

Add knit

Documentation



Table of Contents

[Table of Figures 2](#_Toc158125156)

[1. Introduction 3](#_Toc158125157)

[2. Mesh pooling using ML for segmentation. 3](#_Toc158125158)

[3. Segmentation using Reeb graph. 5](#_Toc158125159)

[Reeb graph: 6](#_Toc158125160)

[Laplacian Matrix: 6](#_Toc158125161)

[Laplacian Smoothing: 7](#_Toc158125162)

[Laplacian Contraction 7](#_Toc158125163)

[Reeb Graph Construction: 7](#_Toc158125164)

[Implementation 7](#_Toc158125165)

[4. Mapper Algorithm for segmentation 13](#_Toc158125166)

[5. Time values, and distances criterion for segmentation: 14](#_Toc158125167)

[6. Time Variance based segmentation. 22](#_Toc158125168)

[7. References: 28](#_Toc158125169)

# Table of Figures

[Figure 1:Example of a 3D structure with undulation and branches (Bent cylinder) 8](#_Toc158125118)

[Figure 2:Implementation of Laplacian contraction on bent cylinder. 8](#_Toc158125119)

[Figure 3: Laplacian Contraction (shown in yellow) after 5 iterations 9](#_Toc158125120)

[Figure 4:Contraction by volume after every iteration 9](#_Toc158125121)

[Figure 5: Skeleton structure of the bent cylinder 10](#_Toc158125122)

[Figure 6: Critical points for segmentation: 11](#_Toc158125123)

[Figure 7:Segmentation lines based on derivative of change in angle 11](#_Toc158125124)

[Figure 8:Time complexity of Laplacian Contraction 13](#_Toc158125125)

[Figure 9: Pyramid and hemi sphere the skeleton doesn’t retain much about the geometry 13](#_Toc158125126)

[Figure 10: Segmentation of bent cylinder using Time & Distance Scalar Fields 15](#_Toc158125127)

[Figure 11: Row building in Bent cylinder 16](#_Toc158125128)

[Figure 12: Row building in Bent cylinder 17](#_Toc158125129)

[Figure 13: Short row building in Bent cylinder 18](#_Toc158125130)

[Figure 14: Time-Distance segmentation algorithm 20](#_Toc158125131)

[Figure 15:Fig: Implementation of the above-mentioned algorithm with w=10 and h=5 21](#_Toc158125132)

[Figure 16:Visualization 21](#_Toc158125133)

[Figure 17:Fig: Implementation of the above-mentioned algorithm with w=0.96 and h=0.42 22](#_Toc158125134)

[Figure 18: Clustering of time values based on thier density 23](#_Toc158125135)

[Figure 19: Formation of full rows and short rows 24](#_Toc158125136)

[Figure 20: Implantation on Bent Cylinder: using time clusters to form full rows 25](#_Toc158125137)

[Figure 21: Un clustered points in bent cylinder 27](#_Toc158125138)

[Figure 22: Time value clustering and overlapping regions due to non-linearity 28](#_Toc158125139)

# Introduction

In the process of Knitting 3D structures, there are multiple steps involved, starting from the 3D representation of the 3D structure to generating a file that is understandable by the knitting machines. One of the critical steps involved in this process is segmentation of the 3D object. Meaning, slicing the structure into smaller structures which when combined gives the most accurate resemblance of the original structure.

Here explained are multiple methods which aid in the process of the segmentation; namely

1. Mesh pooling using ML for segmentation.
2. Mapper algorithm for segmentation.
3. Segmentation using Reeb Graph.
4. Time Value and distance criterion for segmentation
5. Time Variance and distance criterion for segmentation.

# Mesh pooling using ML for segmentation.

Understanding the semantic properties of 3D structures is pivotal for effective segmentation of a given 3D structure. Consequently, employing a mesh pooling method that involves down sampling a 3D mesh while preserving essential features. Mesh pooling identifies irregular structures at corners and edges, signifying the 3D object's crucial points. These irregular structures serve as significant features, indicating critical points on 3D objects. The Edge Collapse Method is utilized to identify these critical points by eliminating less informative features, such as planar and monotonous surfaces on the irregular 3D object. Edge collapse operations helps in simplifying the mesh by removing edges, vertices, and faces, resulting in a coarser and more abstract representation of the 3D structure.

To identify such irregular structures in corners and edges, conducting a geometric or topological analysis would be beneficial. These irregular structures may be found in areas with high curvature or where surface properties change significantly. Down-sampling assists in this identification by helping preserve irregularities that contribute to the overall structure. After down-sampling, we perform pooling or decimation, reducing the number of vertices and faces while retaining essential features.

The down-sampling is performed by implementing Edge Collapse algorithm, specifically using the Quadric Error Metric (QEM). This algorithm is known for its ability to preserve geometric details, ensuring the maintenance of corners and edges while removing planar and less informative regions. It is then fine-tuned on down-sampling parameters and criteria based on the desired level of simplification and feature preservation.

Here the Quadric Error Metric (QEM) evaluates and prioritizes edges for collapse, considering factors like geometric error or simplification cost. This involves establishing error metrics to guide the edge collapse process, measuring how much the simplified surface deviates from the original.

In the Quadric Error Metric (QEM) implementation process, achieving optimal results requires a meticulous and iterative parameter-tuning phase. Experimentation with various parameters, such as thresholds for detecting irregular structures and criteria for edge collapse, is necessary to strike a balance aligned with desired outcomes. This ensures effective identification of corners and edges in a 3D mesh.

Considering computational efficiency, especially with large-scale meshes, optimizing algorithms for speed becomes imperative. As the QEM method is inherently iterative, efficient algorithms are necessary for timely processing. Despite computational challenges, creating analytical or mathematical models for generic skeleton extraction in 3D meshes proves exceedingly difficult. This limitation prompts the adoption of a machine learning (ML)-based approach for extracting 3D object skeletons and critical points, aiding in segmentation with mesh pooling.

An ML-based approach harnesses neural networks' capabilities to autonomously learn intricate patterns and relationships within 3D mesh data. This approach offers a data-driven method for effectively segmenting objects. The ML model is trained to identify critical points, and interpreting the latent space variables becomes crucial for precise segmentation. This methodology employs semantic segmentation on arbitrary shapes, utilizing a Convolutional Neural Network (CNN) with mesh pooling and unpooling layers to enhance feature extraction and representation [6][7][8]

Steps involved in ML based segmentation

I. Data Preparation: Collect or generate a dataset of 3D objects with labeled segments, incorporating knowledge of critical points in the structure. Represent each object as a mesh with associated segment labels.

II. Feature Extraction: Extract relevant features from the 3D mesh data, encompassing geometric features such as curvature, normals, or point-to-point distances, and topological features like connectivity.

III. Mesh Pooling: Apply mesh pooling techniques, as described earlier, to down-sample and simplify the mesh while retaining important features.

IV. ML Model Architecture: Design a neural network architecture suitable for processing mesh data. Utilize layers capable of handling irregular structures, such as Graph Convolutional Layers (Graph ConvNets) or PointNet layers.

V. Training: Train the ML model on the prepared dataset. Input features extracted from the mesh, and output predicted segment labels. Evaluate the segmentation model's performance on a validation set using metrics like precision, recall, F1 score, or Intersection over Union (IoU). Refine the model based on the evaluation results and feedback from these metrics.

VI. Implementation: Utilize the weights obtained in the training process to predict segment-able parts of the 3D structure. If necessary, perform post-processing by refining segmentation results through boundary smoothing or addressing misclassifications.

While training and validation process there are certain considerations are to be taken care:

* 1. Data Augmentation to reduce the effort of data collection and labeling. Therefore, augment with the training dataset with transformations like rotations, translations, and scaling to improve model generalization.
  2. Experiment with hyper parameter settings, such as learning rate and architecture choices, to optimize model performance.
  3. Address class imbalance in the dataset by using techniques like oversampling, under sampling, or class-weighted loss functions.
  4. Consider using interpretable models or visualization techniques to understand how the model is making decisions.
  5. Explore the use of pre-trained models or transfer learning if you have access to models trained on similar tasks.

