This is a previous version of the article published in The International Journal of Advanced Manufacturing Technology. 2023. https://doi.org/10.1007/s00170-023-11361-w

Implementation of efficient surface discretisation algorithms adapted to geometric models specific to the footwear industry

Eduardo Calabuig-Barbero,[a,1] German Martinez-Martinez,[a] Jose-Luis Sanchez-Romero,[b] Antonio Jimeno-Morenilla,[b] Vicente Lopez-Martin,[a] Higinio Mora-Mora [b]

[a] *INESCOP - Footwear Technology Center, Elda, 03600, Spain*
[b] *Department of Computer Technology, University of Alicante, San Vicente del Raspeig, 03690, Spain*

Abstract

In 3D modelling, a surface mesh is a collection of vertices, edges, and faces that define the shape of a 3D object. The surface mesh is typically used to represent the outer surface of an object, as opposed to the internal structure. A surface mesh is usually defined as a polygon mesh, which is a collection of polygons (triangles or quadrilaterals are the most common) that are connected at their vertices. The vertices of the mesh define the shape of the object, and the edges and faces provide the topological information that describes how the vertices are connected. Surface meshes are often used in 3D modeling software to create 3D objects for animation, video games, and other purposes.
Although triangular meshes are the most widely used in various fields, they have a series of disadvantages that quad meshes solve in most cases. This article presents a set of algorithms capable of generating quad meshes from any type of input triangular mesh, preserving the geometric characteristics of the initial model. This process is known as "remeshing". The results obtained by these algorithms on a large number of models related to the footwear industry have been compared, as well as analysing the advantages and disadvantages of each one of them applied to geometric models commonly used in footwear industry.

Keywords: overlocking, quadrangulation, footwear, geometric singularities

1. Introduction

Modelling using triangular meshes (polygonal meshes made up of triangles) is present in a wide variety of fields, such as computer graphics, geometric modelling,

[1] Corresponding author at ecalabuig@inescop.es

computer vision, various fields of medicine and biology, etc. Although triangular meshes are the most widely used representations, quad meshes (polygonal meshes consisting entirely of quadrilaterals) have many advantages and are progressively gaining acceptance in different fields, for example, in computer-aided design (CAD) where significant progress has been made in recent years (Bommes *et al*., 2012).

Due to the high degree of use of triangular meshes in 3D environments and related research, there is a very important strand of studies working on the concept of converting these triangular meshes into quad meshes. This process is also called "quad remeshing" (Alliez *et al*., 2008). Thanks to this remeshing process a type of quad mesh is obtained, less dense than the initial one and capable of preserving the initial geometrical characteristics of the original triangular mesh that is based on established, validated and widely accepted methods.

There are two basic types of quad meshes:

- Semi-regular mesh. It is also called a *multi-block mesh*, in which the blocks correspond to patches, which in turn are regular submeshes formed by regular 2D arrays of quads. In a semi-regular quad mesh, all vertices that are internal to the patches or along their boundary edges are regular (they have valence 4), while only the vertices at the boundaries and corners of the patches can be singular (valence 3 and 2, respectively). These meshes are very useful as base meshes for tensor product spline or NURBS fitting, where a spline patch is defined for each regular patch of the mesh, and the different patches are joined on common boundaries. In general, tensor product patches obtained from quad-type control meshes are much easier to manipulate than triangle-based Bernstein/Bezier bases. These techniques are important because splines, NURBS and subdivision surfaces are the modelling techniques that appear in many industrial applications, such as CAD/CAM applications for splines and NURBS.

- Semi-regular valence mesh. A mesh is said to be *valence semi-regular* if most of its vertices have valence 4. All semi-regular quad meshes are valence semi-regular, but not all valence semi-regular quad meshes can be divided into a small number of patches.

The two types of quad meshes discussed above make it possible to distinguish between between algorithms that produce meshes with a patchy structure, and algorithms that minimise the number of irregular vertices.

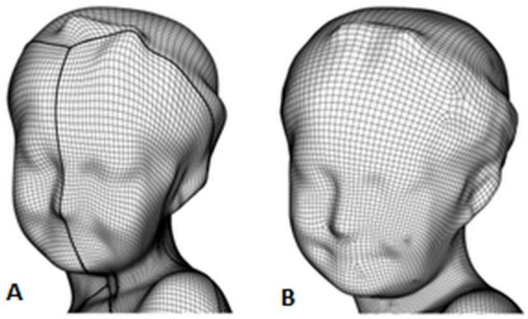


Fig. 1: Different types of meshes. A: Semi-regular mesh. B: Semi-regular valence mesh (Bommes *et al*., 2012).

Some of the main advantages of quad meshes include:

- The quad mesh has a tensor product structure and is, therefore, suitable for fitting splines or NURBS. Since these types of surfaces are very widespread in industrial CAD/CAM applications, quad meshes are ideal for 3D modelling.
- Semi-regular quad mesh patches with a rectangular grid topology (semi-regular valence meshes) naturally match the sampling pattern of all types of textures, from images to displacement maps. Therefore, quad meshes are suitable for texturing.
- Geometries typically have two dominant local directions, usually associated with the principal curvature directions or local sharp features of the surface, to which the quads can be aligned; however, the use of triangular meshes requires an arbitrary choice of one of the edges as the third. The alignment of the elements of a mesh with certain directions is crucial to capture the features of the shape.
- Quad meshes can greatly reduce storage space. Since details are stored in 2D arrays (textures, normal maps, displacement maps) similar to images, they can be reduced in size with standard image compression techniques.

Semi-regular quad meshes have the ability to change the way in which the geometry of different surfaces is stored and processed, thus improving the efficiency of many algorithms. However, the use of this type of meshes also have some disadvantages:

- The number of algorithms that obtain quad meshes that adapt correctly to the given surface is scarce. In addition, they must comply with other aspects that the resulting quad mesh must cover, such as singularities (their good placement has a very important impact on the approximation quality of the

final mesh) and the correct placement of the patches. This is especially evident in geometric models of higher complexity.

- The remeshing times in existing algorithms are usually quite long if high quality results must be achieved. Obviously, this will also depend on the complexity of the geometry.
- Processing geometry on quad meshes is computationally more expensive than on triangle meshes. This is due to the properties of quads, which make them more difficult primitives to process than triangles. A triangle is always convex and flat, can be easily projected onto a plane, and is relatively easy to rasterise. In the case of quads, the operations are more complex since, unlike triangles, a quad may not be flat, flat quads may not be convex, and their rendering requires a more complex rasteriser than in the case of triangles.
- Some of these problems mean that many automatic methods do not achieve results that are obtained by skilled professional modellers. Another option is to use hybrid methods, where an automatic remeshing is performed, after which the user has the option to alter the remeshing with tools such as manual repositioning of singularities or patch boundaries, "guides" to direct the final remeshing, choice of zones to obtain a higher or lower density of quads, etc.

1.1 Use of meshes in footwear design

Most geometric models found in the footwear industry have an organic origin. Therefore, they are made up of soft meshes, without characteristic or reduced line, and also with meshes with a greater number of sharp features. Consequently, the mesh generation and mesh remeshing algorithms must perform well with both types of meshes, in order to obtain the most realistic models possible. The problem is that not all current algorithms obtain quality results for both mesh types.

Using quad meshes for geometric models in the footwear industry offers the same advantages as for models in other industries. In this case, the use of quad meshes makes the manipulation of the model much easier and much less cumbersome, although with the possibility of preserving a high resolution of the surface, helping the designer's subsequent work.

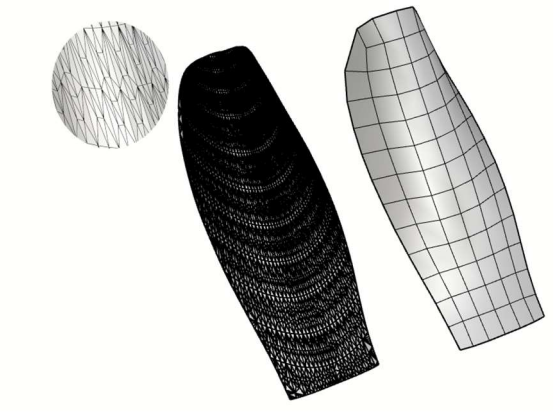


Fig. 2: Left: initial triangular mesh type enlarged. Middle: initial triangular mesh. Right: quad mesh resulting from the remeshing on the previous triangular mesh.

2. State-of-the-art on Meshing an Remeshing

The literature on quad meshing is quite extensive (Bommes *et al*., 2012). In this section, the set of methods which have been tested are detailed. The experimental results can be seen in section 3.

In Jakob *et al*. (2015), the *Instant* Field-*Aligned Meshes* method is presented. This method is able to produce quad or triangular meshes automatically, although there is the option to use some tools to help meshing manually. Meshing algorithms can be classified into local and global methods. Local methods are usually simple, robust and scalable, but due to their locality, they tend to introduce many singularities. Global algorithms solve optimisation problems whose size depends on the entire dataset, which increases quality, but sacrifices scalability, efficiency and simplicity of implementation. In the performed work, ideas from local and global meshing methods have been combined, thus computing a mesh that is globally aligned with a direction field using local smoothing operators of the orientation (determines the directions of the edges of a quad) and position (determines where the vertices of the mesh are placed) field. Unlike global methods, such as the one presented in Bommes *et al*. (2009), the main steps of this algorithm are local to a vertex and its neighbours, based on discontinuous surface fields, whose jumps are solved on the fly by the different local operators provided by the algorithm. In this way, the calculation of a global and continuous parameterisation is avoided. Finally, the mesh of the fields is extracted and, optionally, post-processed.

The execution time of this algorithm increases linearly with the mesh size. As a relatively fast method, it introduces a set of interactive brush tools, which can be used to control the alignment of edges in the final mesh, their exact position on the surface, and the location and number of irregular vertices. With these tools the system combines automatic and manual meshing methods: it allows users to control the positions of elements in critical regions, while automatically re-meshing the rest of the surface.

This algorithm produces quad-dominant meshes (most of the faces that make up the mesh are quads, while there may be a small fraction of these that are not quads, usually being triangles and/or pentagons) of high quality without much distortion. However, it produces quite a few singularities in the position field in addition to the singularities in the orientation field, which in most cases are unnecessary.

In Huang *et al*. (2018), the *Quadriflow* method is shown. This method is able to generate a quad mesh from a triangular mesh, although it can be adapted to accept a point cloud, based on the method from Jakob *et al*. (2015), but obtaining far fewer singularities than the latter. To reduce the number of singularities, which can rarely be completely eliminated, a global method is used to remove them from the position field, while the orientation field is calculated as in Jakob *et al*. (2015), as this field usually has much fewer singularities.

Local optimisation algorithms usually produce meshes with many singularities, while the best algorithms usually require non-local optimisation, and are therefore slow. The method from Huang *et al*. (2018) considers the computation of the position field without singularities as a globally constrained optimisation problem, using a system of linear and quadratic constraints. These constraints are accomplished by solving a global minimum cost network flow problem (for which efficient algorithms exist) and local boolean satisfiability problems (*satisfiability problem*-SAT). Unlike the method from Bommes *et al.* (2009), it does not solve the problem by mixed integer programming. Instead, it divides it into three stages:

1. Calculate the orientation and position fields as in Jakob *et al*. (2015) without imposing additional constraints.
2. Apply constraints by modifying only the integer variables in the position field, changing the integers as little as possible.
3. Re-optimise the continuous variables of the position field.

The second stage is a *Mixed-Integer Programming* (MIP) problem, for which it is difficult to find an optimal solution, but good approximate solutions can be obtained in practice. The problem is simplified to an *integer linear program* (ILP). The ILP is approximated as a simpler *minimum cost flow* problem (MCF) that can be solved in polynomial time, although good approximate solutions are also obtained, as in MIP. The third stage is not difficult and requires the solution of a linear system.

Replacing the MIP solver used by Bommes *et al*. (2009), with an MCF solver that globally reduces the number of singularities, although it is also approximate, plus edge contractions and a SAT solver, which locally imposes triangle orientation constraints, results in a quad remeshing that produces far fewer singularities than the method from Jakob *et al*. (2015).

One limitation of this method is that, when approximating a MIP problem as an MCF, geometric details may be lost when the target mesh density is low. On the other hand, it solves the global problem of removing position singularities much faster than other global methods. It is not as fast as the method from Jakob *et al*. (2015), a purely local method, but it produces far fewer singularities than the latter.

