the corresponding parameters [26]. It was noted in various studies that this method helps avoid high-dimensional designs and can create valid shapes without needing complex geometric or engineering constraints. However, it can be quite challenging to combine this approach with traditional design techniques and the process itself requires some amount of time to create appropriate designs that fits the quality and performance standards [27].

Using parameters and relationships between them to define and modify a model is the basis of parametric modeling. Parameters control properties, like dimension, shape, color or material, while relationships establish rules linking these parameters. Parametric features are components within a model with their own set of parameters and relationships. For instance, a hole has parameters such as diameter, depth and position while a cylinder has parameters like radius, height and orientation. These features can also have connections with elements through alignment or tangency [28], [29]. However, in some cases this can cause some problems when performing design optimization. To be more precise, when dealing with complex systems the number of parameters required to accurately determine whether the model becomes very large and the established relationships and parameters may work against each other. Changing one feature may lead to unwanted alteration of the whole model [26].

The mathematical representation of complex surfaces is a foundation of parametric modeling. There are a lot of frameworks built for this very purpose, and each one has its own use in certain applications.

NURBS (Non-Uniform Rational B-Splines) surfaces are one of the most versatile approaches for modeling free-form surfaces. They are defined by a network of control points, weights and knot vectors, which give us exceptional flexibility when representing complex curved surfaces [7]. All NURBS surfaces of degree (p, q) can be expressed by the following formula:

$$S(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,p}(v)w_{ij}P_{ij}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,p}(v)w_{ij}}$$

Where:

- $N_{i,p}$ and $N_{j,p}$ are the B-spline basis functions defined over the corresponding knotvectors U and V, with $U = \{u_0, u_1, \dots, u_{n+p+2}\}$ and $V = \{v_0, v_1, \dots, v_{m+q+2}\}$
- P_{ij} are control points
- w_{ij} are weights
- u and v are parametric coordinates in $U \times V$

This representation provides great control over curvature and continuity of surface, eventually making NURBS an ideal method for modeling complex surfaces, such as equine back-surfaces. Additionally, T-splines or Catmull-Clark subdivision surfaces can act as suitable alternatives for NURBS surfaces, each of them offering some advantages over NURBS surfaces in terms flexibility [30], [31]. However, these advantages are not particularly relevant to the problem at hand and therefore NURBS, which is a well-established and de facto CAD standard, is preferred over other approaches for this work.

Parametric modeling finds its application in many engineering areas. It displayed its immense value in handling intricate and complicated design problems. For instance, Arapakopoulos et al. created flexible models of marine propellers, which gave them an opportunity to explore variety of other designs by tuning the parameters of the model and optimizing its performance [24]. Their work serves as evidence that parametric modeling can be used for capturing complex curved shapes, while the final design can still be easy to manufacture.

Another work involving the use of parametric modeling in biomedicine is of considerable relevance to our problem in representing horse-back surfaces. Facchini et al. developed parametric models for human ear reconstruction, capturing complex ear morphology [25]. Their approach demonstrates that parametric modeling is suitable for representing biological shapes.

Furthermore, Kostas and Valagiannopoulos implemented parametric modeling in the design of nanomaterials [32]. Their work demonstrated how parametric modeling approaches can be used to solve multi-scale engineering problems. It is a testament that shows how this approach can integrate considerations across different scales of analysis, from microscopic material properties to macroscopic performance characteristics.

Despite all these advantages, parametric modeling has several challenges that must be addressed to efficiently generate representations of equine back-surfaces. When creating complex surfaces with high curvature variation, parametric modeling can become computationally intensive [29]. This may act as potential constraint limiting our ability to manipulate and analyze the model.

Another issue lies in complex relationships and interdependencies between parameters. Modification of single parameter can result in unavoidable adjustments of many other parameters which destroys model's integrity [28]. To ensure design's robustness identifying the minimal set of parameters to accurately represent the model is invaluable.

2.3 Machine Learning in Classification

Machine learning is a powerful tool for complex data analysis. Regarding equine back-surfaces classification, machine learning can identify distinct morphological categories, moreover it can help in developing appropriate prediction models. Clustering algorithms are widely used approaches in machine learning. They are part of unsupervised machine learning techniques that involve identifying natural groups within a complex set of data. In the context of horse-back shapes classification it offers a variety of tools to detect distinct categories where we do not have predefined labels. Many algorithms have proven their effectiveness in shape classification.

K-means is the most well-known and used clustering algorithm. It divides the existing data into a specific number of clusters by minimizing the sum of squared distances between data points and cluster centroids [13]. This method was successfully applied to group spectral profiles according to their shapes [33]. Even though this application is in solar physics, it still highlights the efficiency of k-means in shape classification. Despite all its advantages, k-means has several limitations when dealing with shape classification. For example, it depends strongly on the employed norm and commonly produces cluster of similar shape and size, which might not be applicable to our case. Moreover, it usually requires the number of clusters as input, although there are existing algorithms that address this problem to some extent [34], [35].

There exists a variant of k-means clustering specifically designed to deal with large sets of data. MiniBatch K-means processes random subsets of the data in each iteration rather than the entire dataset. Eventually it helps to reduce computational requirements while at the same

time maintaining clustering quality. This was demonstrated by Sculley in one of his works where he proposed the mini-batch optimization for k-means clustering. As was stated, this approach reduced computational cost by orders of magnitude compared to the classic algorithm while yielding significantly better solutions [36]. Thus, the method is suitable in dealing with large datasets or high-dimensional shape descriptors.

Spectral clustering is another clustering algorithm suitable for shape classification. It employs constructing the similarity matrix, then calculates Laplacian matrix, afterwards performs eigen-decomposition of Laplacian matrix to assign data points to their corresponding clusters in spectral space. Essentially, it reveals underlying patterns or shapes of original data that are not visible from high-dimensional space. It is worth noting that the data points are clustered using k-means or another algorithm [37].

Ng, Jordan and Weiss have done a comprehensive analysis on spectral clustering. In their work they showed how the algorithm is effective in discovering complex clusters with non-convex shapes. They established theoretical foundations of this approach, while at the same time demonstrating its superiority over traditional clustering algorithms [10]. This may come handy if linear approaches fail to capture morphological variation of back surfaces.

Gaussian Mixture Models (GMM) uses Gaussian distribution to represent data. In contrast to other algorithms they calculate the probability density function, assigning to each data point a probability of belonging to every cluster. Furthermore, compared to the traditional k-means algorithm GMM can find clusters of different shapes and sizes. Its ability to identify subtle variations in data points allowed to apply GMM in various fields, such as image segmentation and speech recognition, and this makes it suitable candidate for shape classification [38].

Agglomerative clustering is one of the hierarchical clustering algorithms. As the name implies it uses a hierarchical approach during the formation of clusters starting from separate data points and gradually merging them. To be more specific, after assigning each data point to its own cluster it calculates pairwise distances between all of them, the closest pairs are merged forming one cluster instead of two. Then it computes the distances between newly merged clusters repeating the process several times until a criteria is met. The criteria may be either minimum or maximum distance between points in two clusters, or the variance minimization in the clusters [39]. In our context, agglomerative clustering was successfully

applied to solve anatomical classification problems by Murtagh and Contreras. Their work highlights potential application for classifying equine back-shapes [40].

Ward is a specific case of agglomerative clustering where the algorithm always tries to minimize the variance when merging clusters together. It is particularly useful for data with balanced groupings, because the approach tends to create spherical clusters of similar sizes. This was shown by Murtagh and Legendre. In their work they demonstrated the algorithm's effectiveness in identifying compact, well-separated clusters. Their research highlighted Ward's ability in applications where cluster sizes are similar, this may prove helpful for classifying equine back-surfaces of similar breed or morphologies [12].

Dimensionality reduction plays a crucial role in shape classification by transforming high-dimensional shape descriptors to lower-dimensional representations while preserving essential structural information. Such techniques can improve clustering performance and increase computational efficiency. Dimensionality reduction provides means to efficiently analyze and visualize high-dimensional data. In our case where we may need to use relatively big shape descriptors, we can enhance the clustering efficiency by applying various dimensionality techniques.

Principal Component Analysis (PCA) is the most widely used techniques in machine learning and statistics for dimensionality reduction. PCA simplifies the original data by transforming its original coordinates (used in the shape descriptors) into a set of independent principal components. Each component can capture a certain amount of variance from the data. By projecting data onto these principal components, PCA simplifies the complex datasets, making them easy to analyze and visualize [41], [42]. Slice [43] applied PCA in geometric morphometrics, thus developing methods for analyzing shape variations in biological structures. The research demonstrated PCA's effectiveness in identifying shape variations and relationships between morphological types, therefore suggesting applications for horse-back surface analysis. PCA was specifically implemented by Hosseini et al. to identify morphological relationships between three different horse breeds. Their research successfully identified and classified different morphological traits inherent for each horse breed using PCA and cluster analysis [44]. While PCA is effective when dealing with linearly structured data, it has shortcomings with non-linear relationships.

LLE (Locally Linear Embedding) is an unsupervised approach designed to transform data from its original high-dimensional space into a lower-dimensional representation. It works

by representing each data point as a linear combination of its neighbors [45]. LLE has been applied in various domains, including shape classification. Liu et al. explored the application of LLE to improve classification accuracy of Alzheimer's disease using MRI data. The approach was used to capture geometric structure of data obtained from brain volume and cortical thickness measurements [46]. Their findings provided insights into handling complex geometrical data. This suggests that LLE can be potentially useful in dealing with horse-back shapes.

Another promising non-linear dimensionality technique is Spectral Embedding. To be more precise it is a family of techniques that perform dimensionality reduction through eigen-decomposition of similarity matrices. In a certain way it works similar to Spectral Clustering. Laplacian Eigenmaps – the most common approach in the family – constructs a similarity graph, then calculates Laplacian matrix, and finally solves the resulting generalized eigen-value problem [47], [48]. In the context of shape classification, Laplacian Eigenmaps were applied by Wang et al. to classify 3D models retrieval of 3D shapes. Their research demonstrated the method's capability in capturing complex geometric shapes and facilitating accurate classification [49]. Another study successfully applied Laplacian Eigenmaps to match point clouds by focusing on fine local structures, which is very useful in shape analysis [50].

Multidimensional Scaling (MDS) is another large family of dimensionality techniques. It constructs lower-dimensional representations while preserving pairwise distances or dissimilarities between data points. It minimizes a "stress" function that describes the discrepancy between original distances and those in the embedded space. The process involves creating the dissimilarity matrix, then it tries to assign each data point with a position in lower-dimension space that reflects the original dissimilarity in higher-dimensional space [51]. In a study conducted by Park et al. the authors developed a novel method involving MDS to measure the shape information from MRI scans, providing essential information about structural differences in brain anatomy. Their approach simplified the process of detecting subtle shape disparities [52]. Another work by Park involves extension of MDS called Isometric Feature Mapping (Isomap). Instead of featuring Euclidian distances it incorporates geodesic distances. Park proposed a new shape classification method based on Isomap for brain MRI [53]. This emphasizes its role in shape analysis.

It is very important to determine the features of equine back-surfaces that can accurately describe back shapes. These parameters will provide the foundation for shape classification

through machine learning. For horse-back surfaces identifying geometric features at various scales will be necessary. The length, width and height of the horse's back are the most primitive features that can be used in classification models. Throughout many studies it was proven that measurements in these particular features vary among different breeds and types [54], [55], [56].

For more complex shape descriptors differential geometric features can provide detailed information regarding local surface properties. Among them gaussian and mean curvatures, shape index at various points throughout the surface and curvature integral can provide intrinsic information about surface, and are very useful in shape classification as was proven in various researches [57], [58], [59].

2.5 Research Gap Analysis

While there are multiple approaches for shape classification and many research and projects where these methods were applied in practice, it is evidently clear that we require a new unique Machine Learning algorithm that works specifically on horse-back surfaces. Many studies highlight morphological variation across different horse breeds, but none of these studies attempt to create groups with distinct horse-back variation [54], [55], [56]. The studies that try to classify the horse into generalized groups focus on only specific breeds of horses and don't put enough emphasis on equine-back shape disparity [44]. This research aims to fill this gap and face many challenges surrounding horse-back classification. Unlike other approaches reviewed in past studies, the proposed method will have to deal with limited data and its accuracy rate must be high enough to correctly identify shape categories. This paper and studies reviewed within it give us a promising opportunity to develop a universal sizing system.

CHAPTER 3 – METHODOLOGY

To commence the research, we, first of all, require building appropriate parametric models of horse-back surfaces. To create these parametric models, measurements of the horses are required. The data is produced either by using 3D scanning, semi-automatic measurements, such as using a set of predefined templates, or other manual measurement techniques. The manual measurements are commonly acquired by applying specially designed templates that represent cross-sectional and longitudinal profiles of equine backs. Real life models of these templates can be crafted from wood, cardboard or any other solid material. Typically, there are 3 types of cross-sectional templates: A, B and C. Each type can be distinguished by its shape and represents specific parts of the horse's back. Type A templates correspond to the front part of the back starting from withers extending for approximately 200-250 mm. Type B templates are compatible with the middle part of the back, they may begin from 200-250 mm from withers till 350-400 mms from withers. Lastly type C templates correlate with the back part of the equine's back; the last template may extend all the way to 650 mm from the withers. Of course, these are general guidelines and depend on the breed, age or even posture of the horse. The actually fitted templates may vary significantly among different horses. That said, to ensure the accuracy of measurements, horses must stand straight and still, as was mentioned previously even the slightest change in posture can alter the template matching due to muscle alterations. As for the longitudinal templates, there are 5 different shapes in total. The main difference between them is their curvature to accurately reflect the convexity and concavity of the horse's back in the longitudinal direction.

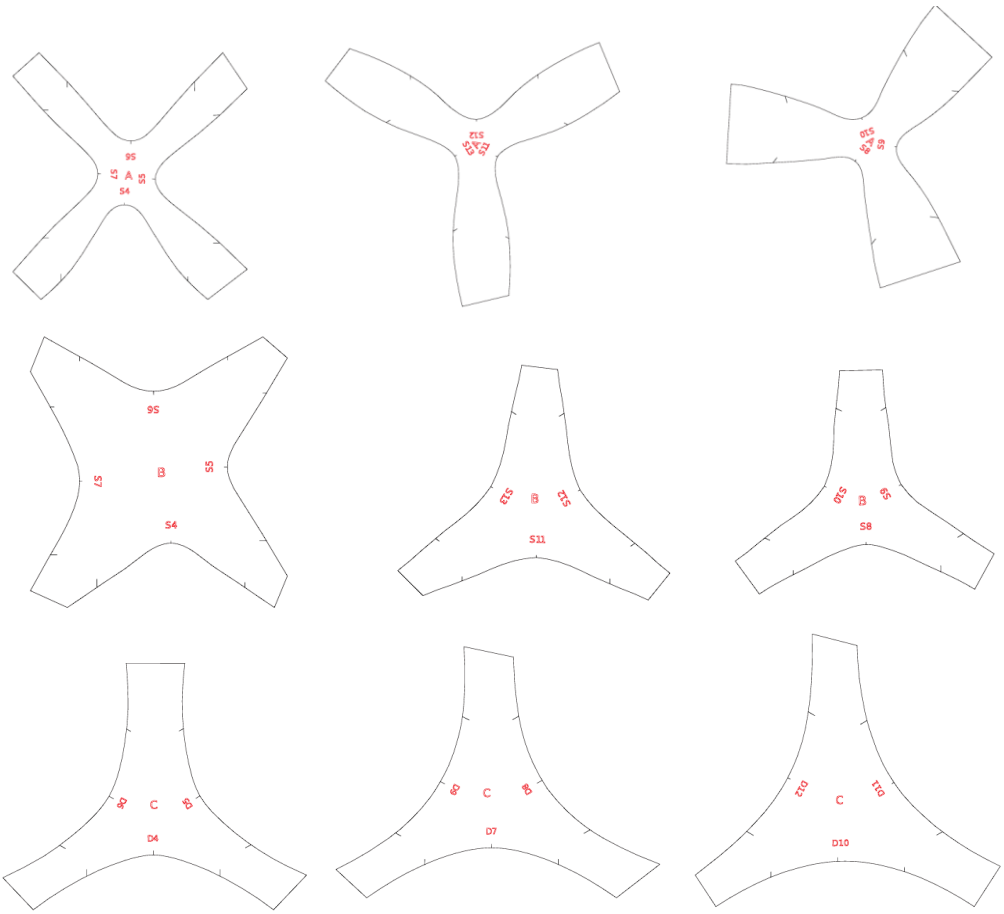

Figure 2 Examples of cross-sectional templates

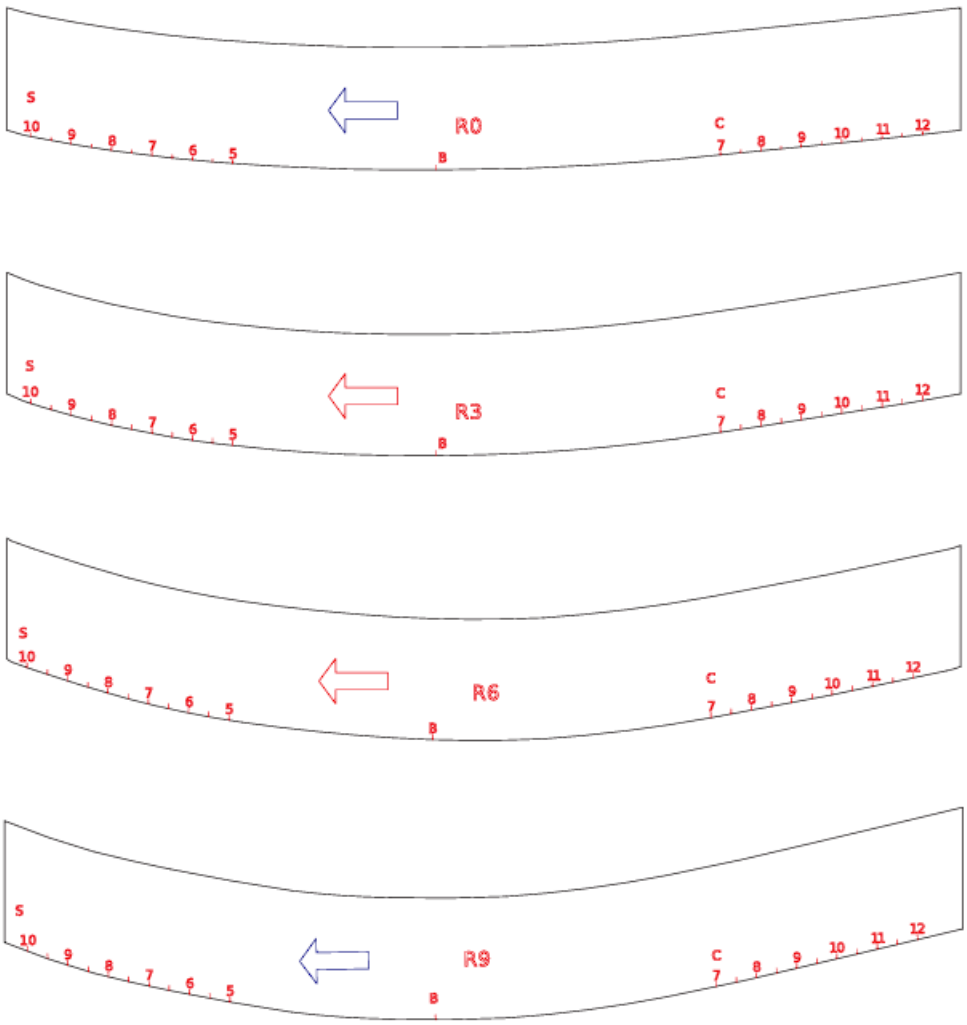

Figure 3 Examples of longitudinal templates

Typically, there will be 13 positions considered along the horse's back. In the beginning we must identify the withers, which is the starting point for placing templates. Each position is approximately 50-55 mm apart from each other, and we start placing the most suitable cross-sectional templates on these positions. Afterwards, we put a ruler on withers so that it stays horizontal without any incline. From there we gather vertical distances for each of these 13 positions and using a flexible ruler we can also measure distances along the spine between successive positions. The longitudinal templates are placed parallel to the spine of the horse approximately 150 mm apart from each other.

The process of taking measurements using predefined set of templates is time-consuming, measuring one horse can take up to one hour depending on the temperament of the

animal, not mentioning that there may be human related errors while taking the measurements. That's why utilizing 3D scanning applications is the most sensible choice. The process is significantly easier and much faster compared to semi-automatic or manual approaches. Despite many advantages of 3D scanning, a large database of measurements taken with templates already exists and therefore the generation of parametric models should take these datasets into account.

We begin the process of generating the parametric model by digitizing all physical templates (cross-sectional and longitudinal) that were kindly provided to us by Mr. Aktan Akhmetov. CAD application Rhino3D offers a range of tools for parametrization and curve fitting. Raw data will be used to properly parametrize the curves defining cross-sectional and longitudinal templates within selected CAD package. For proper pre-processing, the digitized and original templates are compared to ensure that their deviation is within the acceptable tolerances used in the equine community; typically, 2 to 3 mm.

After ensuring the suitability of digitally reconstructed templates, we are able to design the parametric model and ultimately generate horse-back surfaces based on the measurements we possess. In total we have 57 full measurements from horses in Kazakhstan and 242 cross-sectional measurements for horses from Canada. The latter dataset lacks the horizontal and vertical positions of cross-sectional templates. The first dataset will be the baseline on which the initial classification model will be built. The data containing cross-sectional information of Canadian horses will be used to generate artificial horse-back models that can accurately represent real ones and enrich the training set. Using built-in tools in Rhino3D we can automate the process of generating the surfaces; we can write a python script that takes the measurements as input and returns the final surface as output saving them in a local folder. As we have horizontal and vertical positions of templates, as well as which templates correspond to each point for all horses, they can be placed in global space according to the measurements and using variety of curve interpolation methods, such as uniform, chord-length, B-spline approximation or Gordon surfaces, generate the NURBS surface representing the horse back by interpolating the appropriately positioned digital templates. As for 3D scans, the raw data is converted to a 3D point cloud, and the points will be further interpolated using bicubic interpolation to generate surface objects. It is essential to represent all horse-back shapes as surfaces, as we can obtain surface specific features, like area, surface moments or even volume alongside volume moments from these representations.

Apart from horse-back surfaces it is important to generate longitudinal curves representing the horse's spine. Horse spines greatly vary depending on breed and type of the horse. If we can establish the categories of spine forms that each horse can fit into, work on surface classification will be greatly facilitated. The longitudinal curves defining horse spines are simple 2D curves and will be represented as cubic B-splines. Simple geometric features are the only ones we can obtain from them, the main ones are angles, length, height, width.

Since clustering algorithms do not natively operate with surface or curve objects, we will need to represent 3D surfaces of equine-backs as shape vectors (or matrices) with 3D point grids and pointsets along curves being the typical representation for surface and curves, respectively. The coordinates of each point in the grid are part of the features' input used in clustering. Furthermore, having the full surface representation, we can obtain gaussian, mean curvature at each point on the surface, as well as its shape index [60]. These high-order surface attributes can act as powerful additional features further enhancing clustering quality.

