ISSN 2464-4617 (print)
ISSN 2464-4625 (online)
Computer Science Research Notes - CSRN
http://www.wscg.eu
WSCG 2025 Proceedings

Feature-Sensitive Mesh Simplification

Dominik Krobath
Graz University of Technology
Inffeldgasse 16c
8010 Graz, Austria
krobath@tugraz.at

Ursula Augsdörfer
Graz University of Technology
Inffeldgasse 16c
8010 Graz, Austria
augsdoerfer@tugraz.at

ABSTRACT

Feature recognition in meshes is a widely studied topic of computer science and is applied in order to identify and analyse properties of a 3D shape. It finds applications in numerous fields, like, e.g. medical imaging, robotics, cultural heritage, and CAD design. In this paper, we propose a new segmentation-based feature recognition approach for 3D meshes, where features are defined by the intersection of two neighboring segments. The detected features are used to derive very sparse polygonal representations of arbitrary input meshes, even merging features to achieve a high degree of simplification. We compare results for edge-based segmentation and a new normal-based segmentation and demonstrate how to derive good results even for pseudocurved surfaces and meshes affected by artifacts.

Keywords

Feature Recognition, Mesh Segmentation, Mesh Simplification

1 INTRODUCTION

In computer vision and graphics, feature recognition in shapes is a prominent field of research. The focus of this field has changed over the last years from recognizing features in images to detecting features in scanned 3D data. 3D scans tend to be prone to measurement noise which complicates the recognition of features. A common approach to address this problem is to apply a type of smoothing to the noisy data that preserves the features as accurately as possible [Sun02; Wan12; Liu23a; Hur24; Wan24].

In this paper, we present a segmentation-based feature recognition approach to derive a sparse polygonal representation from noisy volumetric data. The goal was for the approach to work on noisy input shapes directly and to be able to handle various types of noise. To achieve this, we apply a greedy algorithm to find larger connected areas that represent segments of the shape. The boundaries between the segments represent the features. From these boundary regions, we extract a sparse polygonal representation which serves as a simplified representative of the shape. The results presented in this paper facilitate the extraction of a sparse polygonal mesh representation from any 3D input geometry.

The paper is structured as follows: In Section 2, we take a closer look at the current state of the art in the field of segmentation-based feature recognition. In Section 3, the new approach proposed in this work is described in detail. The results obtained using two different segmentation algorithms are compared and discussed in Section 4. Finally, in Section 5, we conclude with an outlook on possible future work in this field.

2 STATE OF THE ART

The area of feature recognition in shape data is strongly researched and has numerous applications, e.g., computer-aided design and manufacturing, computer graphics, and structural engineering [Cal23; Hur24]. Typical tasks that require feature recognition include shape denoising, shape simplification, and quality control. Feature-sensitive smoothing of noisy polygonal meshes is also important in the domain of numerical simulation, where it is necessary to find sharp features, because these areas have a large influence on simulation [Kob03; Fan10]. One known way to recognize features is the Hough transform [Has15], which is used to detect various geometric shapes in images. This was extended to detect features in the 3D space [Rom22]. In addition to sharp features, which represent distinct high-curvature changes in geometry, such as edges, corners, creases, and ridges of a shape, there are also so-called smooth or shallow features, which represent gradual, low-curvature variations in

the geometry [He 23]. These two different types of feature are difficult to recognize due to the different properties that characterize them. Typically, algorithms aim to optimize the recognition of one feature while accepting that the results for the other type of feature may not be optimal. An example of this is the work of Hurtado et al. [Hur24] where the geometry of the mesh is used to increase the recognition of sharp features, but shallow features are not represented well. Other works create a dual graph for parts of the mesh to improve the result [Liu23b].

Our work presented in this paper aims to detect sharp features. A typical approach to detect sharp features is by analyzing face normals and the change in normal direction that occurs at sharp features. Various approaches aim to identify the vertices of a mesh that are part of such sharp features by considering their one-ring face neighborhood [Sun02; Wan12; Hur24], or the change between the normals of adjacent faces [Wan24]. Another approach to detecting sharp features is to find the planar areas within the mesh. If planar regions are found, sharp features are located on the border where two planar regions touch [Lee04; Wan24]. In their approach Lee et al. [Lee04] create polygonal meshes from the input mesh, and each edge is labeled with a cost value. Only high-cost polygonal regions are used for the second step where a region growth algorithm is applied. For this region growth, face normals and the area of the region are used as a metric. Wang et al. [Wan24] use two different metrics to assess if two faces belong to the same flat region. One is the intrinsic metric of the average length of the edge. The other metric is the L_2-norm of the dihedral angle between two neighboring faces. By combining these two metrics, a good segmentation is provided. An approach from Cohen-Steiner et al. [Coh04] uses an error-driven approach to segment the faces into regions. They apply two steps iteratively until they reach an error minimum. In the first step they grow a certain number of regions on the mesh and try to minimize the error. In the second step, a representative is calculated for each region to minimize the distortion error for the whole segmentation. Simari et al.[Sim14] also use a k-means style approach. In their approach, they start by selecting k region centers and grow the regions from there. As soon as the whole mesh is partitioned into regions, new starting center faces for the regions are calculated. If these calculated faces remain the same for all regions, the algorithm is terminated. If at least one face changes, a version of Dijkstra's algorithm is applied to each face to find the shortest path to the nearest cluster center, and the face's affiliation is updated to a new region. These two steps are applied alternately.

