SHRINK WRAP MESH GENERATION USING MORPHOLOGICAL OPERATORS WITH SELECTED APPLICATIONS

Vijai Kumar Suriyababu[1,‡] Cornelis Vuik[1] Matthias Möller[1]

[1] *Delft Institute of Applied Mathematics, Delft University of Technology, The Netherlands,*
{v.k.suriyababu,c.vuik,m.moller}@tudelft.nl

[‡] *Corresponding Author*

ABSTRACT

Triangulated meshes discretized from commercial CAD applications often possess a considerable level of complexity. However, when conducting external aerodynamics simulations at an earlier design stage, these meshes are way too complex and contain complex features and topological holes. We propose a practical and fast algorithm to shrink wrap triangulated surfaces with the sole intent of topology and surface simplification. Building upon the concepts of mathematical morphology and newer advancements in geometry processing, such as generalized winding numbers, we show that it is possible to build a straightforward and robust algorithm that can guarantee genus-zero surfaces. Our approach uses a Cartesian background mesh (fixed and adaptive) to approximate an input triangulated surface's interior and exterior volume. We use an octree data structure for adaptive mesh refinement. Although we demonstrate our algorithm exclusively on triangulated meshes, they are equally applicable to general polyhedral meshes. They are also well suited for handling point clouds (oriented and unoriented), and we show some examples of the same with some unoriented point clouds. We built our algorithms with a wide variety of applications in mind. However, we showcase the applicability of our algorithms for aerodynamic simulations, fluid volume extraction, and surface simplification. We also emphasize the practicality and ease of implementation of the proposed algorithms. We also compare our algorithms with existing literature.

Keywords: mesh simplification, shrink wrapping, boolean operations, fluid volume extraction

NOMENCLATURE

$\mathcal{M}$	Triangulated mesh (or surface)
σ	Structuring element
σ_r	Structuring element radius
T_l	Topological sphere level
ω	Solid angle for a query point with respect to an input mesh

1. INTRODUCTION

The shrink-wrapping algorithm can be a helpful tool for remeshing triangulated (or any polyhedral) surfaces. Most algorithms take a volumetric approach to solve the problem. They work by projecting a voxelized approximation of the input surface onto itself. A few papers in the literature focus on this problem, and most work is accomplished in the industry. Attene et al. [1] gives a very detailed comparison of different mesh repair algorithms. Their review article's "Global repair section" gives an excellent overview of the state-of-the-art algorithms with a Global mesh repair and simplification approach. A key highlight of this discussion is that all authors take a volumetric approach to the problem. However, none of these explicitly solve the problem of shrink-wrapping with CAE simulations in mind. They are generic mesh simplification approaches, with the primary focus being

computer graphics applications. Some notable contributions are from Esteve et al.[2] and Nooruddin et al.[3], who also take volumetric approaches. Esteve et al.[2] shrink a discrete membrane to re-mesh surface meshes and point clouds, whereas Noorudin et al.[3] take a strict morphological approach to the problem. One major drawback in their works is a lack of control over the surface genus. In the case of Noorudin et al.[2], they oversimplify the meshes in all of the examples instead of our shrink-wrap mesh simplification. They also do not guarantee a manifold mesh as output, thus rendering their results unsuitable for numerical simulation. Y. K. Lee et al. [4] gave a good summary of the existing techniques that are tailored to the problem of shrink-wrapping with engineering applications in mind. They highlight the effectiveness of these algorithms in closing gaps and removing interior parts of complex geometry along with their potential as remeshing algorithms. They also show that gap detection and bridging are usually achieved with the help of poor / coarser voxelization. There is often a requirement to intersect these voxels with the input surface mesh, increasing computational costs. Some algorithms [5] require explicit tolerance values for the gaps and holes, which could be advantageous in some applications. However, we explicitly focus on fully automated genus simplification. Hence, we propose an algorithm that can inherently close all the gaps in triangulated surfaces and remove all internal structures. The significant contribution in our work is an efficient way of computing genus simplified offset surface with the help of morphological operators. Our algorithms will guarantee an outcome even in the case of imperfect geometries. Imperfections can range from missing triangles to non-manifold edges or completely separated components. We demonstrate the same in our numerical experiments. We also extend the existing shrink-wrap algorithm for selective genus control. An existing semi-heuristic algorithm that we developed [6] for hole detection is used to control the closing operations so that only selective holes are closed.

2. MORPHOLOGICAL OPERATORS

Mathematical morphology is a vibrant subject that finds typical applications in the area of image processing [7]. However, it has been extended to many other application areas, including three-dimensional geometry processing [8]. Broadly, the subject of mathematical morphology is composed of four operators.