Within the same set of tasks, a series of pre-processing filters is required for 3D scans. Firstly, transformation of 3D horse-back scans into point clouds and extraction of above-mentioned feature curves at the same landmark points identified for manual 2D measurements. Matching extracted curves against parametrized curves of the previously determined 2D templates to determine the accuracy of 3D scans. This is essential for generation of 3D horse-back surfaces that exactly match the original ones. Finally, based on the generated horse-back surfaces, appropriate saddle designs will be created.

The next stage of the research involves Machine Learning. Supervised and unsupervised Machine Learning algorithms will be employed to establish equine-back classification and saddle sizing. For this stage it is essential to extract appropriate shape descriptors, such as angles, slopes, dimensions and feature curves. We can obtain simple geometric and higher-order features using built-in Rhino3D tools and functions. To automate the process of extracting such features, python scripts using Rhino's API were implemented. Tables are finally created, each containing all the information about the features.

The shape descriptors are used in the Machine Learning algorithms to assign horse-back surfaces into a finite number of surface categories, which will be further used to establish saddle sizing. Firstly, we will determine the performance of each feature we were able to get. By building and analyzing correlation tables we can gather all possible, workable combinations of features. Different clustering algorithms and dimensionality reduction techniques will be

utilized. In existing literature, classification algorithms were not used on categorizing horse-back shapes, so using multiple clustering approaches and assessing the quality of results to better understand how they handle the data is needed. To better handle high-dimensional data, employing diverse dimensionality reduction methods will help us comprehend which ones will be able to transform the data without losing significant information and to identify the most useful underlying patterns. To properly assess the quality of results series of evaluation metrics will be employed, such as Silhouette score, Calinski-Harabasz index, and Davies-Bouldin index [14], [15], [16]. As the categories will be identified, a generalized representative surface will be developed for each category, and surface metrics will be employed to evaluate the suitability of the representative surface and its deviation from surfaces both within and across previously established surface groups. At the same time, proximity and mismatch metrics will be used to assess how well saddle surfaces fit representative surfaces.

It is worth noting that while we have a limited pool of horse measurements it is still possible to categorize them using machine learning algorithms. But in order to reach satisfactory clustering results we may extract more longitudinal curves to enhance clustering quality and use as validation of generated groups. There are thousands of images of horse's side profiles and by employing image segmentation it is possible to extract accurate horse-back spine representations from images. Deep learning models, such as Segment Anything Model (SAM) or Grounding DINO, can with relative ease detect objects, like horses, on images and extract their masks [61], [62]. After getting object masks, it will be a matter of identifying longitudinal spines and extracting them.

As in the previous stage, the categories for saddle surfaces will be created using Machine Learning, and the averaged-out standard surface will be developed for each category. These surfaces will be used in static and dynamic saddle fit analyses. During static analysis computational geometry techniques and algorithms will be employed to compare horse and saddle surfaces, similar metrics as in previous tasks to estimate proximity and mismatch between the surfaces will be used. As for the dynamic analysis, a "physics engine" will be employed to perform approximate motion simulation and identify collision between saddle and horse surfaces and pressure points of saddle tree with horse's body.

CHAPTER 4 - RESULTS

As the first step of the project, we successfully parametrized all template curves in Rhino3D. The original real-life templates were stored in .DXF format as polygons, we created completely new smooth cubic B-splines and saved them in .IGS format, which is natively supported in many applications, like MATLAB, Rhino3D, etc.

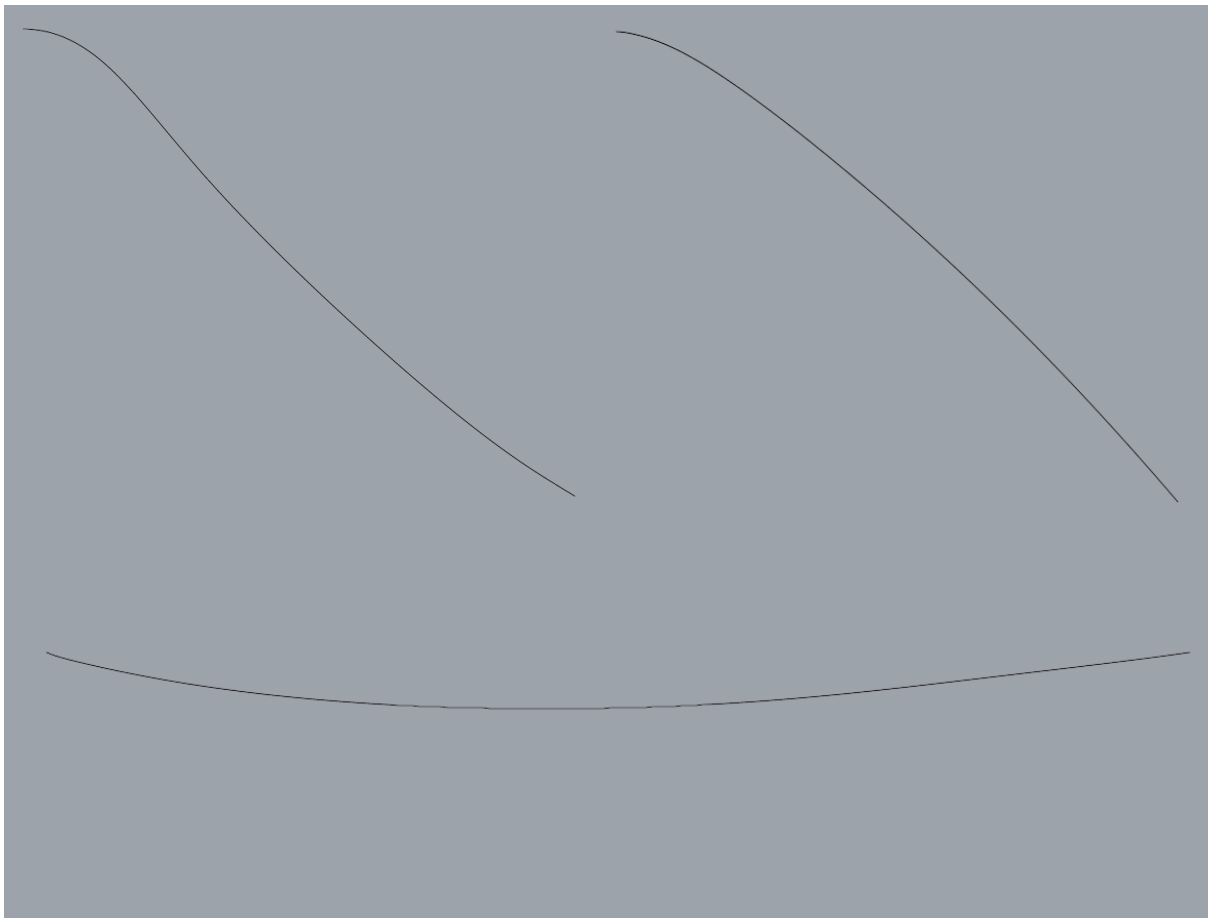

Figure 4 Cubic B-spline representations of cross-sectional and longitudinal templates

To make sure that newly generated curves are accurate, we calculated deviations between original polygons and parametrized curves. Across all 83 template curves the deviations did not exceed 2% of total curve length, we can say for certain that our B-splines represent templates sufficiently accurately, based on tolerances used by saddle makers and the equine community.

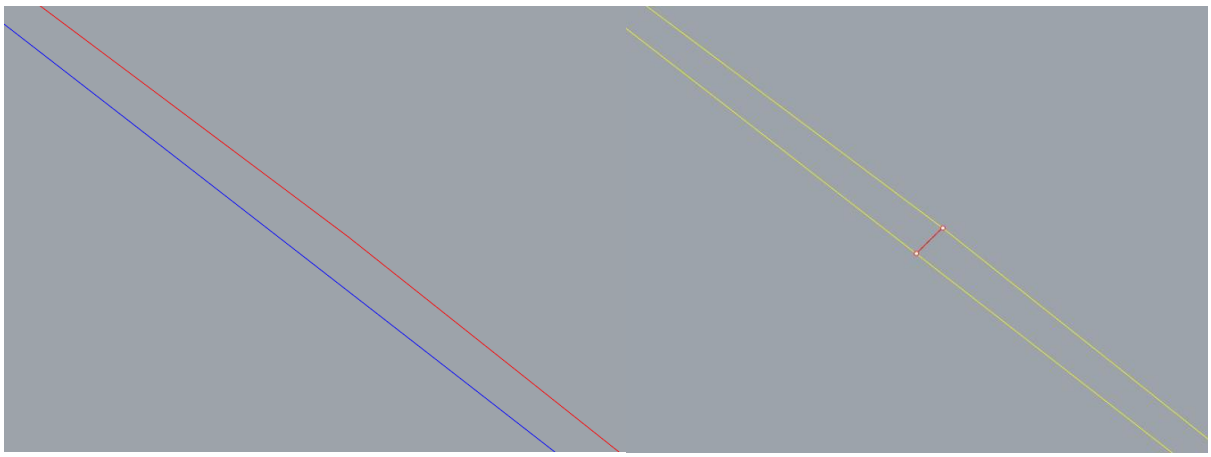

Figure 5 Deviation between original template (red) and parametrized template (blue)

The deviation differs from curve to curve. For instance, in Fig. 5 the total deviation is equal to 0.27 mm, which fits the 2% threshold we set. Similarly in every other curve the total deviation does not exceed the boundary value. The next step involves building horse-back models using the parametrized curves based on measurement data obtained.

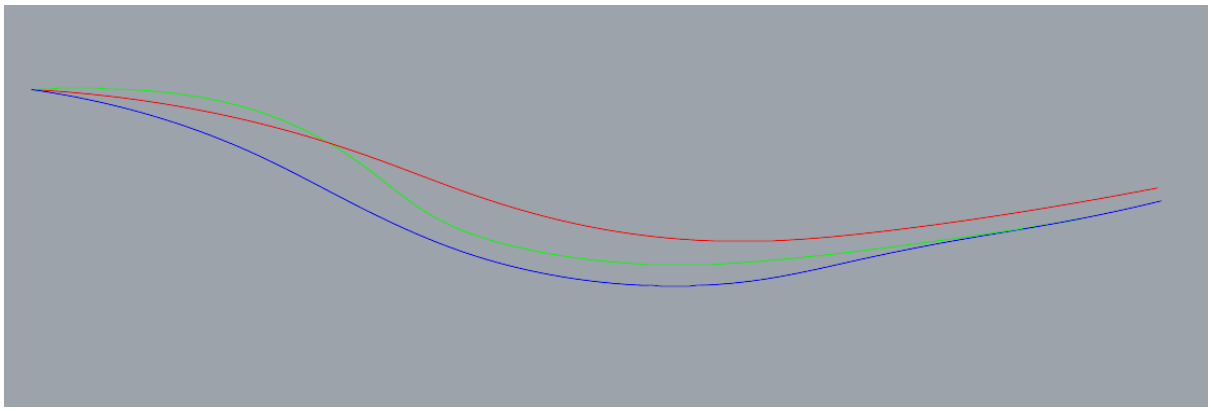

Figure 6 Examples of longitudinal spine representations

We developed a python script that works in Rhino3D environment, it places cross-sectional and longitudinal templates according to their positions stored in measurements dataset. As a result, we get a network of curves that will be further interpolated using NURBS interpolation to create the final surface. Apart from that, we also wrote a python script that can create longitudinal curves to represent horse-back spines.

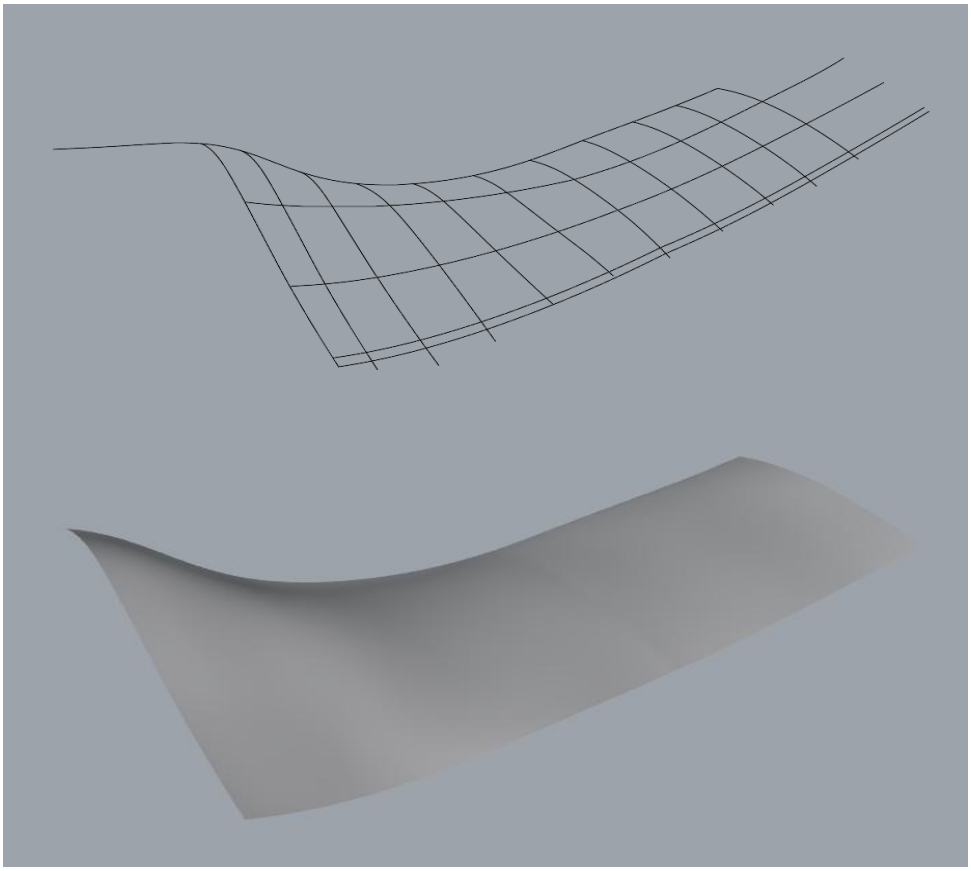

Figure 7 Curve network (top) and horse-back surface (bottom)

After successfully generating parametric models of horse-back surfaces and curve representations of horse-back spines we can start extracting features for classification. We begin by creating a grid of points on the equine-back surfaces. The coordinates of each point on the grid will act as essential features for the clustering algorithms. The point grid can be distributed on surface uniformly on parametric or physical domain, or we can distribute points based on the surface curvature, in other words more points will be distributed on the parts where the geometry changes abruptly. Then, on each point we collect the values of gaussian, mean curvatures and shape index, saving the results in separate tables.

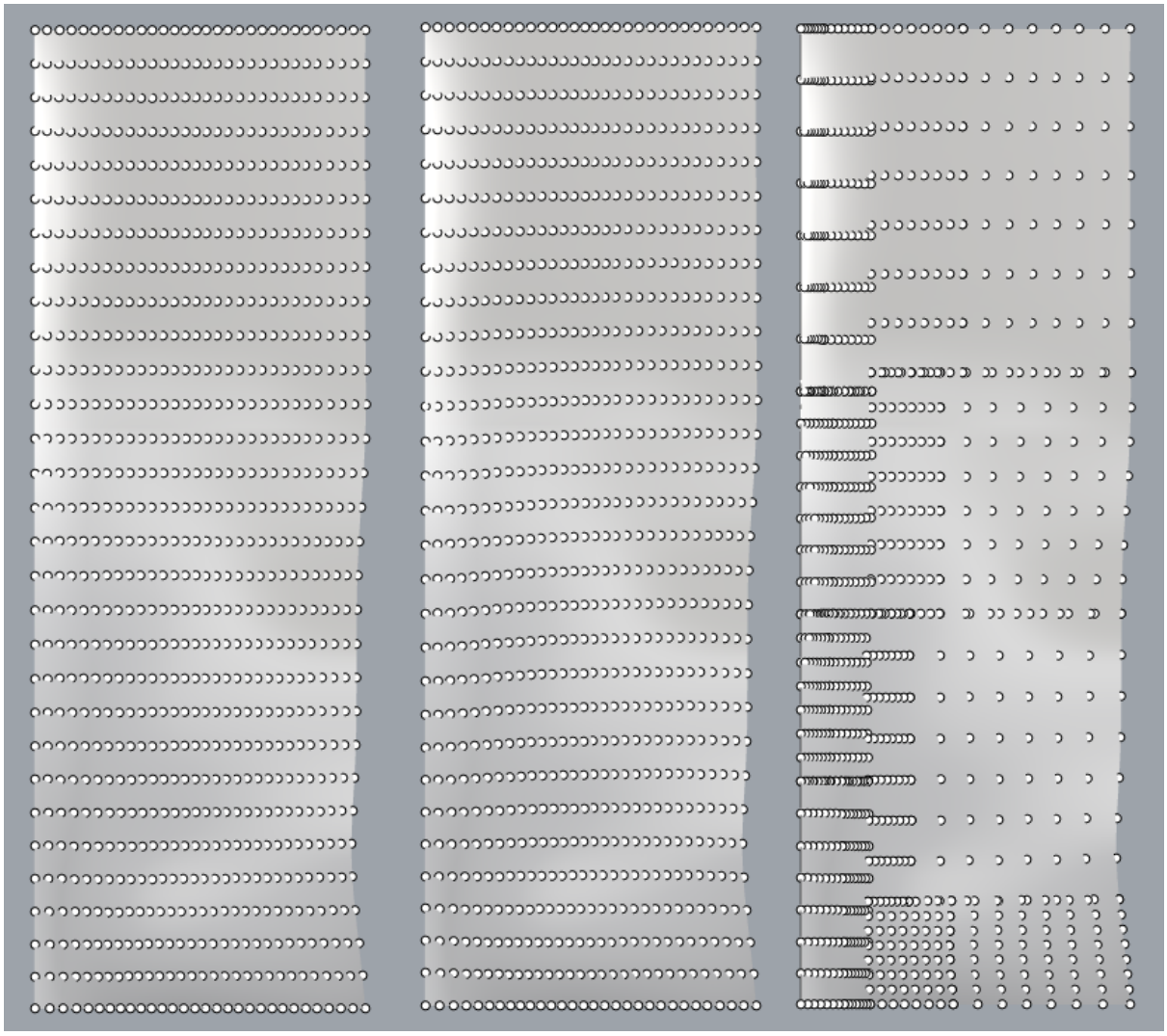

Figure 8 Examples of point distribution: uniform spacing in parametric domain (first); uniform spacing in physical domain (second); curvature based spacing (third)

On top of that we also calculated area and surface area moments using built-in Rhino3D analyzing tools, but we developed python scripts that can also calculate these features. Among surface area moments we specifically calculated first-order, second-order, third-order, fourth-order and product moments. Additionally, after transforming the open horse-back surface into a closed surface we were able to get the surface's volume and volume moments. For volume moments we decided to calculate the first-order, second-order and product moments.

Apart from surface features we calculated curve features specifically for longitudinal spines. The longitudinal curve was divided into a set of successive points, then the coordinates of these points were saved into a table. The point coordinates will act as baseline feature for

classification. The length of the longitudinal curve, its height and width were also calculated and stored in a separate table. Furthermore, the angle between the tangent line of the lowest point on longitudinal curve and the vector from lowest point to the start of the curve, as well as the angle between the tangent and vector from lowest point to the end of curve were calculated; see Fig. 9.

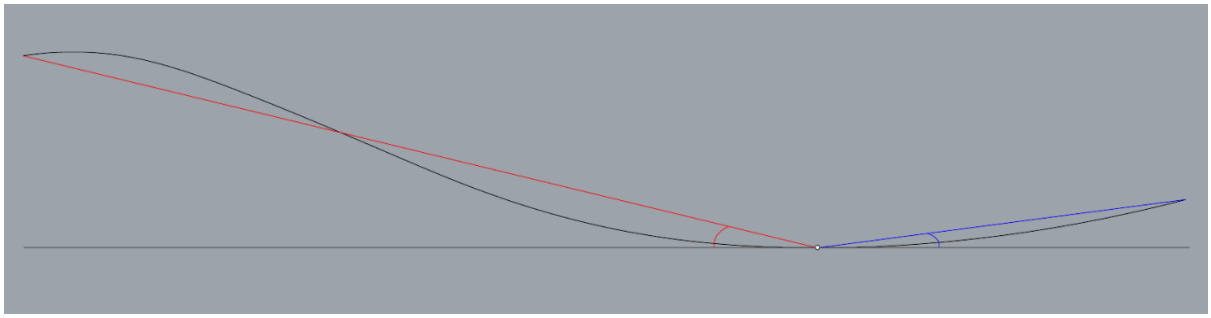

Figure 9 Determening the angle between the tangent of lowest point and the vector from lowest point to the start of the curve(red); and the angle between the tangent of lowest point and the vector from lowest point to the end of the curve(blue)

Information about convexity and concavity can serve as additional features for clustering algorithms. For each cross-sectional template included in a horse-back surface we automated the process of calculating convexity and concavity by writing a python script. A point was placed 50 mm away from the middle top point of the template. 100 mm along the template length an additional point was located and the line connecting both points was drawn; see Fig. 10. The area between this line and the actual curve is the value that describes convexity or concavity of the surfaces at the position of the template. In addition, the angle between this line and the symmetric line on the other side of the template can also serve as a feature for classification; see again Fig. 10. It is worth noting that these points were not taken randomly, but according to traditional design of saddle tree. During the saddle construction most attention is often given to the match or mismatch of the angle between the bars of a saddle, thus calculating this angle can provide additional information regarding horse sizing and enhance the clustering performance.