In Pietroni *et al*. (2021) the *QuadWild* method is detailed. It is an automatic method capable of generating semi-regular quad meshes on a surface, whose objective is to obtain a good quality final mesh, preserving the initial characteristics of the surface. It relies on a coarse design of polygonal patches, where the surface is initially divided into patches that are then tessellated individually but in a globally coherent way. Furthermore, in this method, the boundaries of the patches are forced to be part of the characteristic lines of the surface (which will become edges of the final mesh), allowing for non-quadrilateral patches and T-junctions in the layout. Inside the patches, a globally consistent internal tessellation is achieved, without T-junctions and allowing pure quad meshing.

Usually, T-junctions are not allowed in this class of methods and each patch is rectangular. This leads to a straightforward and easy final quadrangulation, but the restricted nature of the (quadrangular) layout makes its construction difficult. T-junctions make construction easier and provide more degrees of freedom, which can be exploited to optimise the domain in other respects. However, this requires the solution of a non-trivial problem to produce a final conformal mesh and determine a globally consistent subdivision of the layout sides. Ad-hoc strategies are used for this purpose. This ensures that the quadrangulation of the patch coincides with the patch boundaries, resulting in a conformal quadrangulation (without T-junctions).

The method relies on an auxiliary tangent space direction field of the auxiliary tangent space of the input surface to control the orientation of the edges and the location of the irregular vertices, which must be located at the singular points of the field. Briefly, the method is divided into the following steps (Fig. 3):

1. As a preliminary step to quad remeshing, an automatic optimisation of the input triangle mesh is carried out by performing a sequence of local operations aimed to improving the shape of the triangles and the distribution of the vertices of the input mesh. None of these operations disturbs the original characteristic edges.
2. A cross field is then calculated. It is aligned with the characteristic lines and propagates over the entire surface. In the absence of characteristic lines, the field is aligned with the principal curvature directions.
3. A series of trajectories are traced, thus dividing the surface into patches. This is the main stage of the process; its objective is to produce patches that are easily "quadrangulated", i.e. that have a geometric shape and topology that strongly favour the existence of a valid and good quality internal semi-regular quadrangulation of

valence. For this purpose, a series of topological and geometrical criteria are defined to estimate whether the patch is suitable or not.

4. The sides of each patch are tessellated, converting each side into a given number of edges. To determine the exact number, an ILP is solved, driven by an objective function that balances conflicting objectives, such as constant edge length and mesh regularity.
5. The last phase consists of creating each patch with quads. This task is carried out independently for each patch and the result is a pure and conformal quad mesh (without T-junctions). The internal quadrangulation of a patch may have irregular vertices.

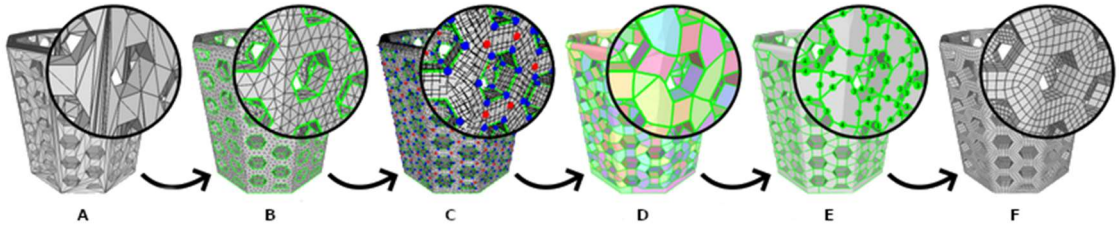


Fig. 3: Phases of the remeshing algorithm from Pietroni *et al*. (2021). A: input mesh. B: optimized input mesh. C: calculation of the cross field and singularities. D: Surface division into patches. E: Tessellation of the edges forming each patch. F: Final quad dominant mesh.

Compared to the results obtained by the methods from Jakob *et al*. (2015) and Huang *et al*. (2018), this method obtains favourable results, both in terms of required number of irregular vertices, i.e., those required for the mesh to fit the given surface correctly, and the quality of the shape, especially in the proximity of the characteristic lines.

One problem to point out is that in many cases symmetries are not respected (see Fig. 4). This piece is symmetrical with respect to the Z axis, but the resulting quad mesh is not similar on both sides of the Z axis, i.e., the final remeshing does not respect the symmetry of the initial piece. In addition, the optimisation of the initial triangular mesh, includes additional time to the process of obtaining the final quad mesh.

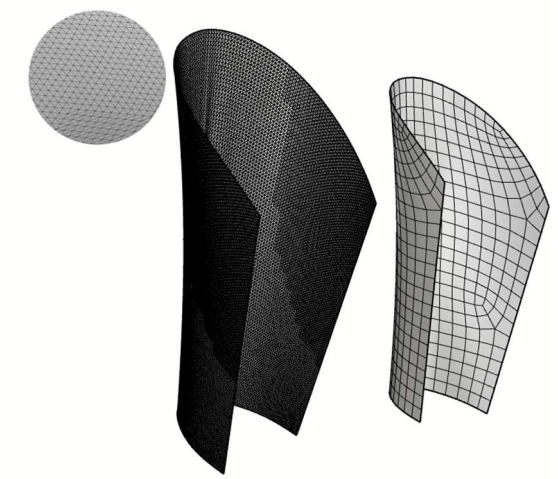


Fig. 4: Piece (part of a heel). Left: initial triangular mesh type enlarged. Middle: initial triangular mesh. Right: quad mesh resulting from the remeshing method from Pietroni *et al*. (2021).

The *Quad Remesher* 1.2 method (Exoside, 2022) is capable of obtaining quad dominant meshes automatically that includes a series of tools that help the designer both to guide the remeshing and to indicate the final density level of the quads by zones in a simple way. It allows any type of polygonal input mesh. This application software belongs to the company EXOSIDE, under a proprietary software licence. It is available for different graphic design applications, such as Blender or Maya.

The *Mixed-Integer Quadrangulation* method (Bommes *et al*., 2009) converts a triangular mesh into a quadrangulation automatically in a two-step process: cross-field generation and a global parameterisation, which are reduced to a MIP problem. After constructing a symmetric cross-field as smooth as possible that satisfies a sparse set of directional constraints to capture the geometric structure of the surface, a globally smooth and seamless parameterisation is computed. The isoparametric lines of the parameterization follow the directions of the cross-field that is given as input. Both steps of the algorithm (cross-field and the parameterisation) can be formulated as a problem that is solved very efficiently by means of a greedy solver. First, a smooth cross-field is obtained and used as input to compute a global parameterisation method. To compute this parameterisation, the mesh is sectioned in such a way as to create a surface patch with a disc-like topology in which all singularities of the cross field lie on the boundary. Then, two piecewise linear scalar fields u and v whose gradients follow the given cross field are computed. Finally, a consistent quadrangulation can be extracted the parameterisation is compatible in the cuts, and all singularities are assigned to integer positions along the boundary of the parameter domain. In both steps of the algorithm, the task can be formulated in terms of a MIP problem which merely consists in a set of linear problems where a subset of the variables belongs the continuous domain while the others are discrete.

Many recent methods use smoothed (discrete) principal curvature directions to guide the meshing of the quadrants. The problem with these approaches is that the final positions of the singularities are determined by the local smoothing operator applied to the initial curvature estimations. Another problem is oversmoothing, which can destroy the original orientation information in the feature regions. To overcome these problems, this method selects only the most relevant and dominant directions (Fig. 5), e.g. by a conservative threshold or by manual selection.

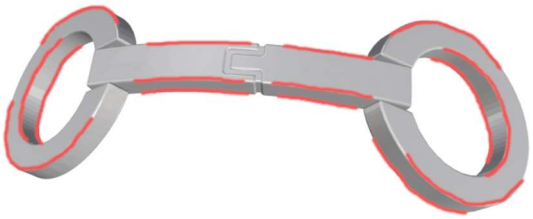


Fig. 5: Example of manual selection of some of the relevant and dominant directions of a model, in this case the edges.

A limitation of this method is that, for coarse quadrangulations of very complex models with many cross-field singularities, the local relocation of the local singularities in the parameterisation step dominates the total computational time. On the other hand, although it produces quad surface meshes with good qualities, it is slow and does not adapt well with large meshes. Another noteworthy fact is that this method completely omits the quad extraction process from the parameterisation obtained by this method. Therefore, a method capable of extracting quads from the obtained parameterisation must be used. The method *QEx* (Ebke *et al*., 2013) is capable of such extraction. It is able to perform quad extraction in a robust way and without the need for any complex tolerance thresholds or disambiguation rules; moreover, it is able to cope with the usual local folds in the parameterisation, which it is important, as quad extractors produce holes or non-square faces when they encounter them.

3. Experimentation

In this section, the experimental results carried out on a set of surfaces related to the footwear industry are shown. Different types of triangular meshes have been chosen (Fig. 6) that act as initial meshes, on which the different algorithms mentioned above have been tested. The experimentation has been carried out on an Intel Core i7-8750H 2.20 GHz computer with 8 GB of RAM.

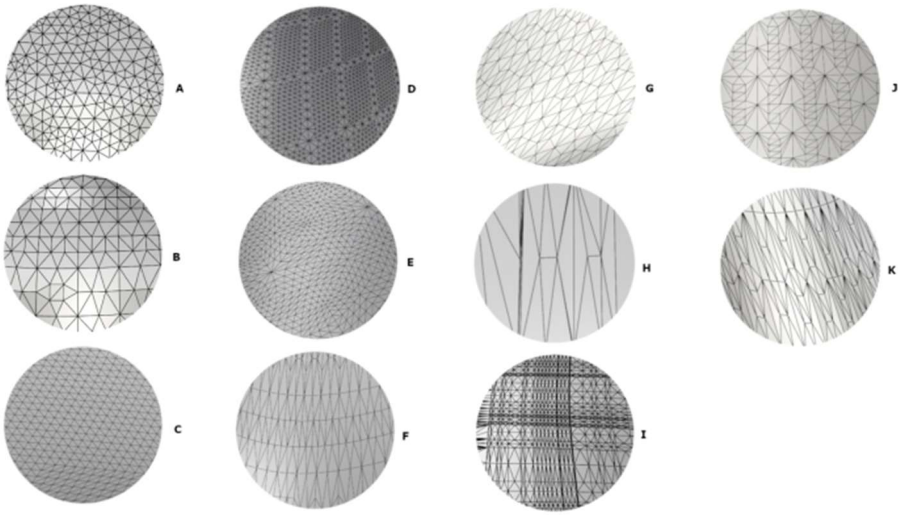


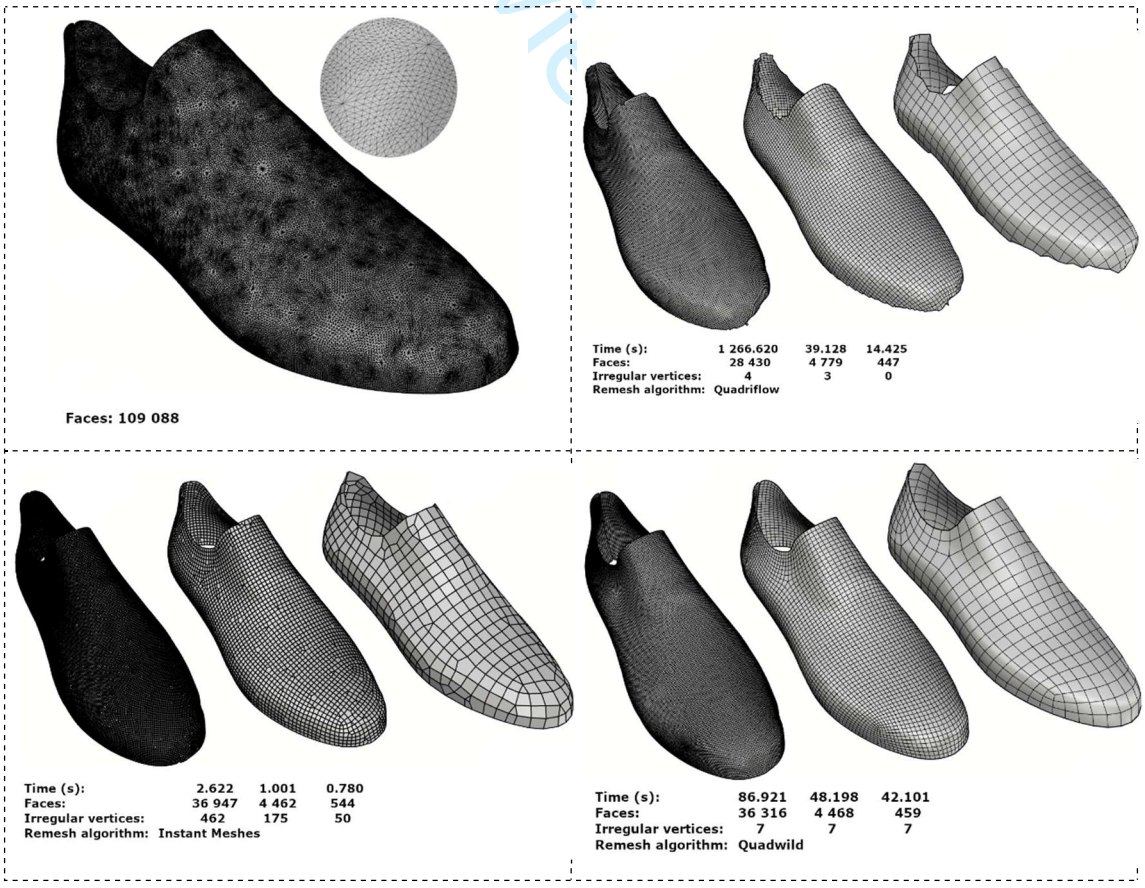
Fig. 6: Types of triangular meshes used as input in the different test models used.