The cost of acquiring data for ML training is contingent on various factors. The method of data collection, whether manual or automated, significantly influences costs. Larger datasets, while enhancing model generalization, may incur higher expenses due to processing and cleaning requirements. Ensuring high data quality through validation and cleaning adds an additional cost layer. Expertise in domain-specific data, such as medical or legal, may necessitate skilled annotators, contributing to overall expenses. The use of advanced technology and infrastructure, including specialized sensors or storage solutions, can also impact costs. Compliance with legal and ethical considerations, such as privacy laws and consent acquisition, adds another dimension. Data processing, augmentation, and long-term storage contribute to expenses, and the iterative nature of ML development may require additional data, compounding costs over multiple iterations. Balancing these considerations is crucial for optimizing the trade-off between data quality, quantity, and budget constraints during the ML training process.

# Segmentation using Reeb graph.

Given that Machine Learning (ML) is data-driven and often computationally demanding, an alternative approach involves employing computational geometry for Reeb graph extraction. The Reeb graph, obtained topologically, serves as a means to analyze critical features within the data. Laplacian contraction, a commonly used method for simplifying meshes or graphs, is applied to a 3D point cloud for Reeb graph extraction [9].

In the context of segmentation, the criteria for categorizing different regions of the data can be established based on specific conditions. These conditions may rely on geometric properties, density, or other relevant metrics tailored to the specific application [10]. This segmentation process allows for a flexible and criteria-driven approach to delineating meaningful regions within the dataset.

### Reeb graph:

Laplacian Contraction is a topological technique, it emerges as a powerful tool for unraveling concealed structures within complex 3D mesh data, proving particularly valuable for simplifying and comprehending intricate features of various structures. In the domain of data analysis, Laplacian contraction plays a crucial role in simplifying graphs or meshes, making it especially relevant in the exploration and understanding of spatial datasets like 3D point clouds.

At its core, Laplacian contraction involves applying the Laplacian operator (denoted by Δ) to a function defined over a space. This operator captures the second spatial derivative, revealing the local variations or smoothness of the function. In any structure, undulations and peaks correspond to different connected components, identifying regions of rapid change and signifying areas of interest for segmentation. The contraction step merges points that are "close" in terms of Laplacian values, producing a simplified representation without compromising critical topological information [11][12].

For instance, in the context of 3D point clouds, Laplacian contraction proves invaluable for extracting the Reeb graph, encapsulating the evolution of connected components as function values change. By contracting regions based on Laplacian criteria, the Reeb graph becomes a concise yet informative map of the data's topological landscape.

Conditional statements play a pivotal role in implementing Laplacian contraction for segmentation, establishing rules or thresholds based on geometric or functional properties. These statements guide the algorithm to discern significant features. For example, a condition might specify that contraction is only applied if the change in Laplacian values is below a certain threshold, ensuring that only subtle variations are merged, as depicted in Fig.2.

In essence, Laplacian contraction functions as a compass in navigating intricate datasets, guiding researchers through the topological terrain. Its application, coupled with conditional statements, provides a dynamic framework for segmentation, enabling the extraction of meaningful information from complex spatial data. The basic mathematical concepts and formulas associated with Laplacian contraction within the context of graph simplification are detailed below [13][14].

### Laplacian Matrix:

The Laplacian matrix of a graph is a square matrix representing the graph's topology. For an undirected graph with vertices, the Laplacian matrix is defined as:

where:

is the degree matrix (a diagonal matrix with vertex degrees on the diagonal).

is the adjacency matrix of the graph.

### Laplacian Smoothing:

In a Laplacian smoothing step, each vertex is updated based on the average of its neighbors:

here:

- is the updated position of vertex

- is the degree of vertex

- represents the neighbors of vertex .

Laplacian Contraction:

Laplacian contraction involves collapsing pairs of connected vertices and into a single vertex at their midpoint:

The graph is then updated by removing and and connecting all neighbors of and to

### Reeb Graph Construction:

In the context of Reeb graphs, Laplacian contraction can be applied iteratively to simplify the graph representation of a scalar field. The Reeb graph is constructed by identifying critical points (locations where the gradient is zero) and tracking the evolution of connected components as the scalar field changes.

Implementation:

The implementation involves linear algebra operations on the Laplacian matrix, updating vertex positions, and managing the graph structure, by gradually obtaining the skeleton structure of a given complex 3D structure. In figure Fig.1 is the given 3D structure, which represents a bent cylinder, upon applying the Laplacian the skeleton or the reeb graph is obtained, which is represented in figure Fig.2.

A colorful curved object with lines and numbers

Description automatically generated with medium confidence

Figure 1:Example of a 3D structure with undulation and branches (Bent cylinder)

A red and blue wireframe of a curved tube

Description automatically generated

Figure 2:Implementation of Laplacian contraction on bent cylinder.

Steps involved in Reeb graph extraction (algorithm)

1. **Skeletal Structure Extraction using Laplacian Contraction**: Apply Laplacian contraction to the 3D mesh to obtain a skeletal structure that emphasizes the main features and connections within the mesh.

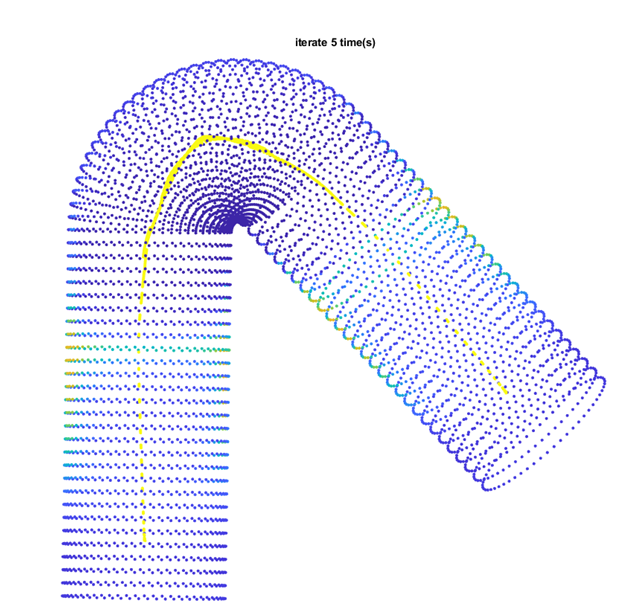


Figure 3: Laplacian Contraction (shown in yellow) after 5 iterations



Figure 4:Contraction by volume after every iteration

Note: Fig 4 indicates the number of iterations required to get a reasonable contraction

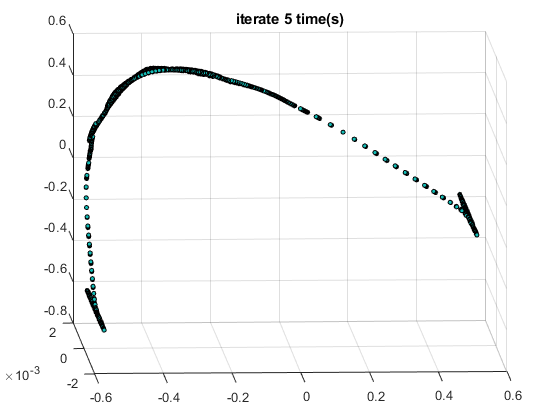


Figure 5: Skeleton structure of the bent cylinder

2. **Identification of Critical Points for Segmentation:** Utilize the obtained skeletal structure to identify critical points such as junctions and angles of deviation. These critical points serve as key markers for subsequent segmentation.