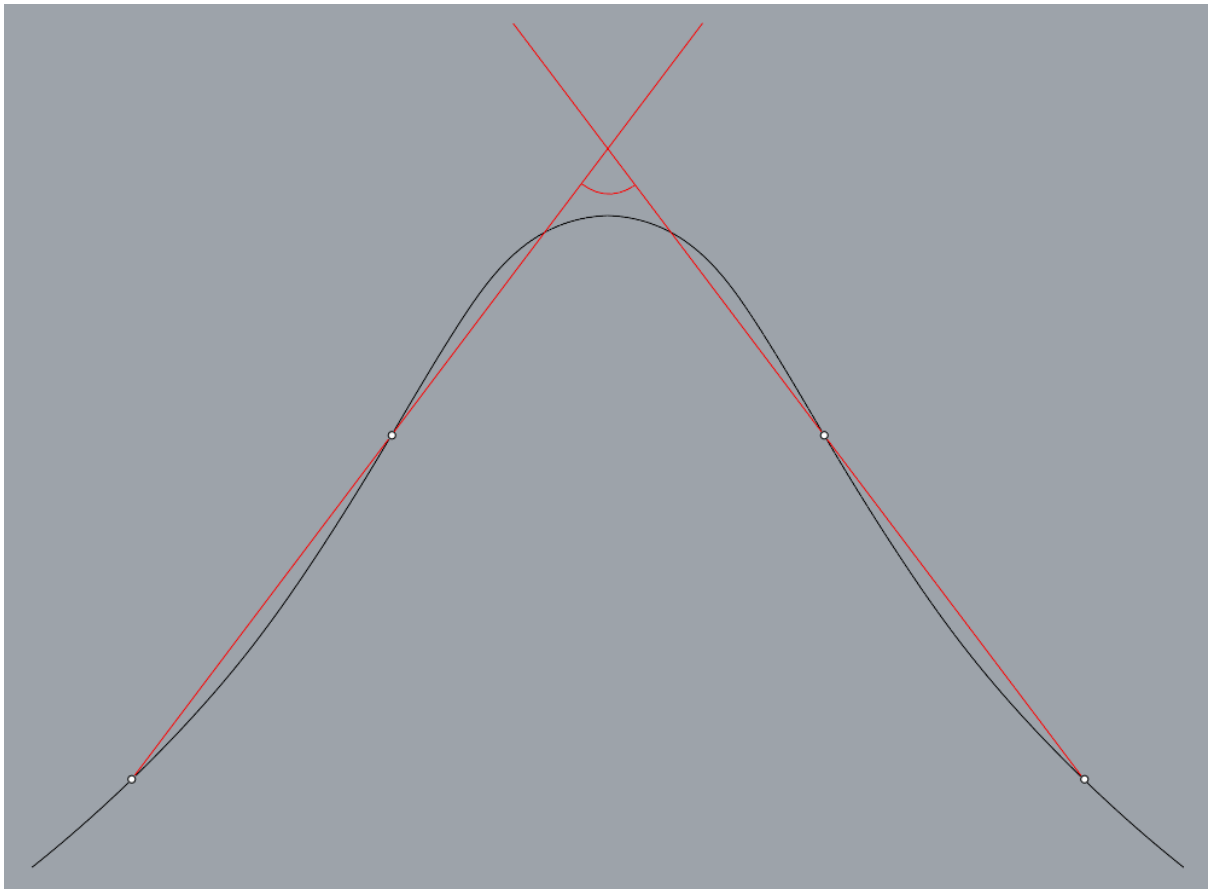

Figure 10 Illustration of the convexity indicator and angle between saddle bars

After gathering all necessary shape descriptors that will prove useful in classification we begin clustering using the available horse population. Many of the features are to be considered high-dimensional descriptors, so before jumping into clustering we employed a variety of dimensionality reduction methods. As was stated previously, they can help to uncover underlying patterns that are usually not visible and can assist with visualizing the scatter plots significantly aiding in visual assessment and analysis. The dimensionality reduction methods used in this research incorporate different approaches spanning multiple families involving both linear and non-linear methods. Similarly, various clustering algorithms were employed as well. To evaluate the quality of results we applied several evaluation metrics, among them Silhouette score, Calinski-Harabasz index, Davies-Bouldin index, as mentioned before.

As the first step in classification of horse-back surfaces we started with classifying longitudinal spine representations. The identified categories for equine-back spines will give

us significant insights into horse-back variability. To this end, we developed a python script that takes the horse features as input, pre-processes it by normalizing the data within, runs through various dimensionality reduction approaches, then employs different clustering algorithms, finally produces histograms and scatter plots for evaluation.

In order to achieve the most adequate result, we must carefully assess the feasibility of features that were used alongside dimensionality reduction methods and clustering algorithms. As shown in the figure below, the dataset containing coordinates of 50 points distributed along the longitudinal spine curve was dimensionally reduced by Spectral Embedding and further clustered using Ward method. In this model we got 5 clusters, silhouette score is equal to 0.59 which implies somewhat fair clustering, but it is evident that the clusters are not distinct enough; see Fig. 11. Comparing this particular cluster's Calinski-Harabasz index of 167.92 to others, it lies in high end, though there are cases with higher values. Davies-Bouldin index of 0.515 suggests that the clustering quality is quite good but can be better.

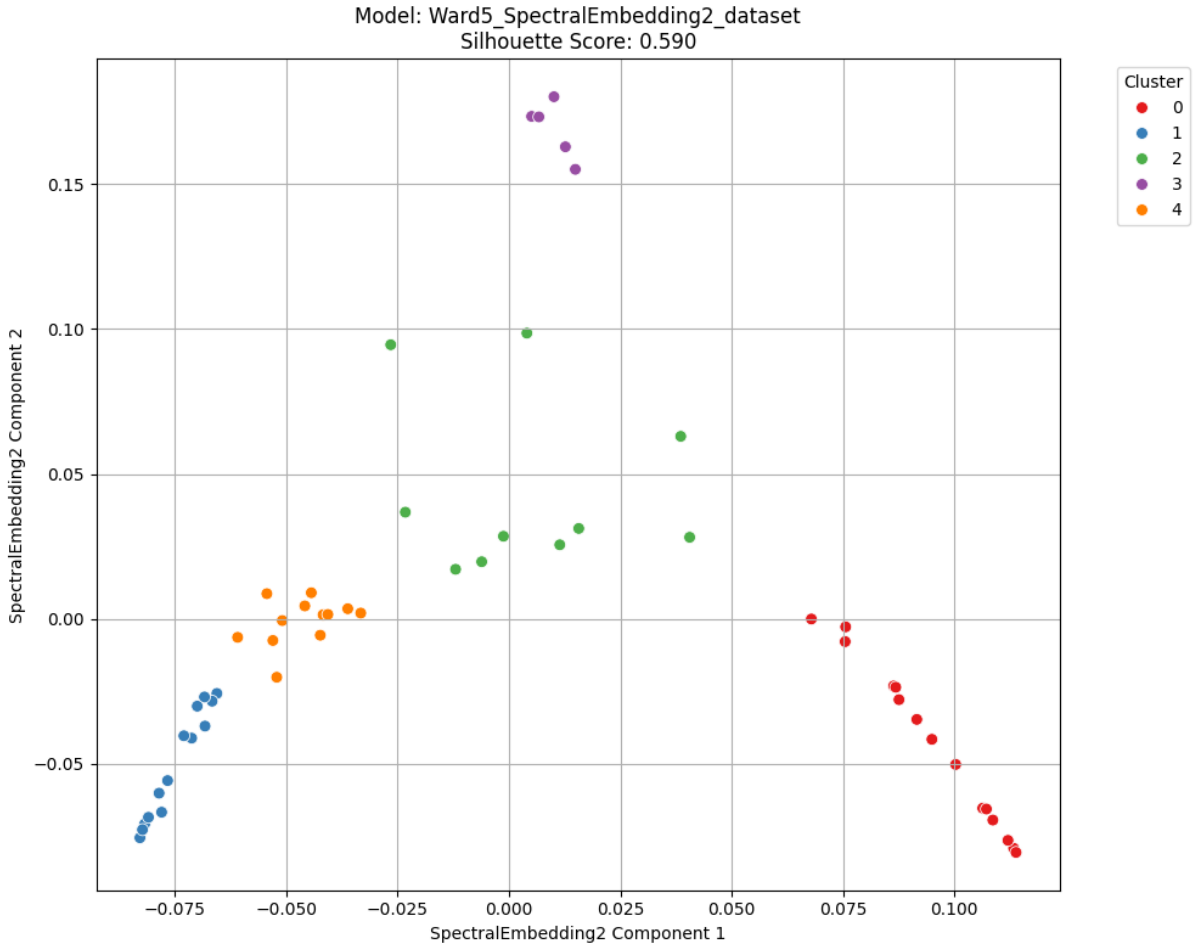

Figure 11 Scatter plot of 5 clusters processed with Ward algorithm for shape vector U0 (U0 - shape vector containing point coordinates)

Moreover, while analyzing the similarities between the representative curves of each cluster with each other by comparing them visually and their values of Hausdorff distance and Frechet distance it becomes clearer that certain cluster representatives are too similar to each other and clustering quality is not good enough; see Fig. 12.

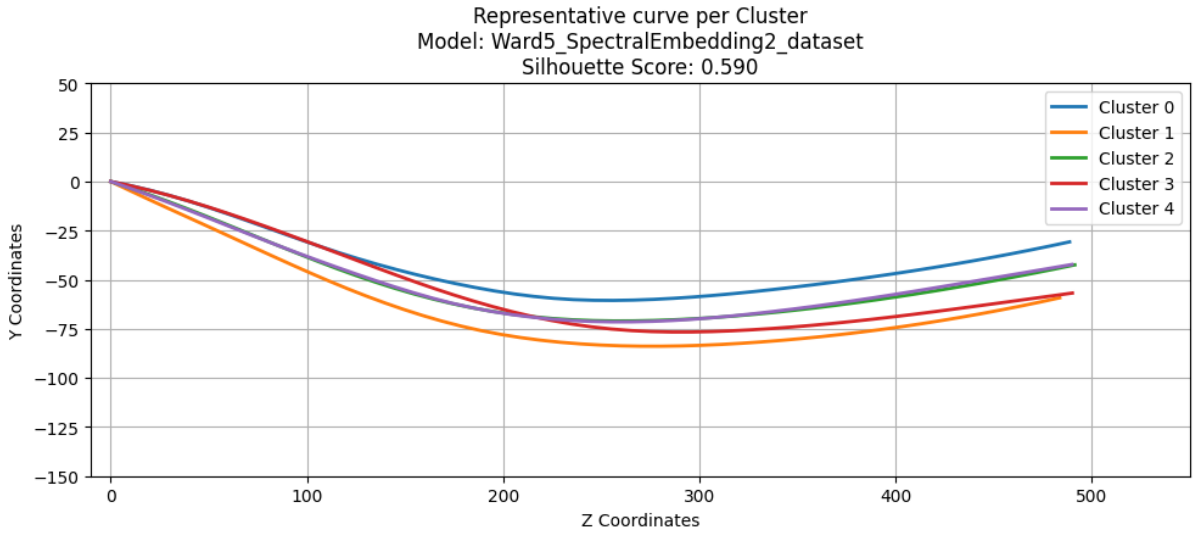

Figure 12 Visualization of representative curves of each cluster for shape vector U0

If we compare both visually and the values of Hausdorff and Frechet distances, we can see that some of the curves within each cluster do not match the shape with other curves in that cluster as well as representative clusters. The unsatisfactory result of this particular dataset suggests that we should try additional combinations of features to get better clusters.

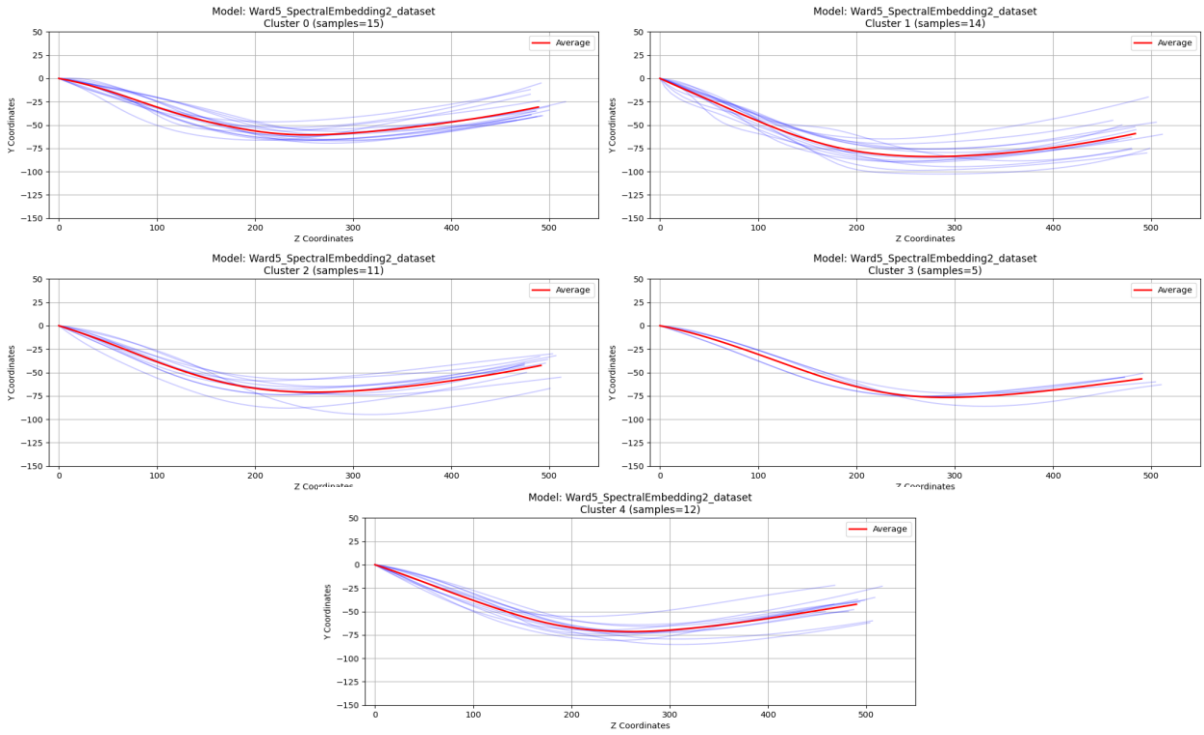

Figure 13 Visualization of representative curves of each cluster with every sample curve in that cluster for shape vector U0

This time we combined point coordinates with the angles between the tangent on given points on the curve and a fixed horizontal line. On top of that we added area moments of first and second order. As we can see in Fig. 14, this time we have 4 clusters that are well defined and clearly separated. Compared to the previous dataset, the silhouette score is increased by significant margin and is equal to 0.722, which suggests high clustering quality. With Calinski-Harabasz index of 245.118 and especially Davies-Bouldin index of 0.365 indicating significant improvement from the previous dataset.

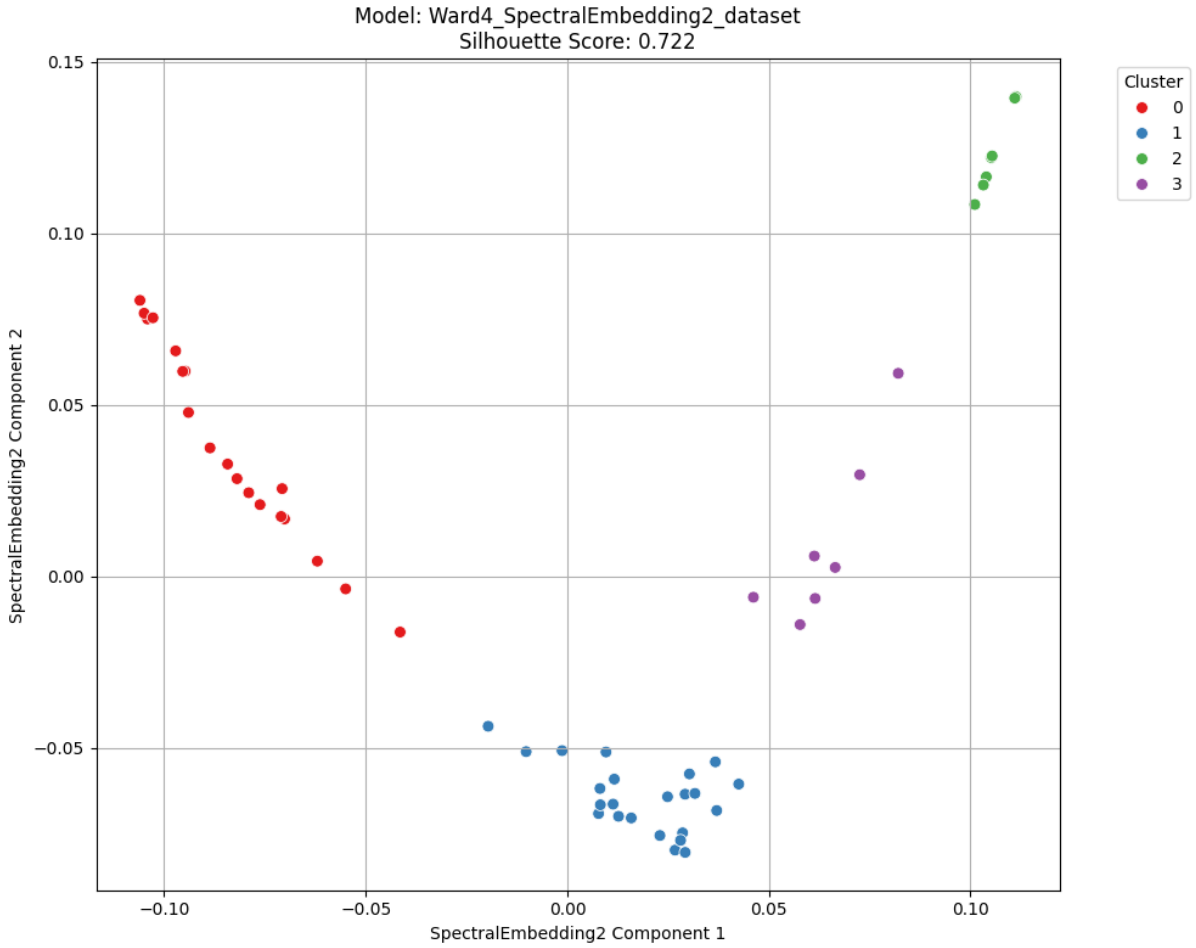

Figure 14 Scatter plot of 4 clusters processed with Ward algorithm for shape vector U1 (U1 - shape vector containing longitudinal curve point coordinates and curvature angles)

If we compare representative curves for these clusters, we can observe that all of them have distinct separation at the front part and at the back part of the spine. Hausdorff and Frechet distances between each curve in the cluster and cluster representative are also satisfactory and therefore we can consider this clustering the best so far. Furthermore, we can compare the average angles between the tangent on the lowest point of the longitudinal curve and the vector from the lowest point to the starting point of the curve; similarly for the ending point of the curve. The average values as shown in Table 1 signify that the representative curves are separated by distinct curve features. The standard deviation within the clusters for these values does not exceed 2 degrees.

Table 1 Average values of angles between tangent of the lowest point on curve and vectors from lowest point to both ends of the curve in degrees

Cluster number	Angle between tangent and vector from lowest point to starting point	Angle between tangent and vector from lowest point to ending point
0	14.11	8.96
1	14.95	7.5
2	13.41	6.37
3	15.64	5.89

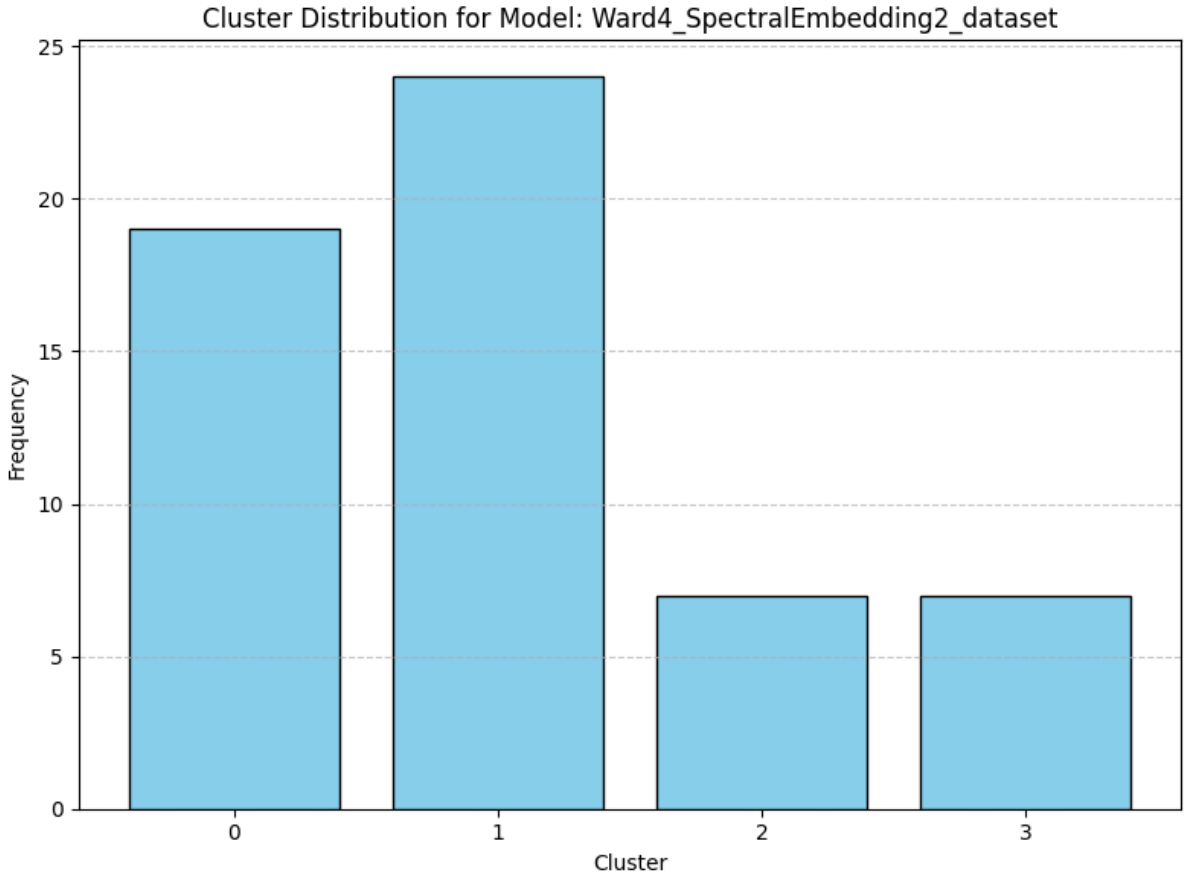

Figure 15 Histogram plot of classification model for shape vector U1

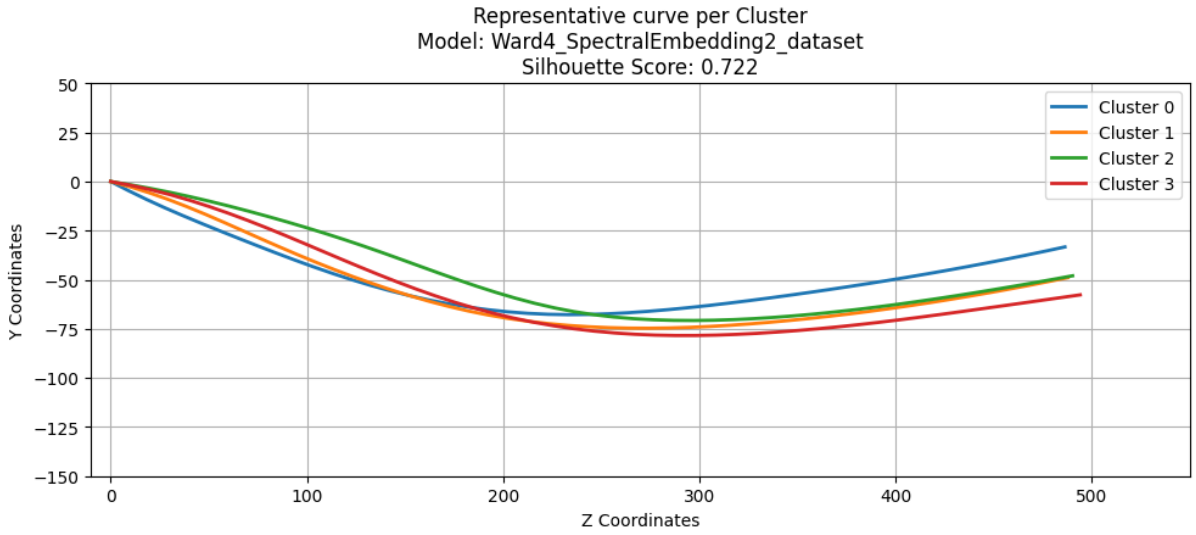

Figure 16 Visualization of representative curves of each cluster for shape vector U1

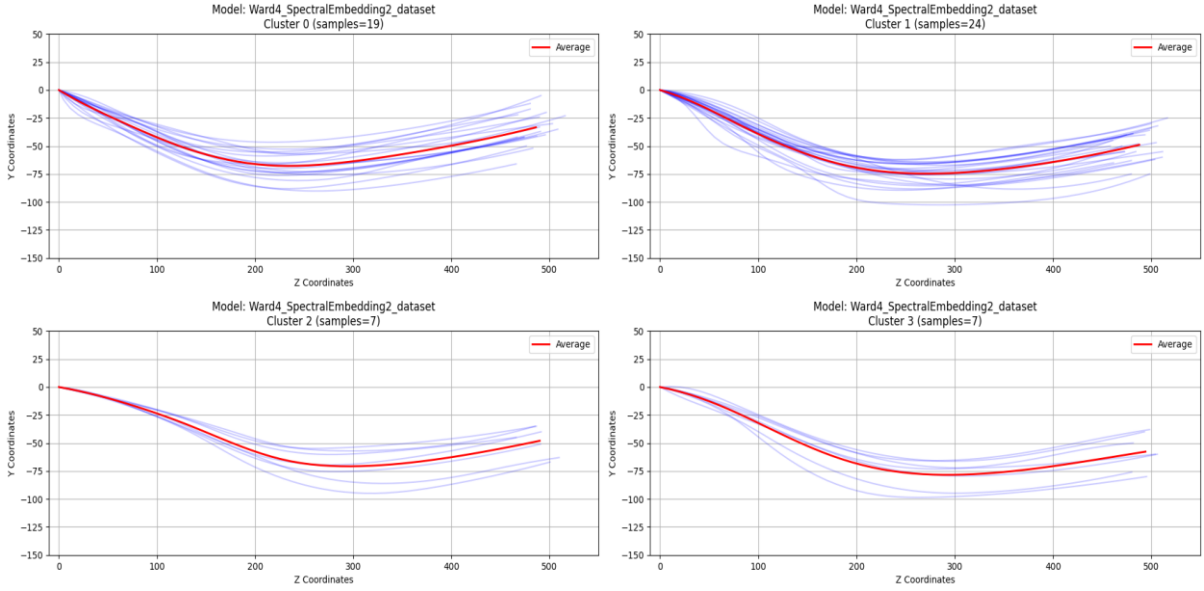

Figure 17 Visualization of representative curves of each cluster with every sample curve in that cluster for shape vector U1

When visually comparing longitudinal curves within each cluster it is evidently clear that they are grouped together by their overall shape. The results obtained from the dataset with point coordinates, curvature angles and area moments are satisfactory and are the best so far, therefore we can use this model as the base classification model.