1. Erosion ($\mathcal{M} \ominus \sigma$)
2. Dilation ($\mathcal{M} \oplus \sigma$)
3. Opening ($(\mathcal{M} \ominus \sigma) \oplus \sigma$)
4. Closing ($(\mathcal{M}(\oplus\sigma) \ominus \sigma$)

Given a triangulated mesh $\mathcal{M}$, a structuring element σ with a radius of σ_r, the erosion operator erodes a surface iteratively for any given number of steps. The dilation operator does the exact opposite and adds to the given mesh. Our work is similar to the approach of Zhen Chen et al. [9] in that they use morphological operations to compute discrete surface offsets. Since our focus here is a restricted form of mesh and topology simplification, we take a lot of freedom to interpret these morphological operators. We also do not compute exact offset surfaces. Instead, we calculate a simplified/approximated offset surface that is only useful for topological simplification. We also use Octrees to represent all operators, allowing the mesh to be reused for numerical simulation. Finally, we introduce a topological sphere level parameter similar to the scaled structuring element radius given by σ_r (scaled by the offset distance). We encourage reading upon the work of Silvia Sellán et al. [10] for a more direct interpretation of these operators in the context of three-dimensional geometry processing. They explore a surface-only approach that only modifies selective areas of a surface mesh. However, as they point out, their work suffers from the flaws of surface flows and isn't suitable for shrink wrapping applications. In our work, we open a surface iteratively until the surface has a spherical topology and then close it by iterative erosion. We explain the different morphological operators in the context of mesh generation in detail in the subsequent sections.

2.1 Mathematical morphology of surface meshes

Morphological operators in our present investigation do not differ very much from their image processing counterparts. In image processing, one represents morphological operators on simple two-dimensional grids. For example, a binary image can be represented on a Cartesian mesh with binary mask values (0 and 1). Therefore, the morphological operators in image processing can be considered as two-dimensional simplification of our morphological operators. First, we represent a surface mesh in a fixed or adaptively refined Cartesian mesh with binary values to identify the geometry's boundary. Then, we perform the morphological closing operation of this approximated geometry followed by surface extraction and projection. One key difference with our erosion operator is that we do not erode the geometry beyond its original boundary to preserve the internal volume. This is clearly shown in subsequent sections.

Let us consider the dilation and erosion operators on binary images and their geometric counterparts. Since other operators such as opening and closing can be built as a combination of these two, it is enough to

understand these two operators.

2.2 Dilation

We first dilate the given image with a square as a structural element. This is the most natural choice since the images are represented as pixels. Furthermore, this approach would only require adjacencies at the edge level to find neighbours from a computational viewpoint. Once the boundary of the image is identified, one step of dilation becomes a straightforward problem of finding the neighbouring faces of the border. The key detail in this approach is that the neighbours should be chosen in the positive normal direction of the boundary. Until a required criterion is achieved, this process can then be repeated for any number of steps.

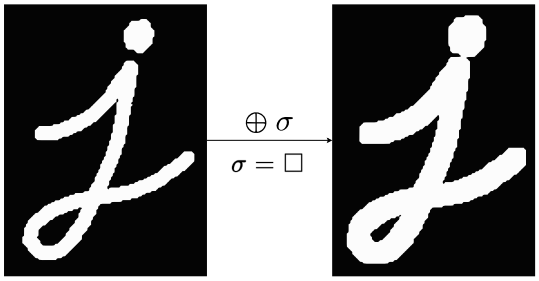


Figure 1: Dilation operation on a binary image

This operation remains the same in mesh processing. As stated earlier, we mark the boundary cells in a Cartesian mesh. Then the dilation operation for mesh processing is a natural extension of image processing to a three-dimensional problem. This operation is explained in detail with algorithmic workflows for computing a dilated surface from a triangulated mesh.

2.2.1 Erosion

The erosion operator also starts from the boundary of a given image. First, however, it finds the neighbours of the border in the negative normal direction. So these cells would usually be part of the image itself. A pivotal contrast to the dilation approach is that an image cannot be eroded infinitely. Since at some point, the erosion operation reaches a singularity.

In the case of shrink wrapping, erosion operation should not destroy the internal volume of a surface mesh. Hence, an erosion operation usually stops at the boundary of a surface mesh.

2.2.2 Closing

The closing operator combines the dilation and erosion operation. In the field of computer vision and image processing, this serves as a tool to close holes (or missing pixels) in a binary image. We perform the same operation on three-dimensional data for closing holes in our algorithms. To provide an easy comparison towards mesh processing, we have extracted a three-dimensional surface of the same geometry (by extruding the contour in the Z direction) and performed the closing operation on the same. The results for the same can be seen in figure 4.

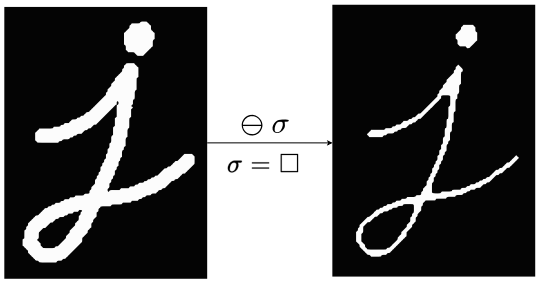


Figure 2: Erosion operation on a binary image

The two differences from the image processing approach are as follows

- We do not erode beyond the boundary of the input geometry
- We project the eroded geometry onto the input geometry

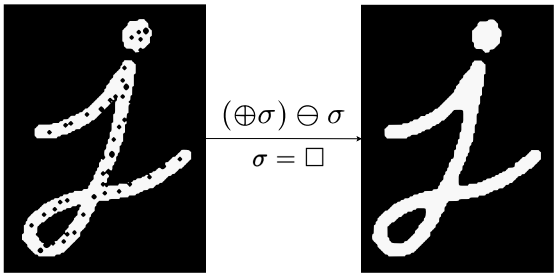


Figure 3: Closing operation on a binary image (Black blobs on the left indicate missing pixels). They are equivalent to holes or missing triangles in a surface mesh.