Instead of planar regions, other geometries may be used. Liu et al. [Liu23a] use the fact that man-made objects can generally be divided into a number of simple geometric shapes. They trained a neural network to recognize simple geometric shapes in a point cloud and then fit a mesh onto the found shapes and define shape features along intersections of these geometric shapes.

Vieria and Shimada [Vie05] applied iterative region growing and fitting of quadratic surfaces. Their approach maximizes the number of connected vertices that can be accurately approximated by a single underlying surface.

Li et al. [Li 18] focused on the curvature change in the area of sharp features, but refined this approach by applying a smoothing algorithm on a copy of the mesh to find the sharp features there and then using the information of the sharp features found to smooth the main mesh with respect to these features.

The aim of this work is to generate a simplified, sparse polygonal representation from a dense, possibly noisy triangle mesh representing man-made or engineering structures, such as mechanical parts in CAD models. Near-planar regions are of particular interest in solving this problem, and therefore, we focussed on growing near-planar regions. The features are then detected at the intersections of planar regions. We will discuss the edge-based simplification by Wang et al. [Wan24] and its limitations in detail and present a new normal-based simplification approach which focuses on creating a sparse polygonal representation. We will refer to these two different approaches as edge-based and normal-based feature extraction, respectively, throughout this paper.

The majority of algorithms which focus on feature detection apply some sort of smoothing to generate planar regions which are always connected to a loss of information. We tried to find a new way to detect sharp features without applying any kind of smoothing. For this, we looked at different approaches. We tried creating new planes in space to represent different sets of faces, but this led to many erroneously detected features from planes that should not intersect. We then experimented with different approaches, focusing on greedily growing regions. This leads to the approach we want to present in Section 3 which uses a sort of probabilistic method to define features.

3 METHOD

In this paper, a new approach is presented to derive a feature-sensitive mesh simplification from corrupted input shape data without any pre-processing. We achieve this by segmenting the input shape into regions of planarity. The segmentation of the input geometry into planar regions will be discussed in Section 3.1. We provide details on how we extract features from these intersections in Section 3.2. The general steps of both approaches discussed in this paper are presented in Table 1.

3.1 Segmentation

We compare two different segmentation approaches. One is a modified approach from their work on feature-sensitive smoothing by Wang et al. [Wan24]. We also propose an approach based on segmenting the input shape into planar segments, which specifically aims to reliably detect sharp features.

3.1.1 Edge-based segmentation

Wang et al. [Wan24] introduce an edge-based approach in their work on mesh simplification. To determine whether two triangles are part of the same region in the mesh, two metrics are used. The first metric is the average cosine between two face normals of neighboring triangles. This is a threshold which is intrinsic to the mesh and therefore does not need any user input. The second metric used is a local metric, $\mathscr{D}(e)$, which is derived from an edge and the four corresponding vertices that make up the two faces separated by the edge. The calculation is performed as follows:

$$\mathscr{D}(e) = \begin{bmatrix} \frac{\triangle_{123}(p_4-p_3)\cdot(p_3-p_1)+\triangle_{134}(p_1-p_3)\cdot(p_3-p_2)}{(|p_3-p_1|)^2(\triangle_{123}+\triangle_{134})} \\ \frac{\triangle_{134}}{\triangle_{123}+\triangle_{134}} \\ \frac{\triangle_{123}(p_3-p_1)\cdot(p_1-p_4)+\triangle_{134}(p_2-p_1)\cdot(p_1-p_3)}{(|p_3-p_1|)^2(\triangle_{123}+\triangle_{134})} \\ \frac{\triangle_{123}}{\triangle_{123}+\triangle_{134}} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} \quad (1)$$

where p_i is the position of one of the four corner vertices of the neighbouring triangles labelled in counterclockwise order as can be seen in Figure 1, and $\triangle_{ijk}$ is the area of the triangle build by corner vertex i, j and k. The L_2-norm of this metric represents the dihedral angle between the two neighboring faces. If the value is equal to zero, then a dihedral angle between the two faces is 180°. If the value is larger than zero, then the angle is smaller than 180°. These two properties are inversely proportional. This metric is compared with a user-given threshold. By combining these two metrics, a good segmentation result is retrieved. For more details, we refer the interested reader to Wang et al. [Wan24].

Wang et al. [Wan24] introduced a post-processing step to improve their segmentation results by removing the so-called "islands", small areas that consist of only very few faces. Next to the user-defined threshold for the local metric $\mathscr{D}(e)$, they therefore introduce a second user-defined threshold, namely the minimum number of faces a segment must contain to not be removed during post-processing. Both parameters may be tuned to improve the segmentation results for a given input. Their approach to avoid small island segments in their segmentation approach is to regroup small islands into segments that are already large enough. Wang et al. are not clear on how their algorithm handles these islands. It is stated that these islands check their neighboring faces for a correct segment, but to which segment the faces are then regrouped is not explained.

