Volume: 11 Issue: V Month of publication: May 2023
DOI: https://doi.org/10.22214/ijraset.2023.51744

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue V May 2023- Available at www.ijraset.com

Mesh Optimization using Python Libraries

Dr. S. K. Ashok[1], Dhanush M[2], Sanjay R[3]
[#]Department of Automobile Engineering, Dr Mahalingam College of Engineering and Technology

Abstract: In this project, the mesh of a 3D model is optimised using Python tools (Pymeshlab and Blender python). This is accomplished by the "mesh decimation" procedure, which involves lowering the mesh's total number of triangles. The goal of this procedure is to reduce the complexity of the mesh while preserving the general form and geometry of the original model. The project's main objective is to boost real-time applications' performance, which must provide quick rendering and visualisation. The project attempts to optimise the mesh in order to achieve higher frame rates and better animations, which can enhance the user experience as a whole. It is important to note that the project's main goal is to optimise the mesh without modifying the model's topology or geometry. This indicates that the original structure and shape.

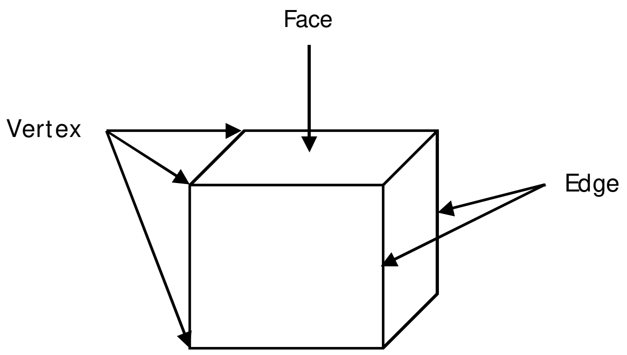
Keywords: Mesh Optimization, Vertex, Edge, Face, Mesh Decimation, Performance, Visualization

I. INTRODUCTION

In the field of 3D modeling, the optimization of meshes is an important task that can greatly improve the performance of rendering and visualization. Mesh decimation is a widely used technique for reducing the complexity of 3D models, while maintaining their topology and geometry. To accomplish this task, various software tools and programming languages are available. In this project, we will be using two popular Python libraries, Pymeshlab and Blender Python, to optimize the mesh of a 3D model. Our goal is to achieve the best possible performance in rendering and visualization for real-time applications by reducing the complexity of the mesh without compromising the model's topology and geometry. This project will showcase the power and flexibility of Python in the field of 3D modeling and its ability to enhance the performance of real-time applications.

II. MESH

A 3D mesh is a structure composed of vertices, edges, and faces that describe the shape of a 3D object in computer graphics. It can be visualized as a network of interlinked triangles or polygons that form the surface of the model. Every vertex represents a coordinate point in 3D space, and each face is formed by connecting vertices that enclose a closed polygon. The edges establish the connections between the vertices, outlining the shape and structure of the mesh. Using various software and programming languages, the mesh can be manipulated to create, edit, and render 3D models.



1) *Face:* The face of a single triangle mesh is a single planar triangle defined by the three vertices.
2) *Edges:* The edges of a single triangle mesh are the three line segments connecting the three vertices.
3) *Vertices:* The vertices of a single triangle mesh are the three points in 3D space that define the corners of the triangle.

III. MESH FORMAT

1) *STL (Stereo Lithography)* - This format is commonly used in 3D printing and rapid prototyping applications. It represents a surface mesh as a collection of triangular facets.
2) *OBJ (Wave front)* - This format is widely used in 3D graphics and animation software. It can represent both surface and volumetric meshes and stores vertex coordinates, texture coordinates, and polygonal face definitions.
3) *PLY (Stanford Polygon Format)* - This format is commonly used in computer vision and robotics applications. It can store both surface and volumetric meshes, along with vertex attributes such as color and normal vectors.

IV. MESH DECIMATION

Mesh decimation is a technique used in computer graphics and computational geometry to simplify a complex 3D mesh by reducing the number of vertices and faces while preserving the essential features of the original mesh. Mesh decimation is also known as mesh simplification or mesh reduction.

The mesh decimation process starts by selecting a subset of vertices and faces in the mesh and collapsing them into a single vertex. This process is repeated iteratively until the desired level of simplification is achieved. During the decimation process, the mesh is checked to ensure that the essential features of the original mesh are preserved, such as surface curvature, topology, and texture coordinates.