It can be observed in figure 4 that a closing operation on a surface not only closes the holes caused by missing triangles. It also seals the topological holes in a mesh. We leverage this benefit for shrink wrapping surface meshes for external aerodynamic simulations. However, the straightforward closing operation on its own is not suitable since it has no stopping criterion. Therefore, our workflow introduces a series of algorithms and data structures such as octrees to use these morphological operators efficiently for shrink

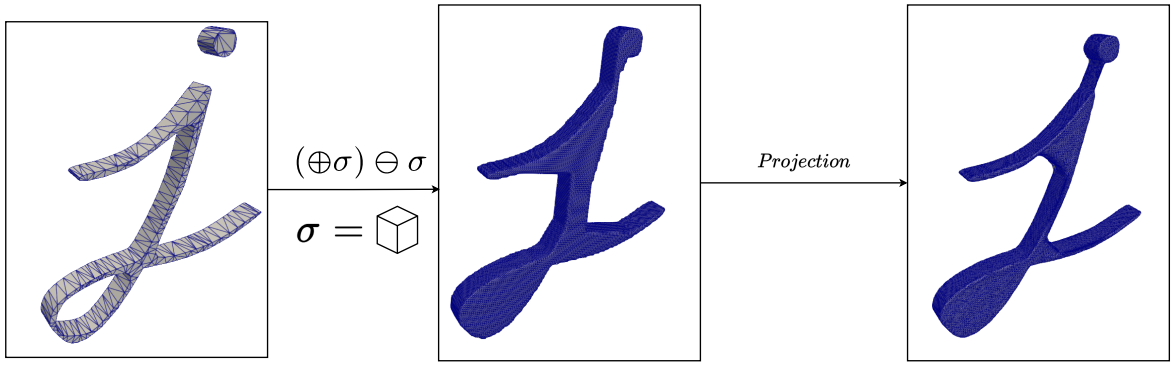


Figure 4: Closing operation on a three dimensional surface. This geometry is equivalent to the two dimensional binary images in figure 3. Closing operation is performed on the left most geometry and then projected onto the ground truth.

wrapping surface meshes. We explain these in detail in the subsequent section.

3. BASIC WORKFLOW

Our algorithm has the following basic components

1. Conversion of Boundary representation(B-Rep) to a volumetric representation (V-Rep)
2. Computation of signed distance function
3. Dilation of the input surface for a given topological sphere level
4. Erode the dilated surface to obtain a topologically hole-free offset surface
5. Iteratively project and smooth the dilated surface
6. Remesh to improve triangle quality (Optional)

We explain these individual components in detail along with the respective algorithms in the subsequent sections.

3.1 B-Rep to V-Rep

It was shown earlier that geometry information needs to be encoded as binary images for efficient computation of morphological operators in computer vision applications. This translates to a boolean value in every voxel of a Cartesian mesh[1] in three dimensions. However, it would require intersecting the surface mesh with the Cartesian mesh. This would only work for watertight surface meshes and lead to some inaccuracies in the case of degenerate geometries. Hence, we do not physically intersect the input geometry with the Cartesian mesh. Since we only need a scalar field to distinguish the inside and outside of the geometry. The usual workflow for Cartesian mesh generation starts with a bounding box computation as shown in figure 5. These can be an axis-aligned or oriented-bounding box. In either case, the generated Cartesian mesh would not be very beneficial for morphological operations. Morphological operators such as erosion and dilation are applied in successive layers. For example, the dilation operator starts from the boundary of a surface and dilates the surface one layer after another, as shown in figure 1. Since we use a cube or a voxel as a structuring element, it is easier to perform these operations successively if the geometry sits approximately in the centre of the Cartesian mesh. If the geometry is moved to the origin of the octree, a dilation operation may not completely dilate the entire surface in a given step. First, the mini ball algorithm [11] is used to obtain a tightly fitting sphere of an input geometry as shown in figure 6. Then we compute a bounding box for this sphere with a specified offset threshold to ensure that our geometry always sits precisely at the centre of our voxelization. Next, we refine all the cells inside the tightly fitting sphere as shown in figure 7. Post refinement, the generalized winding number approach [12] helps distinguish the cells inside and outside the geometry. The generalized winding number algorithm gives a solid angle value at every vertex in the octree mesh. This value is thresholded to mark the cells in the octree as inside, outside or boundary cells. This refinement allows us to get a more accurate surface description during the segmentation process. Spherical refinement also limits the inside-outside queries to the cells within the sphere. As a result, we do not need to query the generalized winding number for cells outside the sphere, thereby

[1]We use the terms Cartesian mesh and octree interchangeably throughout the paper. The reader should be aware that all the computations are performed on an octree, which we consider a special kind of Cartesian mesh.

saving computational time. Finally, we also ensure a 2:1 refinement in our octree for all elements in our workflow. The mesh generation approach referenced in algorithm 1 can be used for any kind of numerical simulation irrespective of the rest of the workflow.

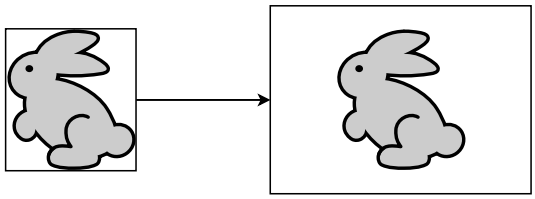


Figure 5: Normal bounding box computation (In case of many geometries, a manual offset threshold may be required for ensuring there are enough layers of mesh for morphological operations. The threshold might also be different for different directions.)