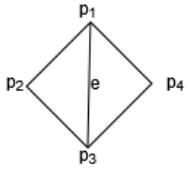


Figure 1: Point orientation for Wang et al. [Wan24]

We evaluated two different regrouping strategies for their suitability in our proposed simplification algorithm. In the first strategy, we refer to as *instant regrouping*, a face belonging to an island first examines its three neighboring faces to determine their affiliation, ignoring those that are also part of islands. Then it checks whether the remaining neighbors belong to different segments. If all belong to the same segment, the face is reassigned to that segment. If there are multiple neighboring segments, the algorithm evaluates which segment shares the most edges with the current face. If a dominant segment is identified, the face is reassigned to it; otherwise, it joins an arbitrary neighboring segment. This process helps minimize the jagged boundaries between the segments. The method is repeated iteratively until all islands are resolved. The second strategy, which we refer to as *iterative regrouping*, is similar to *instant regrouping*, differing only in the way the regrouping is handled. In *instant regrouping* the segment of the face is instantly changed. In *iterative regrouping* the face only saves its new segment but does not change its affiliation yet. Instead, we iterate over all faces and determine the segment for each of them if possible. Only when all faces have determined their new segment will they change their affiliation. This strategy ensures that newly grouped faces do not influence their neighbors, which could otherwise lead to a shift of the borders. If faces are still part of island groups, a new iteration starts. The disadvantage of this strategy is that the borders are more jagged.

3.1.2 Greedy Normal-based Segmentation

In order to find a suitable segmentation of a mesh, different variations of greedy approaches were evaluated. Each approach starts with selecting an arbitrary face, the *reference face* that is currently not part of a segmented group. The face normal of the reference face is compared to the face normal of each of its neighboring faces. If the angle difference between the two normals is smaller than a user-defined threshold, the face is added to the current growing area and is added to a queue that is used for the greedy breadth first search.

The result of this greedy normal-based segmentation approach strongly depends on the position of the face that is arbitrarily chosen as a reference face in each iteration. If a reference face is part of a low-curvature area, it will grow the area on this low-curvature part

Phase	Our Normal-based Extraction	Edge-based Extraction
Input processing	build vertex neighbourhoods, build face neighbourhoods	build vertex neighbourhoods, build face neighbourhoods
⇓	⇓	⇓
Segmentation	face normal-based greedy, multiple seeded segmentations	edge-based greedy, by Wang et al.
⇓	⇓	⇓
Segmentation post-processing	extract intersection of segmentations	merge small regions
⇓	⇓	⇓
Edge recognition	identify the boundary of intersection peel boundary	identify corner vertices, connect corners properly

Table 1: Overview over the steps of the two algorithms in comparison.

of the mesh. However, if the mesh is noisy or contains artifacts that dilute sharp features of the mesh, then the change in normals between each face might be smaller than the user-defined threshold, and the area grows around features in a mesh, which are then not detected. In order to resolve this and make this approach more robust to corrupted input data, multiple iterations of the greedy approach are executed with different randomly selected reference faces. The different results from these iterations are then used to find the edges as described in Section 3.2.

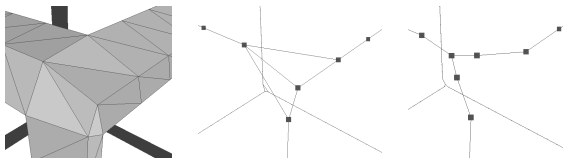
One advantage of this normal-based segmentation strategy is that iterations work independently of each other. This allows us to parallelize this approach, significantly speeding up the segmentation. For the examples presented in this paper, we chose five iterations of this segmentation, which resolved all the features present in each of the example meshes.

For this paper, we decided to use c++ as the implementation language. We did not use special libraries.

3.2 Feature extraction

Various methods for retrieving edges, edge paths, and corner vertices were evaluated.

After applying a normal-based segmentation, we initially used the reference face of the segmentation areas to create planes based on face normals. Although effective for simple meshes like a cube, it lacked clear borders, resulting in well-defined planes but missing edges and corners. Intersecting these planes identified edges (from two-plane intersections) and corners (from three-plane intersections), but this led to many false positives, especially in complex shapes. Filtering them out effectively proved to be difficult. Better results were achieved using segmentation areas to extract border vertices as starting points for edge recognition. This method worked well on clean meshes but failed for meshes which were corrupted by noise or artifacts. To improve robustness, multiple segmentations were combined, each using different random reference faces. A vertex was identified as border vertices only if it was identified as a border vertex in all segmentations. Although this reduced false positives, it also removed valid border vertices due to slight segmentation shifts. To further increase robustness, a one-ring neighborhood was added before intersection, solving the problem of missing border edges. This solution creates broader border areas instead of border regions containing only a polygon path. An example of a segment of this region can be seen in Figure 2a. To refine this, a *peeling* process was applied: in each iteration, vertices on the hull of the region (with both inner and outer neighbors) were marked for removal. This process was iteratively continued until only a thin border in the form of a polygon path remained, which included some triangles which cannot be *peeled* by the algorithm. This can be seen in Figure 2b. To remove the *residual faces* a new vertex is created in the face and used as a representative. Then, all edges of the triangle are deleted. The corner vertices are also deleted if they are not needed for another edge. The final result are connected polygon lines along the features of the input shape. This can be seen in Figure 2c.