Each of these algorithms has different inputs to carry out the remeshing processes. To perform this experimentation, the inputs corresponding to each of these algorithms were as follows:

- Huang *et al*. (2018): this algorithm does not have a graphical interface; therefore, it is managed under command line. The two solvers available for this algorithm have been used: maximum flow solver and SAT solver; in addition, the option to explicitly preserve the sharp edges of the original model has been used in all test cases. Under these options, the target number of approximate faces for the output mesh is given.
- Jakob *et al*. (2015]: this algorithm has a graphical interface that makes it possible to choose a series of options and tools to obtain the final remeshing. On all test cases, the *Extrinsic* and *Align to boundaries* options have been enabled for a better adaptation to the characteristics of the mesh and, when the mesh is not closed, to ensure that the edges of the output mesh follow those of the input mesh. After obtaining the mesh, the algorithm is instructed to perform a Laplacian smoothing process to increase the uniformity of the mesh (3 iterations). All this under an indicated number of approximate target vertices for the final mesh.
- Pietroni *et al*. (2021): This algorithm is also managed under command line. In all use cases, a text type configuration file is given to this algorithm, which includes options such as:
  - Dihedral angle of the sharp features, whose value throughout the experimentation will be between 30º and 90º.
  - Regularity over the output mesh; this value shall always be close to 0.

- Scale of the final quadrangulation, which indicates the size of the quads of the final mesh, this value varies the most, and depends on the number of quads on the final mesh required for each test.

- Exoside (2022): to use this algorithm, an interface has been created with different functionalities available to perform the remeshing. As a goal, a number of approximate quads for the final mesh has been used, in addition to the option that allows the final mesh to be adapted to a greater or lesser extent to the surface topology, thus reproducing the original geometry with a higher precision. This value will always be between 0.5 and 1 in all test cases. An option to automatically detect hard edges on the input mesh geometry is also set. With this option, the algorithm creates guides on these hard edges to obtain an edge loop in the output remeshing. In addition, in symmetrical geometries, the method is indicated on which axis the piece is symmetrical.
- Bommes *et al*. (2009): this algorithm produces a seamless parameterisation. To obtain a quad mesh from this parameterisation, the algorithm from Ebke *et al*. (2013) was chosen. The C++ geometry processing library *libigl*, which implements this algorithm, was used. The algorithm from Ebke *et al*. (2013) is not included in *libigl*, so it was downloaded from another repository.



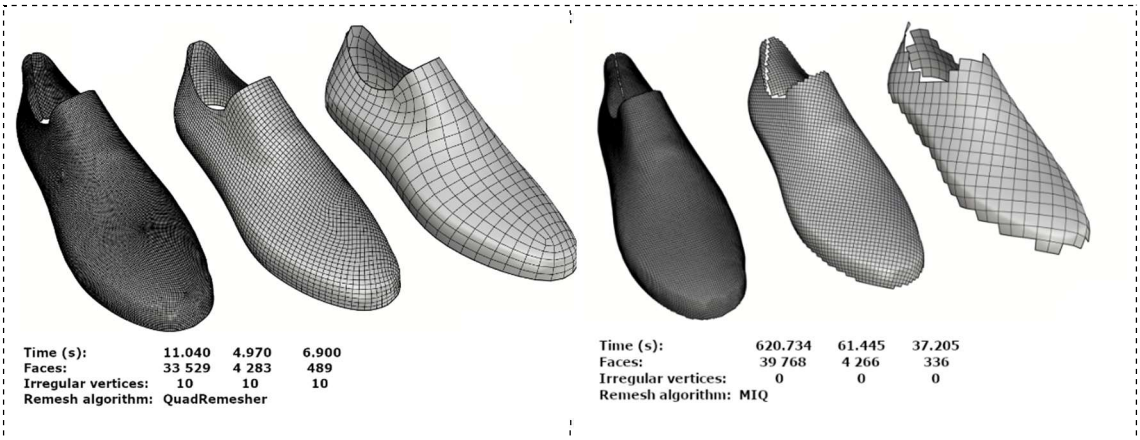


Fig. 7: Processing time, number of final quads and singularities obtained on an input triangular mesh (first mesh) for each of the algorithms seen. Several outputs with different final quad mesh densities are shown (second and third mesh).

Table 1 shows statistical data obtained on a set of experiments performed with the different algorithms. For each of the algorithms on the initial meshes, the following results are detailed:

- Number of vertices (#V): Indicates the number of vertices in the mesh.
- Number of quad and triangular faces (#F): Indicates the number of triangular and quad faces that make up the mesh.
- Number of irregular vertices (#IV): Indicates the number of irregular vertices, i.e., vertices with a valence other than 4. Vertices that form part of the edges of the mesh have not been taken into account as irregular.
- Aspect Ratio (#AR): Indicates the regularity of the quad or triangular elements that make up the mesh. The best numerical accuracy is achieved with a mesh that has perfect and uniform quad elements whose edges have the same length. Obviously, in most cases it is not possible to achieve a mesh of perfect elements, as the surface may be composed of sharp corners, curved parts, etc. Therefore, some of the elements composing the mesh may have much longer edges than others. The range is in [1, +inf); the closer to 1, the more perfect the cell will be.
- Minimum included angle (#MINA),: Indicates the smallest angle included in a quad. The range is [0°, 90°]; angles between 45° and 90° would be an acceptable range for quad cells.
- Maximum included angle (#MAXA),: Indicates the largest angle included in a quad. The range is [90°, 360°]: between 90° and 135° would be an acceptable range for quad cells.
- Warp (#W): Indicates the angle between the normal vectors of the triangles resulting from the diagonalisation of a quad, i.e., it indicates how flat the quads are in the structure. As a quad can be diagonalised in two ways, the value

indicated will be the maximum of the two. The range is [0, +inf); the closer a cell is to 0, the flatter it is.

- Skewness Equiangle (#SK): Difference between the shape of the cell and the shape of an equilateral cell of equivalent volume. Highly asymmetric cells can decrease accuracy and destabilise the solution. For example, optimal quad meshes will have vertex angles close to 90 degrees, while triangular meshes should preferably have angles close to 60 degrees and have all angles less than 90 degrees. The range is [0, 1]; the closer to zero, the more symmetric the cell will be.

These measurements were obtained from Pointwise's free software [Mesh Generation Software for CFD | Pointwise, Inc., n. d.], which allows virtually any 3D mesh file to be viewed, displayed in a variety of styles, and various mesh quality metrics to be calculated.

Table 1. Measurements on the meshes resulting from the different algorithms.

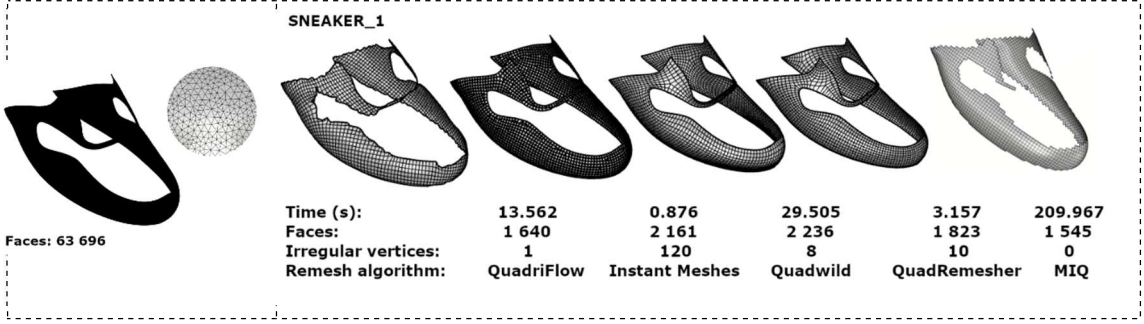
Method	#V	#F (quad/tri)	#IV	#AR	#MINA	#MAXA	#W	#SK	Time (s)
SNEAKER_1									
[Huang *et al*., 2018]	1918	1593/47	1	1,2844	72,9306	106,9664	16,2700	0,2098	13.562
[Jakob *et al*., 2015]	2301	1964/197	120	1,1307	79,6320	94,8634	1,9130	0,1006	0.876
[Pietroni *et al*., 2021]	2434	2236/0	8	1,9306	64,7978	115,8217	2,4529	0,2903	29.505
[Exoside, 2022]	2053	1823/0	10	1,2786	83,8275	96,0733	1,4843	0,0745	3.157
[Bommes *et al*., 2009] + [Ebke *et al*., 2013]	1808	1545/0	0	1,0080	89,3980	90,5384	2,6002	0,0068	209.967
SNEAKER_2									
[Huang *et al*., 2018]	2223	2048/27		1,1599	80,3763	99,5106	7,8031	0,1170	15.030
[Jakob *et al*., 2015]	2724	2447/295	191	1,1346	78,8214	95,0221	2,2661	0,1087	0.877
[Pietroni *et al*., 2021]	2446	2309/0		1,5822	71,4202	109,1198	4,0765	0,2229	34.710
[Exoside, 2022]	1985	1885/7		1,4785	81,4658	98,1644	1,4378	0,1004	0.896
[Bommes *et al*., 2009] + [Ebke *et al*., 2013]	-	-	-	-	-	-	-	-	-
SNEAKER_3									
[Huang *et al*., 2018]	2437	2241/18	0	1,2113	80,0945	99,5637	8,6121	0,1183	11.507
[Jakob *et al*., 2015]	3408	3136/235		1,0891	82,3878	94,0698	0,6735	0,0781	0.565
[Pietroni *et al*., 2021].	3421	3272/0	8	1,2521	70,4391	110,2309	1,0754	0,2270	28.502
[Exoside, 2022]	3050	2922/0	8	1,2439	83,7357	96,1578	0,7418	0,0738	1.727
[Bommes *et al*., 2009] + [Ebke *et al*., 2013]	2958	2768/0	0	1,0073	89,6217	90,35835	1,0612	0,0043	90.590
SNEAKER_4									
[Huang *et al*., 2018]	3372	2898/71		1,3320	75,0260	105,0032	14,6550	0,1829	23.381
[Jakob *et al*., 2015].	3454	2915/399	286	1,1632	77,2859	95,6650	2,1017	0,1232	0,533
[Pietroni *et al*., 2021]	3495	3106/0	63	1,1806	77,1504	103,3007	1,8584	0,1548	69.234
[Exoside, 2022]	2912	2565/7	55	1,3792	79,7049	100,3504	2,6300	0,1252	2.824
[Bommes *et al*., 2009] + [Ebke *et al*., 2013]	3952	3420/0	0	1,0016	89,8494	90,1192	2,0157	0,0017	190.072