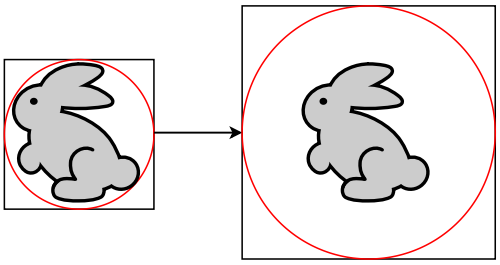


Figure 6: Spherical bounding box computation (In most cases, a threshold of 2 or 3 times the size of the bounding sphere is enough for all morphological operations. The scaling will be uniform irrespective of the geometry since only the sphere is scaled and the bounding box is always a perfect cuboid.

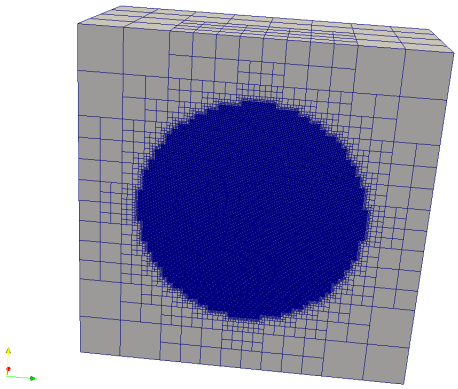


Figure 7: Spherical Refinement

Algorithm 1: B-Rep to V-Rep

Result: Voxelized mesh where every cell has a scalar associated with it (inside / outside)

Initialise Surface;
Compute a tight bounding sphere using the mini ball algorithm and store its radius and centre ;
Calculate the bounding box of the sphere, which is offset at a user-specified distance *(2.0 in our experiments)*;
Initialize a Cartesian mesh with a specified cell size or number of cells *(64 * 64 * 64 in all of our experiments)*;
forall *Cells of Cartesian mesh* **do**
  if *Cell inside bounding sphere* **then**
    Mark for refinement;
  end
  else
    Mark for coarsening;
  end
end
forall *Cells in bounding sphere* **do**
  Compute the Generalized winding number (This indirectly gives us the solid angle for all the cells in the octree);
end
Mark cells outside bounding sphere as outside and store this in the respective cells;
forall *Cells in bounding sphere* **do**
  if *Solid angle is higher than 0.9 steradians (based on our experimental observation)* **then**
    Mark the cell as inside and store this in the respective cell;
  end
  else
    Mark the cell as outside and store this in the respective cell;
  end
end

3.2 Computation of signed distance function

It is evident that once the cells of the octree are classified into inside and outside (using any approach such as generalized winding numbers in our case), an artificial signed distance function can be bestowed upon the voxelization as shown in figure 9. We rely on Generalized winding numbers since they are swift even on a CPU only computational environment and are immune to imperfections in the input surface to a large degree. A brief overview of this approach can be seen in appendix A. However, for the rest of the algorithmic workflow, one only needs to categorize the cells in the octree as inside or outside cells. These will

be used to build an approximate surface boundary which can be used for morphological operations described in the consequent sections. Our experimental observation has shown that a solid angle value of 0.9 steradians indicates cells inside a surface mesh, and everything else can be marked as outside. The bounding sphere computed in the mesh generation algorithm can be used to automatically mark all the cells outside the sphere as outside cells.

We use the term artificial since we do not compute the exact distance here. We only use an integer that indicates a particular voxel's relative position with its respective boundary voxel. As will be evident later, we do not need an exact signed distance field for computing a Genus simplified offset surface. A similar approach has been used by other researchers [13] to calculate intersection-free offset surfaces. We achieve the same by outward propagation from the zero level set voxels. This outward propagation is done along the normal outward direction of the surface mesh. Since we mark all the cells in the octree as inside or outside, zero level set voxels or boundary voxels can be determined by finding cells that contain faces that are part of both inside and outside cells. As opposed to the usual approaches, which intersect the surface mesh with the octree mesh, our proposed method is highly computationally efficient. All the morphological operations are explained with the help of a maple leaf geometry shown in figure 8.