The benefits of mesh decimation include reduced storage requirements, faster rendering times, and improved performance of algorithms that rely on the mesh data. Mesh decimation is commonly used in computer games, virtual reality, and other interactive applications where real-time rendering is required. It is also used in scientific and engineering simulations where large meshes need to be processed efficiently.

V. PYTHON LIBRARY

Mesh decimation is a technique used in computer graphics and computational geometry to simplify a complex 3D mesh by reducing the number of vertices and faces while preserving the essential features of the original mesh. Mesh decimation is also known as mesh simplification or mesh reduction.

A. Pymeshlab

PyMeshLab is a Python library that enables users to interact with MeshLab, a portable, open-source system for processing, analyzing, and visualizing 3D meshes. This library provides a simple interface for Python users to perform a variety of mesh processing tasks, including cleaning, simplification, smoothing, and feature extraction. Additionally, PyMeshLab includes advanced algorithms for mesh alignment, registration, and segmentation. The library supports multiple file formats for meshes, including STL, OBJ, PLY, and OFF. PyMeshLab is a valuable tool for researchers, engineers, and artists working with 3D data due to its integration with MeshLab and access to a wide range of advanced mesh processing algorithms.

B. Blender Python

Blender Python API provides functions for decimating a mesh, such as bpy.ops.mesh.decimate(), which can be used to reduce the polygon count while maintaining the overall shape of the mesh.

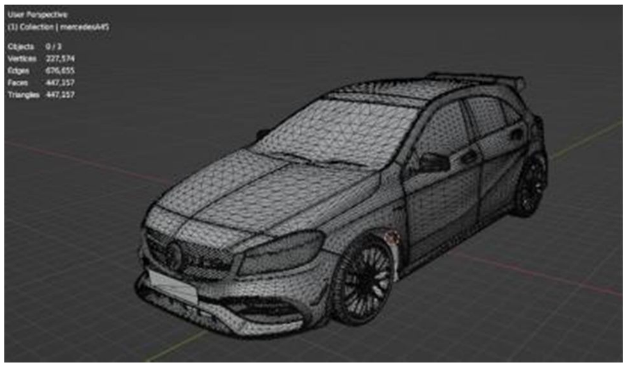
VI. PYMESHLAB AND BLENDER PYTHON

Pymeshlab and Blender are both powerful tools for 3D modeling and mesh optimization, but they have different capabilities. Pymeshlab is a Python library specifically designed for mesh processing and provides a broad range of functions including mesh simplification, smoothing, remeshing, and Boolean operations. It also allows users to convert between different mesh representations and supports various file formats. Pymeshlab is particularly useful for automating complex mesh processing tasks and can be used as a standalone library or integrated into other Python applications.

In contrast, Blender is a 3D modeling and animation software with its own Python API. The Blender Python API enables users to perform various operations on objects, meshes, materials, and other features within Blender. These tasks include creating and modifying objects and meshes, adding materials and textures, rendering images and animations, and developing custom tools and add-ons. Both Pymeshlab and Blender libraries offer mesh optimization capabilities such as simplifying meshes, removing redundant geometry, and optimizing topology.

VII. INPUTS

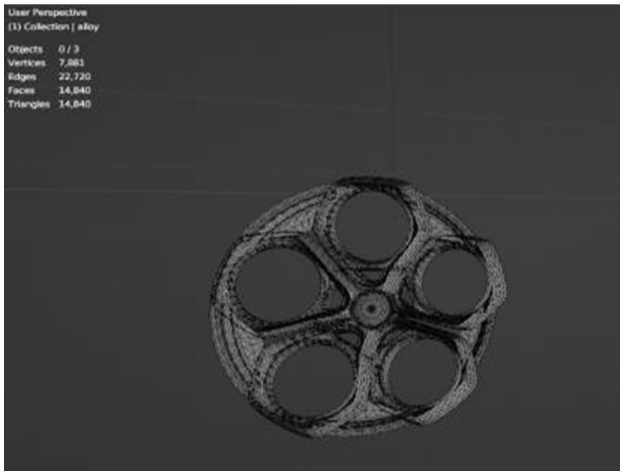
As input we have chosen a STL file of a Mercedes A45. The input mesh file images are displayed below,



```
User Perspective
(1) Collection | mercedesA45

Objects    0 / 3
Vertices   227,574
Edges      676,655
Faces      447,157
Triangles  447,157
```