SNEAKER_5									
[Huang *et al*., 2018]	2374	1968/46	31	1,2758	66,3436	113,5020	31,3577	0,2801	20.069
[Jakob *et al*., 2015]	2995	2487/291	193	1,1550	78,8259	95,0941	6,5382	0,1092	0.580
[Pietroni *et al*., 2021]	2948	2682/0		2,0470	63,9144	115,9969	13,2647	0,3082	197.691
[Exoside, 2022]	2259	2255/4	55	3,3208	82,7013	97,3719	6,2721	0,0922	0.879
[Bommes *et al*., 2009] + [Ebke *et al*., 2013]	3778	3204/0	5	1,0818	81,9519	95,5173	23,1094	0,0934	82.222
SALON_ABOTINADO_1									
[Huang *et al*., 2018]	973	892/10	0	1,2176	82,4449	97,4724	7,9150	0,0919	4.694
[Jakob *et al*., 2015]	1080	981/76		1,0926	80,7561	95,5993	0,9313	0,0968	0.479
[Pietroni *et al*., 2021]	1147	1080/0		1,1520	76,5365	103,7488	1,7359	0,1565	19.349
[Exoside, 2022]	1008	949/0		1,1598	84,7330	95,0469	0,5887	0,0613	1.066
[Bommes *et al*., 2009] + [Ebke *et al*., 2013]	1401	1315/0	0	1,0057	89,6346	90,3580	1,6615	0,0042	49.334
SALON_ABOTINADO_2									
[Huang *et al*., 2018]	288	204/10	0	1,3246	66,4351	114,2445	25,2612	0,3022	2.436
[Jakob *et al*., 2015]	337	252/70		1,2140	69,8415	99,1034	6,4049	0,1976	0.198
[Pietroni *et al*., 2021]	439	352/0		1,9937	33,5699	148,2162	30,7302	0,6916	61.479
[Exoside, 2022]	347	306/0		2,7862	77,9303	102,0141	3,2057	0,1483	0.800
[Bommes *et al*., 2009] + [Ebke *et al*., 2013]	-	-	-	-	-	-	-	-	-
SALON_ABOTINADO_3									
[Huang *et al*., 2018]	666	576/11		1,2186	71,5857	108,7371	14,3626	0,2241	2.695
[Jakob *et al*., 2015]	691	572/107	52	1,1684	75,0590	96,7806	2,5656	0,1468	0.289
[Pietroni *et al*., 2021]	705	601/0		1,8171	53,0275	129,1585	15,5621	0,4607	60.369
[Exoside, 2022]	578	525/0		1,3235	81,4377	98,2492	2,0473	0,1028	0.675
[Bommes *et al*., 2009] + [Ebke *et al*., 2013]	-	-	-	-	-	-	-	-	-
SALON_ABOTINADO_4									
[Huang *et al*., 2018]	273	160/35	10	2,5003	52,2393	128,5582	50,5424	0,4805	10.290
[Jakob *et al*., 2015]	296	170/88	52	1,3532	62,2668	99,9616	2,9432	0,2628	0.334
[Pietroni *et al*., 2021]	424	299/0		2,5637	27,8209	151,7751	31,7001	0,7388	51.079
[Exoside, 2022]		197/15	31	1,4168	70,6380	106,4487	1,9202	0,2289	0.808
[Bommes *et al*., 2009] + [Ebke *et al*., 2013]	-	-	-	-	-	-	-	-	-
SALON_ABOTINADO_5									
[Huang *et al*., 2018]	3496	3460/0		1,1667	77,0720	102,9235	9,3285	0,1653	65.403
[Jakob *et al*., 2015]	4070	3716/635	496	1,2179	74,1384	97,3859	7,7828	0,1591	1.254
[Pietroni *et al*., 2021]	3899	3863/0	148	1,1177	79,6498	101,6773	5,7946	0,1445	181.562
[Exoside, 2022]	2339	2245/116	125	5,6049	65,7868	113,0169	3,0122	0,2829	1.320
[Bommes *et al*., 2009] + [Ebke *et al*., 2013]	-	-	-	-	-	-	-	-	-
SALON_ABOTINADO_6									
[Huang *et al*., 2018]	2223	1964/60		1,2390	78,5869	101,8211	9,9740	0,1412	39.700
[Jakob *et al*., 2015]	2290	2018/274		1,1437	78,0354	95,9290	1,1749	0,1204	0.758
[Pietroni *et al*., 2021]	2521	2226/0	95	1,5347	66,3074	114,1316	6,2759	0,2834	116.171
[Exoside, 2022]	2279	2061/59		4,0637	77,8471	101,3480	1,2513	0,1427	4.513
[Bommes *et al*., 2009] + [Ebke *et al*., 2013]	-	-	-	-	-	-	-	-	-

SALON_ABOTINADO_7									
[Huang *et al*., 2018]	1710	1708/0	8	1,2030	82,1193	98,0300	1,8191	0,09561 1691	29.642
[Jakob *et al*., 2015]	1969	1769/152	101	1,7775	76,5567	98,4339	7,1596	0,1458	1.001
[Pietroni *et al*., 2021]	2322	2320/0		1,2458	81,0752	99,2321	2,0255	0,1118	957.109
[Exoside, 2022]	2238	2236/0	8	3,4394	85,0629	95,2982	0,7203	0,0621	6.850
[Bommes *et al*., 2009] + [Ebke *et al*., 2013]	812	810		1,3119	72,9263	106,5801	11,6387	0,20087	1048.705
BOTA									
[Huang *et al*., 2018]	-	-	-	-	-	-	-	-	-
[Jakob *et al*., 2015]	7475	5882/639	311	1,1545	79,3997	95,2711	2,8536	0,1062	1.932
[Pietroni *et al*., 2021]	7182	5887/0	563	2,2439	44,4347	135,8416	22,7834	0,5336	350.874
[Exoside, 2022]	6922	6615/106	848	2,8005	66,5147	112,0747	12,3944	0,2735	30.300
[Bommes *et al*., 2009] + [Ebke *et al*., 2013]	-	-	-	-	-	-	-	-	-
OBJETO_CREMALLERA									
[Huang *et al*., 2018]	226	139/0		1,2951	50,3995	130,8181	37,1035	0,4848	1.282
[Jakob *et al*., 2015]	257	163/31		1,2354	72,8033	99,1834	0,9857	0,1764	0.217
[Pietroni *et al*., 2021]	335	243/0	21	2,8911	69,1854	110,8332	10,6661	0,2551	50.393
[Exoside, 2022]	251	251/4		1,5033	79,1056	100,1946	1,0664	0,1350	0.597
[Bommes *et al*., 2009] + [Ebke *et al*., 2013]	-	-	-	-	-	-	-	-	-
PARTE_CREMALLERA									
[Huang *et al*., 2018]	-	-	-	-	-	-	-	-	-
[Jakob *et al*., 2015]	4413	1798/597	264	1,3184	66,6447	98,5874	7,3675	0,2198	2.187
[Pietroni *et al*., 2021]	3197	2164/0	309	2,1892	54,6080	123,0786	22,8788	0,4120	351.459
[Exoside, 2022]	2854	2681/26	634	1,6486	77,6398	102,8663	12,0364	0,1643	1.305
[Bommes *et al*., 2009] + [Ebke *et al*., 2013]	-	-	-	-	-	-	-	-	-
HEBILLA									
[Huang *et al*., 2018]	2054	2046/0		1,2065	81,6378	98,5535	5,1716	0,1019	25.494
[Jakob *et al*., 2015]	2114	2005/129		1,0807	83,1998	93,6120	3,1636	0,0681	0.509
[Pietroni *et al*., 2021	2159	2109/0		1,1928	82,0941	98,1387	3,1189	0,0953	57.185
[Exoside, 2022]	1966	1968/0		2,5160	82,0757	98,0237	2,5343	0,0945	1.872
[Bommes *et al*., 2009] + [Ebke *et al*., 2013]	-	-	-	-	-	-	-	-	-
PISO_2									
[Huang *et al*., 2018]	-	-	-	-	-	-	-	-	-
[Jakob *et al*., 2015]	5220	4164/585	311	1,1658	77,5327	95,7889	2,1829	0,1236	1.224
[Pietroni *et al*., 2021]	5337	4668/0	268	3,0357	60,4365	120,1319	12,3293	0,3574	313.820
[Exoside, 2022]	3946	3936/16	93	5,9301	71,8434	108,0972	5,5874	0,2100	5.149
[Bommes *et al*., 2009] + [Ebke *et al*., 2013]	-	-	-	-	-	-	-	-	-
DEPORTIVA_PARTE_SUPERIOR									
[Huang *et al*., 2018]	-	-	-	-	-	-	-	-	-
[Jakob *et al*., 2015]	1491	1254/205		1,1586	76,4042	96,2133	1,8151	0,1333	0.490
[Pietroni *et al*., 2021]		1430/0	35	1,2836	75,7644	105,2842	3,1795	0,1785	74.352
[Exoside, 2022]	1375	1236/10	44	1,3805	80,0436	99,5759	3,2686	0,1202	0.911
[Bommes *et al*., 2009] +	-	-	-	-	-	-	-	-	-

[Ebke *et al*., 2013]									
DEPORTIVA_PARTE_TRASERA									
[Huang *et al*., 2018]	21	19/0	10	1,5641	35,9132	128,1096	75,0128	0,6377	0.886
[Jakob *et al*., 2015]		43/7		1,1563	74,3609	98,7836	1,9027	0,1586	0.407
[Pietroni *et al*., 2021]	75	57/0	8	1,6451	56,9035	124,9006	12,0205	0,4157	28.204
[Exoside, 2022]	26	16/2		1,5142	68,5990	102,4167	5,1058	0,2286	4.647
[Bommes *et al*., 2009] + [Ebke *et al*., 2013]	-	-	-	-	-	-	-	-	-
ZAPATO_ADORNO_SUPERIOR									
[Huang *et al*., 2018]	-	-	-	-	-	-	-	-	-
[Jakob *et al*., 2015]	283	109/20		1,2894	73,6368	98,1457	0,0092	0,1604	0.970
[Pietroni *et al*., 2021]	3848	1964/0	357	11,8714	45,0493	134,7928	2,2766	0,5136	526.811
[Exoside, 2022]	399	397/6		2,5037	53,9315	124,4888	21,8944	0,4242	1.597
[Bommes *et al*., 2009] + [Ebke *et al*., 2013]	-	-	-	-	-	-	-	-	-

Fig. 8 shows a comparison on a set of parts belonging to the same shoe. It can be seen that the methods from Huang *et al*. (2018) and Bommes *et al*. (2009) did not respect the edges of the meshes, which are open. In the remaining open meshes, they will also not keep the edges as they are in the initial mesh. In addition, the method from Bommes *et al.* (2009) did not manage to remesh the SNEAKER_2 and SNEAKER_5 pieces. It can also be seen that in pieces SNEAKER_2 and SNEAKER_5, only the methods from Pietroni *et al*. (2021) and Exoside (2022) respected the internal "holes" of the initial mesh. Regarding the quantitative values, it can be observed that the methods from Huang *et al*. (2018) and Bommes *et al*. (2009) managed to obtain a dominant quad mesh with a reduced number of singularities. On the other hand, *Instant Meshes* algorithm obtained a final mesh with a suitable aspect ratio and symmetry. Despite these values, the method from Exoside (2022) obtained better overall results, both from a qualitative and quantitative point of view. Although this method generates irregular vertices (#IV), it distributes them in areas of the surface where they are considered necessary (curved, sharp areas, etc.), as does also the method from Pietroni *et al*. (2021) and unlike *Instant Meshes*, which includes irregular vertices in unnecessary areas. Apart from maintaining an aspect ratio and symmetry on the quads of the final mesh adequate to the initial surface, as does the method from Pietroni *et al*. (2021), the fundamental difference with respect to the latter method is the remeshing times.