After properly defining general categories for the longitudinal curves, we start classifying horse-back surfaces. The approach will be similar to curve classification, we will use the same python algorithm that we used when classifying longitudinal curves. By implementing all 3 surface discretization methods we got several point grids of different sizes. For uniform spacing in parametric and physical domains the 57 surfaces of horses from Kazakhstan were divided into a grid of points: 20-by-20, 30-by-30, 20-by-40, 30-by-60, 40-by-40, 60-by-60. For curvature based spacing we do not adjust the number of points in each direction but rather divide the surface into patches and distribute points on these patches. We match the total number of points to be the same as in the previous point grids. Then, verify which discretization method and which grid size will be most optimal for further surface classification.

Table 2 Silhouette score and Calinski-Harabasz index of clustering models representation with varying point grids

Discretization method and grid size	Silhouette score	Calinski-Harabasz index
Parametric 20x20	0.6684	255.67
Parametric 20x40	0.6396	286.407
Parametric 30x30	0.6516	267.077
Parametric 30x60	0.6476	275.706
Parametric 40x40	0.6516	267.077
Parametric 60x60	0.6583	277.808
Physical 20x20	0.6925	279.258
Physical 20x40	0.6745	140.753
Physical 30x30	0.6936	148.907
Physical 30x60	0.6971	154.635
Physical 40x40	0.6823	137.437
Physical 60x60	0.6773	137.876
Curvature 16 patches 5x5	0.5854	137.35
Curvature 32 patches 5x5	0.504	75.68
Curvature 32 patches 6x6	0.53	85.076

Curvature 64 patches 5x5	0.544	102.348

Silhouette score and Calinski-Harabasz index of best performing clustering models for each point grid are displayed in Table 2. We can easily observe that out of all 3 discretization methods the distribution of points equally in physical domain accomplished the best results. Notice the pattern of silhouette score when the grid size increases it increases accordingly until it reaches a bottleneck, and the clustering performance starts to gradually decrease. From this information we can safely assume that the grid size of 30-by-60 is optimal for classification when using uniform distribution in physical domain.

We only possess 57 full measurements of horses from Kazakhstan, which is not enough to produce any feasible results and make concrete conclusion. We also have 243 cross-sectional profiles of Canadian horses without their vertical or horizontal positions. But since we successfully identified 4 categories of horse-back spines, we can use them to create artificial surfaces by placing cross-sectional templates along these curves. In total, by combining cross-sectional profiles of horses from Kazakhstan and Canada and distributing them along 4 spine types we got 1196 artificial surfaces. With nearly 1200 surfaces we can start the surface shape classification.

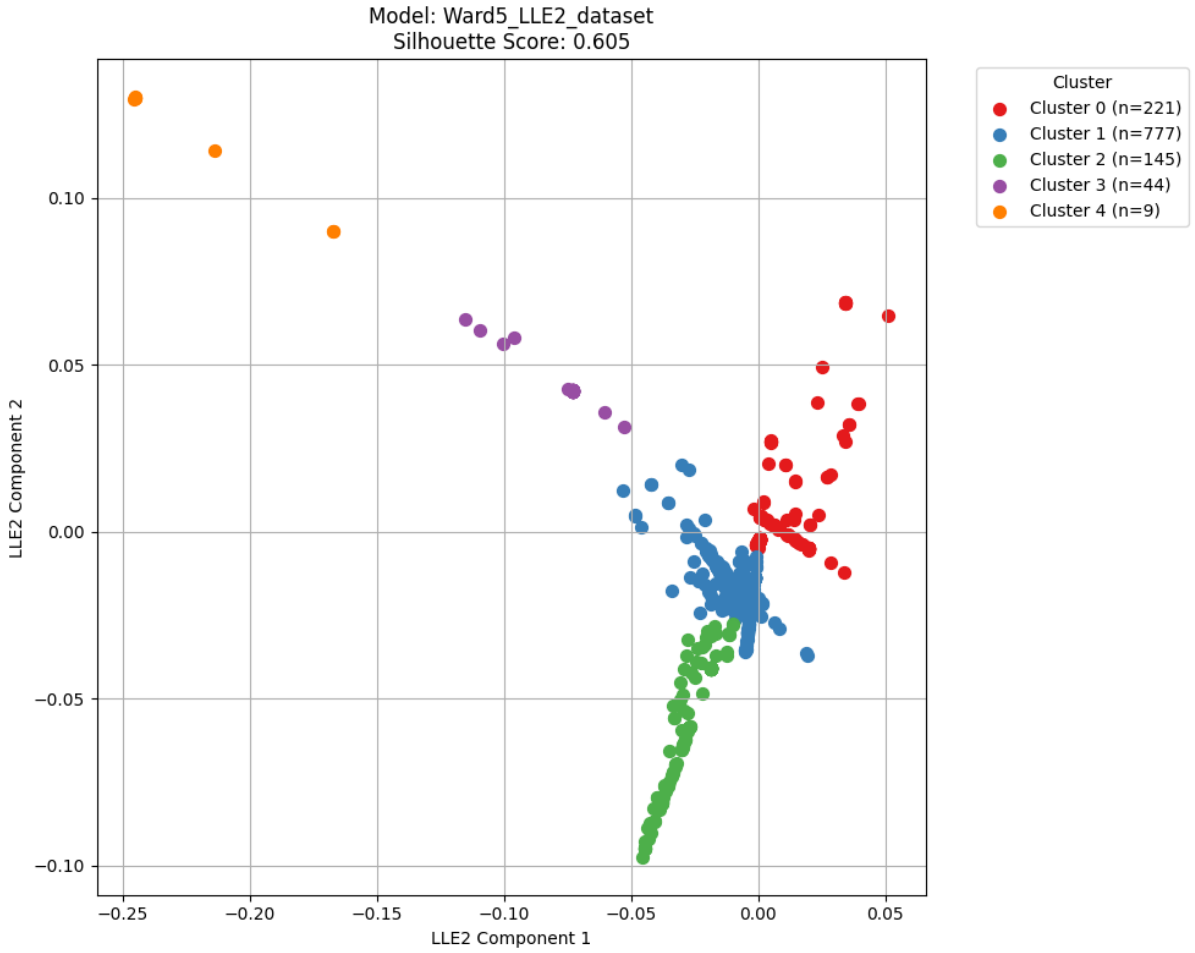

Figure 18 Scatter plot of 5 clusters processed with Ward algorithm for shape vector U2 (U2 - shape vector containing surface point coordinates and surface area moments of first and second order)

Due to the fact that surfaces are higher-dimensional entities than curves, the shape descriptors that were used in curve classification may not be very effective when clustering surfaces. It can be seen in Fig. 17 where the dataset with point coordinates and surface area moments of first and second order was processed. With the silhouette score of 0.605 indicating that the results can be improved.

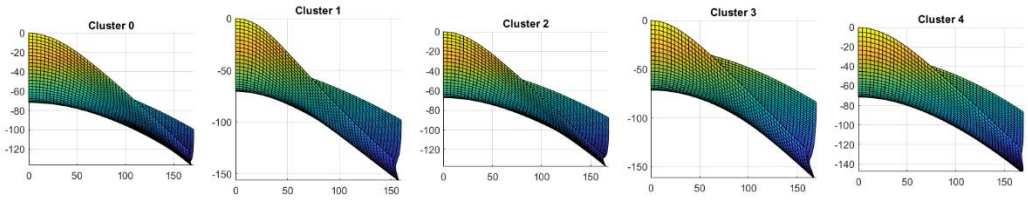

Figure 19 Front view of representative surfaces for shape vector U2

Comparing the representative surfaces side-by-side in Fig. 19, we observe some variation between them. These results are not the best, but they are fair enough overall. To improve the surface classification model, we should try additional combinations of features.

To improve the clustering quality, we combined point coordinates and surface area moments of fourth order. As visible in Figure 19, while the scatter plot does not show the clear separation between the clusters, although silhouette score of 0.715 suggests promising results.

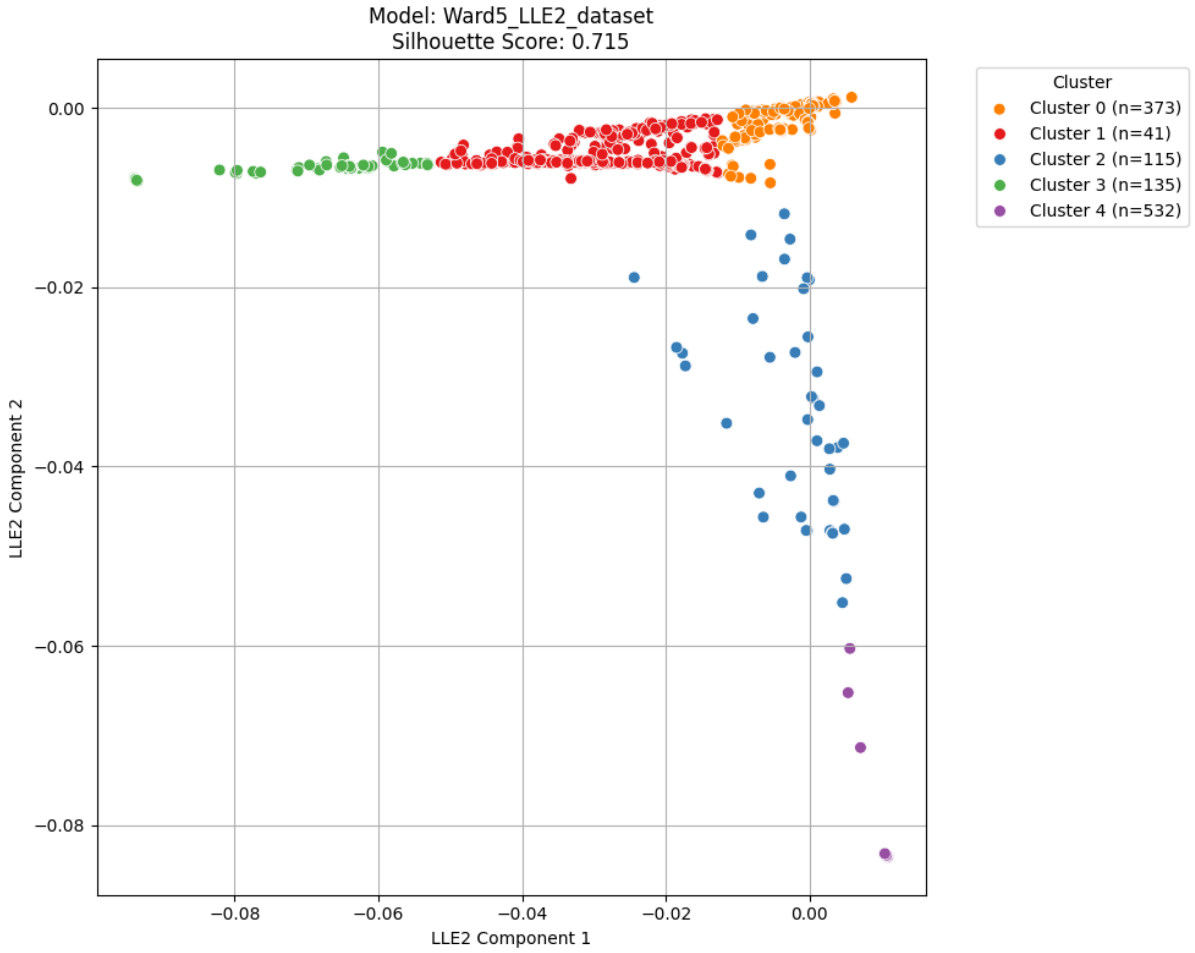

Figure 20 Scatter plot of 5 clusters processed with Ward algorithm for shape vector U3 (U3 - shape vector containing surface point coordinates and surface area moments of fourth order)

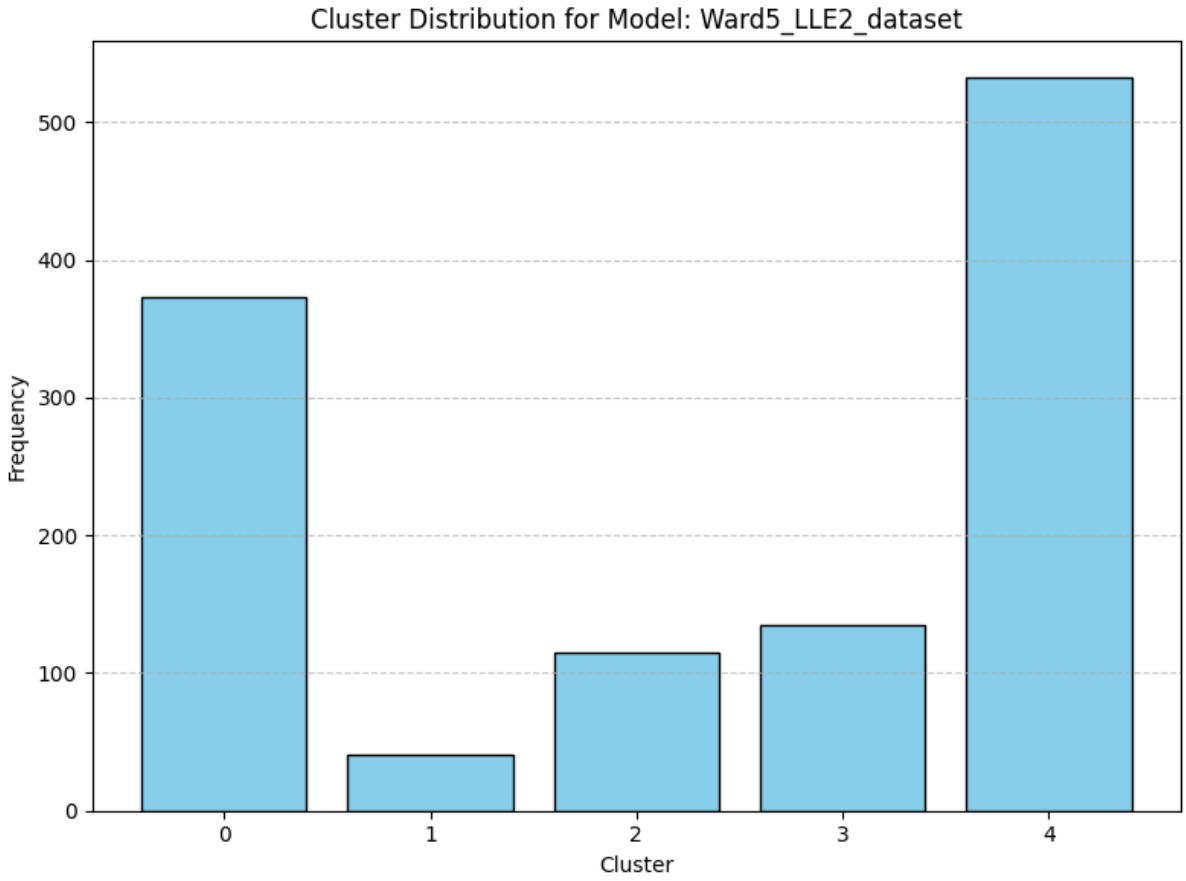

Figure 21 Histogram plot of classification model for shape vector U3

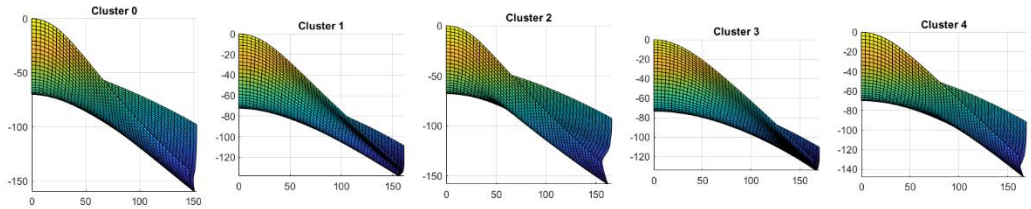

Figure 22 Front view of representative surfaces for shape vector U3

If we compare the cross-sectional profiles of representative surfaces in Figure 21, we can conclude that they are distinguishable from each other. For example, surfaces 0 and 2 may seem similar at the first glance, but at the front part surface 0 is more concave compared to surface 2. Same with surfaces 1 and 3, where surface 3 is more convex.

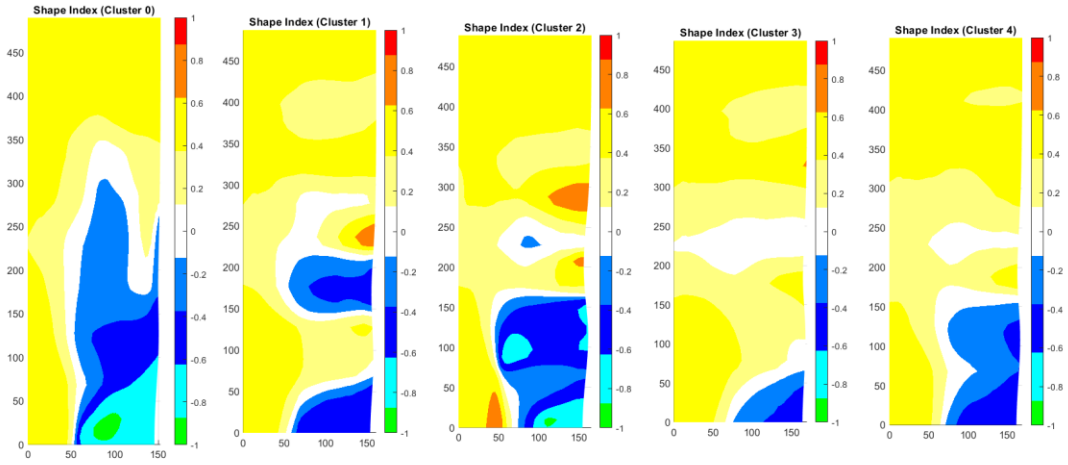

Figure 23 Shape index of cluster representatives for shape vector U3

In Fig. 23 we observe shape index distribution on the representative surfaces, where the lower part of the images is horse's front part and upper part is horse's aft part. From it we are able to discern even the slightest differences in shape between the surfaces [60]. For interpreting the correspondence between color and shape index we can refer to Fig. 24 below, where green corresponds to spherical cup, cyan to rut, blue to saddle, light blue to ridge, white to planar, light yellow to trough, yellow to saddle rut, orange to saddle ridge and red to dome. From Fig. 23 we can discern that surfaces 1 and 3 have some similarities at the front part, moving towards the middle of the surfaces there are recognizable differences between the two. All other surfaces are very distinct shape wise.