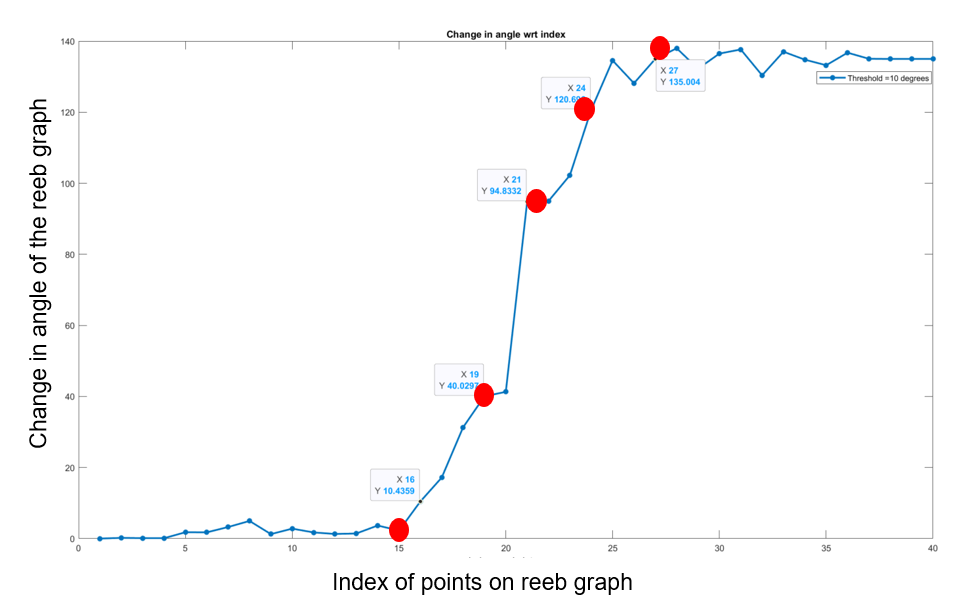


Figure 6: Critical points for segmentation for bent cylinder

3. **Creation of Segmentation Rule**: Based on the identified critical points, establish a segmentation rule. This rule could involve defining thresholds for certain properties, angles, or distances to guide the segmentation process. For instance, here the the change in angle is taken as the segmentation rule to create the critical points defined use red dots on the figure below.

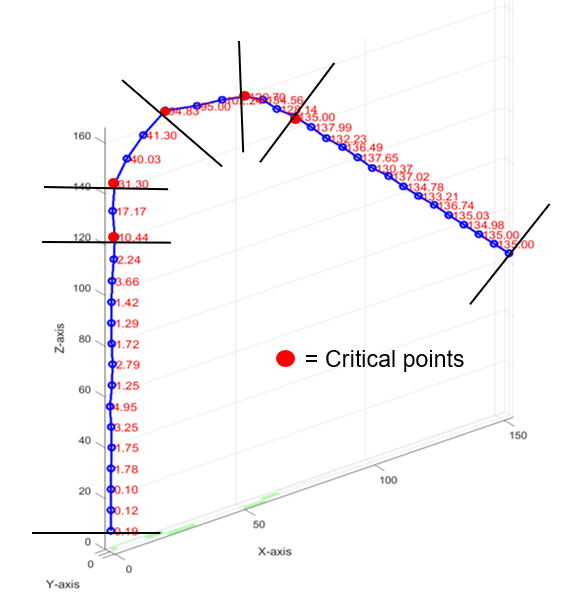


Figure 7:Segmentation lines based on derivative of change in angle

For Angle Calculation, one has to calculate the angle between each directed edge in the weighted graph and its preceding edge. Utilize trigonometric functions to determine these angles, aiding in the identification of critical points. When the calculated angle surpasses the defined threshold, mark the corresponding point on the Reeb graph as a segmentation point. Similarly, change in angle or derivative of the change in angle calculation can be evaluated.

Combining both the criteria an estimate of segmentation can be performed. Upon careful consideration of the threshold should be chosen, therefore threshold tuning is a critical step. Experiment with different threshold values to strike an optimal balance between sensitivity and specificity.

4. **3D Mesh Segmentation**: Implement the segmentation rule on the 3D mesh, dividing it into distinct segments or regions based on the identified critical points and the established criteria.

The above steps outlines a comprehensive process, from skeletal structure extraction using Laplacian contraction to the extraction of segments based on Reeb graph properties and segmentation criteria.

The Laplacian contraction process presents several advantages. Firstly, it maintains the nature of the original mesh throughout the procedure. This ensures that the final curve-skeleton derived from the process remains homotopic to the original object, preserving the essential topological features. Additionally, the method exhibits an inherent ability to handle noise, rendering it insensitive to variations and disturbances in the data. This robustness to noise contributes to the reliability of the Laplacian contraction process, making it a valuable approach in scenarios where data may contain uncertainties or irregularities.

Despite its advantages, the Laplacian contraction process comes with several drawbacks. Firstly, being an iterative process, it demands significant computational power to perform efficiently. The need for substantial computational resources can be a limitation, particularly in situations where computational capabilities are constrained.

Moreover, the Laplacian contraction process exhibits a large time complexity, with almost 7 seconds required for just 5 iterations. This extended processing time may hinder its practicality in applications where real-time performance is crucial.

Additionally, the method is limited to closed and branched mesh models, rendering it ineffective for other geometries. In the case of pyramid-shaped models, the skeleton extraction fails to retain crucial information about the geometry. This limitation prevents the identification of critical points and hampers the ability to perform accurate segmentation. As a result, the Laplacian contraction process may not be suitable for scenarios involving diverse geometric structures.