(a) (b) (c)

Figure 2: Peeling process

Similarly, we explore two approaches to extract a sparse polygon from the segmentation by Wang et al. [Wan24]: In the first approach, we describe the edges by defining all vertices that are part of at least two segments as edge vertices. In a second approach, we define all vertices that lie on the border of three different segments as corner vertices. Edges between corner vertices are added if they share at least two regions. Defining corner vertices typically leads to the

shape being represented by sparse polygon meshes. However, this approach fails if a mesh does not contain corners, as can be seen in the example of a cylinder in Section 4.1. For this paper, we used the edge approach to derive a reliable feature detection for any input shape and generate results robustly independent of the quality of the shape data. We use the *instant regrouping* approach for post-processing to generate smoother borders, as discussed in Section 3.1.1.

4 RESULTS

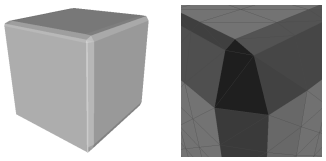
We carried out experiments on a range of simple geometries, each illustrating different shape feature scenarios.

In order to test the proposed algorithm and compare the results with the example algorithm from Wang et al. [Wan24], both algorithms are applied to triangular meshes of different geometric shapes which are corrupted by various levels of noise or are warped by artifacts. As an input value for D, which is needed for the edge-based approach, we tested several values and came to the conclusion that $D = 0.001$ creates the best results. For the normal-based approach,different angles were used for the examples, since the angle that worked well for one approach did not produce good results for all the other meshes.

4.1 Experiments on clean meshes

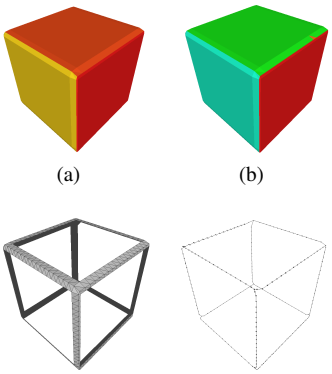
We conducted experiments on a cube mesh. To add a degree of difficulty typically found in real-world engineering data, the edges of the cube were flattened. Such scenarios make it difficult to convert shape data into the very sparse representations which we aimed for in this work. The input mesh for this experiment is shown in Figure 3a. An enhancement of one corner is shown in Figure 3b. To increase the visibility of the flattened corner, a shader was applied. The angle difference we used for this mesh is 40°. Two results for the segmentation approach discussed in Section 3.1.2 can be seen in Figure 4a and Figure 4b, respectively. For both results, the main faces of the cube are segmented correctly but differences in the area of the edge features can be seen. The reason for this is the flattened edges of the input mesh which lead to different representations. In Figure 4c, the intersection of all segmentation boundary regions was extracted as explained in Section 3.1.2. The core geometry of a cube is already visible there. The result of the peeling process and the removal of the triangles as described in Section 3.2 is shown in Figure 4d.

In comparison, the segmentation result of Wang et al. [Wan24] can be seen in Figure 5. The segmentation result is shown before, Figure 5a, and after post-processing, Figure 5b. Every vertex that is part of at least three segments is regarded as a corner vertex of the sparse polygon. We introduced edges between these corner vertices if two of at least three segments are the same. The sparse polygonal representation, which is generated in this way, can be seen in Figure 5c. The polygonal representation of the shape shows some degree of distortion, which originates from the flattened edges of the input mesh, which causes a shift of the corner vertices. The derived polygonal representations here demonstrate that features are detected, which are captured as polygon chains. The results shown in the section are therefore not true polygon meshes. However, from these polygonal representations we can derive sparse polygon meshes simply by converting polygon chains to single edges. An example in which this has been done is shown in Figure 11d for the polygonal representation example shown in Figure 11c.



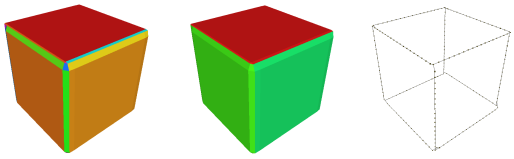
(a) (b)

Figure 3: Input mesh



(a) (b) (c) (d)

Figure 4: Results for the cube input mesh with the proposed algorithm described in Section 3.1.2.



(a) (b) (c)

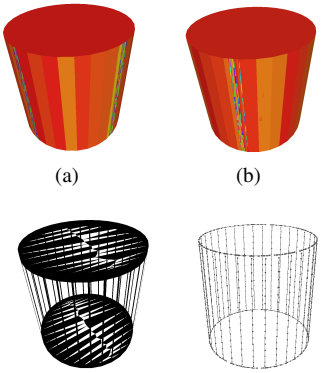
Figure 5: Results for the cube input mesh with the algorithm from Wang et al. [Wan24].