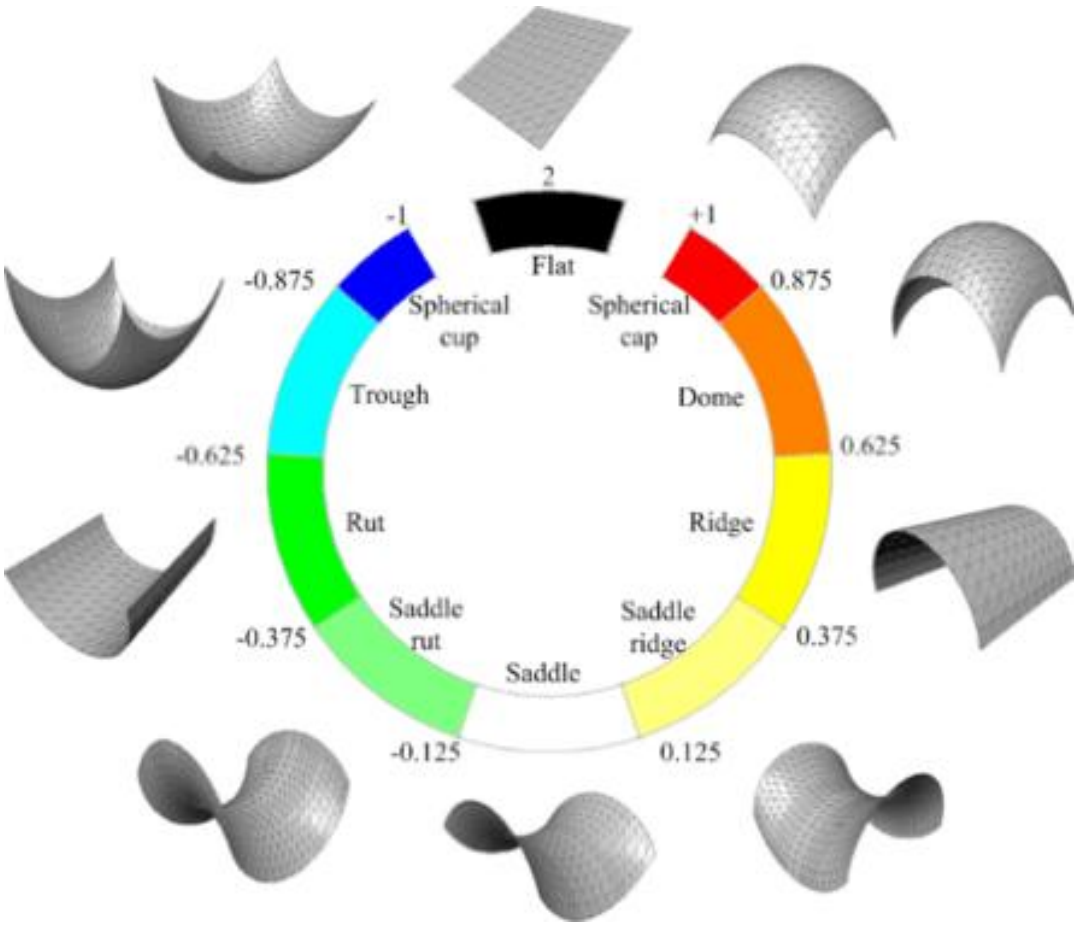

Figure 24 Illustration of shape index scale divided into nine categories [63]

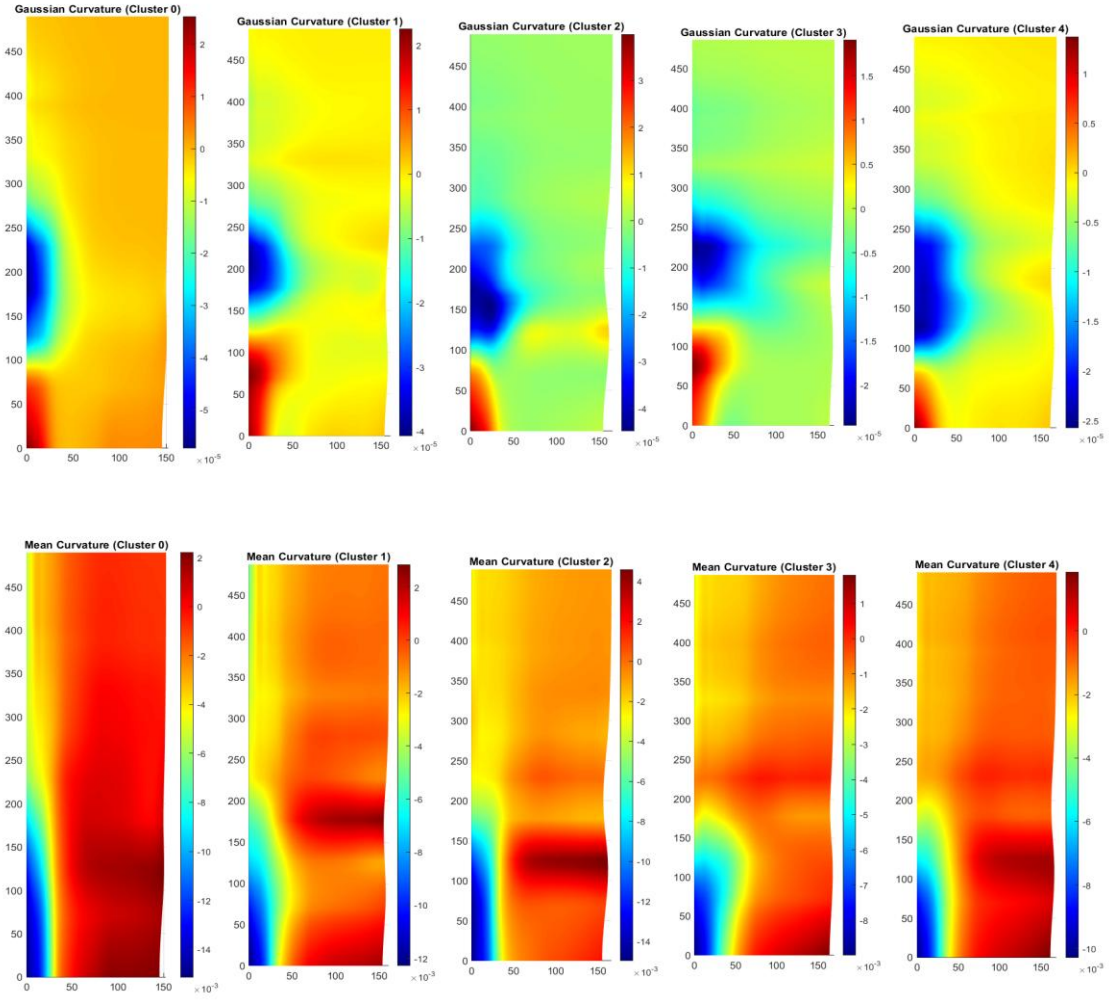

Figure 25 Gaussian (top) and mean (bottom) curvature distribution on representative surfaces for shape vector U3

We can also discern surface shape differences by analyzing gaussian and mean curvature distribution on surfaces. As shown in Fig. 25 all representative surfaces have visible variations in curvature, which hints shape wise diversity of surfaces.

Table 3 L2-Norm distance between representative surfaces for shape vector U3

	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Cluster 0	0	0.001072	0.000614	0.003725	0.001867
Cluster 1	0.001072	0	0.000162	0.001129	0.000626
Cluster 2	0.000614	0.000162	0	0.001334	0.000495
Cluster 3	0.003725	0.001129	0.001334	0	0.000534
Cluster 4	0.001867	0.000626	0.000495	0.000534	0

Another way to detect shape variation between surfaces is by calculating L2-Norm of two surfaces. It is the value of the volume between surfaces, which essentially describes the shape difference between them. As visible in Table 3 all representative surfaces distinctly vary from each other, with the only exception being representative surface of cluster 4, where it stands very close to surfaces 1, 2 and 3. Overall, these visualizations give us plenty of evidence regarding shape variation of representative surfaces.

As was mentioned earlier, the 1196 artificial surfaces that were clustered came from the 249 cross-sectional profile and only 4 longitudinal spine categories. There may be cases where surfaces with similar cross-sectional profiles were categorized into different groups solely based on their spine type. To check this, we will classify the surfaces with certain longitudinal spines separately.

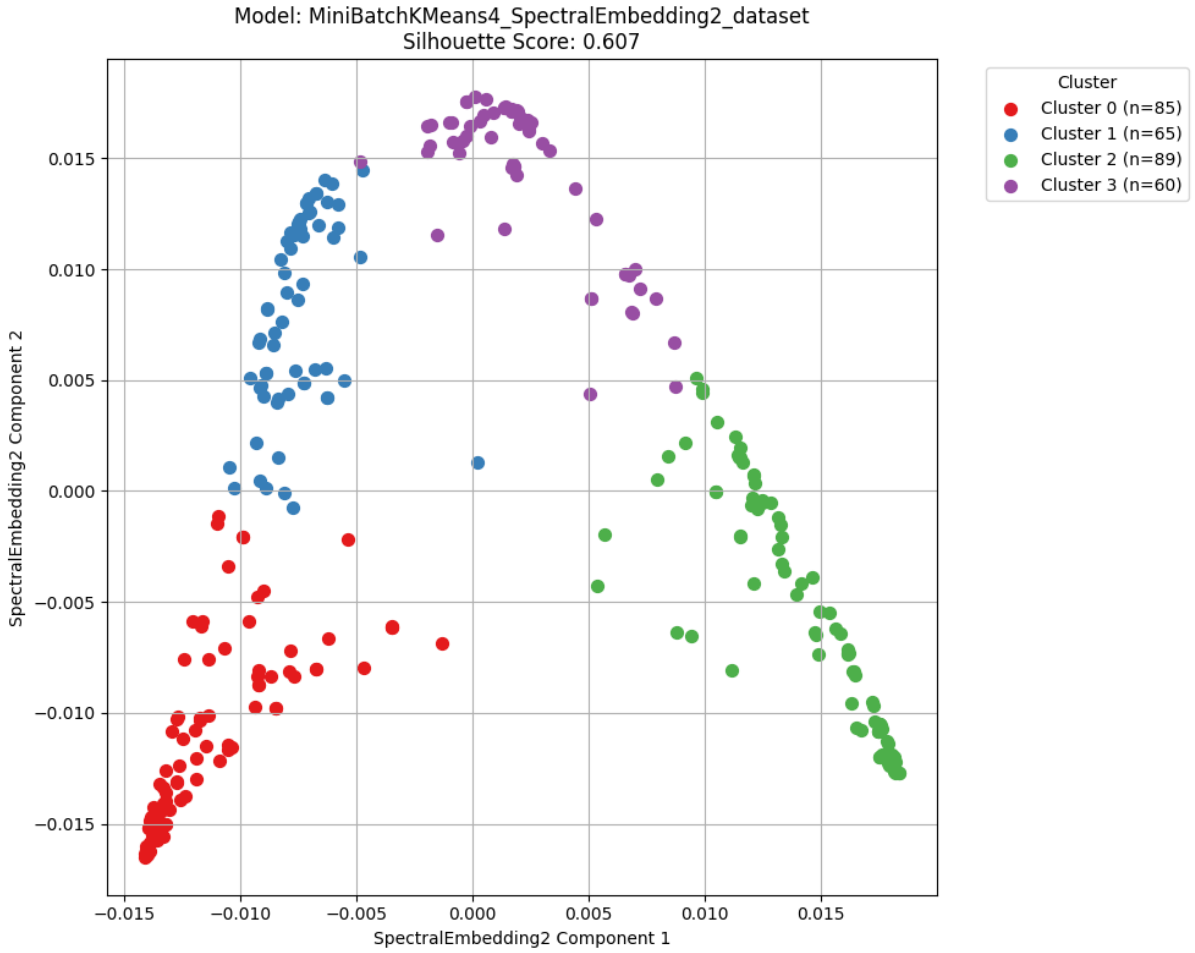

Figure 26 Scatter plot of 4 clusters based on longitudinal spine type 1 processed with MiniBatch k-means algorithm for shape vector U3

The clustering of surfaces built using the 1st longitudinal spine type are visualized in Fig. 26. The silhouette score of 0.607 indicates fair clustering, although Calinski-Harabasz index of 730.93, which is not that high if put into context with other models and Davies-Bouldin index of 0.566 suggests that clustering quality lacks in performance. Furthermore, as we can see from the scatter plot the clusters themselves are not that well defined. Considering that this particular model is the best performing compared to others and the dataset that was used in clustering contained high dimensional features like fourth order surface are moments with point coordinates the quality of clusters is suboptimal.

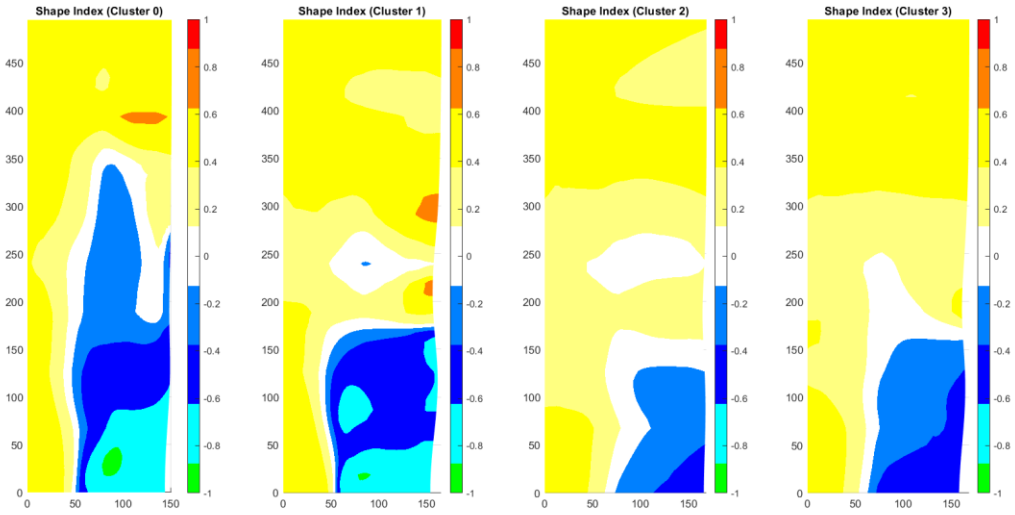

Figure 27 Shape index of cluster representatives based on longitudinal spine type 1 for shape vector U3

If we analyze shape index distribution on representative surfaces as shown in Fig. 27, we see that clusters 3 and 4 are relatively close to each other in the front part with minor differences in the middle of the surface. Apart from these two other clusters exhibit noticeable differences, especially the 2nd cluster, which is significantly distinct at the back part.

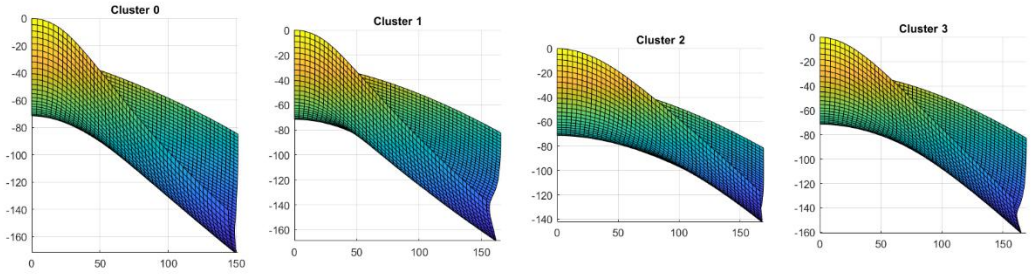

Figure 28 Front view of representative surfaces based on longitudinal spine type 1 for shape vector U3

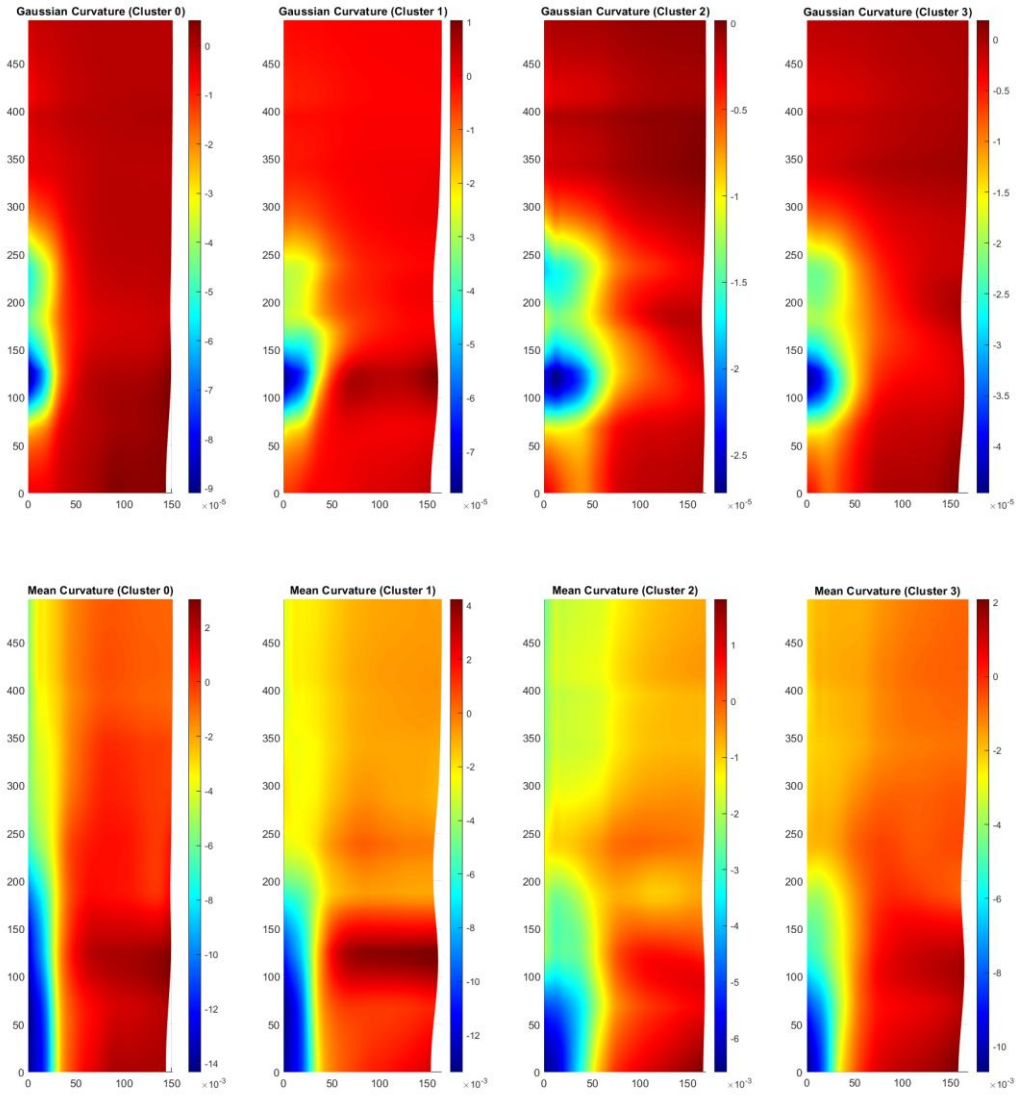

Figure 29 Gaussian (top) and mean (bottom) curvature distribution on representative surfaces based on longitudinal spine type 1 for shape vector U3

The gaussian and mean curvature distribution on representative surfaces does not indicate any noticeable differences. All clusters are very similar, apart from minor differences in terms of curvature.

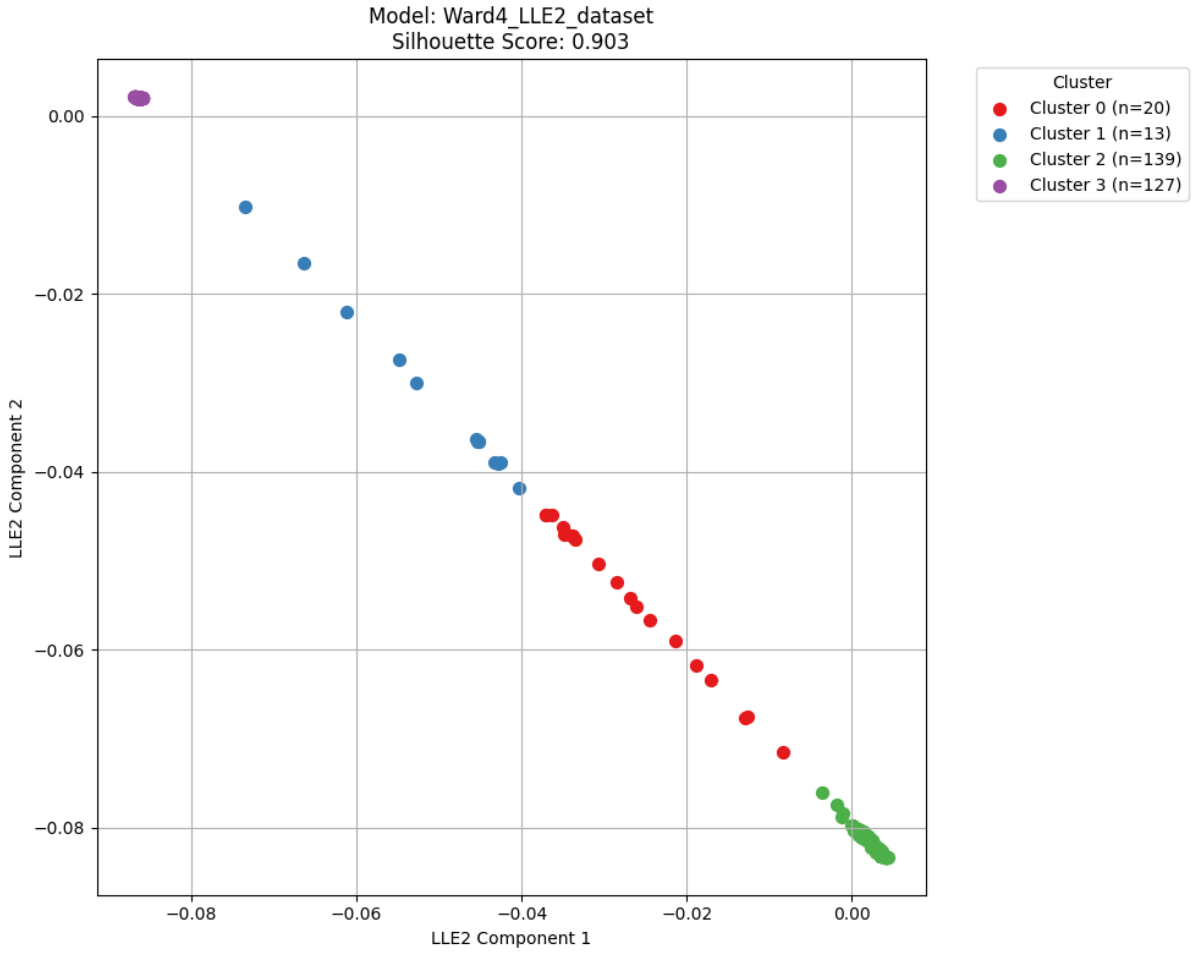

Figure 30 Scatter plot of 4 clusters based on longitudinal spine type 2 processed with MiniBatch k-means algorithm for shape vector U3

We continue by processing the surfaces based on the 2nd longitudinal spine type and this time we observe very high scores of evaluation metrics, as well as unusual shapes of scatter plots. In this case the model reached silhouette score of 0.903, Calinski-Harabasz index of 16201.24 and Davies-Bouldin index of 0.479. Silhouette score suggests that this model exhibits high clustering quality, however if we consider Calinski-Harabasz index, which is pretty average among other models and Davies-Bouldin index, then the picture doesn't look that impressive.

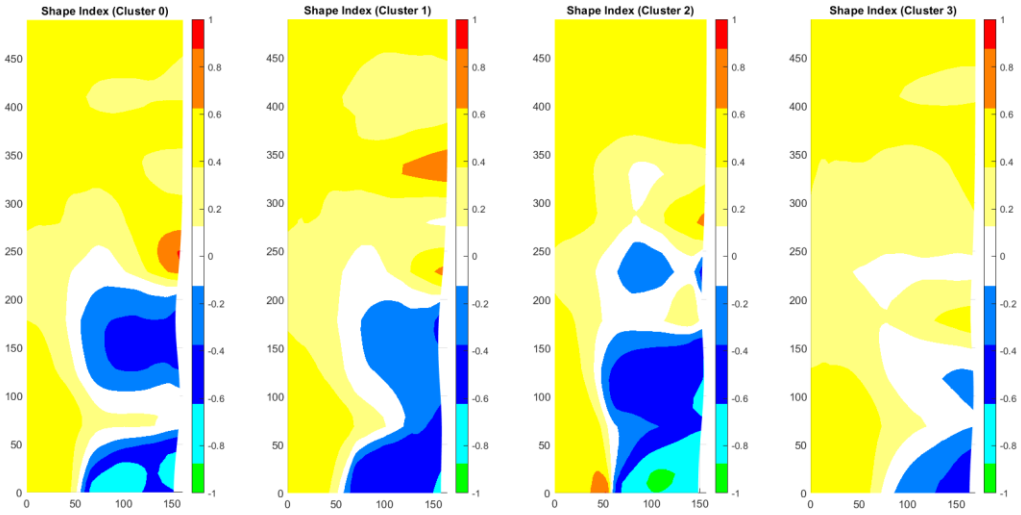

Figure 31 Shape index of cluster representatives based on longitudinal spine type 2 for shape vector U3

Shape index distribution on representative surfaces on this model demonstrates noticeable differences among all clusters with the 2nd and 3rd clusters being relatively close shape wise to each other. All clusters do not have any variability at the back part with the 4th cluster being an exception.