Fig. 1 Input STL file

The above loaded STL file has initial face count of 447,157 faces.
Secondly we have chosen an alloy wheel as our input,



```
User Perspective
(1) Collection | alloy

Objects    0 / 3
Vertices   7,881
Edges      22,720
Faces      14,840
Triangles  14,840
```

Fig. 2 Input STL file

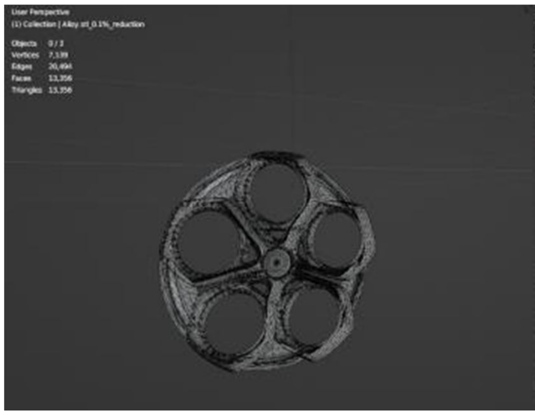
The above loaded STL file has initial face count of 14,840 faces.

VIII. PYMESHLAB MESH DECIMATION

The PyMeshLab function ms.meshing_decimation_quadric_edge_collapse() is a mesh simplification algorithm that employs the "quadric edge collapse" technique to reduce the number of polygons in a 3D mesh while retaining its overall appearance and shape. This method involves calculating a "quadric error metric" for each vertex in the mesh, which considers the position of neighboring vertices and the angles between adjacent edges. This metric determines how much the mesh's shape would be altered if a particular vertex were removed. The algorithm then systematically collapses edges, beginning with those that have the smallest quadric error metric, resulting in a mesh with a reduced number of polygons while still retaining its structural integrity.

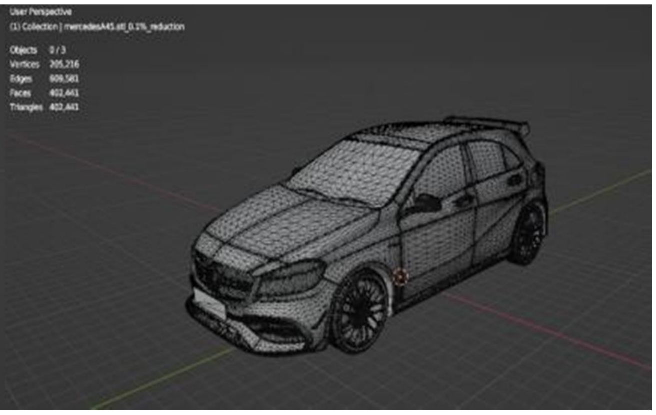
The meshing_decimation_quadric_edge_collapse() function in PyMeshLab is a mesh simplification algorithm that reduces the complexity of a 3D mesh by collapsing edges based on a specified ratio value. This can be useful for improving the performance of the mesh during 3D visualization and rendering processes. The algorithm works by calculating a "quadric error metric" for each vertex in the mesh, which determines how much the mesh's shape would be altered if a particular vertex were removed. The algorithm then collapses edges with the smallest quadric error metrics until the desired ratio of mesh is achieved. The final decimated output will have a reduced number of faces and vertices while preserving the overall appearance and shape of the original mesh.

The below images represents the 10% , 50% and 90% of reduction in mesh using pymeshlab.



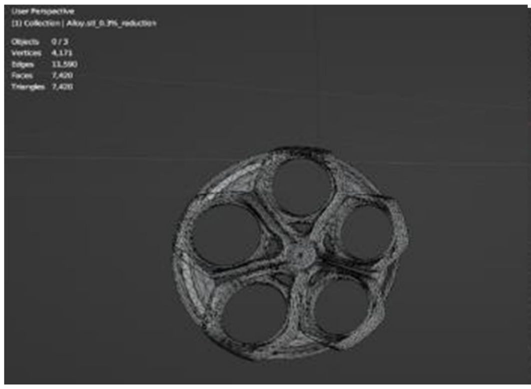
```
User Perspective
(1) Collection | Alloy.stl_0.1%_reduction

Objects    0 / 3
Vertices   7,139
Edges      20,494
Faces      13,356
Triangles  13,356
```