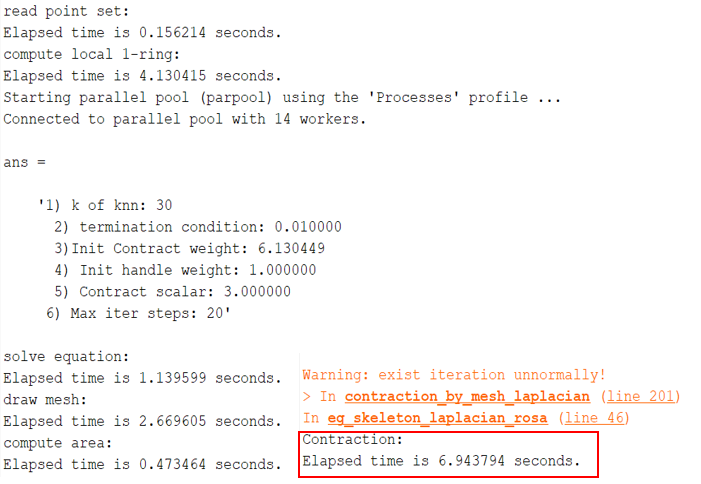


Figure 8:Time complexity of Laplacian Contraction

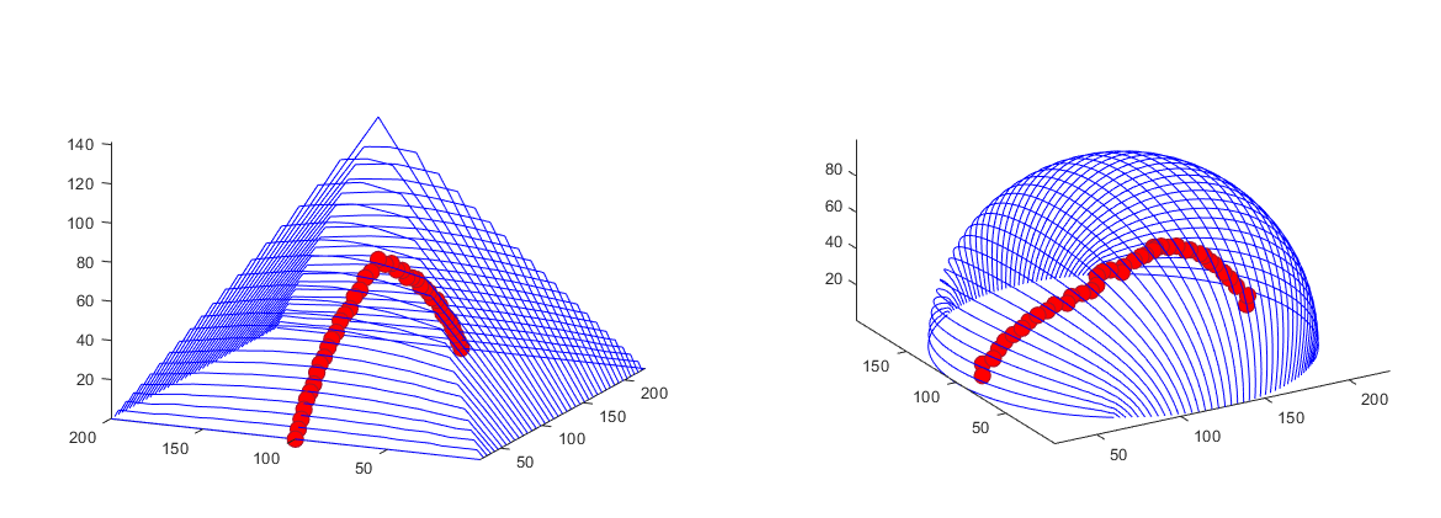


Figure 9: Pyramid and hemi sphere the skeleton doesn’t retain much about the geometry

# Mapper Algorithm for segmentation

The Mapper Algorithm serves as a powerful method for the qualitative analysis, simplification, and visualization of extensive high-dimensional datasets. Its primary objective is to transform high-dimensional data into simplicial complexes with fewer points, offering a tool for generalized coordinatization. Unlike conventional real-valued coordinates, Mapper utilizes a discrete and combinatorial object—a simplicial complex—to represent data meaningfully.

Grounded in topological principles, Mapper preserves nearness while permitting distortion in large-scale distances, a valuable characteristic when dealing with intricate and vast datasets. The method leverages a real-valued function on the dataset to construct a graph, which can be adjusted for various parameter spaces.

The pivotal concept within Mapper is partial clustering, wherein standard clustering algorithms are applied to subsets of the dataset. The interactions between these partial clusters are then employed to construct a simplicial complex, creating a multiscale representation of the dataset. This approach enables the assessment of features' validity across different coarseness levels.

It's important to note that Mapper doesn't strive for a fully accurate representation but rather focuses on providing a low-dimensional, easily understandable image that highlights areas of interest. The method implicitly defines a parameter space, determining the upper bound on the dimension of the studied simplicial complex. This concept draws an analogy to the Postnikov tower in algebraic topology, named after the mathematician Mikhail Postnikov. Just as the Postnikov tower breaks down a topological space into simpler components, the Mapper method decomposes high-dimensional data into simplified structures, offering a low-dimensional representation for analysis.

Implementing the Mapper algorithm for 3D data segmentation follows the following steps. First, prepare real-valued 3D point cloud data. Then, apply the Mapper algorithm to create a simplicial complex that captures the data's topological and geometric features at a chosen resolution. The key is selecting a suitable real-valued function that reflects relevant properties, like density or critical points.

Next, use standard clustering algorithms on different subsets of the 3D data for partial segmentation. These partial clusters contribute to building a multiscale simplicial complex, offering segmentation at various coarseness levels.

Examine the resulting complex to identify meaningful segments. Robust segments persist across different resolutions, while those appearing only at specific coarseness levels may be artifacts.

Fine-tune parameters, like the resolution of the Mapper algorithm and clustering parameters, to optimize segmentation for your specific 3D dataset. However, keep in mind that successful implementation requires thoughtful choices regarding the real-valued function, clustering algorithm, and Mapper construction parameters. Experimentation and validation are crucial, though this can be a computationally intensive trial-and-error process.

# Time values, and distances criterion for segmentation:

The previous algorithms, while been effective, has been hindered by computational expenses and the necessity for human input in selecting real-valued functions. In response to these challenges, a simpler segmentation algorithm has been employed, prioritizing the construction of knitting patterns through a rule-based approach. This alternative method relies on specific point properties, including time values and distances, to guide the creation of knitted rows and short rows within a 3D object. By focusing on the inherent characteristics of the points in the dataset, this segmentation algorithm aims to streamline the segmentation process and mitigate the computational demands associated with more complex approaches. This introduction explores a simplified knitting-pattern-based segmentation algorithm, emphasizing its potential to offer a more accessible and efficient solution for certain applications.

Algorithm Overview:

This segmentation algorithm operates on the basis of two key criteria: time and distance. To form a full row, the time criterion is assessed. If the maximum time value in the preceding layer or row is less than all the time values in the potential current row, the time criterion is considered satisfied. Following this, the algorithm checks the distance criterion. For a full row, the distance between all points in a row and short rows must be less than two times the designated knitting width.

Consequently, when the time criterion is met, a full row is generated. Conversely, if the time criterion is not satisfied, short rows are created. In essence, this algorithm dynamically determines whether a full row or a short row is more appropriate based on the temporal sequencing of the data points and the permissible distances between them. This dual criterion approach ensures a nuanced segmentation process, facilitating the construction of knitting patterns in a manner that is both clear and adaptable to varying datasets.