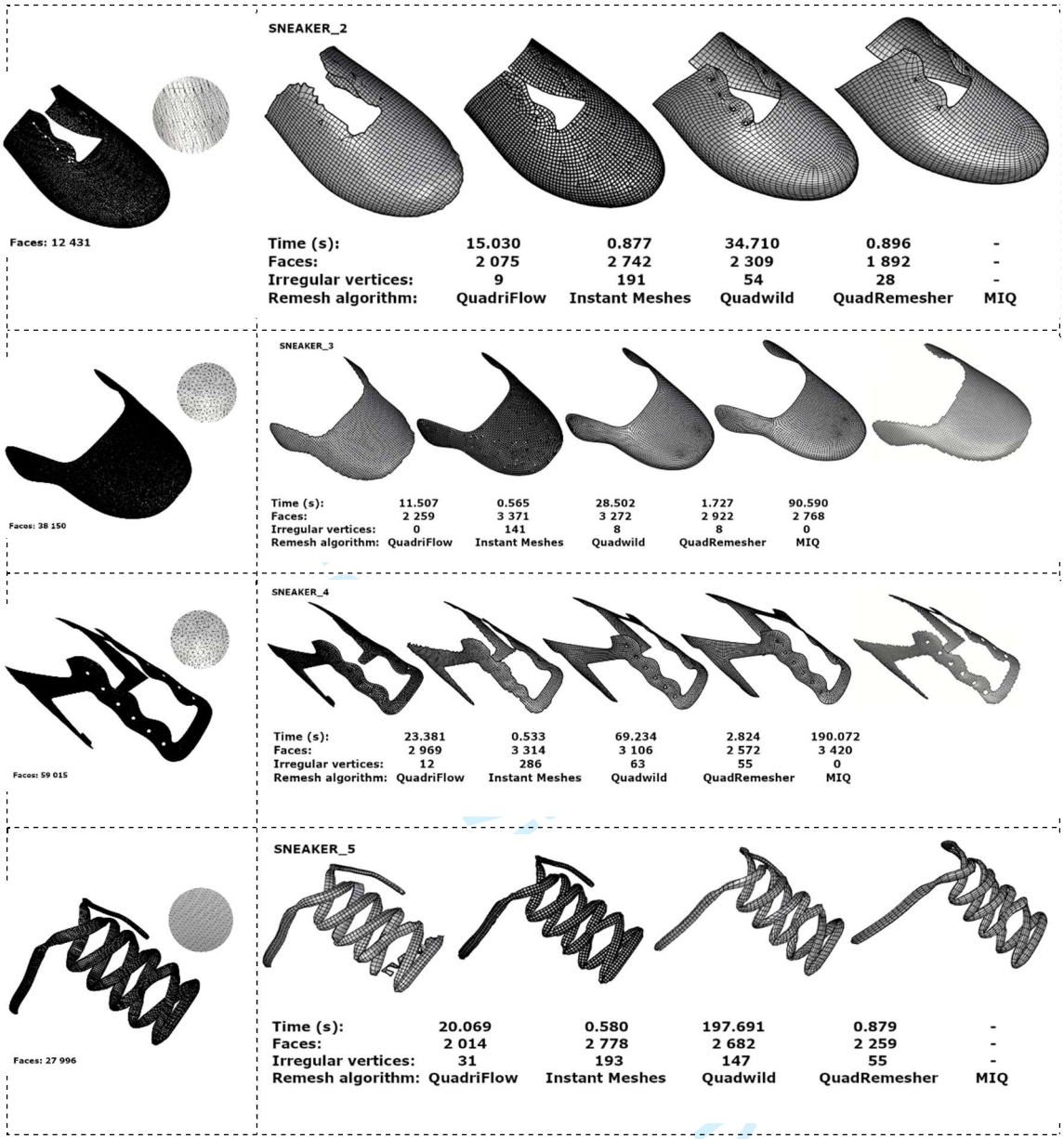


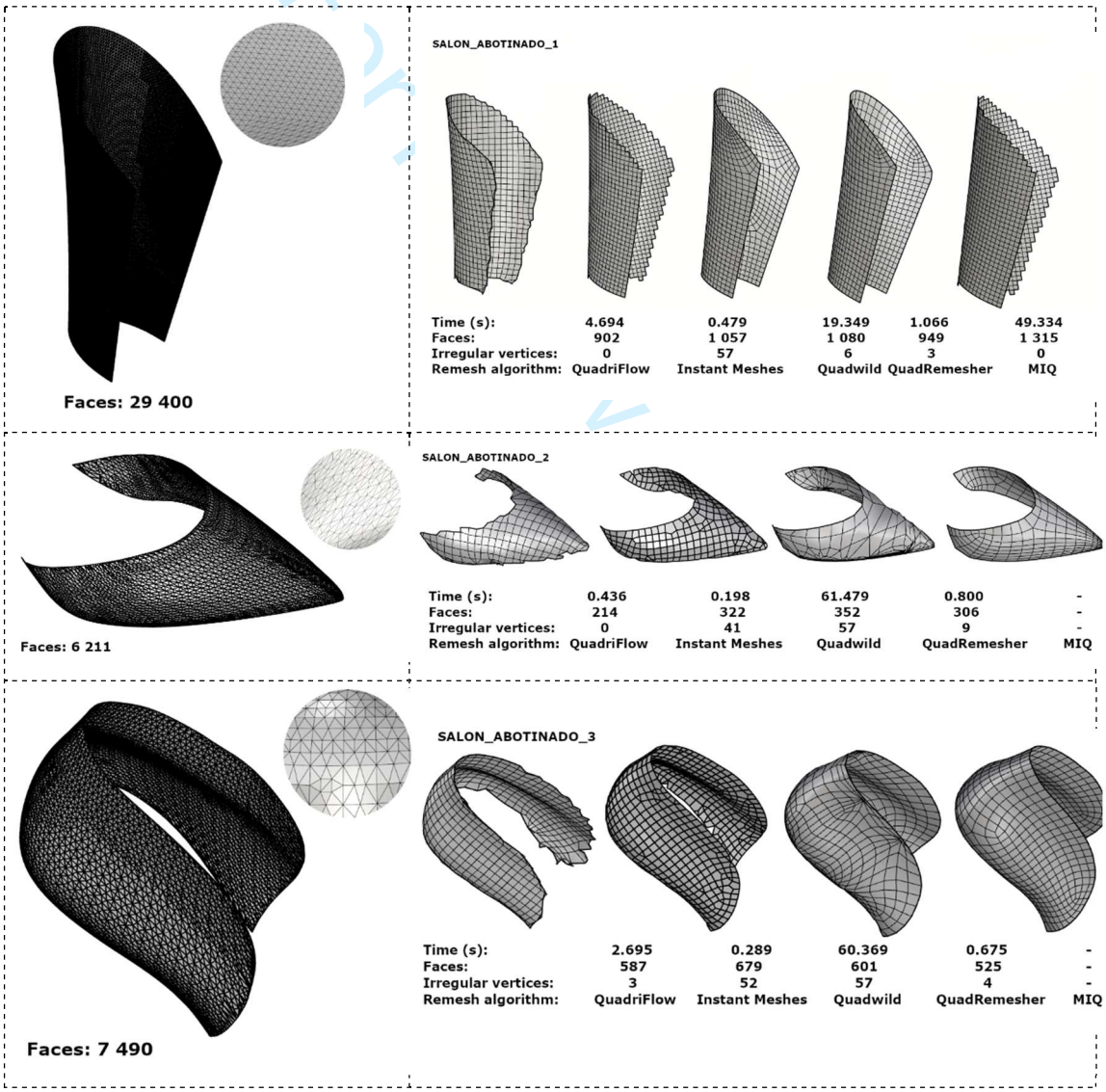
Fig. 8: Visual results of the quad remeshing as shown in Table 1 (SNEAKER_1 to SNEAKER_5).

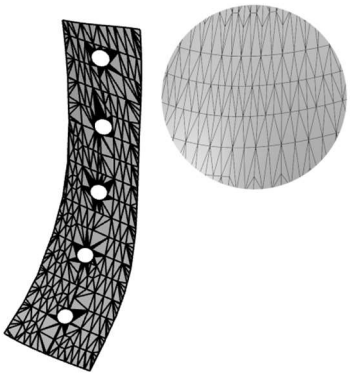
Fig. 9 shows another comparison between a series of pieces belonging to the same shoe, with different types of triangular input meshes. For the piece SALON_ABOTINADO_1, its initial mesh is similar to the SNEAKER_5 pieces, where Huang *et al*. (2018), in spite of generating a pure quad mesh, still does not respect the initial edges; Bommes *et al*. (2009) only manages to remesh two of the proposed parts, which still do not respect the initial edges; Jakob *et al*. (2015) still generates singularities in unnecessary areas; Pietroni *et al*. (2021) and Exoside (2022) obtain the best results, but with a much better remeshing time in the latter.

For the pieces from SALON_ABOTINADO_2 to SALON_ABOTINADO_4, the quality of overlocking both in qualitative and quantitative values is lower than for SALON_ABOTINADO_1, especially in the first three overlocking methods. This is due

to the initial input mesh. In the piece SALON_ABOTINADO_5, something similar to pieces SALON_ABOTINADO_2 to SALON_ABOTINADO_4 occurs, with the difference that it is a non-connected mesh; this means that the final overlocking between the different parts that make up the mesh is not homogeneous between all its non-convex components.

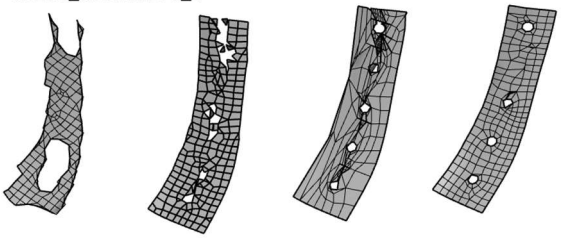
Finally, the pieces SALON_ABOTINADO_6 and SALON_ABOTINADO_7 are similar with the only difference in their triangular input mesh. This alters the final result, as can be seen in another example in Fig. 10. In these last two pieces of the Fig. 9, the results obtained by the methods from Pietroni *et al*. (2021) and Exoside (2022) are shown; it can be observed that they are quite different with respect to the initial input mesh in these last two cases. It is worth mentioning that for the part SALON_ABOTINADO_7, the method from Huang *et al*. (2018) was not able to remesh.



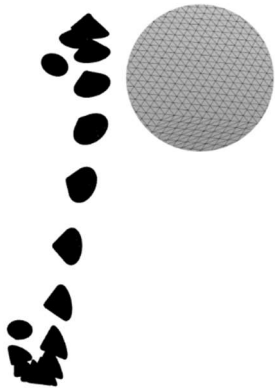


Faces: 735

SALON_ABOTINADO_4

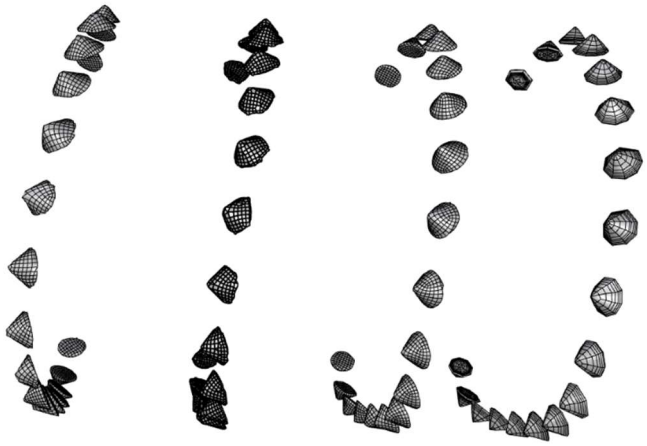


Time (s):	10.290	0.334	51.079	0.808	-
Faces:	195	258	299	212	-
Irregular vertices:	10	52	72	31	-
Remesh algorithm:	QuadriFlow	Instant Meshes	Quadwild	QuadRemesher	MIQ

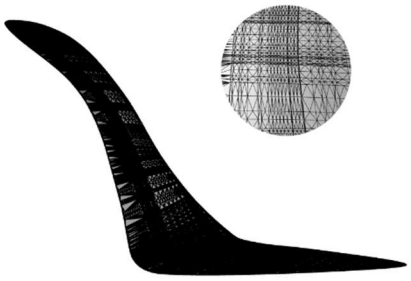


Faces: 43 200

SALON_ABOTINADO_5



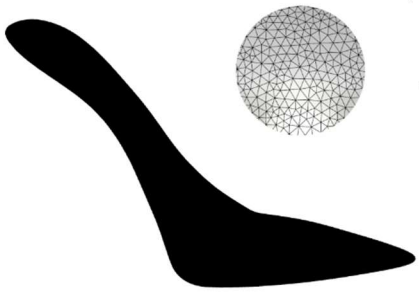
Time (s):	65.403	1.254	181.562	1.320	-
Faces:	3 440	4 351	3 863	2 361	-
Irregular vertices:	144	496	148	125	-
Remesh algorithm:	QuadriFlow	Instant Meshes	Quadwild	QuadRemesher	MIQ



Faces: 60 292

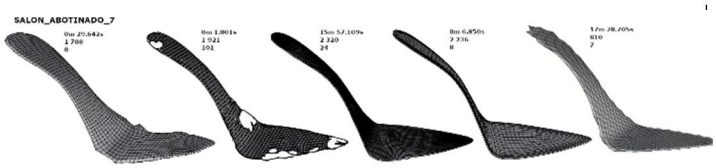
SALON_ABOTINADO_6

Time (s):	39.700	0.758	116.171	4.513	-
Faces:	2 024	2 292	2 226	2 120	-
Irregular vertices:	11	154	95	65	-
Remesh algorithm:	QuadriFlow	Instant Meshes	Quadwild	QuadRemesher	MIQ



Faces: 141 228

SALON_ABOTINADO_7



Time (s):	29.642	1.001	900.109	6.850	1 048.705
Faces:	1 708	1 921	2 320	2 236	810
Irregular vertices:	8	101	24	8	7
Remesh algorithm:	QuadriFlow	Instant Meshes	Quadwild	QuadRemesher	MIQ

Fig. 9: Visual results of quad remeshing as shown in Table 1 (SALON_ABOTINADO_1 to SALON_ABOTINADO_7).