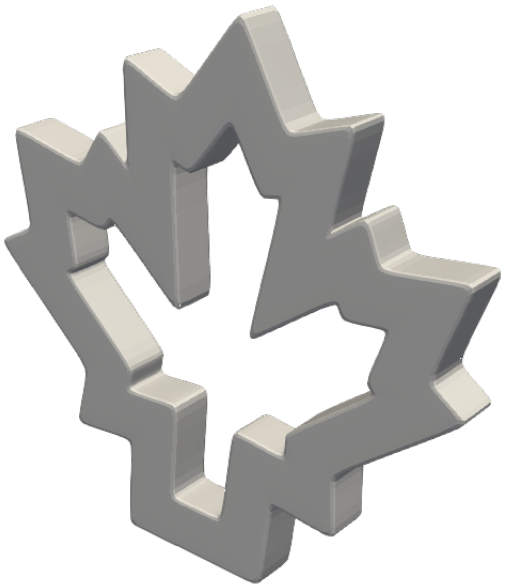


Figure 8: Maple leaf geometry

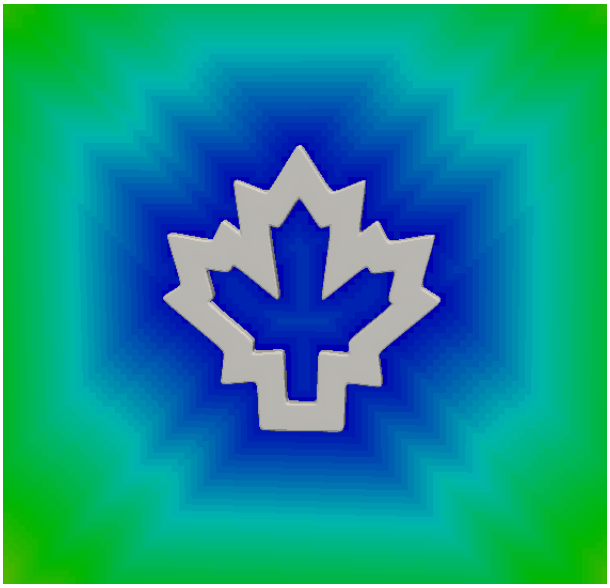


Figure 9: Artificial signed distance function of a maple leaf (Computed using our approach)

3.3 Dilation driven approximated offsets

In the previous section, we proposed a straightforward method to determine the zero level set or boundary voxels of a given surface mesh inside an octree. We already established that a dilation operation followed by an erosion operation performed in a sequence leads to the morphological closing operation as shown in figure 3. Our investigation also reveals that one does not need to dilate the boundary voxels across the entire voxelization. Instead, we only need to dilate the input surface until we achieve a spherical topology. Here, we use a user-specified parameter called **"Topological sphere level"**. This parameter is the only user-controlled input in the algorithm, and the choice of topological sphere level dictates the number of outward propagation levels as shown in algorithm 2. The bigger the hole in the geometry, the larger the topological sphere level. Effects of different Topological sphere levels are clearly shown in the numerical experiments. For example, the dilated maple leaf geometry can be seen in figure 10. It is clearly evident that a spherical topology is achieved after approximately 15 levels.

If complete automation is required from input to projection, the topological sphere level can be ignored, and the geometry can be dilated to the maximum level.

3.4 Erosion of dilated offset surface

The dilated surface should now have a spherical topology, and it needs to be eroded towards the input surface. This process is similar to the dilation except for

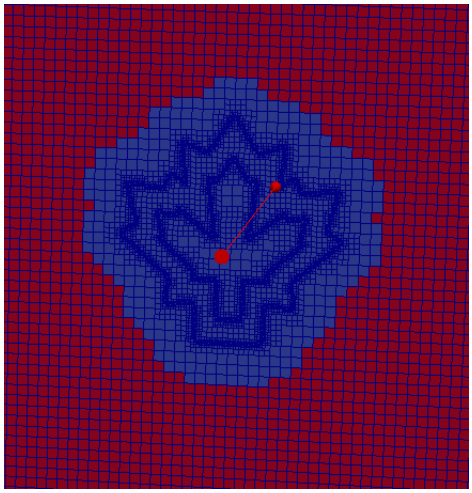


Figure 10: Dilated Maple leaf geometry with a spherical topology

Algorithm 2: Dilation of the input surface

Result: Dilated surface stored in the voxelized mesh
Initialize **interior_cells** as **seed_cells**;
Initialize current_topological_sphere_level to 0;
while ***current_level*** $\leq$ ***topological_sphere_level*** **do**
  Initialize a newer_seeds_cells_id vector;
  forall *cells in seed_cells* **do**
    forall *cell_neighbours in voxelized_mesh* **do**
      if *Neighbour is outside cell* **then**
        Add neighbour to newer_seeds_cells_id;
      end
    end
    Set newer_seeds_cells_id as seed_cells;
    Increment current_topological_sphere_level;
    if *current_topological_sphere_level eq topological_sphere_level* **then**
      Store these cells as topological_sphere_cells;
    end
  end
end

the marching direction. The number of levels would be the same as the topological sphere level chosen during the previous step of the algorithm. Once eroded, this will give a hole-free approximation of the input geometry. Once the surface is eroded to the given "Topological Sphere Level", the operation becomes straightforward. It is explained in detail in algorithm 3. The scalar field can be directly eroded until it hits the boundary voxels. This is where the erosion operation differs from the erosion operation in computer vision algorithms. In the case of surface mesh, the geometry is never eroded beyond the boundary voxels for volume preservation. This approach is relatively simple since a manifold mesh can be easily extracted without the need for any additional algorithms [14].

The offset surface is still embedded inside a volumetric mesh as shown in figure 11, and a surface needs to be extracted. Due to the artificial nature of the signed levels in the volumetric mesh, it is easy to distinguish the region where the genus simplified offset meets the external offset surface. Surface extraction becomes a simple task with this information. This is similar to the approach proposed for identifying boundary voxels from inside and outside voxels. The detailed algorithm for surface extraction is listed in algorithm 4.