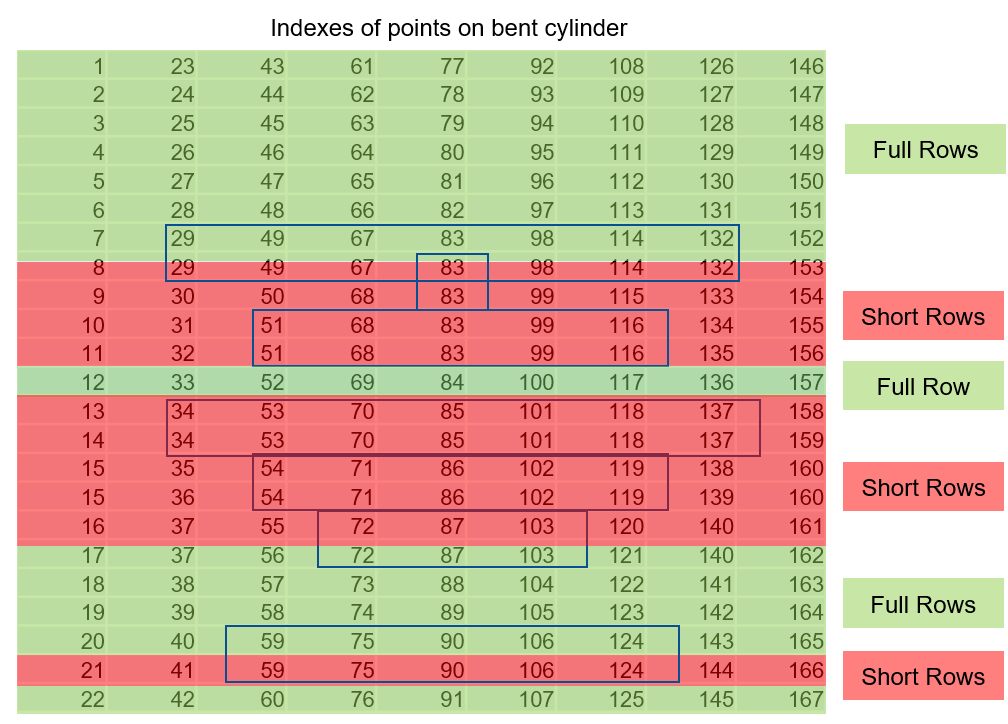


Figure 10: Segmentation of bent cylinder using Time & Distance Scalar Fields

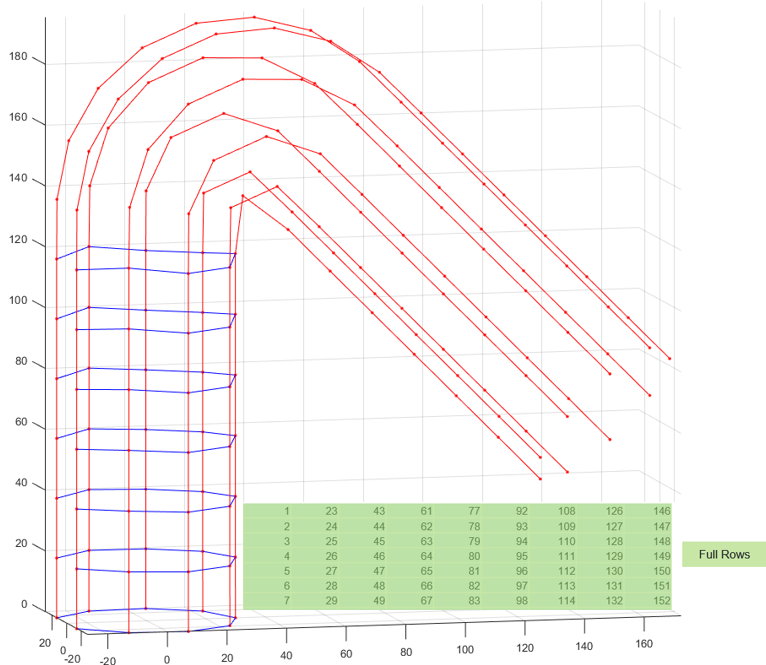


Figure 11: Row building in Bent cylinder

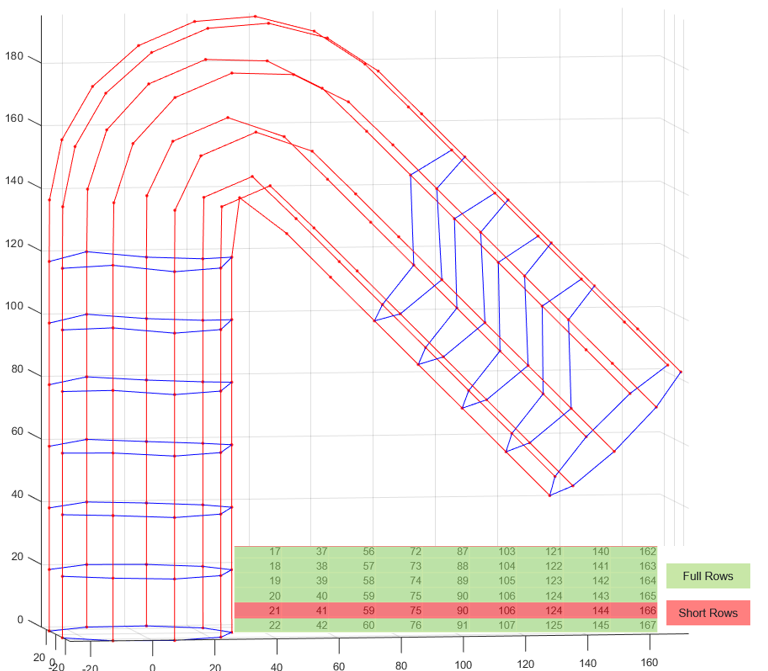


Figure 12: Row building in Bent cylinder

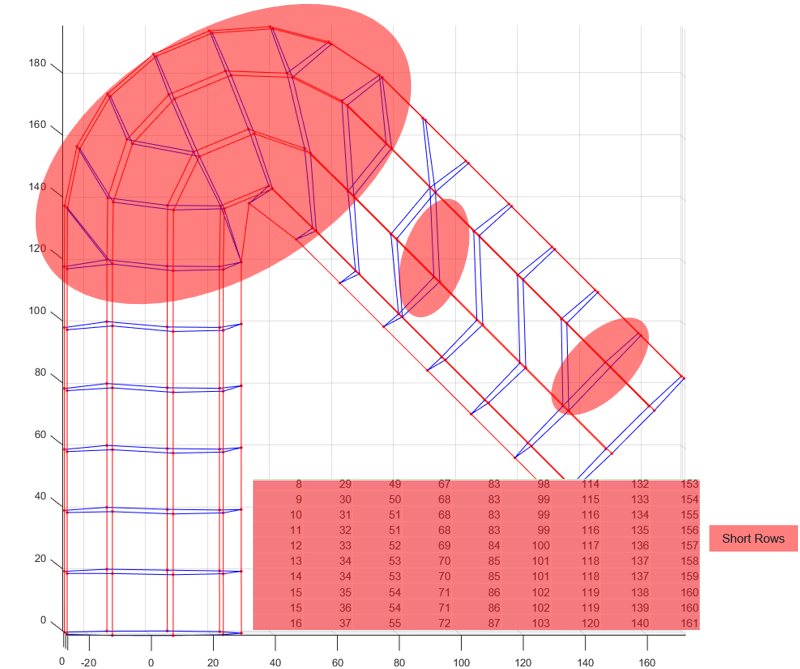


Figure 13: Short row building in Bent cylinder

Segmentation through the utilization of Time and Distance Scalar Fields offers distinctive advantages, primarily characterized by a layer-by-layer row-building strategy. This approach enhances the precision of segmentation, leading to a more localized and nuanced segmentation process. By systematically evaluating each layer individually, the algorithm achieves a finer level of detail, ensuring that the segmentation is tailored to the specific characteristics of each layer. This layer-by-layer row-building methodology enhances the algorithm's ability to adapt to variations in time and distances, resulting in a more accurate and locally focused segmentation process. Overall, these advantages contribute to the effectiveness and precision of the segmentation technique, particularly in scenarios where localized segmentation is paramount.

The segmentation method employing Time and Distance Scalar Fields exhibits certain drawbacks that need consideration. Firstly, due to truncation errors in distance values, a single node can connect to multiple points based on the nearest distance criterion. This leads to the creation of multiple short rows associated with the same node, introducing redundancy and potential inaccuracies.

Secondly, the algorithm faces challenges in generalization, particularly when the number of points in a full row varies. This issue becomes pronounced in structures like pyramids, where the start and end lines have fewer points compared to the other lines in between. The dynamic nature of point counts impedes effective generalization.

Furthermore, the sequential layer-by-layer building of rows results in substantial computational time, particularly when dealing with intricate structures. The algorithm's efficiency is compromised when applied to complex geometries, leading to prolonged processing times. Addressing these identified drawbacks is crucial for enhancing the robustness and applicability of the segmentation approach using Time and Distance Scalar Fields.

This detailed procedure outlines how the algorithm initializes data structures, visualizes the initial state, constructs knitting courses based on specified conditions, and iteratively builds the knitting pattern. Figure Fig.3 shows the complete knit pattern where the red lines are the columns, and the blue lines are the rows of the knitting pattern. There some nodes connect to two other nodes at the same time, this gives rise to short rows.

A graph of a slide

Description automatically generated

Figure 15:Fig: Implementation of the above-mentioned algorithm with w=10 and h=5

A green and white table with numbers

Description automatically generated

Figure 16:Visualization

Figure Fig.5 is analytical representation of the knitting pattern, the repeated rows represents short rows.

A red and blue graph

Description automatically generated

Figure 17:Fig: Implementation of the above-mentioned algorithm with w=0.96 and h=0.42

Implementation of the above-mentioned algorithm for different sampling data is shown in figure Fig.6. In figure Fig.6 just before the bent part of the cylinder, there are distortions which are a result of the cumulative addition of time and distance inequality.

# Time Variance based segmentation.

The segmentation algorithm proposed in this study centers around the intricate task of constructing knitting patterns through a meticulous analysis of point properties within a 3D object. The primary focus lies on leveraging attributes such as time values density and distances between points. An essential insight guiding the segmentation process is that points potentially belonging to the same row share the same time level.