Fig. 3 Reduction percentage – 10% (13,356 faces)



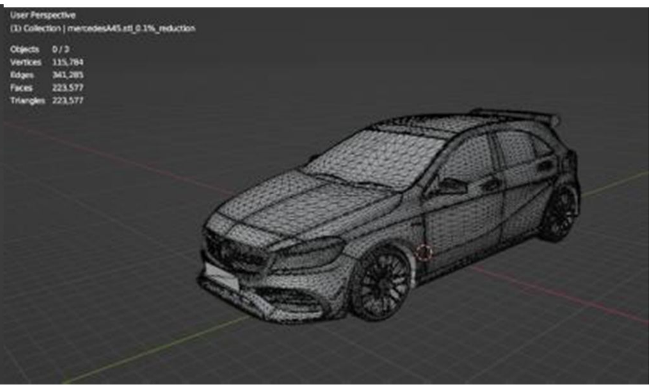
```
User Perspective
(1) Collection | mercedesA45.stl_0.1%_reduction

Objects    0 / 3
Vertices   205,216
Edges      609,581
Faces      402,441
Triangles  402,441
```

Fig. 4 Reduction percentage – 10% (402,441 faces)



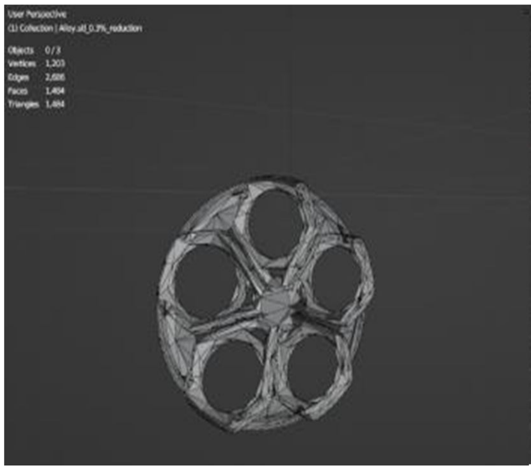
```
User Perspective
(1) Collection | Alloy.stl_0.3%_reduction

Objects    0 / 3
Vertices   4,171
Edges      11,590
Faces      7,420
Triangles  7,420
```

Fig. 5 Reduction percentage – 50% (7,420 faces)



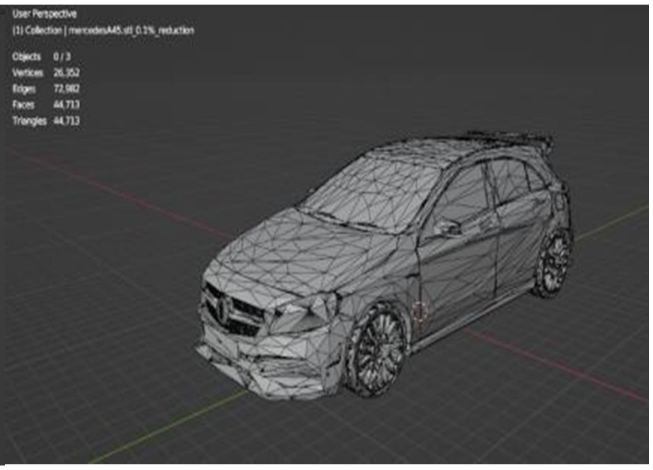
```
User Perspective
(1) Collection | mercedesA45.stl_0.1%_reduction

Objects    0 / 3
Vertices   115,784
Edges      341,285
Faces      223,577
Triangles  223,577
```

Fig. 6 Reduction percentage – 50% (223,577 faces)



```
User Perspective
(1) Collection | Alloy.stl_0.3%_reduction

Objects    0 / 3
Vertices   1,203
Edges      2,606
Faces      1,484
Triangles  1,484
```

Fig. 7 Reduction percentage – 90% (1484 faces)



```
User Perspective
(1) Collection | mercedesA45.stl_0.1%_reduction

Objects    0 / 3
Vertices   26,352
Edges      72,982
Faces      44,713
Triangles  44,713
```

Fig. 8 Reduction percentage – 90% (44,713 faces)

The above mentioned images display about the decimated face mesh file. The decimation is performed for various ratio parameters, The ratio displayed are from 10 percentage to 90 percentage of reduction in face count of the mesh. Displayed images are of ratio 10 percentage reduction to 90 percentage of reduction in mesh.