To test our approach for curved shapes, we applied the algorithms to a triangulated cylinder mesh shown in Figure 6. For this mesh, we used an angle difference of 10°. Figure 7a and Figure 7b show two of the segmentation iterations. It can be seen that some of the faces

are fragmented in this segmentation. However, this is resolved through the intersection of all boundary sets, which can be seen in Figure 7c. The result of the peeling and removal of the triangles can be seen in Figure 7d.

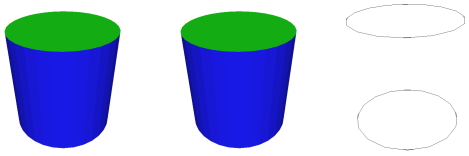
In comparison, the segmentation results for the approach by Wang et al. [Wan24] are shown in Figure 8c, before 8a and after 8b post-processing, respectively. Their algorithm is designed to segment curved and pseudo-curved faces together, which leads to the whole lateral surface of the cylinder being treated as one segment. This is a good result for some fields of application, but it leads to only extracting two not connected circles as can be seen in Figure 8c.

Figure 6: Cylinder input mesh



(a) (b) (c) (d)

Figure 7: Results for the cylinder input mesh with the proposed algorithm described in Section 3.1.2.



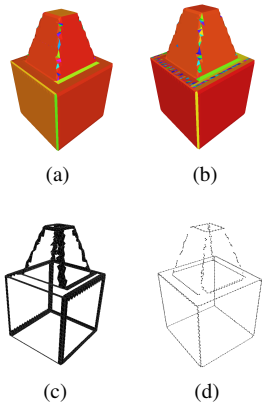
(a) (b) (c)

Figure 8: Results for the cylinder input mesh with the algorithm from Wang et al. [Wan24].

To assess the detection of surface features, we analyzed a cube with a truncated pyramid on its top surface. The input mesh can be seen in Figure 9. The angle difference we choose for this mesh is 30°. Two segmentation results of the proposed algorithm can be seen in Figure 10a and Figure 10b. The generated intersection of these border regions can be seen in Figure 10c. The final border polygon, which still consists of some triangles, can be seen in Figure 10d.

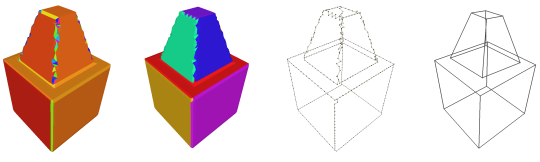
The segmentation results of Wang et al. [Wan24], are shown in Figure 11a. This segmentation still contains some small islands on the abraded edges of the input mesh. These islands are resolved in the post-processing step, Figure 11b. The edge polygon recovered from this segmentation can be seen in Figure 11c. The simpler corner-based polygon can be seen in Figure 11d. In this case, both algorithms result in two polygonal representations that are not connected. Deriving a polygonal mesh from this result is less straightforward than in previous examples, since the algorithm cannot extract a connected polygonal representation from the input. An approach could be to use topological information to extract these edges. Another way could be to check connectivity in the extracted polygons and, if they are disconnected, add edges.

Figure 9: Cube with truncated pyramid input mesh



(a) (b) (c) (d)

Figure 10: Results for the truncated pyramid input mesh with the proposed algorithm described in Section 3.1.2.



(a) (b) (c) (d)

Figure 11: Results for the truncated pyramid input mesh with the algorithm from Wang et al. [Wan24].

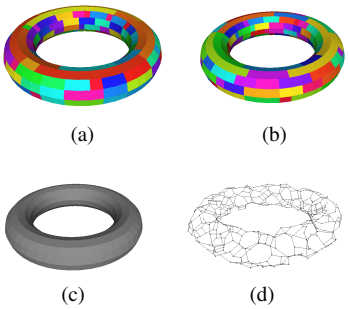
The last clean shape, on which the algorithms are tested, is a classic torus, Figure 12.

Two of the segmentation results for the normal-based segmentation algorithm can be seen in Figure 13a and Figure 13b. The segmentation results consist of many different patches of regions. The reason for this is the

way that this algorithm greedily finds the regions by comparing them to the reference face where this region started to grow. Due to the curvature, the normals for faces that should be part of the same region can strongly change, which results in this segmentation. The strong fragmentation of the regions leads to the classification of every face as a border region, as seen in Figure 13c. This results in a problem where the algorithm tries to peel the border but only generates a not very useful result. The peeling on this border region does not give any useful results because there is no outer hull where the peeling can start, as shown in Figure 13d.

If the segmentation approach by Wang et al. is applied to the torus, a less partitioned result is created, as shown in Figure 14a. The post-processing for this segmentation also yields the same result as without post-processing as shown in Figure 14b. From here, several rings are found, but without connections between them, as shown in Figure 14c. Similarly to the previous example, to derive a true polygon mesh from this, additional edges would need to be introduced carefully.

Figure 12: Torus input mesh



(a) (b) (c) (d)

Figure 13: Results for the torus input mesh with the proposed algorithm described in Section 3.1.2.