In contrast to prior methods that sorted data solely based on time and distances, the introduced approach acknowledges and addresses distortions introduced during this sorting process. To overcome such distortions, the algorithm incorporates the time variance of each layer, paving the way for more accurate segmentation. Specifically tailored for the realm of knitting patterns, the 3D data segmentation technique elucidated herein involves grouping time values based on predetermined ranges or variances across the mean of each row.

Within the context of the knitting algorithm, this segmentation process is executed on the time field values associated with sampled points. The overarching objective is to discern and cluster time values that share similar characteristics, thereby constructing a well-organized representation of the temporal aspects inherent in the knitted pattern. This innovative approach integrates insights from previous methods while introducing a nuanced consideration of time variance, ultimately contributing to a more refined and accurate segmentation of 3D data in the context of knitting pattern construction [18][19][20].

Overview of the algorithms:

The segmentation method outlined in this approach combines time values density and distances within the context of a 3D mesh structure. The process commences with the calculation of the number of potential rows, a critical parameter denoted as "h," alongside the determination of the largest length in the 3D mesh. This calculation serves as a pivotal precursor for the subsequent clustering steps.

The implementation of the k-means clustering algorithm follows, utilizing equi-spaced time values ranging from -1 to 1. The number of clusters is determined by the previously computed potential rows. This clustering operation effectively groups time values based on their proximity, initiating an initial segmentation of the temporal data.

Following the clustering phase, the algorithm proceeds to construct full rows and short rows utilizing specific distance criteria. This comprehensive approach not only clusters time values but also refines the segmentation by incorporating distance considerations, contributing to a more nuanced and accurate segmentation of the 3D mesh data.

Steps in 3D Data Segmentation:

1. Data Arrangement: The algorithm arranges the input data based on wale indices and time values. The arranged data is stored in the matrix `arranged\_data`.
2. Time Sorting: The algorithm organizes time values into a matrix called `wale\_sorted\_time`. This matrix is structured based on wale indices, creating a temporal sequence of time values for each wale.

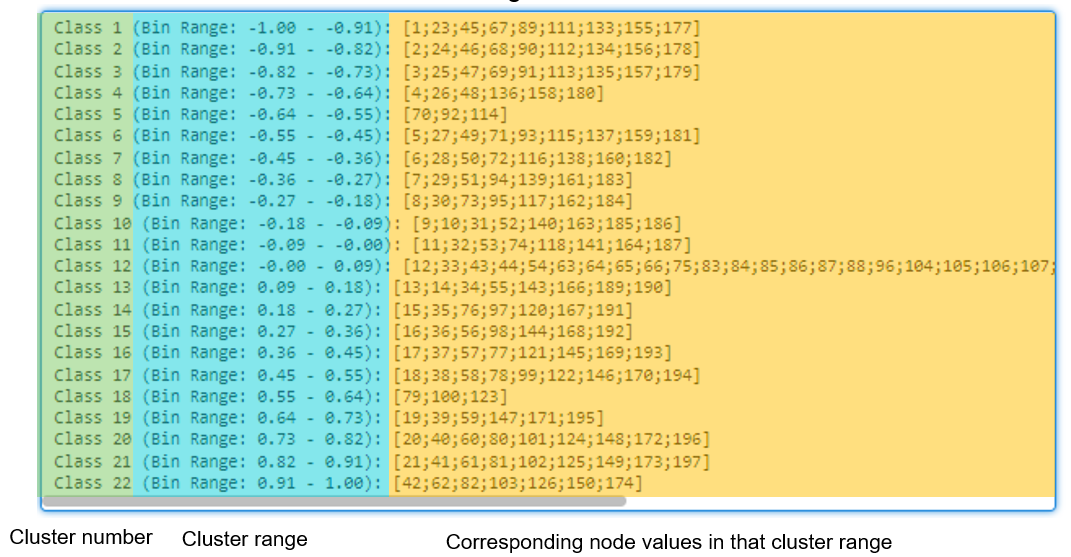


Figure 18: Clustering of time values based on thier density

1. Histogram Generation: The code generates a histogram of time values to analyze their distribution. The histogram provides insights into the frequency of occurrence of different time values.
2. Data Analysis: The algorithm then performs an analysis on the generated histogram. It identifies ranges or classes of time values that are closely grouped together.
3. Visualization Based on Classes: For each identified class or range of time values, the algorithm retrieves the corresponding indices from the arranged data. These indices represent points in the knitted structure.
4. Point Visualization: The algorithm plots the points associated with each identified class in blue on the existing 3D scatter plot. This visualization helps in understanding the temporal patterns within the knitted structure.