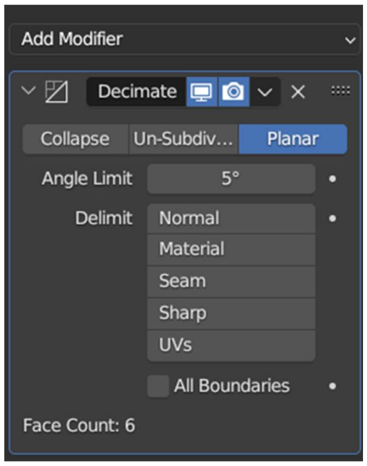
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue V May 2023- Available at www.ijraset.com

IX. BLENDER MESH DECIMATION

In Blender, mesh decimation can be achieved through the use of the Decimate modifier or the Simplify tool. Both of these tools can be accessed through the Blender Python API, allowing for automated or scripted decimation of meshes.

Planar decimation is a technique used in Blender to simplify complex 3D meshes with a large number of vertices that may cause performance issues during rendering. This process involves reducing the number of vertices while retaining the planar surfaces of the mesh. The technique works by identifying and removing redundant vertices that lie close to each other and are on the same plane. As a result, the final mesh is simplified, but the original shape and topology are still preserved.

Planar decimation takes angle difference as input. The angle represents the adjacent face angles which are less than the respective angles are treated as coplanar faces.



Angle Limit: 5°
Face Count: 6

Fig. 9 Planar Decimation option

A. *Blender Decimation Outputs*

The Angle limit parameter in Blender's planar decimation determines the greatest angle at which two neighbouring faces are regarded as coplanar. The Decimate modifier, which may be used to simplify a 3D mesh by reducing the number of vertices while maintaining its general shape and topology, includes this option.

When dealing with meshes that contain non-planar faces, the angle limit parameter can be quite helpful. In certain circumstances, employing the Planar option alone to reduce the number of vertices may produce deformed or uneven surfaces. The Decimate modifier can remove vertices that are not required for the overall shape of the mesh while preserving nearby faces with similar normals by specifying an acceptable angle limit.

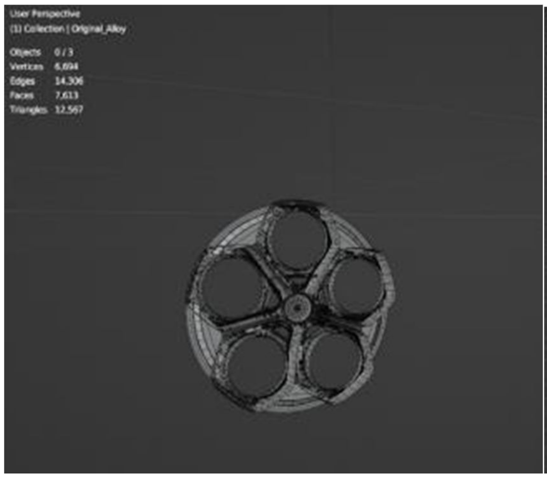

```
User Perspective
(1) Collection | Original_Alloy

Objects    0 / 3
Vertices   6,694
Edges      14,306
Faces      7,613
Triangles  12,567
```

Fig. 10 Angle limit – 1 degree (7613 faces)