(a) (b) (c)

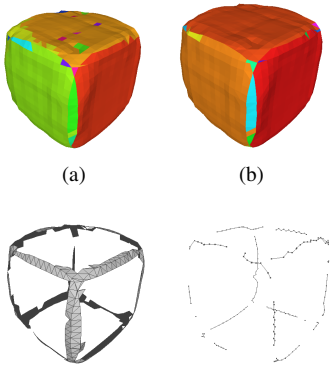
Figure 14: Results for the torus input mesh with the algorithm from Wang et al. [Wan24].

4.2 Experiments on corrupted meshes

In this section a closer look is taken on how both approaches handle corrupted data. The corrupted cube is shown in Figure 15. The cube was not subject to particular noise, but the edges have been smoothed during the scanning process, and additional shallow features appear. The goal is to derive a polygonal representation that captures the overall shape but ignores shallow features. We choose an angle difference from 40°. The results for the normal-based approach and the edge-based approach are shown in Figures 16 and 17, respectively. Two results of the segmentation of the normal-based algorithm can be seen in Figure 16a and Figure 16b. The intersection of the border regions of different segmentations can be seen in Figure 16c. It can be seen that the border region is already a good representation of the polygon, but there are some holes in these borders. After the peeling only a small border is left. This border also contains some holes and some triangles, but this is already a good representation of the sparse polygon. An advantage is that the polygon shown in Figure 16d has nearly no distortion.

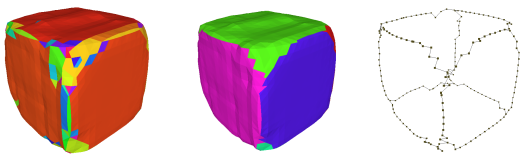
In comparison, the result of the edge-based approach algorithm gives a smoother polygon, see Figure 17a before and with post-processing, shown in Figure 17b. However, edge-based segmentation also leads to a stronger distortion in the final polygonal representation, as can be seen in Figure 17c.

Figure 15: Cube with artefacts input mesh



(a) (b) (c) (d)

Figure 16: Results for the cube input mesh with artifacts with the proposed algorithm described in Section 3.1.2.



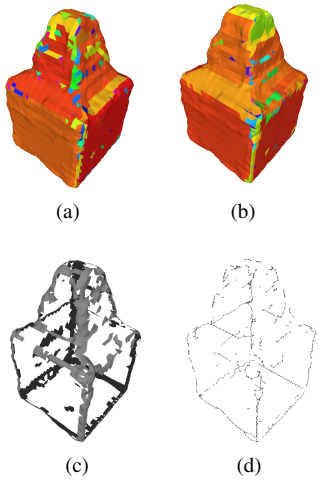
(a) (b) (c)

Figure 17: Results for the cube input mesh with artifacts with the algorithm from Wang et al. [Wan24].

Next, we analyzed both algorithms applied to the corrupted cube with a truncated pyramid on its top surface, as can be seen in Figure 18. We choose the same angle difference as for the clean mesh with 30°. Again, the artifacts by which this shape is corrupted lead to smoothing of the sharp edges, which will not lead to all edges being identified correctly. In Figure 19a and Figure 19b, two iterations of the segmentation can be seen. The result of the intersection can be seen in Figure 19c. The intersection of the boundary regions of the normal-based segmentations contains some holes in the edges, as well as some additional islands due to the artifacts. After peeling, the algorithm returns features that collectively resemble a sparse polygon, but some of the needed edges are missing, as shown in Figure 19d.

The segmentation of this input shape by the edge-based algorithm can be seen in Figure 20a. The corners of the cube part of the mesh are too rounded to be reliably identified. The upper part of the truncated pyramid is strongly fragmented due to artifacts. Post-processing only partly repairs this, so that the result, which can be seen in Figure 20b. By applying the same approach as before to the segmented mesh, the polygon can be seen in Figure 20c is created.

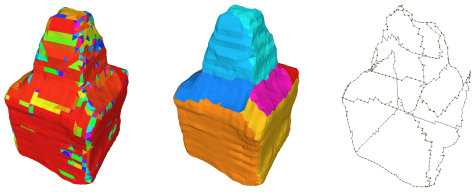
Figure 18: Truncated pyramid with artifacts input mesh



(a) (b) (c) (d)

Figure 19: Results for the truncated pyramid input mesh with artifacts with the normal-based algorithm described in Section 3.1.2.

The last mesh to which the algorithms are applied is the mesh of the cube with the truncated pyramid, but here, the shape is corrupted by Gaussian noise; see Figure 21. For this mesh, we also used the same angle difference at 30°. Two iterations of the segmentation of the proposed algorithm can be seen in Figure 22a and Figure 22b. It can already be seen that the noise leads to stronger fragmentation. This leads to the result that the border region shows only a small reduction compared to the input mesh as shown,as shown in Figure 22c. Peeling can be applied to this border region but does not produce a very good result, as shown in Figure 22d.



(a) (b) (c)

Figure 20: Results for the truncated pyramid input mesh with artifacts with the algorithm from Wang et al. [Wan24].

The edge-based segmentation algorithm is not better suited for this challenge. The result of the segmentation as well as the corresponding post-processing can be seen in Figure 23a and 23b respectively. It can be seen that uniformly distributed noise leads to a wrong segmentation where nearly all faces are classified as one area. Due to this misclassification, it is not possible to extract a polygon from the segmentation result.