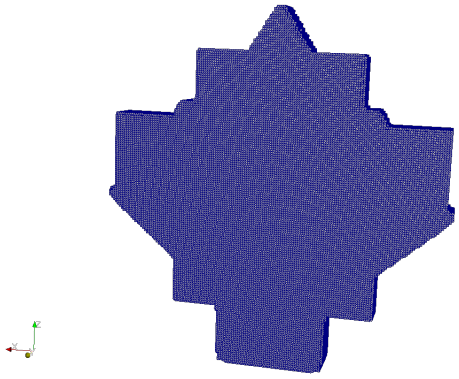


Figure 11: Eroded offset surface (Unsmoothed & Hole Free)

Algorithm 3: Erosion of the dilated offset surface

Result: Eroded surface stored in the voxelized mesh
Initialize topological_sphere_cells as seed_cell_ids;
forall *topological_sphere_levels* **do**
  Initialize a newer_seeds_cells_id vector;
  forall *cells in seed_cells* **do**
    forall *cell_neighbours in voxelized_mesh* **do**
      if *Neighbour is from lower topological_sphere_level* **then**
        Add neighbour to newer_seeds_cells_id;
      end
    end
    Set newer_seeds_cells_id as seed_cells;
  end
end
Final seed_cells form the basis for the Genus simplified offset surface;

Algorithm 4: Surface Extraction

Result: Genus simplified watertight surface
Initialize topological_sphere_cells as seed_cell_ids;
forall *topological_sphere_levels* **do**
  Initialize a newer_seeds_cells_id vector;
  forall *cells in seed_cells* **do**
    forall *cell_neighbours in voxelized_mesh* **do**
      if *Neighbour is from lower topological_sphere_level* **then**
        Add neighbour to newer_seeds_cells_id;
      end
    end
    Set newer_seeds_cells_id as seed_cells;
  end
end
Final seed_cells form the basis for the Genus simplified surface;

Algorithm 5: Projection and Smoothing

Result: Projected and smoothed mesh
Sample a uniform point cloud on the input surface;
Build a kD tree on the uniformly sampled point cloud;
forall *vertices in the Extracted surface* **do**
  Find the nearest vertex in the kD tree and move the vertex;
end
forall *vertices in the projected mesh* **do**
  Find one ring neighbourhood;
  Average the position of the current vertex with the vertices from the one ring neighbourhood ;
end

3.5 Projection and smoothing

The eroded surface needs to be projected onto the input geometry. With practicality in mind, we chose a point cloud- based approach over direct projection on the triangulated surface. In realistic industrial geometries, the construction of an AABB tree is costly and time-consuming and leads to failure in many cases. Since we chose to allow input meshes that are not perfectly two-manifold, the point cloud-based approach will support a broader range of input meshes, including those that are entirely degenerate, as shown in the later section. We approximate the input geometry as a uniformly sampled point cloud and then construct a kD tree [15] on it. We also extended our algorithm for point cloud due to this projection approach. However, one can choose a more sophisticated method that projects directly onto the triangles in the input surface. This might produce erroneous results if the surface mesh is completely degenerate. We experimented with both a direct projection approach and the one using point cloud sampling as shown in algorithm 12 and the results were satisfactory for our point cloud approach.

We can find the nearest neighbour in this point cloud for every vertex in the eroded surface and move the vertex to this position. Unfortunately, results do not look good at this stage, and the mesh seems slightly tangled. However, our experiments show that a few cycles of Laplacian smoothing followed by projection will immediately provide better quality results, as seen in figure 5.

Our investigation also shows that the geometry can be double wrapped to achieve better quality results. In double wrapping, the final result from the first run

Figure 12: Projected & Smoothed Mesh

of the algorithm can be passed back onto the same workflow to produce a better quality approximation. The mesh at this stage is already analysis suitable. If required, an optional remeshing step can be included for improving the mesh quality further. The geometry practitioner is not required to follow our heuristic-based approach. They can choose any remeshing algorithm (commercial or public domain). However, the proposed algorithms provide satisfactory results in our investigation.

3.6 Remeshing and quality improvement (Optional)

This step is entirely optional. The projected surfaces are well suited for analysis, and we show the same in numerical experiments for various surface and volumetric PDEs. However, the smoothed surface may still have a few tangled edges and triangles with lousy quality. Hence, we propose a heuristic-based remesh-

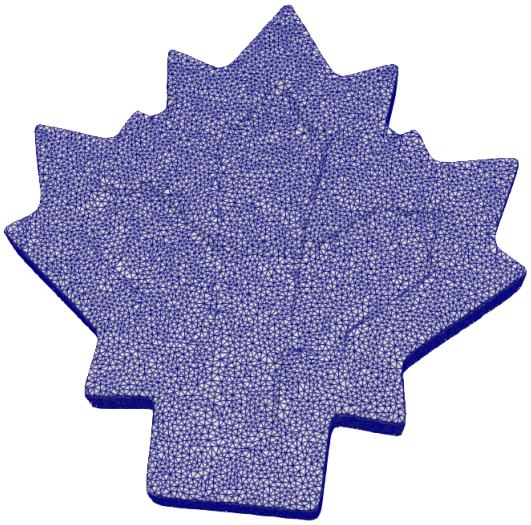


Figure 13: Remeshed and quality improved maple leaf geometry

ing approach to improve triangle quality quickly. An overview of the same can be seen in algorithm 6. In all our numerical experiments, this approach seems to improve the mesh quality vastly.