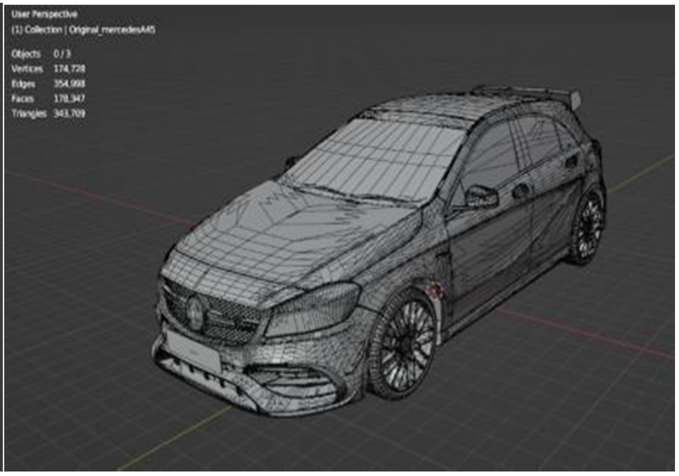
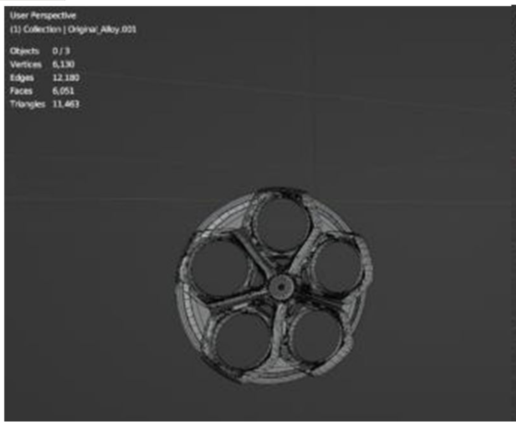
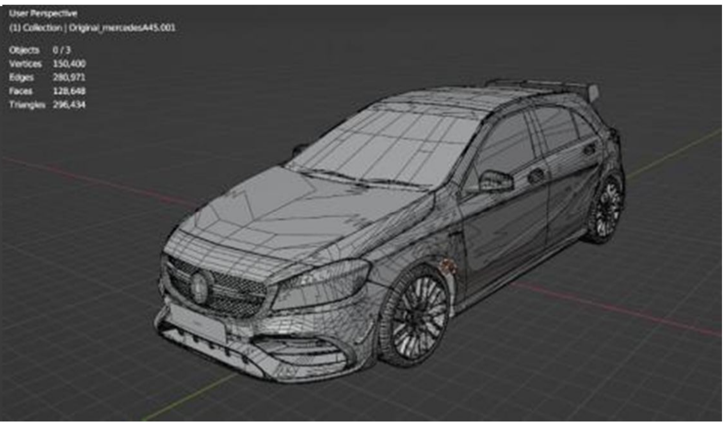

```
User Perspective
(1) Collection | Original_mercedesA45

Objects    0 / 3
Vertices   174,728
Edges      354,998
Faces      178,347
Triangles  343,709
```

Fig. 11 Angle limit – 1 degree (178,347 faces)



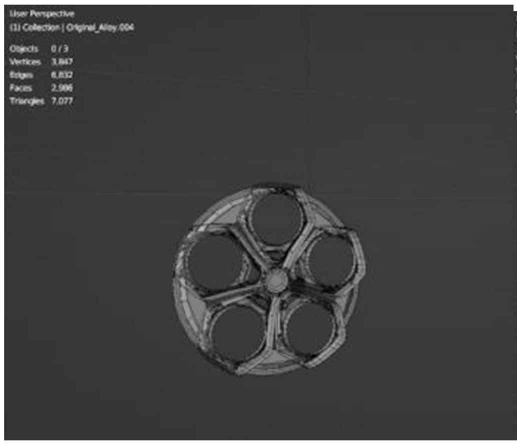
```
User Perspective
(1) Collection | Original_Alloy.001

Objects    0 / 3
Vertices   6,130
Edges      12,180
Faces      6,051
Triangles  11,463
```



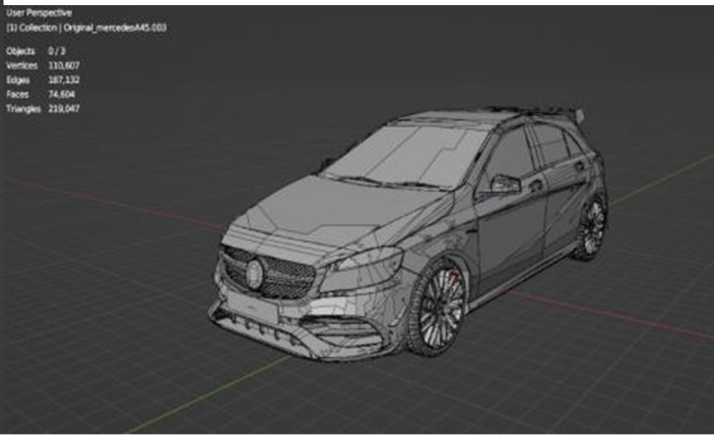
```
User Perspective
(1) Collection | Original_mercedesA45.001

Objects    0 / 3
Vertices   150,400
Edges      280,971
Faces      128,648
Triangles  296,434
```