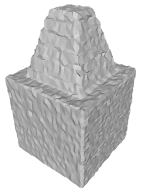
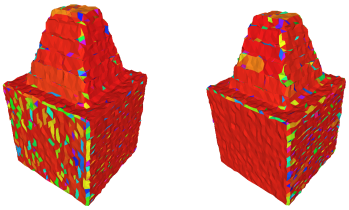
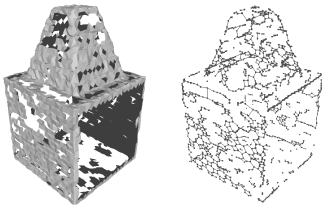


Figure 21: Truncated pyramid with noise input mesh

In order to improve the results, we looked at the effects of adjusting the user-defined normal-angle threshold for the normal-based segmentation. We used the mesh of the cube with the truncated pyramid as an input and added different levels of Gaussian noise. The standard deviation of the noise was derived from the average length of the edges of the input mesh divided by 10, 25, or 50, respectively. The angle thresholds of the normals are 30°, 45°, and 60°. The results for these different configurations can be seen in Table 2. The results for the noise with the highest standard deviation (division factor of 10) still contain a significant amount of noise, and even lowering the normal angle threshold did not lead to good feature identification. However, we were able to derive the most important edges by lowering the normal angle threshold if there was less noise in the input shape. In the image with an angle difference of 30°some artifacts caused by the noise are still present. If the angle difference is increased to 45°the

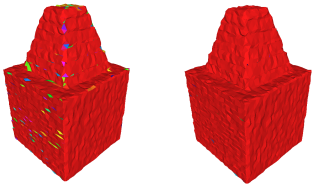


(a) (b)



(c) (d)

Figure 22: Results for the truncated pyramid input mesh with Gaussian noise with the proposed algorithm described in Section 3.1.2.



(a) (b)

Figure 23: Results for the truncated pyramid input mesh with Gaussian noise with the algorithm from Wang et al. [Wan24].

majority of artifacts caused by the noise are filtered out and relevant border edges are found. As soon as the angle difference increases, holes in the border edges are produced. For even smaller noise, the result is already quite similar to the result for the noiseless mesh.

5 CONCLUSION AND FUTURE WORK

In this paper, we introduce the segmentation-based feature extraction algorithm which returns a very sparse polygonal representation for a dense input mesh. To segment the input shape, we compare a normal- and an edge-based approach to find sharp features of a given mesh at intersections of different segments. The goal was to find a robust approach that reliably identifies the overall shape from shape data, which may be corrupted by artifacts or noise. The presented approach therefore focuses on sharp features. One advantage that can be seen in this paper is the higher stability in regard to noise input meshes. Compared to Wang et al. [Wan24], our algorithm extracted some features from the noise input that could be improved by post-processing. The result still extracts some false positives, but this can be a field of future research. One way to improve this algorithm would be to increase the number of segmentation iterations and to determine an appropriate percentile that ensures that all features are identified almost surely.

The results presented are the first and most important step towards extracting very sparse polygon meshes, which represent the overall shape of corrupted input data. Future work will derive clean, very sparse polygon meshes from the polygonal representations derived in this work.

REFERENCES

[Sun02] Sun, Yiyong and Page, David L. and Paik, Joonki and Koschan, Andreas and Abidi, Mongi A. “Triangle mesh-based edge detection and its application to surface segmentation and adaptive surface smoothing”. In: *Proceedings. International Conference on Image Processing*. Vol. 3. 2002, pp. 825–828.

[Wan12] Wang, Xiao-chao and Cao, Jun-jie and Liu, Xiu-ping and Li, Bao-jun and Shi, Xi-quan and Sun, Yi-zhen. “Feature detection of triangular meshes via neighbor supporting”. In: *Journal of Zhejiang University SCIENCE C* 13 (2012), pp. 440–451.

[Liu23a] Liu, Qi and Xu, Shibiao and Xiao, Jun and Wang, Ying. “Sharp Feature-Preserving 3D Mesh Reconstruction from Point Clouds Based on Primitive Detection”. In: *Remote Sensing* 15.12 (2023), p. 3155.

[Hur24] Hurtado, Jan and Gattass, Marcelo and Raposo, Alberto and Lopez, Cristian. “Sharp feature-preserving mesh denoising”. In: *Multimedia Tools and Applications* 83 (2024), pp. 69555–69580.

[Wan24] Wang, Weijia and Pan, Wei and Dai, Chaofan and Dazeley, Richard and Wei, Lei and Rolfe, Bernard and Lu, Xuequan. “Segmentation-driven feature-preserving mesh denoising”. In: *The Visual Computer* 40 (2024), pp. 6201–6217.

[Cal23] Calatroni, Luca and Huska, Martin and Morigi, Serena and Recupero, Giuseppe Antonio. “A Unified Surface Geometric Framework for Feature-Aware Denoising, Hole Filling and Context-Aware Completion”. In: *Journal of Mathematical Imaging and Vision* 65 (2023), 82â98.