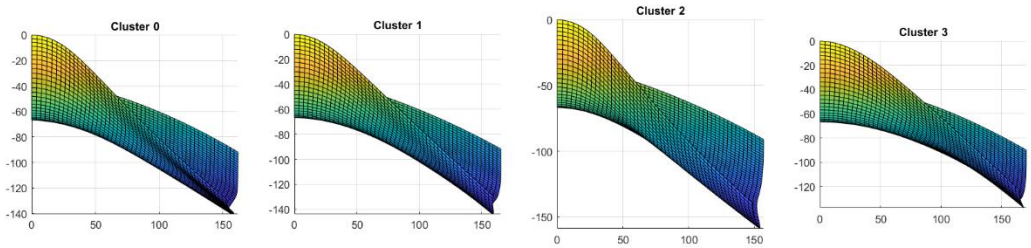

Figure 32 Front view of representative surfaces based on longitudinal spine type 2 for shape vector U3

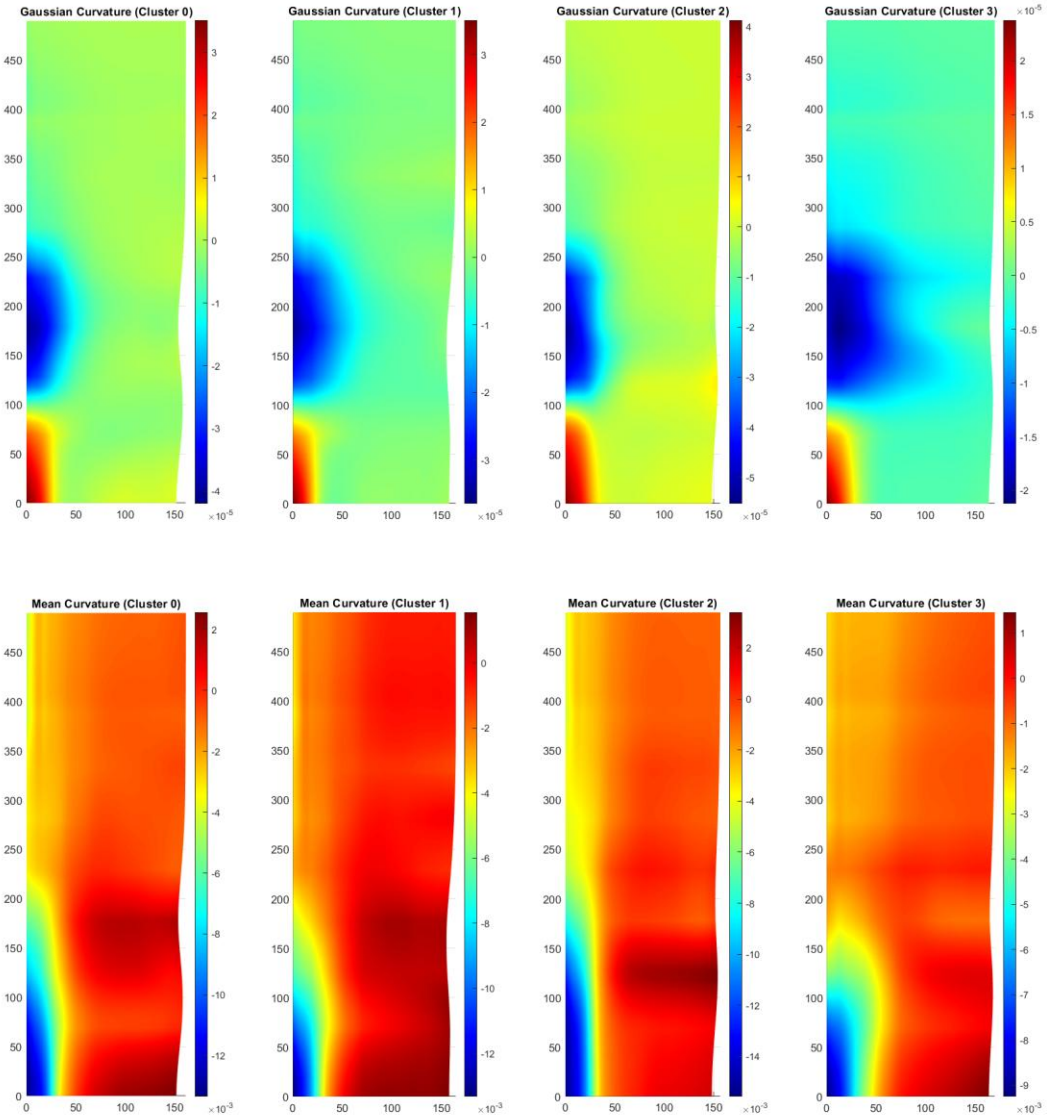

Figure 33 Gaussian (top) and mean (bottom) curvature distribution on representative surfaces based on longitudinal spine type 2 for shape vector U3

Same as Fig. 29 the Fig. 33 does not reveal any intricate variation in surface curvature distribution. All clusters are very close to each other.

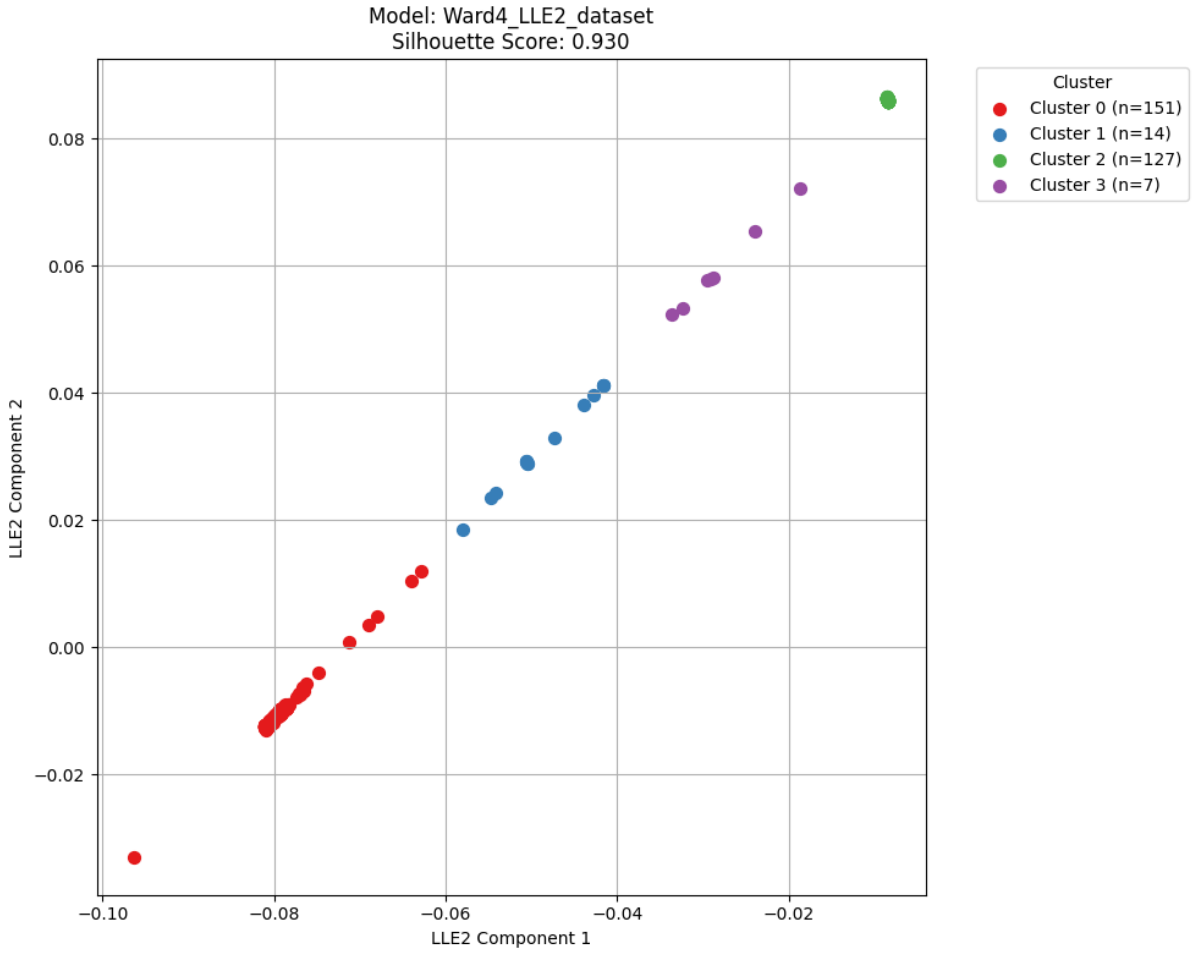

Figure 34 Scatter plot of 4 clusters based on longitudinal spine type 3 processed with MiniBatch k-means algorithm for shape vector U3

The classification model of surfaces based on longitudinal spine type 3 looks similar to the previous one. Silhouette score is equal to 0.93 with Calinski-Harabasz index of 18747.49 and Davies-Bouldin index of 0.305 indicate high clustering quality.

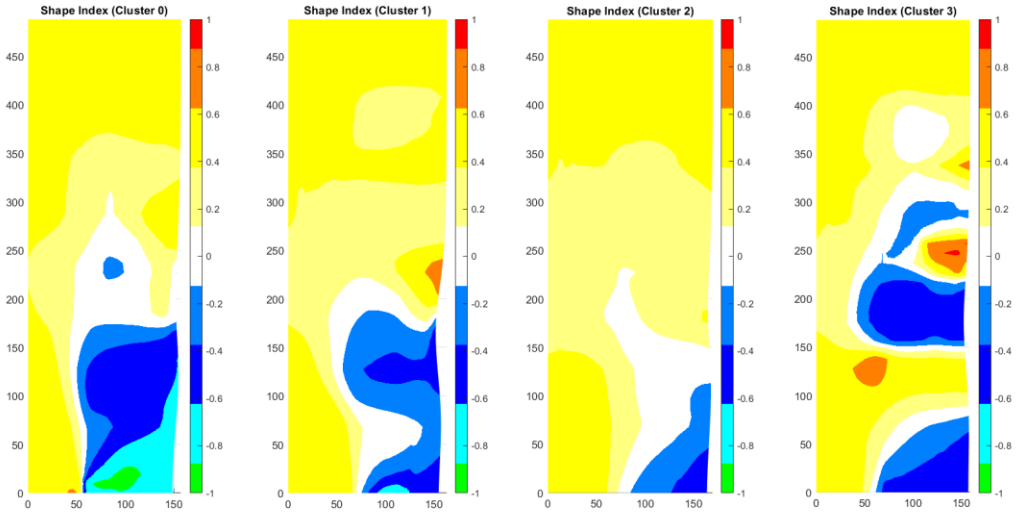

Figure 35 Shape index of cluster representatives based on longitudinal spine type 3 for shape vector U3

In Fig. 35, which displays shape index distribution on representative surfaces, we clearly see that all clusters demonstrate noticeable differences both at front and middle parts of the surface.

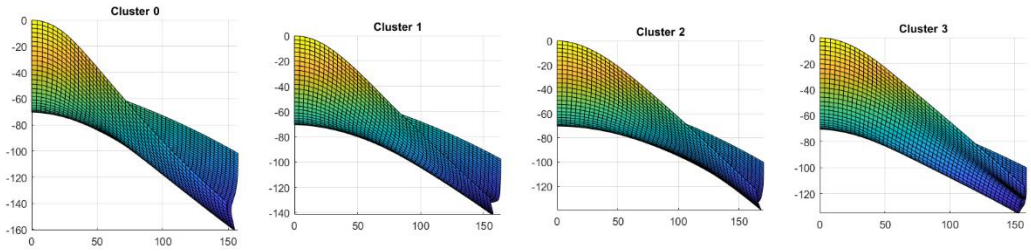

Figure 36 Front view of representative surfaces based on longitudinal spine type 3 for shape vector U3

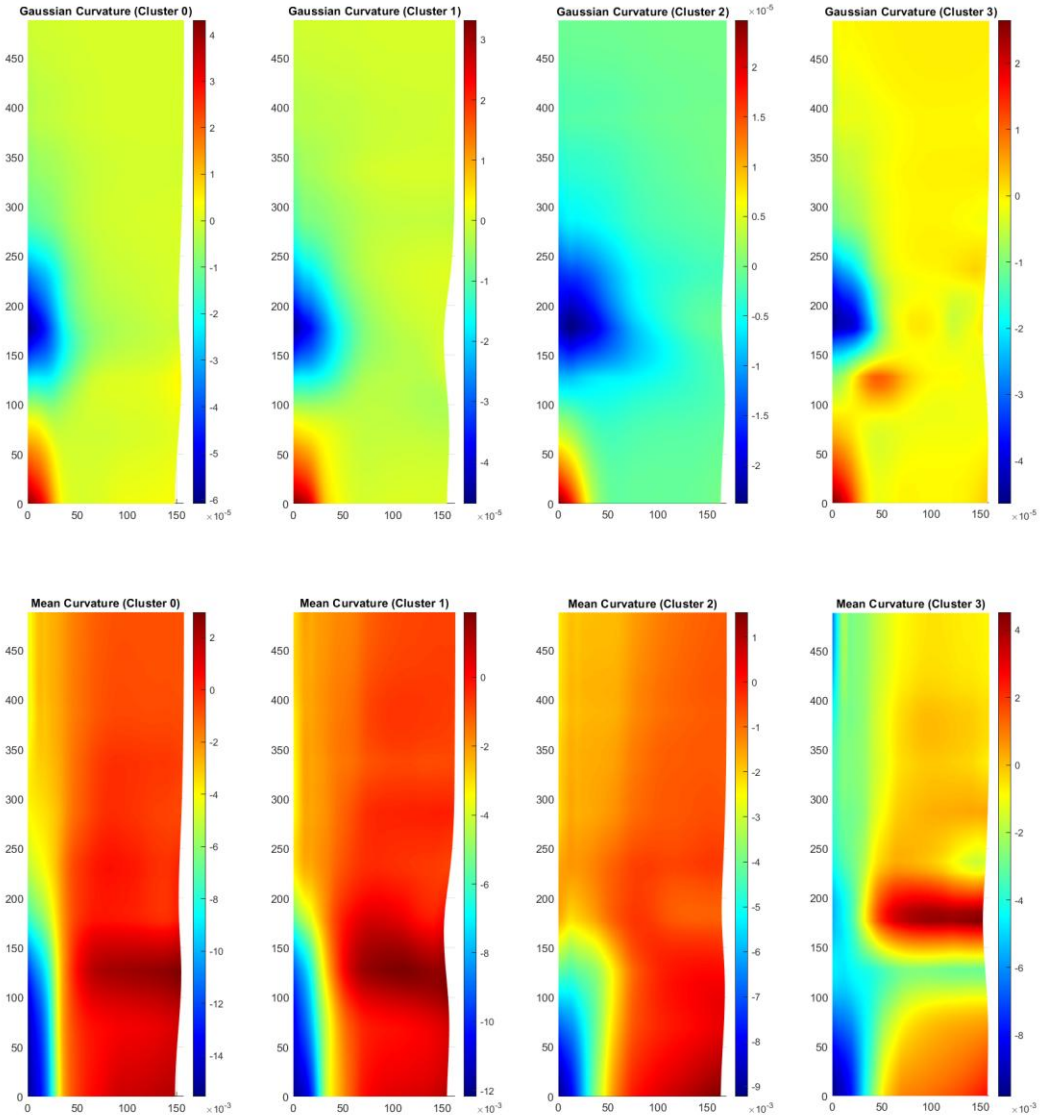

Figure 37 Gaussian (top) and mean (bottom) curvature distribution on representative surfaces based on longitudinal spine type 3 for shape vector U3

In the case of curvature distribution clusters 1 and 2 are relatively similar with cluster 3 slightly different. Cluster 4 is not akin to the other 3 surfaces.

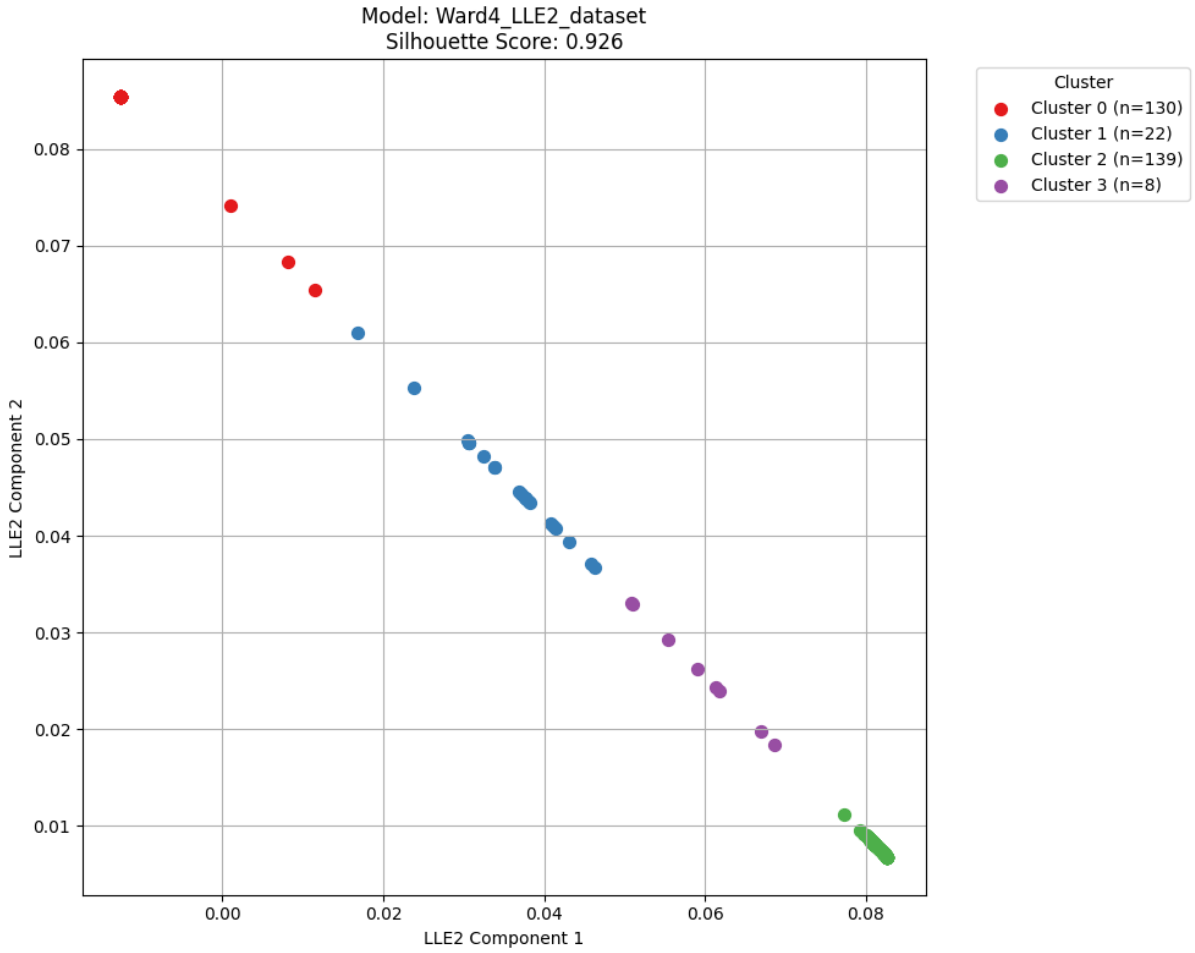

Figure 38 Scatter plot of 4 clusters based on longitudinal spine type 4 processed with MiniBatch k-means algorithm for shape vector U3

Lastly, the clusters originated from the surfaces based on longitudinal spine type 4 showcase similar results to the previous two models. Silhouette score of 0.926 hint to the high clustering quality, which is supported by Davies-Bouldin index of 0.325. Although, Calinski-Harabasz index of 22238.138 is not the best one compared to other models.

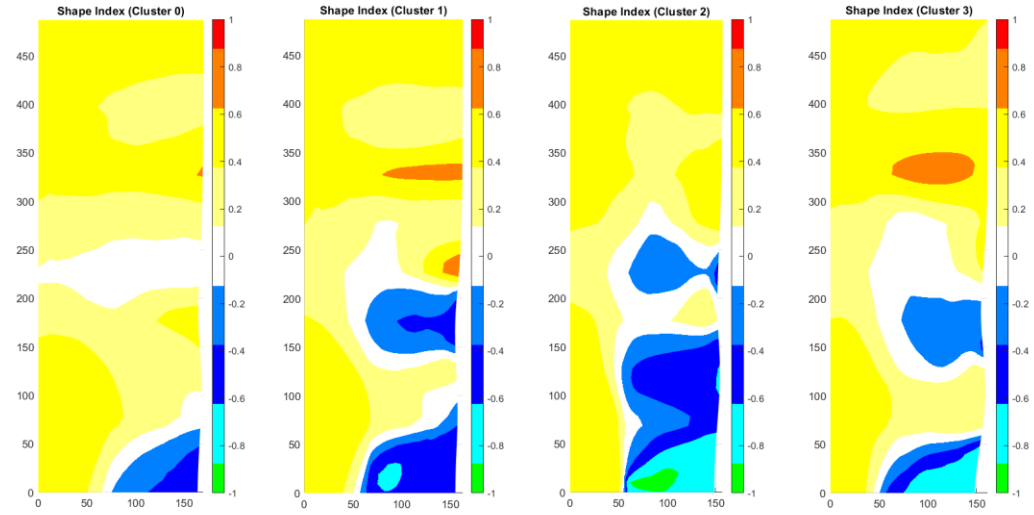

Figure 39 Shape index of cluster representatives based on longitudinal spine type 4 for shape vector U3

All representative surfaces exhibit very noticeable differences shape wise with very small similarities between clusters 2 and 4. All 4 surfaces indicate high variability on the front and middle parts of the surface.