Fig. 12 Angle limit – 2 degree (6051 faces)

Fig. 13 Angle limit – 2 degree (128648 faces)



```
User Perspective
(1) Collection | Original_Alloy.004

Objects    0 / 3
Vertices   3,847
Edges      6,832
Faces      2,986
Triangles  7,077
```



```
User Perspective
(1) Collection | Original_mercedesA45.003

Objects    0 / 3
Vertices   110,607
Edges      187,132
Faces      74,604
Triangles  219,047
```

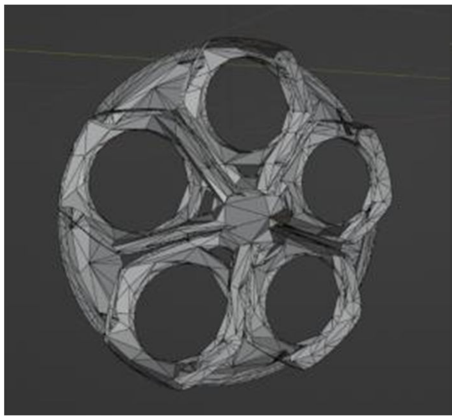
Fig. 14 Angle limit – 5 degree (2986 faces)

Fig. 15 Angle limit –5 degree (74604 faces)

The above mentioned images represent the angle limit 1 degree, 2 degree and 5 degree in blender mesh decimation.

X. DEVIATION COMPARISON

The decimated mesh files are compared here with the most deviated surface or region in pymeshlab as well as blender python. Both the library has deviation in geometry but the comparison images displays you the visual appearance of the decimated mesh after decimation is performed.



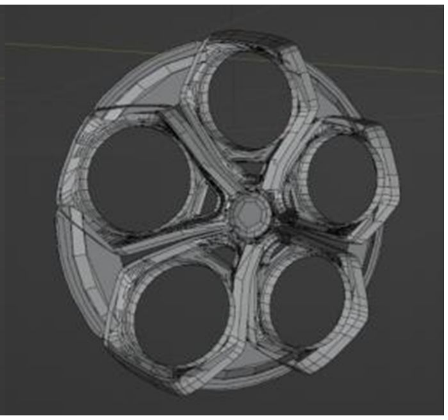


Fig. 14 Pymeshlab Deviation – 90%

Fig. 14 Blender Deviation – 5degree

XI. CONCLUSION

The output mesh's correctness and quality might be significantly impacted by the software tool you choose, according to the conclusion. Comparatively to whole mesh decimation, planar decimation was found to be more effective at reducing the number of polygons in a mesh. Additionally, compared to Pymeshlab, Blender generated a mesh with less distortion or deviation. So selecting the right software tool is essential if you want accurate and high-quality output meshes.

The difference in deviation between the two software programmes demonstrates how choosing the right tool may have a significant impact on the accuracy and calibre of the output mesh. The results produced by pymeshlab and blender were compared, and it was found that using planar decimation rather than decimating the entire mesh was more effective at reducing the number of polygons in a mesh. Additionally, using blender as opposed to pymeshlab resulted in a mesh with less distortion or deviation.

REFERENCES

[1] Ruud M. Bolle and Baba C. Vemuri. On three-dimensional surface
[2] T. DeRose, H. Hoppe, T. Duchamp, J. McDonald, and W. Stuetzle.
[3] Gene Golub and Charles Van Loan. Matrix Computations. John Hopkins University Press, 2nd edition, 1989.
[4] J.V. Miller, D.E. Breen, W.E. Lorensen, R.M. O’Bara, and M.J. Wozny.Geometrically deformed models: A method for extracting closed ge-ometric models from volume data. Computer Graphics (SIGGRAPH’91 Proceedings), 25(4):217–226, July 1991.
[5] William Schroeder, Jonathan Zarge, and William Lorensen. Decima-tion of triangle meshes. Computer Graphics (SIGGRAPH ’92 Pro-ceedings), 26(2):65–70, July 1992.
[6] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle.Mesh optimization. TR 93-01-01, Dept. of Computer Science andEngineering, University of Washington, January 1993
[7] R. B. Schudy and D. H. Ballard. Towards an anatomical model ofheart motion as seen in 4-d cardiac ultrasound data. In Proceedingsof the 6th Conference on Computer Applications in Radiology andComputer-Aided Analysis of Radiological Images, 1979.