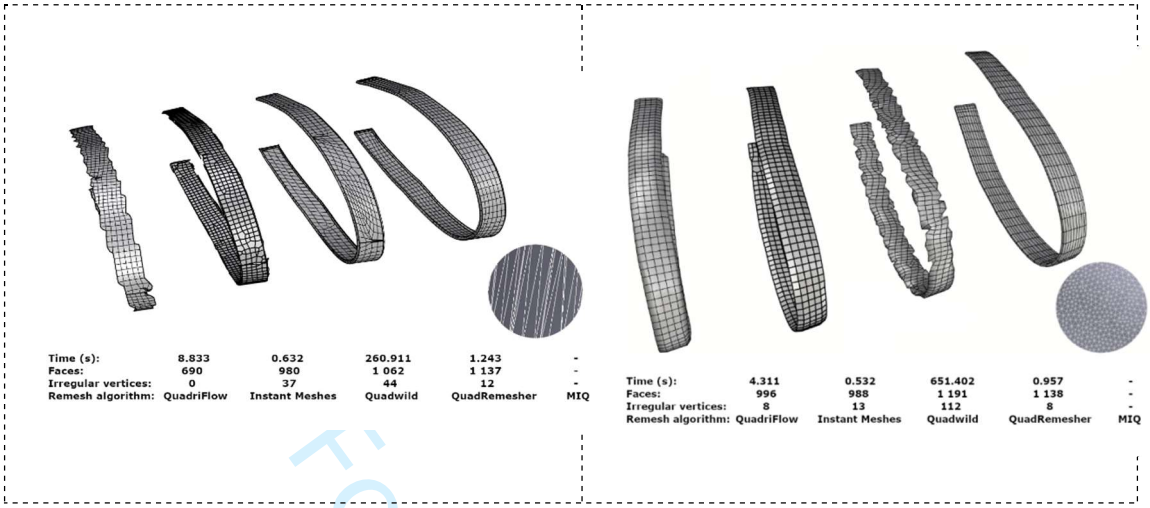
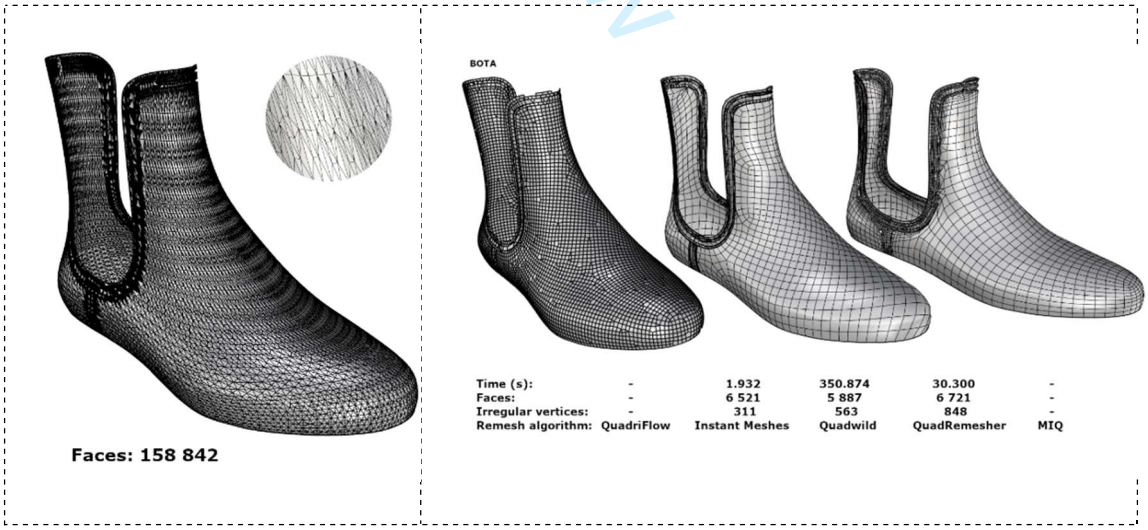


Fig. 10: Example of quadratic meshing on the same model but with a different input triangular mesh. It can be seen that the results are different between the same methods.

Fig. 11 shows pieces belonging to a boot. It can be seen that the final overlocking in this piece is of lower quality in all methods, mainly due to the areas with indentations where the triangular mesh is much denser. The other two pieces have similar results to some of the pieces analysed previously.



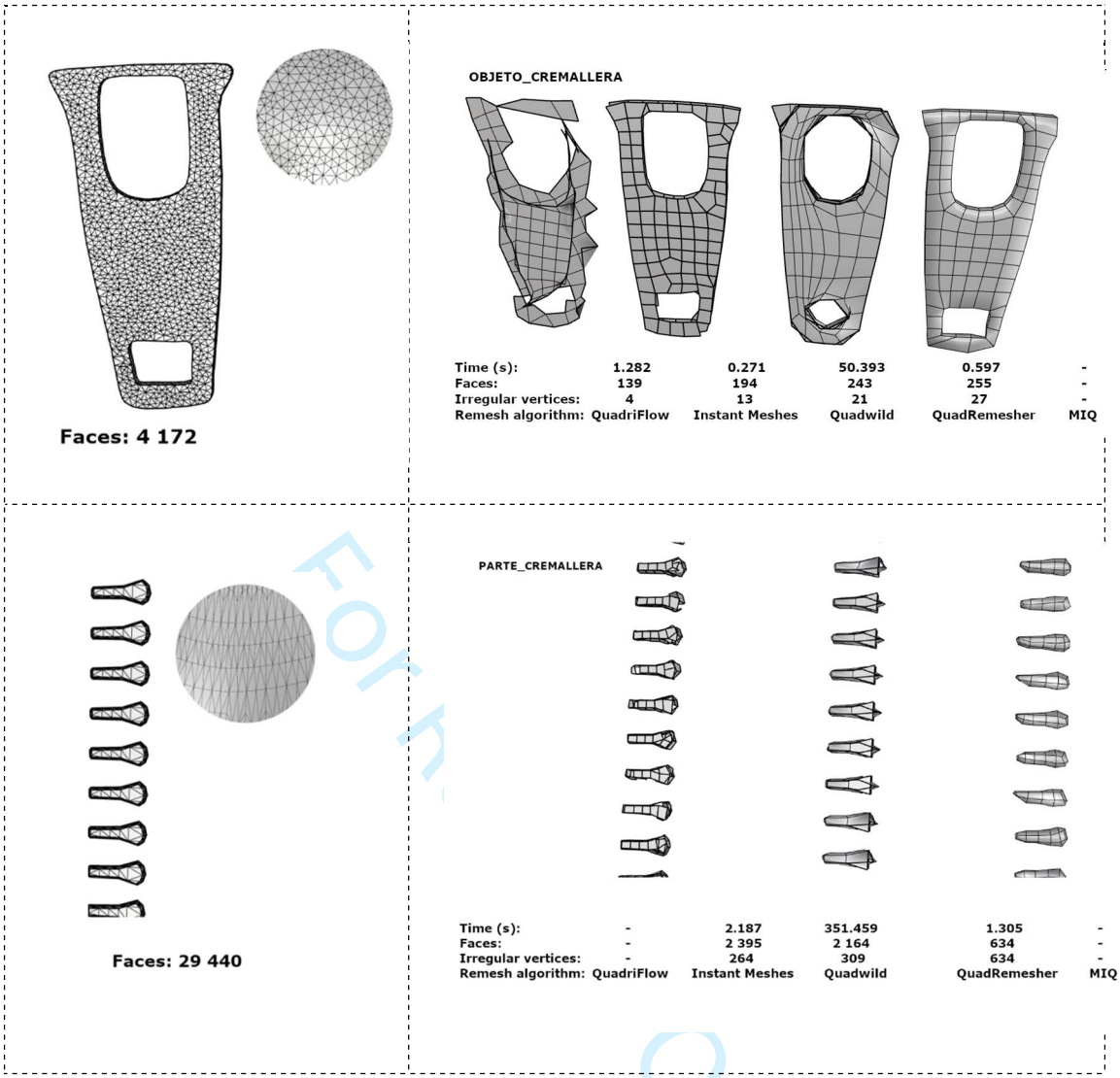
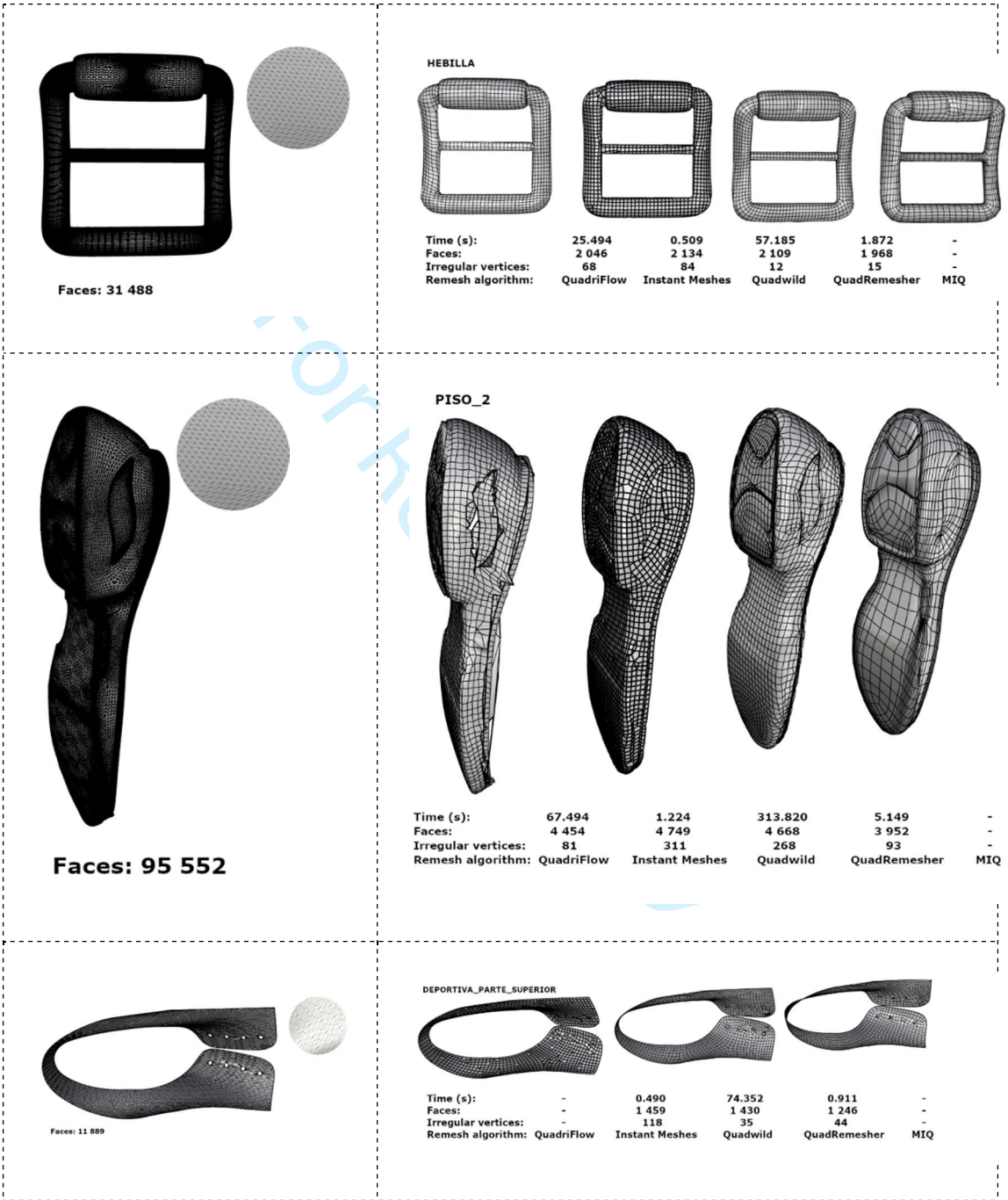


Fig. 11. Visual results of the quad remeshing shown in Table 1 (BOTA to PARTE_CREMALLERA).