Algorithm 6: Remeshing and quality improvement algorithm (Heuristic driven)

Result: Quality improved mesh
Compute the bounding box of the wrapped mesh and its diagonal length and offset the diagonal length by 0.005 (Based on our experimental observation);
Remove degenerated triangles;
Split long edges (edges longer than the diagonal length with our offset);
Store the **num_vertices** at this level;
while ***current_num_vertices*** ! = ***previous_num_vertices*** **do**
    Collapse short edges (threshold of 1e-06);
    Remove obtuse triangles (Above 150.0);
end
Resolve self intersections;
Remove duplicated faces;
Remove isolated vertices;

An additional stopping criterion can ensure the termination of the algorithm. One can also replace this heuristic with a more sophisticated remeshing algorithm such as the one driven by different error metrics. However, we understand that this heuristic-based approach is not very elegant. One can replace our algorithm with a black-box remeshing algorithm from libraries like CGAL[16]. Since we produce a genus zero surface in most cases, spherical parameterization based remeshing approaches can also be considered an alternative. However, we found meshes at the projection stage suitable for numerical simulations. It is not within the scope of our work to investigate a dedicated remeshing approach. In fact, for the numerical experiments shown in the subsequent section, we use the geometries from the projection stage and ignore the remeshing routine altogether.

4. NUMERICAL EXPERIMENTS

We perform experiments on a wide variety of input geometries that help underscore the robustness of our algorithm. We noticed that even in the case of entirely ill-formed artefacts from industry, we could guarantee some form of a Genus simplified geometry. A wide variety of surfaces and their shrink-wrapped counterparts are shown in appendix B. In all cases, our algorithm produced a valid two-manifold surface mesh without any holes, and the Hausdorff distance was within a reasonable range (99% of the vertices are close to the original mesh).

4.1 Effect of topological sphere level

As stated earlier, **"Topological sphere level"** is the only parameter in the algorithm. In simpler terms, this is the number of layers the algorithm needs to travel along the positive normal direction of an input surface. It can be increased or decreased depending on the size of the biggest hole in the geometry. Since the algorithm is fast, the topological sphere level value can be chosen even on a trial and error basis. The algorithm could be allowed to propagate to the maximum possible level. However, this might significantly increase the algorithm's run time, which is entirely unnecessary in our case. We show some examples of the same in section 5.2.1 and some of its pleasant side effects.

4.2 Effect on bad quality geometries

We show a car geometry in figure 16 which is missing most of its bottom. We use a coarser grid to produce a simplified approximation of the car geometry. It can be seen that our algorithm produces a tight wrap even in this case and simplifies the geometry. Since the offset computation does not require an exact segmentation of the geometry boundary, the hole free offset computation works even in such extremely poor quality geometries. The topological sphere level can be tuned on a trial and error basis for such geometries until the complete geometry is wrapped. In the case of skull geometry shown in figure 14, it has many non-manifold edges and has many disconnected components. There is no pre-processing requirement

on either of the geometries, and their shrink-wrapped results are shown in figure 15 a perfect two-manifold mesh without any leaks.

4.3 Boosting projection quality using external sources

We mentioned earlier that we double wrap the geometries to achieve a better projection quality. This is primarily due to the geometric structure of morphological erosion. It leads to a competitive projection which can be beneficial in many cases. For example, if the bottom is entirely missing, rather than failing to close the hole in the bottom, vertices are projected to the next closest area, which would be the boundary of the bottom hole. This ensures a good priori for the next wrapping stage. Hence, the double wrapping stage would provide a better distribution of triangles. This is an undesirable yet pleasant side effect of the competitive nature of the projection of the algorithm to stick to whatever comes first. However, if the geometry practitioner/end-user in the industry would like to have better conditioning for such holes / even topological holes, using external sources would help. In our earlier work [6], we proposed a semi heuristic algorithm to detect topological and geometric cavities in a triangulated mesh. Just like our present algorithm, it does not need a perfect two manifold mesh. We show later in section 5.2 that this can also be useful for selective Genus closing.

4.4 Runtime analysis

The algorithm proposed in this paper is memory bound, and memory usage depends entirely on the size of the octree and the number of triangles in the input mesh. Even though we mentioned the Topological sphere level as the only parameter in the algorithm, different geometric parameters related to octree could also be tuned towards requirements. All the investigations in the paper are carried out with a fixed initial grid size of 64 x 64 x 64 and 3 levels of spherical refinement, and it gives satisfactory results. The code was implemented in C++ with OpenMP parallelization, and the software is available as a binary (for Linux) for reproduction. However, the source code is not available due to commercial interests. All the results were obtained on a laptop with a 12 core Intel Core i7 CPU with 64 gigabytes of RAM. The program runtime (in minutes) versus the number of triangles in the input mesh is shown in figure 17. Our algorithm can scale linearly with the increasing number of triangles. The largest triangulation in our investigation had 23 million triangles, and the algorithm converged to a manifold surface in under 10 minutes.

5. APPLICATIONS AND VARIANTS

5.1 Surface PDE

Surface-based finite element methods[17] and Boundary element methods (BEM) have gained a lot of momentum in recent years among the geometry processing and engineering research community in general. Our goal here is to show the shrink-wrapped geometries suitability for surface PDE computations on standard benchmark problems. For the first example, we obtain selective eigenmodes of the Laplace equation (using Laplace Beltrami operator) based on the work of Vallet, B et al. [18]. For the second example, we solve for Geodesic distance using the heat method on our shrink-wrapped geometries. There are numerous PDE's and applications besides the ones shown in our investigation. We make no effort to suggest that we modified or improved the methodology in either of these examples. We only use these as an example to show that our geometries are suitable for analysis.