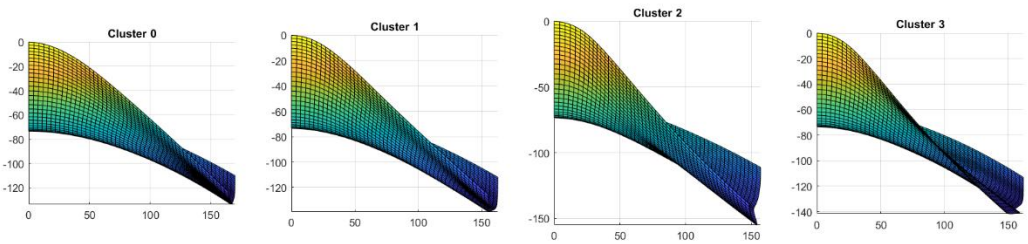

Figure 40 Front view of representative surfaces based on longitudinal spine type 4 for shape vector U3

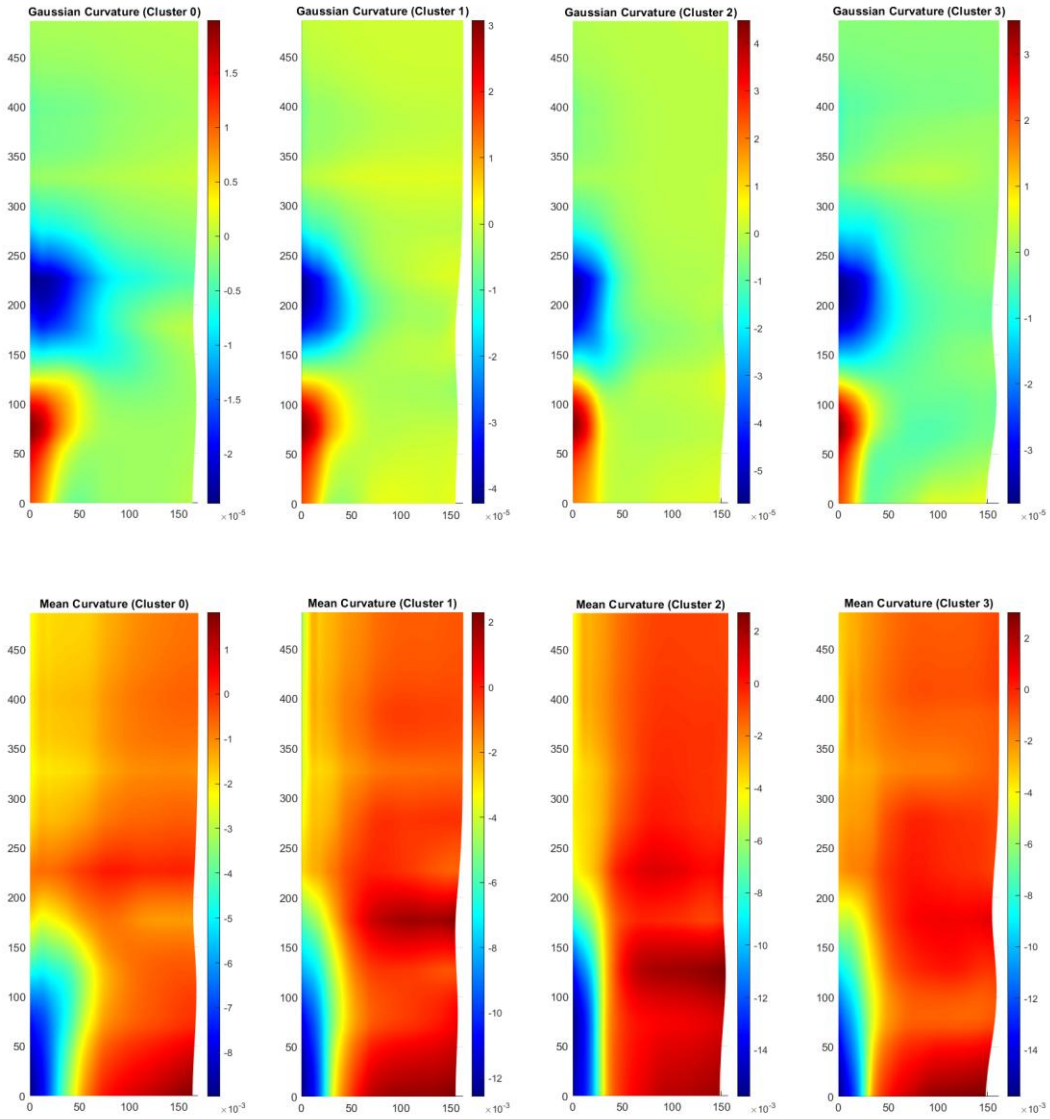

Figure 41 Gaussian (top) and mean (bottom) curvature distribution on representative surfaces based on longitudinal spine type 4 for shape vector U3

Similar to previous cases this one does not demonstrate any subtle variation in terms of curvature distribution.

In all cases, differences are mainly located at the front part of the horseback surface. There are some differentiations on the back part of the surfaces starting from longitudinal spine type 2. In all cases, there are differences between representative clusters, but in some cases two of the clusters are relatively close to each other when compared to the rest. Though for all spine types the same set of cross-sectional templates were used, there are major differences between

resulting representative surfaces across all 4 longitudinal spine types. This proves that the horse-back spine significantly influences the overall shape of the back surface.

Admittedly, 57 horse-back curves representing back spines are not enough to finalize the generation of 4 spine categories. Therefore, extraction more spines from images of horses was done.

Figure 42 Horse mask (green) extraction from the image with longitudinal spine starting from the withers point (red) ending with croup point (yellow)

Python script implementing segmentation with Meta AI's SAM model was written to identify horse silhouette and extract its mask. The side profiles of pictures vary in terms of the direction horse faces. To make all results uniform we mirrored all the pictures, so that horses faced to the left side. On the horse mask withers and croup points were identified by calculating slope direction. However, in some cases the withers and croup points are not represented accurately in the pictures. Therefore, to finalize the spine extraction process a more deliberate

approach is required. After identifying these points, we can place a net grid on the resulting horse-back spine and get the horizontal and vertical positions of any point on the curve. This will allow us to reconstruct the longitudinal spine in Rhino3D and use them to enhance our classification model.

CHAPTER 5 – CONCLUSION

The research study provides significant advancements and valuable contributions in horse-back shape classification. The research opens a new opportunity for creating universal saddle sizing system with appropriate horse-back surface categories. The project is based on scientific principles; thus, it does not require expensive equipment.

The successful implementation of machine learning techniques for surface classification highlights the potential of computational methods in saddle-fitting. Based on existing dataset 4 categories for longitudinal spine representations were identified. The clustering results for horse-back surfaces indicate 5 general groups for nearly 1200 horses, as well as 4 categories for horse-back surfaces based on each of the longitudinal curves. From the results obtained it is possible to conclude that the horse-back spine shape greatly affects the overall shape of the entire back surface. However, improving the classification model requires larger datasets with a diverse horse population. Despite its benefits, there are limitations, in particular the size of current dataset prevents us from making any conclusions about horse-back spine type categories. Future efforts should focus on collecting new datasets with diverse horse population, refining classification models, and developing standardized deployment strategies.

5.1 Future Plan

The future work involves enriching the current dataset or collecting new datasets containing horses of varying breeds and age. This will allow us to further enhance classification models and verify existing categories of horse-back surfaces and spines. The main focus should be on collecting 3D scans of horse-back surfaces, due to its undeniable advantages compared to manual approaches. Simultaneously, horse-back spine extraction from the pictures of horses should be finalized and implemented to get more samples for training and testing sets.

In the next phase of this research, for each identified surface category appropriate saddle design will be developed. By implementing both static and dynamic simulations, the suitability of saddle design will be investigated. The results from the simulations will be used to further improve the saddle designs. The goal is to create a saddle model for each of the identified categories appropriate for manufacturing. This will significantly contribute to horses' welfare by making saddle-fitting affordable and easy to use for the equestrian community.

REFERENCES

[1] L. Greve and S. J. Dyson, 'The interrelationship of lameness, saddle slip and back shape in the general sports horse population', *Equine Veterinary Journal*, vol. 46, no. 6, pp. 687–694, 2014, doi: 10.1111/evj.12222.

[2] R. Mackechnie-Guire *et al.*, 'Relationship Between Saddle and Rider Kinematics, Horse Locomotion, and Thoracolumbar Pressures in Sound Horses', *Journal of Equine Veterinary Science*, vol. 69, pp. 43–52, Oct. 2018, doi: 10.1016/j.jevs.2018.06.003.

[3] B. Riccio, C. Fraschetto, J. Villanueva, F. Cantatore, and A. Bertuglia, 'Two Multicenter Surveys on Equine Back-Pain 10 Years a Part', *Front. Vet. Sci.*, vol. 5, Aug. 2018, doi: 10.3389/fvets.2018.00195.

[4] M. Pérez-Ruiz, D. Tarrat-Martín, M. J. Sánchez-Guerrero, and M. Valera, 'Advances in horse morphometric measurements using LiDAR', *Computers and Electronics in Agriculture*, vol. 174, p. 105510, Jul. 2020, doi: 10.1016/j.compag.2020.105510.

[5] M. R. Story *et al.*, 'Equine Cervical Pain and Dysfunction: Pathology, Diagnosis and Treatment', *Animals*, vol. 11, no. 2, Art. no. 2, Feb. 2021, doi: 10.3390/ani11020422.

[6] C. Lesimple, C. Fureix, E. D. Margerie, E. Sénèque, H. Menguy, and M. Hausberger, 'Towards a Postural Indicator of Back Pain in Horses (Equus caballus)', *PLOS ONE*, vol. 7, no. 9, p. e44604, Sep. 2012, doi: 10.1371/journal.pone.0044604.

[7] L. Piegl and W. Tiller, *The NURBS Book*. Springer Science & Business Media, 2012.

[8] R. M. & Associates, 'Rhinoceros 3D', www.rhino3d.com. Available: https://www.rhino3d.com/en/asia/

[9] T. M. Kodinariya, 'Review on determining number of cluster in K-means clustering', *International Journal of Advance Research in Computer Science and Management Studies*, vol. 1. p. 90, 2013. Available: https://cir.nii.ac.jp/crid/1370584339778882466

[10] A. Y. Ng, M. I. Jordan, and Y. Weiss, 'On spectral clustering: analysis and an algorithm', in *Proceedings of the 15th International Conference on Neural Information Processing Systems: Natural and Synthetic*, in NIPS'01. Cambridge, MA, USA: MIT Press, Jan. 2001, pp. 849–856.

[11] M. R. Ackermann, J. Blömer, D. Kuntze, and C. Sohler, 'Analysis of Agglomerative Clustering', *Algorithmica*, vol. 69, no. 1, pp. 184–215, May 2014, doi: 10.1007/s00453-012-9717-4.

[12] F. Murtagh and P. Legendre, 'Ward's Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward's Criterion?', *J Classif*, vol. 31, no. 3, pp. 274–295, Oct. 2014, doi: 10.1007/s00357-014-9161-z.

[13] S. Na, L. Xumin, and G. Yong, 'Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm', in *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, Apr. 2010, pp. 63–67. doi: 10.1109/IITSI.2010.74.

[14] K. R. Shahapure and C. Nicholas, 'Cluster Quality Analysis Using Silhouette Score', in *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, Oct. 2020, pp. 747–748. doi: 10.1109/DSAA49011.2020.00096.

[15] X. Wang and Y. Xu, 'An improved index for clustering validation based on Silhouette index and Calinski-Harabasz index', *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 569, no. 5, p. 052024, Jul. 2019, doi: 10.1088/1757-899X/569/5/052024.

[16] J. Xiao, J. Lu, and X. Li, ‘Davies Bouldin Index based hierarchical initialization K-means’, *Intelligent Data Analysis*, vol. 21, no. 6, pp. 1327–1338, Nov. 2017, doi: 10.3233/IDA-163129.
[17] J. Clutton-Brock, ‘The process of domestication’, *Mammal Review*, vol. 22, no. 2, pp. 79–85, Jun. 1992, doi: 10.1111/j.1365-2907.1992.tb00122.x.
[18] C. Peham, T. Licka, H. Schobesberger, and E. Meschan, ‘Influence of the rider on the variability of the equine gait’, *Human Movement Science*, vol. 23, no. 5, pp. 663–671, Nov. 2004, doi: 10.1016/j.humov.2004.10.006.
[19] G. Tabor, ‘Veterinary physiotherapy for back pain in the horse’, *UK-Vet Equine*, vol. 6, no. 4, pp. 168–174, Jul. 2022, doi: 10.12968/ukve.2022.6.4.168.
[20] G. F. Tabor, D. J. Marlin, and J. M. Williams, ‘Use and repeatability of 3D light scanning to measure transverse dorsal profile size and symmetry in the thoracic region in horses’, *Comparative Exercise Physiology*, vol. 18, no. 5, pp. 445–451, Nov. 2022, doi: 10.3920/CEP220016.
[21] ‘WOW Saddles | Home Page’, Wow Saddles. Available: https://www.wowsaddles.com
[22] ‘Custom Saddlery - Dressage Saddles’, My Saddle. Available: https://mysaddle.com/en
[23] ‘Home | ReactorPanel Saddle Company’. Available: https://www.reactorpanel.com/
[24] A. Arapakopoulos, R. Polichshuk, Z. Segizbayev, S. Ospanov, A. I. Ginnis, and K. V. Kostas, ‘Parametric models for marine propellers’, *Ocean Engineering*, vol. 192, p. 106595, Nov. 2019, doi: 10.1016/j.oceaneng.2019.106595.
[25] F. Facchini, A. Morabito, F. Buonamici, E. Mussi, M. Servi, and Y. Volpe, ‘Autologous ear reconstruction: Towards a semiautomatic cadbased procedure for 3D printable surgical guides’, 2020, doi: 10.14733/cadaps.2021.357-367.
[26] P. Janssen and R. Stouffs, ‘Types of Parametric Modelling’, pp. 157–166, 2015, doi: 10.52842/conf.caadria.2015.157.
[27] N. C. Brown and C. T. Mueller, ‘Design variable analysis and generation for performance-based parametric modeling in architecture’, *International Journal of Architectural Computing*, vol. 17, no. 1, pp. 36–52, Mar. 2019, doi: 10.1177/1478077118799491.
[28] J. Carlos Dos Santos Júnior, J. Manoel Almeida Santos, and M. Rocha Almeida Santos, ‘Parametric modeling using the BIM methodology for the process of pathology identification in buildings’, *J Build Rehabil*, vol. 8, no. 1, p. 62, Jun. 2023, doi: 10.1007/s41024-023-00311-4.
[29] H. G. Matthies and R. Ohayon, ‘Analysis of parametric models’, *Adv Comput Math*, vol. 45, no. 5, pp. 2555–2586, Dec. 2019, doi: 10.1007/s10444-019-09735-4.
[30] E. Catmull and J. Clark, ‘Recursively generated B-spline surfaces on arbitrary topological meshes’, *Computer-Aided Design*, vol. 10, no. 6, pp. 350–355, Nov. 1978, doi: 10.1016/0010-4485(78)90110-0.
[31] T. W. Sederberg, D. L. Cardon, G. T. Finnigan, N. S. North, J. Zheng, and T. Lyche, ‘T-spline simplification and local refinement’, *ACM Trans. Graph.*, vol. 23, no. 3, pp. 276–283, Aug. 2004, doi: 10.1145/1015706.1015715.
[32] K. V. Kostas and C. Valagiannopoulos, ‘Optimally shaped nanotubes for field concentration’, *Engineering Analysis with Boundary Elements*, vol. 169, p. 106022, Dec. 2024, doi: 10.1016/j.enganabound.2024.106022.
[33] T. E. Moe, T. M. D. Pereira, F. Calvo, and J. Leenaarts, ‘Shape-based clustering of synthetic Stokes profiles using k-means and k-Shape’, *A&A*, vol. 675, p. A130, Jul. 2023, doi: 10.1051/0004-6361/202346724.

[34] E. Schubert, ‘Stop using the elbow criterion for k-means and how to choose the number of clusters instead’, *SIGKDD Explor. Newsl.*, vol. 25, no. 1, pp. 36–42, Jun. 2023, doi: 10.1145/3606274.3606278.
[35] K. P. Sinaga and M.-S. Yang, ‘Unsupervised K-Means Clustering Algorithm’, *IEEE Access*, vol. 8, pp. 80716–80727, 2020, doi: 10.1109/ACCESS.2020.2988796.
[36] D. Sculley, ‘Web-scale k-means clustering’, in *Proceedings of the 19th international conference on World wide web*, in WWW ’10. New York, NY, USA: Association for Computing Machinery, Apr. 2010, pp. 1177–1178. doi: 10.1145/1772690.1772862.
[37] U. von Luxburg, ‘A tutorial on spectral clustering’, *Stat Comput*, vol. 17, no. 4, pp. 395–416, Dec. 2007, doi: 10.1007/s11222-007-9033-z.
[38] D. Reynolds, ‘Gaussian Mixture Models’, in *Encyclopedia of Biometrics*, S. Z. Li and A. K. Jain, Eds., Boston, MA: Springer US, 2015, pp. 827–832. doi: 10.1007/978-1-4899-7488-4_196.
[39] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 2009.
[40] F. Murtagh and P. Contreras, ‘Algorithms for hierarchical clustering: An overview’, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 86–97, Dec. 2012, doi: 10.1002/widm.53.
[41] I. T. Jolliffe and J. Cadima, ‘Principal component analysis: a review and recent developments’, *Philos Trans A Math Phys Eng Sci*, vol. 374, no. 2065, p. 20150202, Apr. 2016, doi: 10.1098/rsta.2015.0202.
[42] N. Salem and S. Hussein, ‘Data dimensional reduction and principal components analysis’, *Procedia Computer Science*, vol. 163, pp. 292–299, Jan. 2019, doi: 10.1016/j.procs.2019.12.111.
[43] D. E. Slice, ‘Geometric Morphometrics’, Dec. 05, 2007, *Social Science Research Network, Rochester, NY*: 1041381.
[44] M. Hosseini, H. M. Shahrbabak, M. B. Zandi, and M. Fallahi, ‘A Morphometric Survey Among Three Iranian Horse Breeds with Multivariate Analysis’, *MP*, vol. 39, no. 3, pp. 155–160, 2016.
[45] S. T. Roweis and L. K. Saul, ‘Nonlinear Dimensionality Reduction by Locally Linear Embedding’, *Science*, vol. 290, pp. 2323–2326, Dec. 2000, doi: 10.1126/science.290.5500.2323.
[46] X. Liu, D. Tosun, M. W. Weiner, and N. Schuff, ‘Locally Linear Embedding (LLE) for MRI based Alzheimer’s Disease Classification’, *Neuroimage*, vol. 83, p. 10.1016/j.neuroimage.2013.06.033, Dec. 2013, doi: 10.1016/j.neuroimage.2013.06.033.
[47] M. Belkin and P. Niyogi, ‘Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering’, in *Advances in Neural Information Processing Systems*, MIT Press, 2001.
[48] M. Belkin and P. Niyogi, ‘Laplacian Eigenmaps for Dimensionality Reduction and Data Representation’, *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, Jun. 2003, doi: 10.1162/089976603321780317.
[49] X. Wang, F. Gu, G. Liu, and Z. Chen, ‘Application of Semantic-Based Laplacian Eigenmaps Method in 3D Model Classification and Retrieval’, in *Cloud Computing and Security*, Z. Huang, X. Sun, J. Luo, and J. Wang, Eds., Cham: Springer International Publishing, 2015, pp. 550–559. doi: 10.1007/978-3-319-27051-7_47.
[50] M. Bastico, E. Decencière, L. Corté, Y. Tillier, and D. Ryckelynck, ‘Coupled Laplacian Eigenmaps for Locally-Aware 3D Rigid Point Cloud Matching’, in *2024 IEEE/CVF*

Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2024, pp. 3447–3458. doi: 10.1109/CVPR52733.2024.00331.
[51] M. C. Hout, M. H. Papesh, and S. D. Goldinger, 'Multidimensional scaling', *WIREs Cognitive Science*, vol. 4, no. 1, pp. 93–103, 2013, doi: 10.1002/wcs.1203.
[52] H. Park, J. Seo, and ADNI, 'Application of multidimensional scaling to quantify shape in Alzheimer's disease and its correlation with Mini Mental State Examination: a feasibility study', *J Neurosci Methods*, vol. 194, no. 2, pp. 380–385, Jan. 2011, doi: 10.1016/j.jneumeth.2010.10.019.
[53] H. Park, 'ISOMAP induced manifold embedding and its application to Alzheimer's disease and mild cognitive impairment', *Neuroscience Letters*, vol. 513, no. 2, pp. 141–145, Apr. 2012, doi: 10.1016/j.neulet.2012.02.016.
[54] S. A. Brooks *et al.*, 'Morphological variation in the horse: defining complex traits of body size and shape', *Anim Genet*, vol. 41 Suppl 2, pp. 159–165, Dec. 2010, doi: 10.1111/j.1365-2052.2010.02127.x.
[55] E. T. Chu, J. J. Allen, C. L. Streeter, N. B. Sutter, and S. A. Brooks, 'Skeletal Size and Shape Diversity in the Horse', *Journal of Equine Veterinary Science*, vol. 29, no. 5, pp. 323–324, May 2009, doi: 10.1016/j.jevs.2009.04.031.
[56] E. A. Staiger, R. R. Bellone, N. B. Sutter, and S. A. Brooks, 'Morphological Variation in Gaited Horse Breeds', *Journal of Equine Veterinary Science*, vol. 43, pp. 55–65, Aug. 2016, doi: 10.1016/j.jevs.2016.04.096.
[57] G. P. T. Choi, D. Qiu, and L. M. Lui, 'Shape analysis via inconsistent surface registration', *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 476, no. 2242, p. 20200147, Oct. 2020, doi: 10.1098/rspa.2020.0147.
[58] K. Matsuyama, T. Shimizu, and T. Kato, 'Systematic Classification of Curvature and Feature Descriptor of 3D Shape and Its Application to "Complexity" Quantification Methods', *Entropy*, vol. 25, no. 4, p. 624, Apr. 2023, doi: 10.3390/e25040624.
[59] S. Tang and A. Godil, 'An evaluation of local shape descriptors for 3D shape retrieval', in *Three-Dimensional Image Processing (3DIP) and Applications II*, SPIE, Jan. 2012, pp. 217–231. doi: 10.1117/12.912153.
[60] J. J. Koenderink and A. J. van Doorn, 'Surface shape and curvature scales', *Image and Vision Computing*, vol. 10, no. 8, pp. 557–564, Oct. 1992, doi: 10.1016/0262-8856(92)90076-F.
[61] A. Kirillov *et al.*, 'Segment Anything', Apr. 05, 2023, *arXiv*: arXiv:2304.02643. doi: 10.48550/arXiv.2304.02643.
[62] S. Liu *et al.*, 'Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection', Jul. 19, 2024, *arXiv*: arXiv:2303.05499. doi: 10.48550/arXiv.2303.05499.
[63] S. Bendjebla, N. Cai, N. Anwer, S. Lavernhe, and C. Mehdi-Souzani, 'Freeform Machining Features: New Concepts and Classification', Procedia CIRP, vol. 67, pp. 482–487, Jan. 2018, doi: 10.1016/j.procir.2017.12.248.