Fig. 12 shows several pieces of different types of footwear. We can highlight that the PISO_2 piece, even though it is a closed piece like the HEBILLA piece, the method from Huang *et al*. (2018) did not managed to maintain the initial edges. In the piece DEPORTIVA_PARTE_SUPERIOR, the method from Pietroni *et al*. (2021) was the only one that manages to maintain the eyelets of the piece to some extent, even ahead of the method of Exoside (2022). The piece DEPORTIVA_PARTE_TRASERA clearly shows how the triangular input mesh completely conditions these remeshing methods; indeed, since this piece does not present any kind of "rough" area, it should be easy to obtain a pure quad mesh; the method from Exoside (2022) was the one that obtained the best result, using a smaller number of singularities than the rest. Finally, for the piece ZAPATO_ADORNO_SUPERIOR, the method from Exoside (2022) was the only one that obtained a mesh without holes although without completely respecting the sharp edges; this is also due to the fact that the resulting mesh is much less dense than the initial

one, therefore, quite a lot of geometrical details are lost. It is worth mentioning that the resulting mesh of the method from Pietroni *et al*. (2021) was much denser in this part, because it was not possible to reduce its quad density to the level of the other methods. The method from Huang *et al*. (2018) was not been able to obtain a valid result.



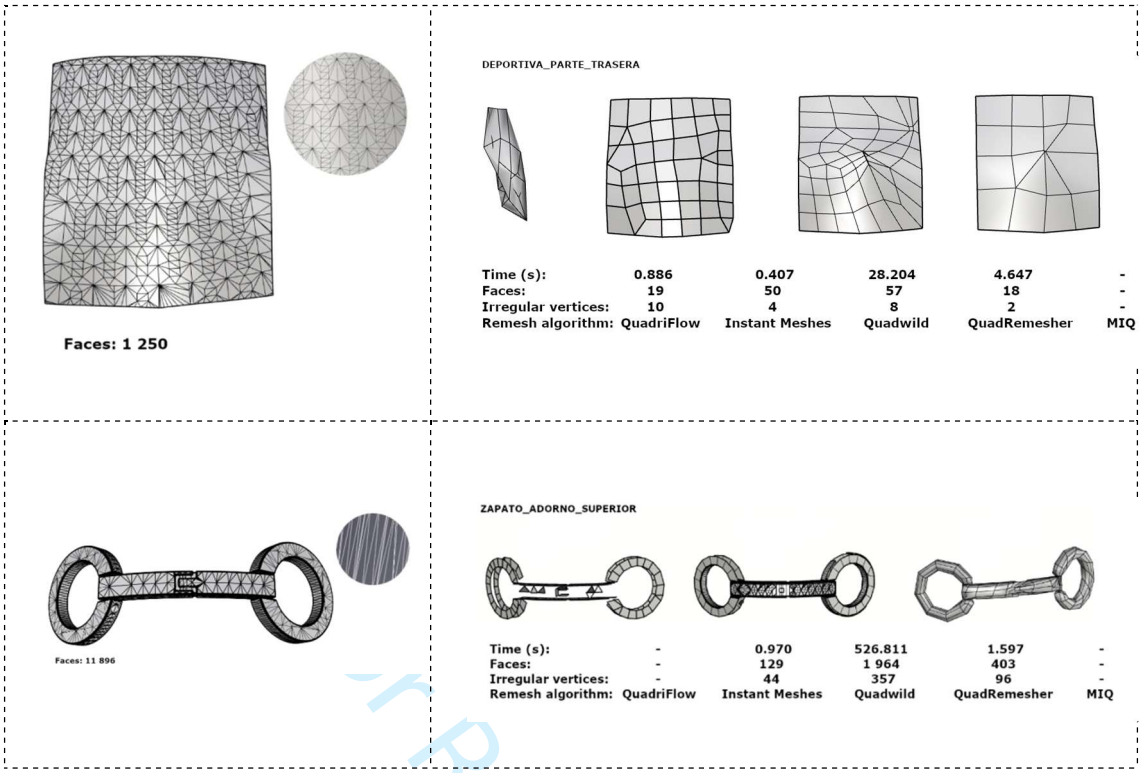


Fig. 12: Visual results of the quad riveting shown in Table 1 (HEBILLA to ZAPATO_ADORNO_SUPERIOR).

After analysing these tests, it is observed that the methods of Exoside (2022) and Pietroni *et al*. (2021) are the ones that obtain the best results with regard to the number and distribution of singularities, preservation of geometric characteristics and remeshing times. Although Jakob *et al*. (2015) obtains quite acceptable meshes in many cases, however, due to being a local method it produces many unnecessary singularities, although it achieves the lowest execution times over the rest, in some cases, by far. Finally, the method from Huang *et al*. (2018) does not manage to remesh all the proposed test cases; moreover, it does not manage to preserve the edges of the meshes, especially the open ones; therefore, for this type of parts in the field of footwear, this method is not very useful.

Moreover, with regard to the automatic remeshing, the methods form Exoside (2022) and Jakob *et al*. (2015) offer a number of user tools. The first offers two tools: one to focus the final quad density on user-specified areas of the surface, and a second to guide the final remeshing with a series of user-specified curves. These tools are easy to use and useful in some specific cases and depending on the final quad mesh required. On the other hand, Jakob *et al*. (2015) offers a series of tools that allow the user to intervene in the process of guiding the orientation fields and the movement/removal of some of the singularities. These tools are more tedious to use for a non-specialist user. Fig. 13 shows an example of the result of the tools on a model. Finally, it is worth mentioning that the method from Pietroni *et al*. (2021) allows receiving a text file indicating the sharp features of the piece, although the algorithm automatically detects the edges as sharp features by

default. In Fig. 14, some examples on the remeshing achieved with each of the algorithms on different types of models and input meshes are shown.

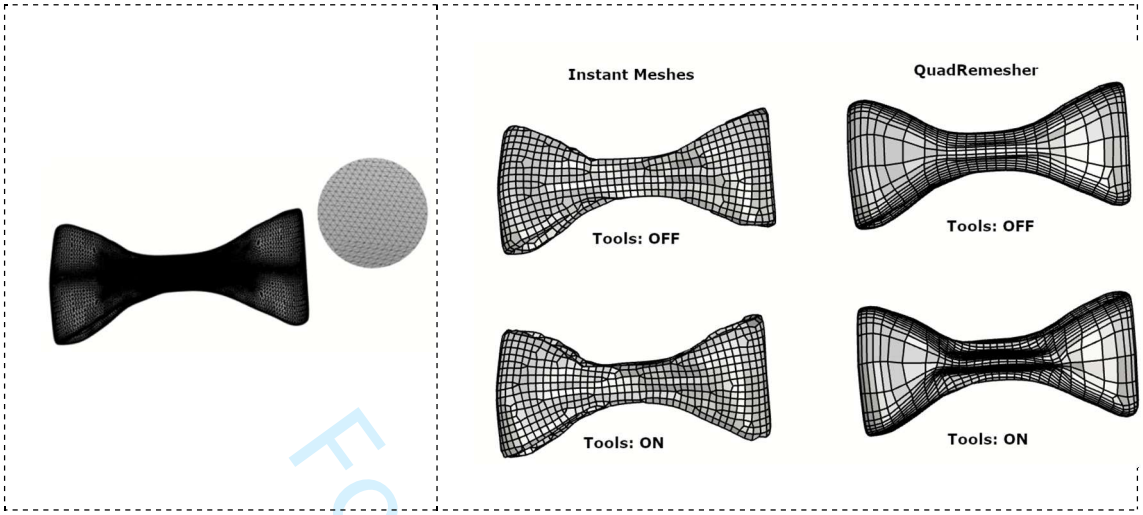
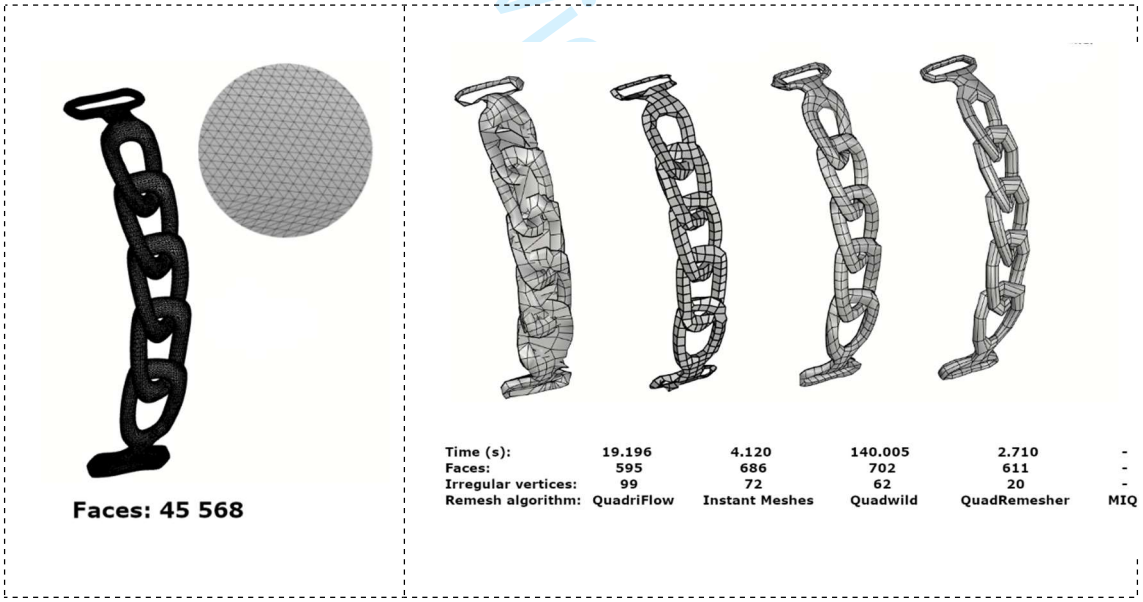


Fig. 13. Remeshing on an initial triangular mesh with the algorithms from Jakob *et al.*, (2015) and Exoside (2022), without using (Tools: OFF) and using (Tools: ON) density tool and singularities and field orientation tool, respectively.