[Kob03] Kobbelt, Leif and Botsch, Mario. “Feature sensitive mesh processing”. In: *Proceedings of the 19th Spring Conference on Computer Graphics*. Association for Computing Machinery, 2003, 17â22.

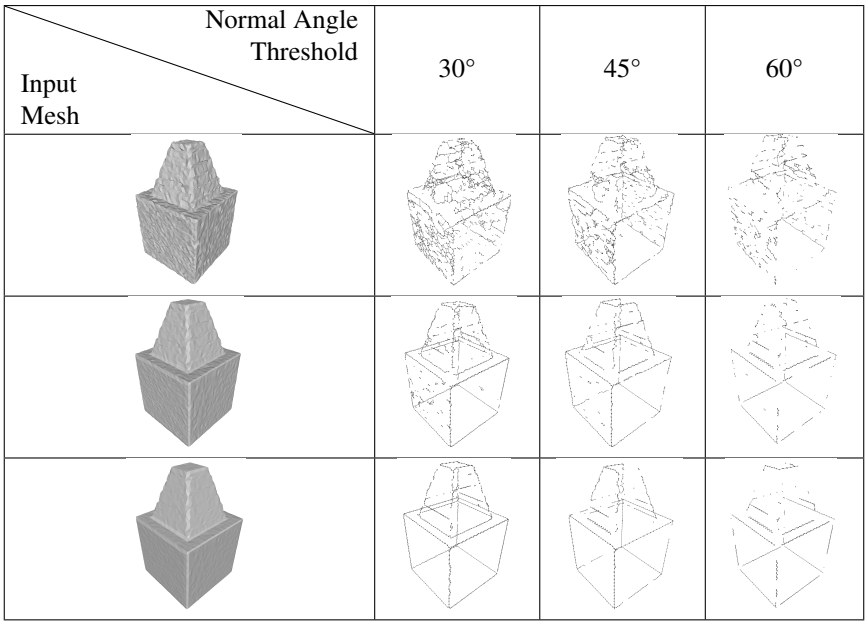


Table 2: Overview over the influence of the standard deviation of the noise and the angle difference on the feature extraction result

[Fan10] Fan, Hanqi and Yu, Yizhou and Peng, Qunsheng and Ragab, Mohammad Ehab. "Robust Feature-Preserving Mesh Denoising Based on Consistent Subneighborhoods". In: *IEEE Transactions on Visualization and Computer Graphics* 16.2 (2010), pp. 312–324.

[Has15] Hassanein, Allam Shehata and Mohammad, Sherien and Sameer, Mohamed and Ragab, Mohammad Ehab. "A Survey on Hough Transform, Theory, Techniques and Applications". In: *International Journal of Computer Science Issues* 12(1).2 (2015), pp. 139–156.

[Rom22] Romanengo, Chiara and Biasotti, Silvia and Falcidieno, Bianca. "Hough Transform for Detecting Space Curves in Digital 3D Models". In: *Journal of Mathematical Imaging and Vision* 64 (Mar. 2022). DOI: 10.1007/s10851-021-01066-8.

[He 23] He, Qi and Zhao, Qianqian and Zhao, Danfeng and Liu, Bilin and Chu, Moxian. "Fusing Local Shallow Features and Global Deep Features to Identify Beaks". In: *Animals* 13.18 (2023). ISSN: 2076-2615. DOI: 10.3390/ani13182891. URL: https://www.mdpi.com/2076-2615/13/18/2891.

[Liu23b] Liu, Bin and li, bo and Cao, Junjie and Wang, Weiming and Liu, Xiuping. "Adaptive and propagated mesh filtering". In: *Computer-Aided Design* 154 (Sept. 2023), p. 103422. DOI: 10.1016/j.cad.2022.103422.

[Lee04] Lee, Y. and Park, S. and Jun, Yongtae and Choi, W.C. "A robust approach to edge detection of scanned point data". In: *International Journal of Advanced Manufacturing Technology* 23 (Feb. 2004), pp. 263–271. DOI: 10.1007/s00170-003-1695-x.

[Coh04] Cohen-Steiner, David and Alliez, Pierre and Desbrun, Mathieu. "Variational shape approximation". In: *ACM Trans. Graph.* 23.3 (Aug. 2004). ISSN: 0730-0301. DOI: 10.1145/1015706.1015817.

[Sim14] Simari, Patricio and Picciau, Giulia and De Floriani, Leila. "Fast and Scalable Mesh Superfacets". In: *Comput. Graph. Forum* 33.7 (Oct. 2014), 181â190. ISSN: 0167-7055. DOI: 10.1111/cgf.12486.

[Vie05] Vieira, Miguel and Shimada, Kenji. "Surface mesh segmentation and smooth surface extraction through region growing". In: *Computer Aided Geometric Design* 22.8 (2005), pp. 771–792.

[Li 18] Li, Tao and Liu, Wei and Liu, Hao and Wang, Jun and Liu, Ligang. "Feature-Convinced Mesh Denoising". In: *Graphical Models* 101 (Dec. 2018). DOI: 10.1016/j.gmod.2018.12.002.