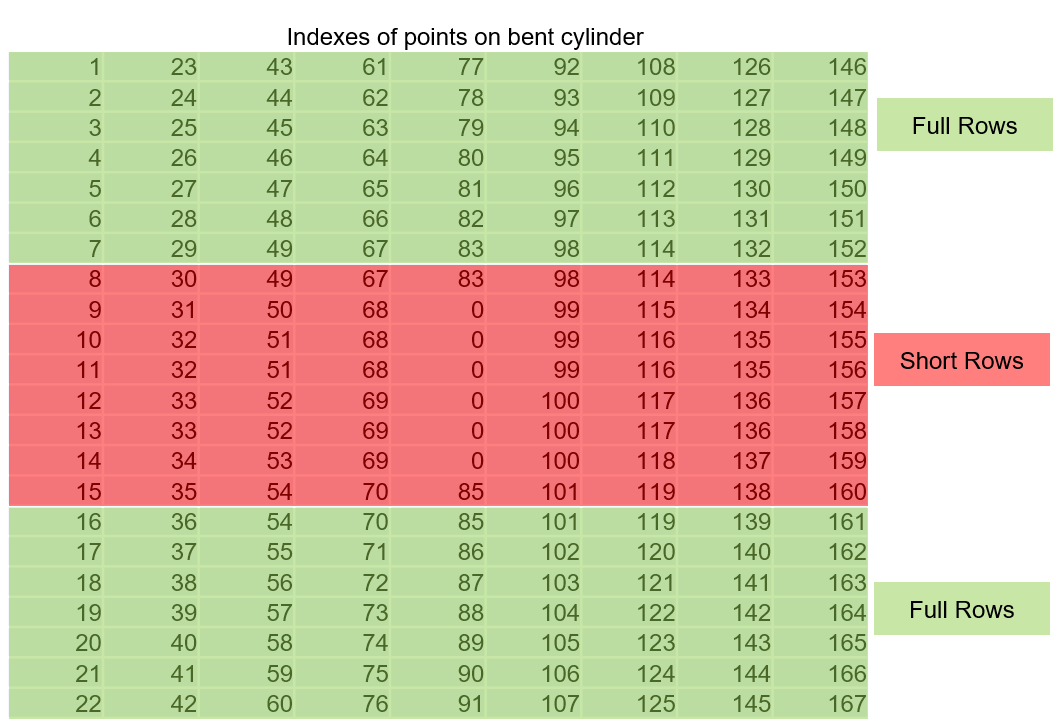


Figure 19: Formation of full rows and short rows

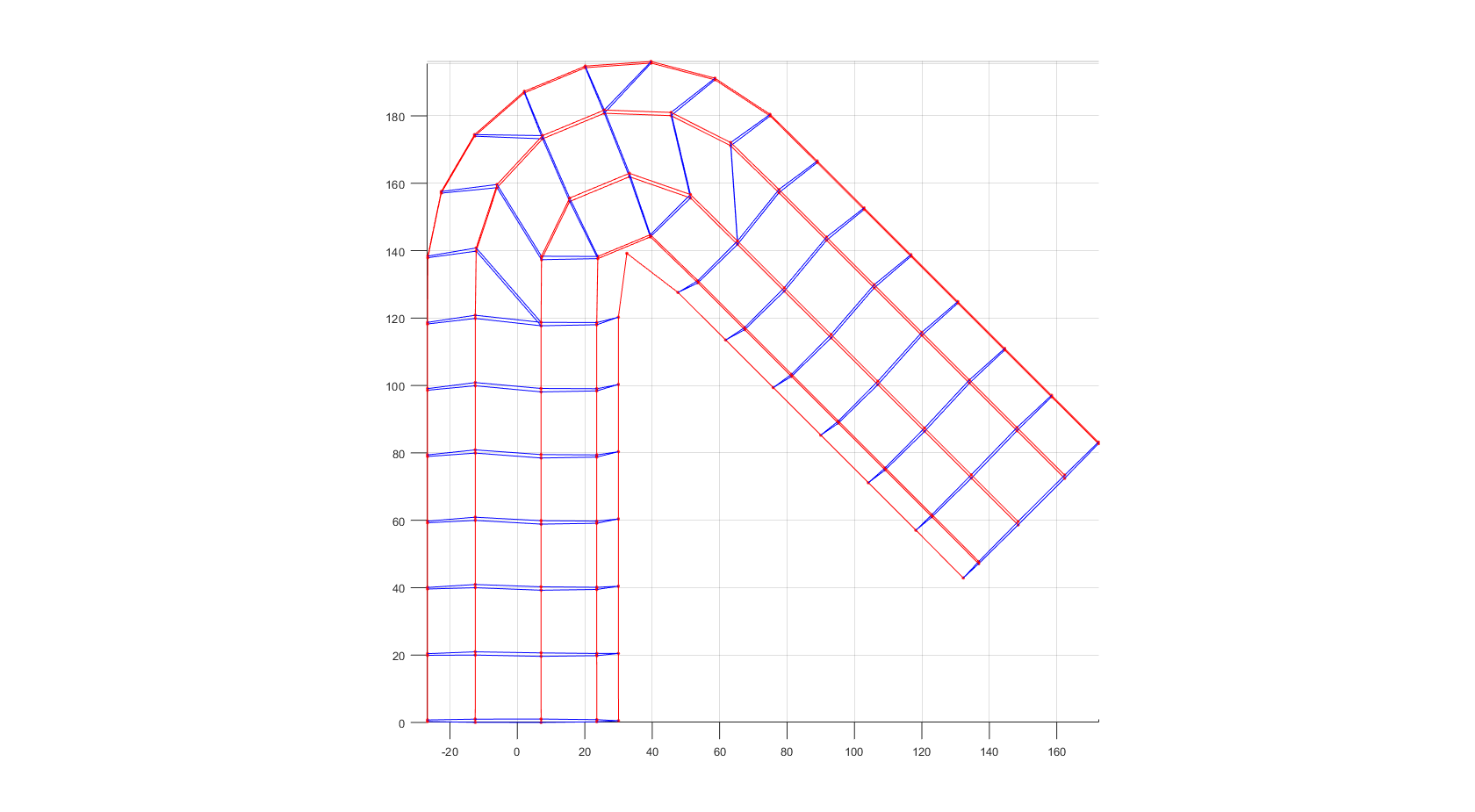


Figure 20: Implantation on Bent Cylinder: using time clusters to form full rows

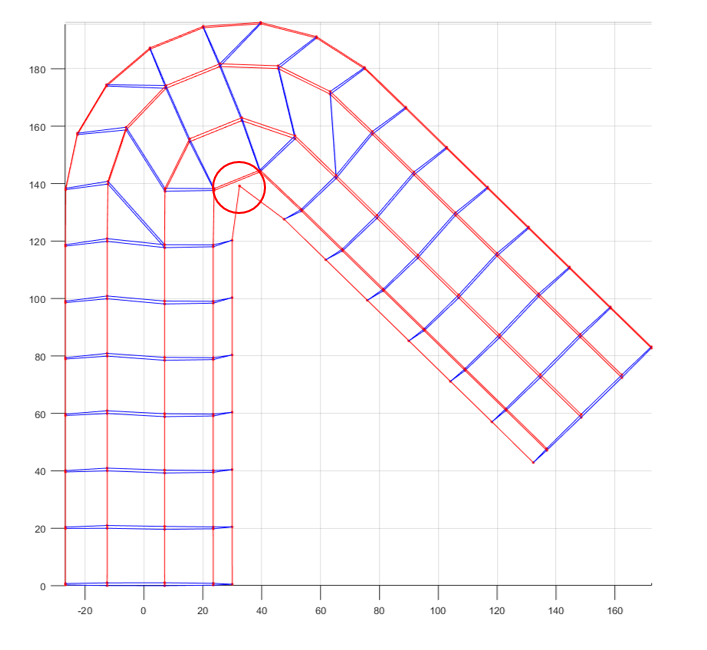
The segmentation of time values plays a pivotal role in streamlining and organizing the time information embedded within the knitted structure. The primary objective is achieved by grouping similar time values together, a process that effectively accentuates specific temporal patterns or features inherent in the knitted pattern. Through this clustering mechanism, the algorithm mitigates distortion, offering a clearer understanding of the temporal dynamics involved in the knitting process.

This segmentation proves to be particularly valuable for unraveling the intricacies of the knitting timeline. It aids in the identification of recurring patterns, opening avenues for potential optimizations or refinements in knitting techniques. By enhancing the interpretability of temporal aspects within the knitted structure, the segmentation contributes to a more comprehensive analysis and visualization of the knitted pattern. Overall, this approach enriches our ability to delve into the temporal intricacies of the knitting process, fostering a deeper understanding of its dynamic nature.

Drawbacks of segmentation using Time values density and distances :

The segmentation method utilizing time values density and distances is not without its identified drawbacks. One notable limitation arises during the creation of bins for clustering classes, where the non-linearity of the distribution places marked points on the boundary of the bins. Consequently, these boundary points are overlooked when creating short rows, introducing a potential gap in the segmentation process.

The omission of these points is contingent upon the geometry of the object, specifically the density of points between the start line and end line. This dependence on the object's geometry underscores the need for an exhaustive list of diverse geometries to enhance the generalization capability of the segmentation method. Addressing this limitation becomes imperative to ensure the method's robustness across various object shapes and distributions, emphasizing the ongoing importance of refining and expanding the segmentation approach for broader applicability.



The unclustered points in the segmentation process signify instances where a point lies on the boundary of clusters and is equidistant from all neighboring points. These unclustered points, being unable to satisfy both time and distance conditions, are consequently designated as zeros due to their unavailability for assignment to specific clusters.

The impact of this phenomenon is contingent upon the geometry of the object and the knitting parameters, specifically the height (h) and width (w). The interplay of these factors contributes to the emergence of unclustered points, highlighting the intricate relationship between geometric characteristics and the chosen knitting parameters within the segmentation framework. Addressing this aspect becomes crucial for refining the segmentation process and improving its adaptability to diverse geometries and parameter configurations.