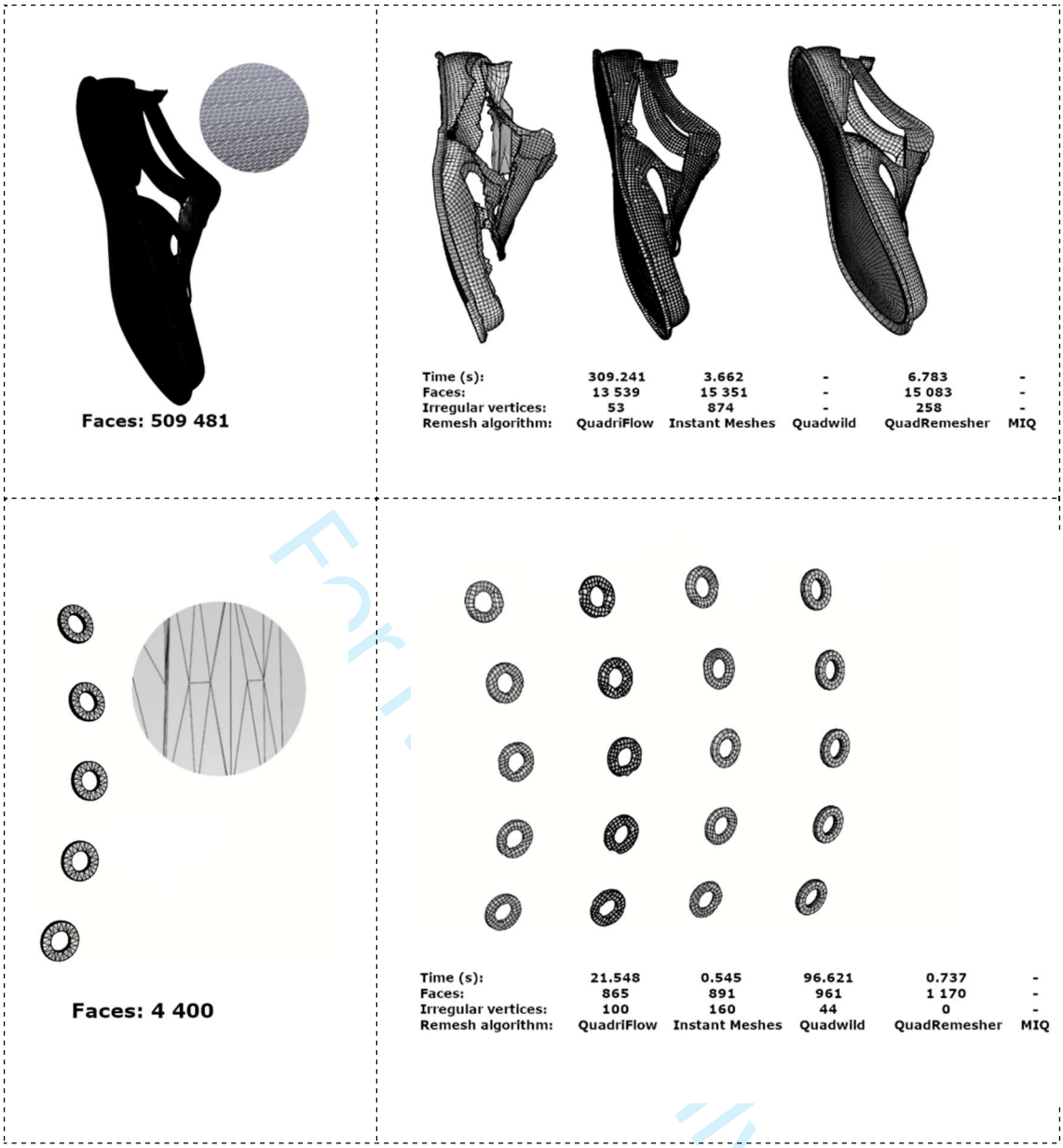


Fig. 14: Examples remeshed with each of the algorithms on different types of models and input meshes.

4. Conclusions

In this work, several methods have been analyzed which are aimed at obtaining a quad mesh from an input, such as any polygonal mesh or a point cloud. The aim is to obtain a dominant and structured quad mesh from input models in the footwear domain, in order to facilitate at a later stage, the manipulation of these meshes by a designer, as well as to obtain lightweight models for their storage and exploitation on the web and new techonologies.

Five algorithms capable of obtaining dominant and structured quad meshes have been tested and several measurements have been collected in order to compare these algorithms, such as the number of singularities generated in the final mesh, symmetry of

the quads that make up the mesh, etc. In addition, it has been found that, on the same input surface with a different type of initial mesh, different results are obtained, even in some cases, a final mesh of poor quality. This indicates that it is important that the input mesh is correctly structured. It is worth mentioning that, in non-convex input meshes, the remeshing between the different parts of the mesh is often different, sometimes even losing geometrical characteristics in certain parts.

After analysing and comparing these methods, it can be concluded that the algorithms from Pietroni *et al*. (2021) and Exoside (2022) obtained good quality meshes in most of the tested models, the quality of the latter being superior in some of them, as well as obtaining significantly lower times with denser and/or more complex inputs. The method from Exoside (2022) has several options and tools to improve meshing, while that from Pietroni *et al*. (2021) is more limited in this aspect. It should be noted that the latter method is open source, while the method from Exoside (2022) is not. Regarding the rest of the algorithms, the algorithm from Jakob *et al*.(2015), although it has the best execution times, tends to generate meshes with too many singularities in areas where they are not necessary a priori; this makes the mesh more difficult to handle in post-processing. The method from Huang *et al*. (2018), with quite short times, failed to maintain the initial edges of the open meshes, in addition to failing in the remeshing process in several proposed cases. Finally, the methods from Bommes *et al*. (2009) and Ebke *et al*. (2013) failed to obtain a final mesh in most cases; in almost all of them, the parameterisation process of Bommes *et al*. (2009) ended up with some kind of exception, although there have also been some cases where the parameterisation has been achieved but Ebke *et al*. (2013) failed to extract the quad mesh.

The choice of one of these algorithms will depend on the characteristics of the final mesh to be obtained: if it is desired to obtain a quad mesh that maintains the characteristics of the initial surface, with a controlled number of singularities and reduced execution times, the method from Exoside (2022) is the best option. If an open-source algorithm is desired, Pietroni *et al*. (2021) may be an option. On the other hand, if singularities do not matter and the initial surface features are desired to be preserved, the method of Jakob *et al*. (2015) may be an acceptable option. Finally, if the edges of the initial surface are preserved, especially in open meshes, and some tools are included to improve the meshing, Huang *et al*. (2018) is a good option.

As future lines of research, some of the analysed methods could be integrated in real industrial design platforms in the field of footwear, as well as to implement and test in a more exhaustive way, the different tools that offer some of these methods, such as Exoside (2022) and Jakob *et al*. (2015).

References

Alliez, P., Ucelli, G., Gotsman, C., Attene, M. (2008). Recent Advances in Remeshing of Surfaces, in L. De Floriani and M. Spagnuolo (eds) Shape Analysis and Structuring. Berlin, Heidelberg: Springer (Mathematics and Visualization), pp. 53-82. Available at: https://doi.org/10.1007/978-3-540-33265-7_2.

Bommes, D., Levy, B., Pietroni, N., Puppo, E., Silva, C., & Zorin, D. (2012). State of the Art in Quad Meshing. Eurogr. STARS 2012, 20, 1–24.

Bommes, D., Zimmer, H., & Kobbelt, L. (2009). Mixed-integer quadrangulation. ACM Transactions on Graphics, 28(3), 77:1-77:10, https://doi.org/10.1145/1531326.1531383

Ebke, H.-C., Bommes, D., Campen, M., & Kobbelt, L. (2013). QEx: Robust quad mesh extraction. ACM Transactions on Graphics, 32(6), 168:1-168:10, https://doi.org/10.1145/2508363.2508372.

Exoside (2022, Novembre 15) Quad Remesher-Auto Retopology. https://exoside.com/quadremesher/ Accessed 15 November 2022.

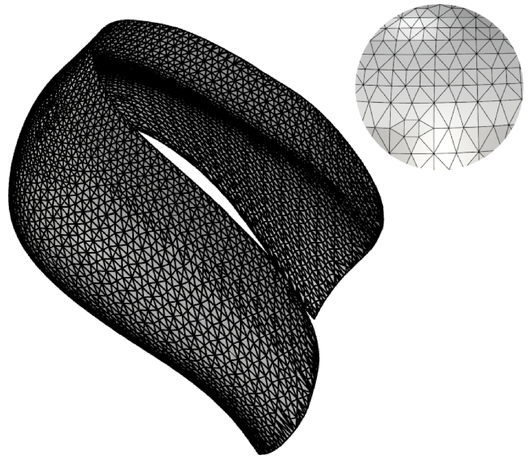
Huang, J., Zhou, Y., Niessner, M., Shewchuk, J. R., & Guibas, L. J. (2018). QuadriFlow: A Scalable and Robust Method for Quadrangulation. Computer Graphics Forum, 37(5), 147-160, https://doi.org/10.1111/cgf.13498.

Jakob, W., Tarini, M., Panozzo, D., & Sorkine-Hornung, O. (2015). Instant field-aligned meshes. ACM Transactions on Graphics, 34(6), 189:1-189:15, https://doi.org/10.1145/2816795.2818078

Pietroni, N., Nuvoli, S., Alderighi, T., Cignoni, P., & Tarini, M. (2021). Reliable feature-line driven quad-remeshing. ACM Transactions on Graphics, 40(4), 155:1-155:17. https://doi.org/10.1145/3450626.3459941.

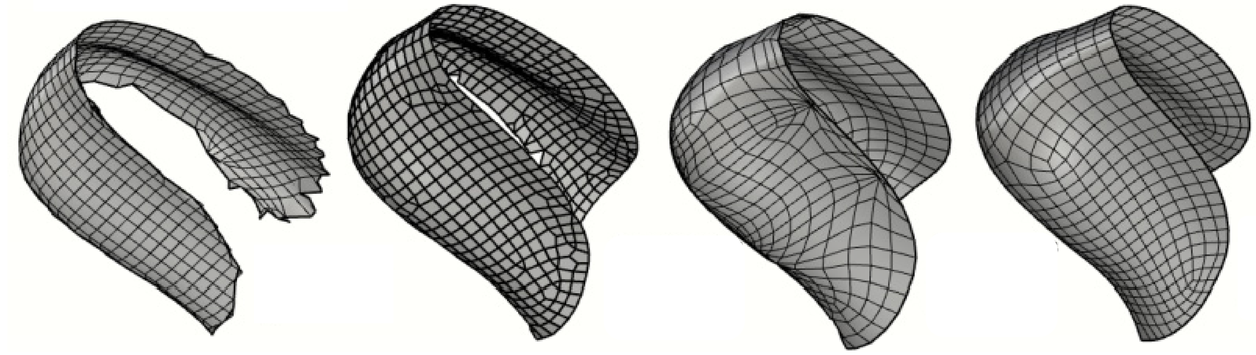
Pointwise (2022, November) Mesh Generation Software for CFD - Pointwise, Inc. https://www.pointwise.com/ Accessed 15 November 2022.

Triangular mesh



Different Transform Algorithms

Quadrangular (*quad*) mesh

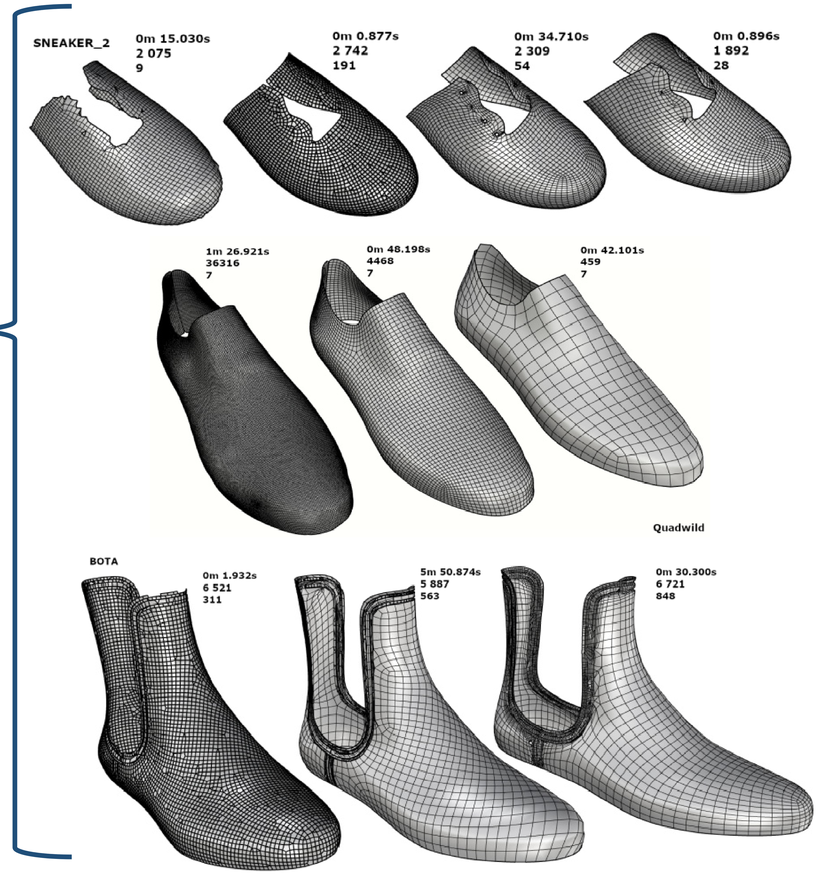


Implementation of the Algorithms

Experimentation on real footwear manufacturing pieces

Visual Results

Statistical Robustness Results



HIGHLIGHTS

- Different algorithms for transforming triangular meshes into quadrangular meshes are analyzed.
- The algorithms are tested on different pieces of real objects related to footwear manufacturing.
- The results of the experimentation are shown in terms of statistical data and also visualization.
- The conclusions allow manufacturers to have a more advanced knowledge about the most suitable method with respect to the user's requirements.