Appendix A

The main Python scripts for Rhino3D

```
#! python3

import pandas as pd
import math
import rhinoscriptsyntax as rs
import scriptcontext as sc

import System
import System.Collections.Generic
import Rhino

from scipy import integrate
from scipy.optimize import minimize_scalar
import locale
import os
import sys
locale.setlocale(locale.LC_ALL, 'en_US')
sys.path.append(os.path.abspath(
   'C:\\Users\\Abulkhair\\.rhinocode\\py39-rh8\\Scripts'))

def fuExportObject(name, path):
   igs_file_name = f"{name}.igs"
   igs_file_path = os.path.join(path, igs_file_name)
   rs.AllObjects(True)
   rs.Command('_-Export "{}" _Enter'.format(igs_file_path), True)


def fuHorseMeasurements(file_path, sheet_name, first_point=None, last_point=None):
   # Parse measurements from Excel/CSV file and return a dictionary of horse measurements
   file_extension = os.path.splitext(file_path)[1].lower()

   try:
      if file_extension == '.xlsx' or file_extension == '.xls':
         if not sheet_name:
            raise ValueError("Sheet name is required for Excel files")
         df = pd.read_excel(file_path, sheet_name=sheet_name)
      elif file_extension == '.csv':
         df = pd.read_csv(file_path)
      else:
         raise ValueError(f"Unsupported file type: {file_extension}")

      if 'horseID' not in df.columns:
```

```
            raise ValueError("File must contain 'horseID' column")

        df.set_index('horseID', inplace=True)

        if first_point and last_point:
            if first_point not in df.columns or last_point not in df.columns:
                raise ValueError(
                    f"Columns {first_point} and/or {last_point} not found in file")
            df = df.loc[:, first_point:last_point]

        measurements_dict = df.apply(lambda x: x.tolist(), axis=1).to_dict()
        return measurements_dict

    except Exception as e:
        print(f"Error reading file {file_path}: {str(e)}")
        return None

def fuTemplateDistance(index, distance):
    return distance * index - (distance / 2 if index >= 3 else 0)

def fuImportTemplates(across, along, up_down, template_path, option):
    # Import and position IGES templates based on measurements
    arrCurves = []
    arrPoints = []

    if option == 1:
        move_vectors = [(0, -up_down[i], fuTemplateDistance(i, 50.8))
                        for i in range(len(across))]
    elif option == 2:
        delta_h = [up_down[0]]
        delta_h += [up_down[i] - up_down[i-1] for i in range(1, len(up_down))]
        delta_along = [along[0]]
        delta_along += [along[i] - along[i-1] for i in range(1, len(along))]

        horizontal_positions = [0]
        for i in range(1, len(along)):
            d = math.sqrt(delta_along[i]**2 - delta_h[i]**2)
            horizontal_positions.append(horizontal_positions[-1] + d)

        move_vectors = [(0, -up_down[i], horizontal_positions[i])
                        for i in range(len(along))]

    for i, file_name in enumerate(across):
        igs_template_path = os.path.abspath(
            os.path.join(template_path, file_name + '.igs'))
        if os.path.isfile(igs_template_path):
            try:
```

```
            success = rs.Command(
                '_-Import "{}" _Enter'.format(igs_template_path), True)
            if success:
                curve = rs.LastCreatedObjects(select=False)
                if curve:
                    rs.MoveObject(curve, move_vectors[i])
                    point = rs.AddPoint(move_vectors[i])
                    arrCurves.append(curve)
                    arrPoints.append(point)
                else:
                    print(f"No objects created from file: {igs_template_path}")
            else:
                print(f"Failed to import file: {igs_template_path}")
        except Exception as e:
            print(f"Error importing file {igs_template_path}: {e}")
    else:
        print(f"File not found: {igs_template_path}")

return arrCurves, arrPoints

def fuNormalizeKnotVectors(knots):
    min_knot = min(knots)
    max_knot = max(knots)
    return [(knot - min_knot) / (max_knot - min_knot) for knot in knots]

def fuSurfaceScaling(surface_id):
    # Scale and normalize surface to unit cube
    bbox = rs.BoundingBox(surface_id)
    if not bbox:
        return None, None

    centroid = [(bbox[0][i] + bbox[6][i])/2 for i in range(3)]
    dimensions = [
        bbox[1][0] - bbox[0][0],
        bbox[3][1] - bbox[0][1],
        bbox[4][2] - bbox[0][2]
    ]

    scale_factors = [1.0/dim if dim != 0 else 1.0 for dim in dimensions]
    scaled_surface_id = rs.ScaleObject(
        surface_id, centroid, scale_factors, copy=False)
    if not scaled_surface_id:
        return None, None

    target_centroid = [0.5, 0.5, 0.5]
    new_bbox = rs.BoundingBox(scaled_surface_id)
    if not new_bbox:
```

```
        return None, None

    new_centroid = [(new_bbox[0][i] + new_bbox[6][i])/2 for i in range(3)]
    translation_vector = [target_centroid[i] - new_centroid[i]
                          for i in range(3)]
    rs.MoveObject(scaled_surface_id, translation_vector)

    surface_knots = rs.SurfaceKnots(scaled_surface_id)
    u_knots = surface_knots[0]
    v_knots = surface_knots[1]
    u_knots_normalized = fuNormalizeKnotVectors(u_knots)
    v_knots_normalized = fuNormalizeKnotVectors(v_knots)

    degree = rs.SurfaceDegree(scaled_surface_id)
    points = rs.SurfacePointCount(scaled_surface_id)
    control_points = rs.SurfacePoints(scaled_surface_id)
    weights = rs.SurfaceWeights(scaled_surface_id)

    new_surface_id = rs.AddNurbsSurface(
        points, control_points, u_knots_normalized, v_knots_normalized, degree, weights)

    rs.DeleteObject(scaled_surface_id)
    return new_surface_id, scale_factors

def fuSurfaceDistance(surface1, surface2, intU, intV):
    # Calculate average distance between two surfaces using numerical integration
    intU = intU + 1 if intU % 2 == 0 else intU
    intV = intV + 1 if intV % 2 == 0 else intV
    hu = 1.0 / intU
    hv = 1.0 / intV

    def integrate_surfaces(surface_from, surface_to):
        sum_out = 0
        for i in range(0, intU - 1, 3):
            val_out = [0] * 3
            for k in range(3):
                u = (i + k) * hu
                sum_in = 0
                for j in range(0, intV - 1, 3):
                    val_in = [0] * 3
                    for l in range(3):
                        v = (j + l) * hv
                        point1 = rs.SurfaceEvaluate(surface_from, [u, v], 1)
                        closest_param = rs.SurfaceClosestPoint(
                            surface_to, point1[0])
                        point2 = rs.EvaluateSurface(
                            surface_to, closest_param[0], closest_param[1])
```

```
                distance = rs.Distance(point1[0], point2) ** 2
                surface_metric = rs.VectorLength(
                    rs.VectorCrossProduct(point1[1], point1[2]))
                val_in[l] = distance * surface_metric
            sum_in += val_in[0] + 4 * val_in[1] + val_in[2]
        val_out[k] = hv / 3 * sum_in
    sum_out += val_out[0] + 4 * val_out[1] + val_out[2]
  return sum_out * hu / 3

sum_out1 = integrate_surfaces(surface1, surface2)
sum_out2 = integrate_surfaces(surface2, surface1)
avg_distance = (sum_out1 + sum_out2) / 2
print(f"Average Distance: {avg_distance}")
return avg_distance

def fuUniformSpacingParametricDomain(surface, intU, intV):
  # Generate uniform point distribution in parametric domain with curvature analysis
  arrU = rs.SurfaceDomain(surface, 0)
  arrV = rs.SurfaceDomain(surface, 1)

  point_x, point_y, point_z = [], [], []
  gauss, mean, shape_indices = [], [], []

  for i in range(intU):
    for j in range(intV):
      u = arrU[0] + i * (arrU[1] - arrU[0]) / (intU - 1)
      v = arrV[0] + j * (arrV[1] - arrV[0]) / (intV - 1)

      point = rs.EvaluateSurface(surface, u, v)
      if point is None:
        continue
      curvature = rs.SurfaceCurvature(surface, (u, v))
      if curvature is None:
        continue

      k1 = curvature[2]
      k2 = curvature[4]
      shape_index = 2 / math.pi * math.atan((k2 + k1) / (k2 - k1))

      point_x.append(point[0])
      point_y.append(point[1])
      point_z.append(point[2])
      gauss.append(curvature[6])
      mean.append(curvature[7])
      shape_indices.append(shape_index)

  return point_x, point_y, point_z, gauss, mean, shape_indices
```

```
def fuUniformSpacingPhysicalDomain(surface, intU, intV):
    # Generate uniform point distribution in physical domain with curvature analysis
    point_x, point_y, point_z = [], [], []
    gauss, mean, shape_indices = [], [], []

    longitudinal_curve = rs.ExtractIsoCurve(surface, [0, 0], 1)
    if longitudinal_curve is None:
        return point_x, point_y, point_z, gauss, mean, shape_indices

    pointsU = rs.DivideCurve(longitudinal_curve, intU-1, False, True)
    if pointsU is None:
        return point_x, point_y, point_z, gauss, mean, shape_indices

    paramU = []
    for point in pointsU:
        param = rs.CurveClosestPoint(longitudinal_curve, point)
        paramU.append(param)

    cross_sectional_curves = [rs.ExtractIsoCurve(
        surface, [0, param], 0) for param in paramU]
    for curve in cross_sectional_curves:
        pointsV = rs.DivideCurve(curve, intV-1, False, True)
        if pointsV is None:
            continue

        for point in pointsV:
            point_x.append(point[0])
            point_y.append(point[1])
            point_z.append(point[2])

            param = rs.SurfaceClosestPoint(surface, point)
            curvature = rs.SurfaceCurvature(surface, param)
            if curvature is None:
                continue

            gauss.append(curvature[6])
            mean.append(curvature[7])

            k1 = curvature[2]
            k2 = curvature[4]
            shape_index = 2 / math.pi * math.atan((k2 + k1) / (k2 - k1))
            shape_indices.append(shape_index)

    return point_x, point_y, point_z, gauss, mean, shape_indices

def gauss_curvature_integral(surface, u_bounds, v_bounds, absolute=True, n=20):
```

```
    def f_gauss(u, v):
        curvature_data = rs.SurfaceCurvature(surface, [u, v])
        if curvature_data is None:
            return 0.0
        gauss_curvature = curvature_data[6]
        return abs(gauss_curvature) if absolute else gauss_curvature

    def simpsons_double_integral(f, ax, bx, ay, by, nx, ny):
        if nx % 2 != 0 or ny % 2 != 0:
            raise ValueError("Number of intervals must be even")

        hx = (bx - ax) / nx
        hy = (by - ay) / ny

        x = np.linspace(ax, bx, nx + 1)
        y = np.linspace(ay, by, ny + 1)

        f_xy = np.array([[f(xi, yj) for yj in y] for xi in x])

        sx = np.array([1] + [4, 2] * (nx // 2 - 1) + [4, 1])
        sy = np.array([1] + [4, 2] * (ny // 2 - 1) + [4, 1])

        return (hx * hy / 9) * np.sum(np.outer(sx, sy) * f_xy)

    return simpsons_double_integral(f_gauss, u_bounds[0], u_bounds[1], v_bounds[0],
v_bounds[1], n, n)

def split_surface(surface, n_sub=6, eps=1e-3):
    u_range = surface.Domain(0)
    v_range = surface.Domain(1)

    nodes = [[] for _ in range(n_sub + 1)]

    def Node(): return {"left": None, "right": None,
                        "val_u": None, "val_v": None}

    nodes[0] = [Node()]
    nodes[0][0]["val_u"] = [u_range.T0, u_range.T1]
    nodes[0][0]["val_v"] = [v_range.T0, v_range.T1]

    for i in range(n_sub):
        for j, node in enumerate(nodes[i]):
            a, b = node["val_u"]
            c, d = node["val_v"]

            if i % 2 == 0:
                def f2(x):
```

```
                return abs(gauss_curvature_integral(surface, [a, x], [c, d], 10) -
                           gauss_curvature_integral(surface, [x, b], [c, d], 10))
            x = minimize_scalar(f2, bounds=(a, b), method='bounded').x
            nodes[i+1].extend([
                {"val_u": [a, x], "val_v": [c, d]},
                {"val_u": [x, b], "val_v": [c, d]}
            ])
        else:
            def f1(x):
                return abs(gauss_curvature_integral(surface, [a, b], [c, x], 10) -
                           gauss_curvature_integral(surface, [a, b], [x, d], 10))
            x = minimize_scalar(f1, bounds=(c, d), method='bounded').x
            nodes[i+1].extend([
                {"val_u": [a, b], "val_v": [c, x]},
                {"val_u": [a, b], "val_v": [x, d]}
            ])

    return nodes[-1]

def extract_surface_points(surface, patches, points_per_patch=4):
    points = []
    for patch in patches:
        u1, u2 = patch["val_u"]
        v1, v2 = patch["val_v"]
        u_step = (u2 - u1) / (points_per_patch - 1)
        v_step = (v2 - v1) / (points_per_patch - 1)

        for i in range(points_per_patch):
            u = u1 + i * u_step
            for j in range(points_per_patch):
                v = v1 + j * v_step
                point = surface.PointAt(u, v)
                if point:
                    points.append([point.X, point.Y, point.Z])

    return points

def calculate_point_count(n_sub, points_per_patch):
    num_patches = 2 ** n_sub
    points_per_patch_total = points_per_patch * points_per_patch
    total_points = num_patches * points_per_patch_total
    return num_patches, points_per_patch_total, total_points

def process_surface(surface, n_sub, points_per_patch):
    try:
        surface_obj = rs.coercesurface(surface)
        if not surface_obj:
```

```
            return None

        patches = split_surface(surface_obj, n_sub)
        points = extract_surface_points(surface_obj, patches, points_per_patch)
        return points
    except Exception as e:
        print(f"Error processing surface: {str(e)}")
        return None

def fuSurfaceThirdMoments(surface):
    try:
        # Calculate centroid first
        centroid = fuCalculateCentroid(surface)
        if not centroid:
            return None

        # Get surface domain
        u_domain = rs.SurfaceDomain(surface, 0)
        v_domain = rs.SurfaceDomain(surface, 1)
        if not u_domain or not v_domain:
            return None

        # Initialize moments array [XXX, XXY, XXZ, XYY, XYZ, XZZ, YYY, YYZ, YZZ, ZZZ]
        moments = [0] * 10
        indices = [(0, 0, 0), (0, 0, 1), (0, 0, 2), (0, 1, 1), (0, 1, 2), (0, 2, 2),
                   (1, 1, 1), (1, 1, 2), (1, 2, 2), (2, 2, 2)]

        # Simpson's rule parameters
        nu = nv = 50  # number of intervals
        hu = (u_domain[1] - u_domain[0]) / nu
        hv = (v_domain[1] - v_domain[0]) / nv

        # Double integration using Simpson's rule
        for i in range(nu + 1):
            for j in range(nv + 1):
                u = u_domain[0] + i * hu
                v = v_domain[0] + j * hv

                # Get point and derivatives
                point, derivatives = fuSurfaceDerivatives(surface, u, v)
                if not point or not derivatives:
                    continue

                # Calculate surface element
                surface_element = rs.VectorLength(
                    rs.VectorCrossProduct(derivatives[0], derivatives[1]))
```

```
            # Weight for Simpson's rule
            wu = 1 if i == 0 or i == nu else (2 if i % 2 == 0 else 4)
            wv = 1 if j == 0 or j == nv else (2 if j % 2 == 0 else 4)
            w = wu * wv

            # Calculate moments about centroid
            for k, idx in enumerate(indices):
                moments[k] += w * (point[idx[0]] - centroid[idx[0]]) * \
                    (point[idx[1]] - centroid[idx[1]]) * \
                    (point[idx[2]] - centroid[idx[2]]) * \
                    surface_element

        # Apply Simpson's rule coefficients
        coef = hu * hv / 9
        moments = [m * coef for m in moments]

        return moments

    except Exception as e:
        print(f"Error calculating third-order moments: {str(e)}")
        return None

def fuSurfaceFourthMoments(surface):
    try:
        # Calculate centroid first
        centroid = fuCalculateCentroid(surface)
        if not centroid:
            return None

        # Get surface domain
        u_domain = rs.SurfaceDomain(surface, 0)
        v_domain = rs.SurfaceDomain(surface, 1)
        if not u_domain or not v_domain:
            return None

        # Initialize moments array [XXXX, XXXY, XXXZ, XXYY, XXYZ, XXZZ, XYYY, XYYZ,
XYZZ, XZZZ,
        #                           YYYY, YYYZ, YYZZ, YZZZ, ZZZZ]
        moments = [0] * 15
        indices = [(0, 0, 0, 0), (0, 0, 0, 1), (0, 0, 0, 2), (0, 0, 1, 1), (0, 0, 1, 2), (0, 0, 2, 2),
                   (0, 1, 1, 1), (0, 1, 1, 2), (0, 1, 2, 2), (0, 2, 2, 2),
                   (1, 1, 1, 1), (1, 1, 1, 2), (1, 1, 2, 2), (1, 2, 2, 2), (2, 2, 2, 2)]

        # Simpson's rule parameters
        nu = nv = 50  # number of intervals
        hu = (u_domain[1] - u_domain[0]) / nu
        hv = (v_domain[1] - v_domain[0]) / nv
```

```
    # Double integration using Simpson's rule
    for i in range(nu + 1):
        for j in range(nv + 1):
            u = u_domain[0] + i * hu
            v = v_domain[0] + j * hv

            # Get point and derivatives
            point, derivatives = fuSurfaceDerivatives(surface, u, v)
            if not point or not derivatives:
                continue

            # Calculate surface element
            surface_element = rs.VectorLength(
                rs.VectorCrossProduct(derivatives[0], derivatives[1]))

            # Weight for Simpson's rule
            wu = 1 if i == 0 or i == nu else (2 if i % 2 == 0 else 4)
            wv = 1 if j == 0 or j == nv else (2 if j % 2 == 0 else 4)
            w = wu * wv

            # Calculate moments about centroid
            for k, idx in enumerate(indices):
                moments[k] += w * (point[idx[0]] - centroid[idx[0]]) * \
                    (point[idx[1]] - centroid[idx[1]]) * \
                    (point[idx[2]] - centroid[idx[2]]) * \
                    (point[idx[3]] - centroid[idx[3]]) * \
                    surface_element

    # Apply Simpson's rule coefficients
    coef = hu * hv / 9
    moments = [m * coef for m in moments]

    return moments

except Exception as e:
    print(f"Error calculating fourth-order moments: {str(e)}")
    return None
```

Appendix B

The Machine Learning script implemented in Python

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

import time
from itertools import cycle, islice

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import Normalizer

from sklearn import cluster, mixture
from sklearn.cluster import KMeans, MeanShift, MiniBatchKMeans, AffinityPropagation,
SpectralClustering, AgglomerativeClustering, DBSCAN, OPTICS, Birch
from sklearn.mixture import GaussianMixture
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
from sklearn.metrics import calinski_harabasz_score
from sklearn.metrics import davies_bouldin_score

from sklearn.neighbors import kneighbors_graph
from sklearn.manifold import Isomap, LocallyLinearEmbedding, MDS, TSNE,
SpectralEmbedding
```

```
def fuDimRedCluster(datasets, n_clusters_list, names_df, dataset_names, random_state=42):
    dim_red_methods = {
        'Isomap': Isomap(n_components=2),
        'SpectralEmbedding': SpectralEmbedding(n_components=2,
random_state=random_state),
        'LLE': LocallyLinearEmbedding(n_components=2, random_state=random_state,
eigen_solver='auto'),
        'MDS': MDS(n_components=2, random_state=random_state),
        'PCA': PCA(n_components=2, random_state=random_state),
        'TSNE': TSNE(n_components=2, random_state=random_state),
    }


    unique_results = []
    labels_results = []

    for dataset_idx, dataset in enumerate(datasets):
        dataset_name = dataset_names[dataset_idx]

        for dim_red_name, dim_reducer in dim_red_methods.items():
            try:
                data_transformed = dim_reducer.fit_transform(dataset)

                connectivity = kneighbors_graph(
                    data_transformed,
                    n_neighbors=7,
                    include_self=False
                )
                connectivity = 0.5 * (connectivity + connectivity.T)

                for n_clusters in n_clusters_list:
                    clustering_algorithms = {
```

```
"MiniBatchKMeans": MiniBatchKMeans(
    n_clusters=n_clusters,
    random_state=random_state
),
"SpectralClustering": SpectralClustering(
    n_clusters=n_clusters,
    eigen_solver="arpack",
    affinity="nearest_neighbors",
    random_state=random_state
),
"AgglomerativeClustering": AgglomerativeClustering(
    linkage="average",
    metric="cityblock",
    n_clusters=n_clusters,
    connectivity=connectivity
),
"Ward": AgglomerativeClustering(
    n_clusters=n_clusters,
    linkage="ward",
    connectivity=connectivity
),
"OPTICS": OPTICS(
    min_samples=7,
    xi=0.05,
    min_cluster_size=0.1
),
"Birch": Birch(n_clusters=n_clusters),
"GaussianMixture": GaussianMixture(
    n_components=n_clusters,
    covariance_type="full",
    random_state=random_state
)
```

```
        }

        for clust_name, algorithm in clustering_algorithms.items():
            try:
                algorithm.fit(data_transformed)

                if hasattr(algorithm, "labels_"):
                    y_pred = algorithm.labels_.astype(int)
                else:
                    y_pred = algorithm.predict(data_transformed)

                if len(set(y_pred)) > 1:
                    silhouette = silhouette_score(data_transformed, y_pred)
                    calinski_harabasz = calinski_harabasz_score(data_transformed, y_pred)
                    davies_bouldin = davies_bouldin_score(data_transformed, y_pred)
                else:
                    silhouette = np.nan
                    calinski_harabasz = np.nan
                    davies_bouldin = np.nan

                dunn = dunn_index(data_transformed, y_pred)

                model_name =
f"{clust_name}{n_clusters}_{dim_red_name}2_{dataset_name}"

                unique_results.append({
                    "model_name": model_name,
                    "n_clusters": n_clusters,
                    "clustering_algorithm": clust_name,
                    "silhouette_score": silhouette,
                    "calinski_harabasz_score": calinski_harabasz,
                    "davies_bouldin_index": davies_bouldin,
```

```
                    #"dunn_index": dunn
                })

                for idx, label in enumerate(y_pred):
                    labels_results.append({
                        "model_name": model_name,
                        "n_clusters": n_clusters,
                        "clustering_algorithm": clust_name,
                        "horseID": names_df.iloc[idx]['horseID'],
                        "referenceCluster": label,
                        "silhouette_score": silhouette,
                        "calinski_harabasz_score": calinski_harabasz,
                        "davies_bouldin_index": davies_bouldin,
                        #"dunn_index": dunn
                    })

            except Exception as e:
                print(f"Clustering failed for {clust_name} on {dataset_name}: {str(e)}")

        except Exception as e:
            print(f"Dimension reduction failed for {dim_red_name} on {dataset_name}:
{str(e)}")

    clusterModelsUniqueDF = pd.DataFrame(unique_results)
    clusterModelsLabelsDF = pd.DataFrame(labels_results)

    return clusterModelsUniqueDF, clusterModelsLabelsDF
```