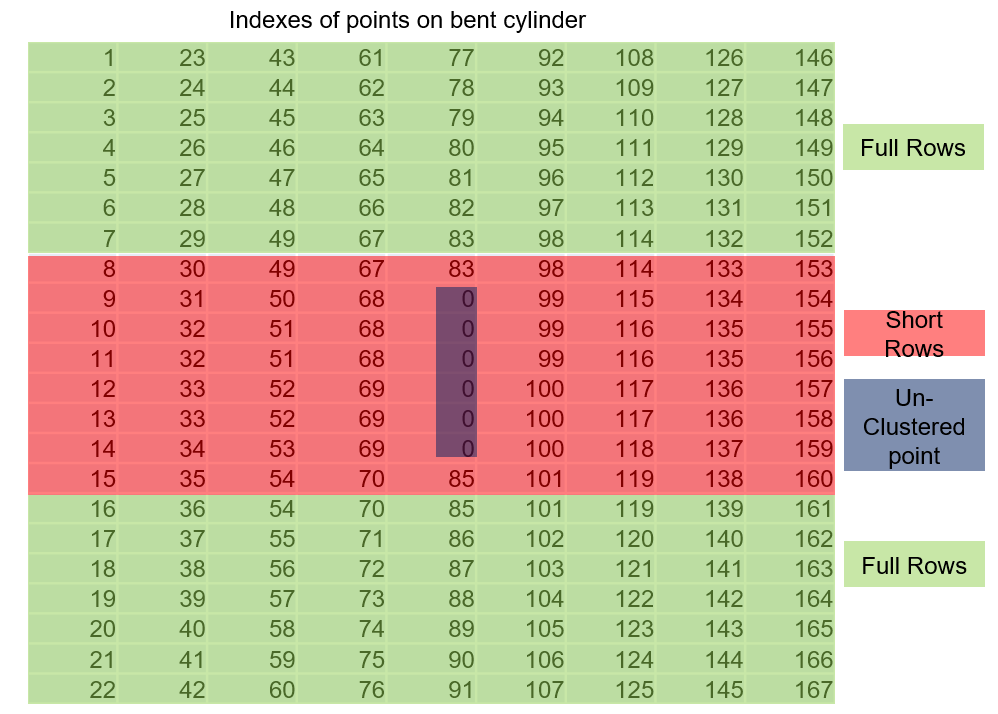


Figure 21: Un clustered points in bent cylinder

The segmentation method utilizing time values density and distances exhibits certain drawbacks. Notably, the non-linearity present in structures can lead to clustering between points that are spatially distant but temporally close. An illustrative example is evident in the bent section of a cylindrical object, where non-linearity results in clustered points spanning multiple locations within a single cluster. This clustering behavior adversely affects subsequent rows, creating undesired short rows due to a diminished number of points available for the following row.

Moreover, iterating over these points to rectify the issue proves computationally costly, impacting the efficiency of the segmentation process. Another challenge lies in the initialization of class centers, as it significantly influences the clustering of time values. The generalization of these initialization steps is intricately tied to the specific geometry of the object, further emphasizing the need for robust initialization strategies to enhance the method's adaptability across various structures. Addressing these challenges is crucial for refining the segmentation approach, improving accuracy, and reducing computational overhead.

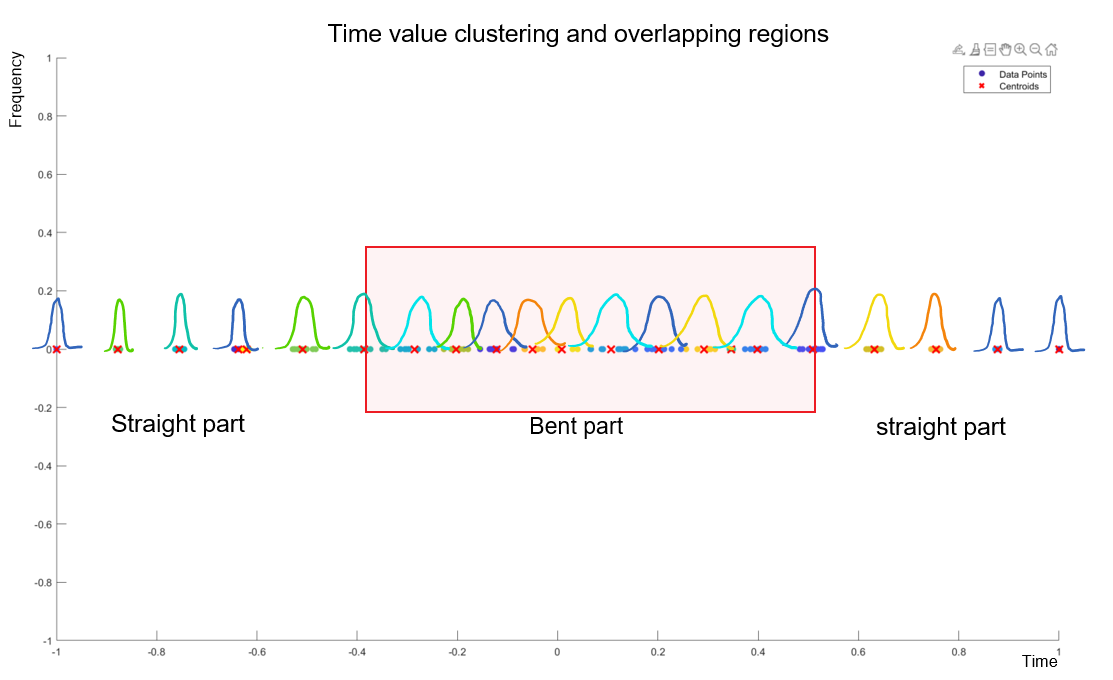


Figure 22: Time value clustering and overlapping regions due to non-linearity

# References:

* 1. Garland, M., & Heckbert, P. S. (1997). Surface Simplification Using Quadric Error Metrics. ACM Transactions on Graphics (TOG), 17(2), 209-216.
  2. Hoppe, H. (1996). Decimation of Triangle Meshes. ACM Transactions on Graphics (TOG), 15(3), 205-236.
  3. Hoppe, H. (1996). Progressive Meshes. ACM Transactions on Graphics (TOG), 15(3), 99-108.
  4. Borodin, P., Pasko, A., & Adzhiev, V. (2009). A survey of mesh simplification algorithms. Computer-Aided Design, 41(1), 65-85.
  5. Garland, M., Kondo, Y., & Hart, J. C. (2002). Simplification Envelopes. ACM Transactions on Graphics (TOG), 21(3), 365-371.
  6. Fang, Y., Xu, Y., et al. (2018). DeepSkeleton: Learning Multi-task Scale-associated Deep Side Outputs for Object Skeleton Extraction in Natural Images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
  7. Lan, S., Ma, R., et al. (2018). SkeletonNet: Mining Deep Part Features for 3D Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
  8. Yang, W., Ouyang, W., et al. (2016). End-to-End Learning of Deformable Mixture of Parts and Deep Convolutional Neural Networks for Human Pose Estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
  9. Sorkine, O., & Cohen-Or, D. (2004). Laplacian Surface Editing. Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, 175-184.
  10. De Floriani, L., Magillo, P., & Puppo, E. (2002). Reeb Graphs for Shape Analysis and Applications. ACM Transactions on Graphics (TOG), 21(2), 169-174.
  11. Edelsbrunner, H., Letscher, D., & Zomorodian, A. (2002). Topological Persistence and Simplification. Discrete and Computational Geometry, 28(4), 511-533.
  12. Edelsbrunner, H., & Musin, O. R. (2010). Reeb Space of Piecewise Linear Functions. Topology and its Applications, 157(5), 870-889.
  13. Dey, T. K., & Sun, J. (2008). Efficient Computation of Homology of Subdivisions and Cell Complexes. Computational Geometry, 41(3), 128-144.
  14. Dey, T. K., & Fan, F. (2009). Fast Construction of the Vietoris-Rips Complex. Discrete & Computational Geometry, 42(1), 139-158.
  15. Singh, G., Mémoli, F., & Carlsson, G. (2007). Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition. In Symposium on Point-Based Graphics.
  16. Damodaran, H., & Wang, Y. (2019). A Local Framework for Mapper on Metric Spaces. Journal of Computational and Graphical Statistics, 28(2), 426-435.
  17. Xia, Y., & Morozov, D. (2015). Persistence-Based Clustering in Riemannian Manifolds. Applied and Computational Harmonic Analysis, 38(2), 235-259.
  18. Ester, M., Kriegel, H. P., et al. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In KDD.
  19. Arthur, D., & Vassilvitskii, S. (2007). K-means++: The Advantages of Careful Seeding. In SODA.
  20. Arthur, D., & Vassilvitskii, S. (2012). On the Convergence of the K-means Clustering Algorithm. arXiv preprint arXiv:1202.1585.
* <https://topology-tool-kit.github.io/installation-windows-sources.html>
* <https://arxiv.org/pdf/1809.05910v2.pdf>
* <https://paperswithcode.com/task/3d-part-segmentation>