5.1.1 Eigenmode decomposition of the Laplace equation

Eigenmode decomposition of the Laplace equation[18] can be helpful to represent functions on surfaces. It is most useful in real-time deformation problems. We use the famous Laplace-Beltrami operator to obtain its eigen decomposition on shrink-wrapped geometries. A generalized eigenvalue problem is of the form.

$$Ax = \lambda Bx \tag{1}$$

Here we use Laplace-Beltrami operator L for A and use per vertex mass matrix $\mathbf{M}$ of the shrink-wrapped surface as B. The Eigen mode functions can then be represented as follows[18]:

$$\mathbf{f} = \sum_{i=1}^{n} a_i \phi_i \tag{2}$$

where a_i represents the scalar coefficients and ϕ_i represents the Eigen functions which satisy $\Delta\phi_i = \lambda_i \phi_i$. Hence, we end up with an equation as follows

$$\mathbf{L}\phi_i = \lambda_i \mathbf{M}\phi_i \tag{3}$$

For low-frequency modes, it produces smooth and slowly changing functions on the shrink-wrapped mesh. For a more thorough understanding of the theory of manifold harmonics, it is best to refer to Vallet, B et al. [18]. We only chose Eigen decomposition as an example problem for showcasing the analysis suitability of our shrink-wrapped surfaces, and a detailed explanation of the underlying theory is beyond the scope

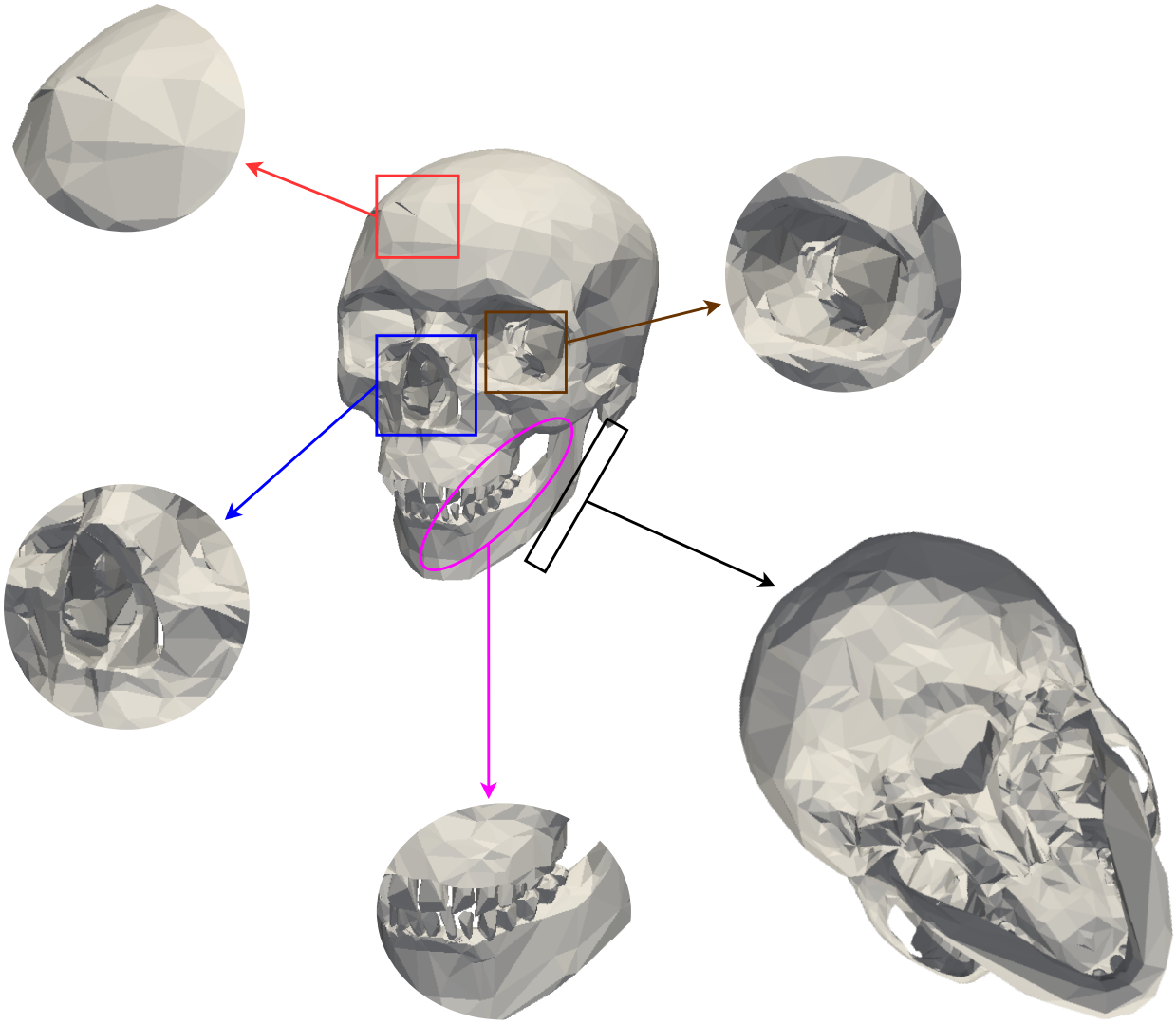


Figure 14: A human skull geometry with atleast 15 non-manifold edges and many disconnected components. Various defects in a skull geometry (gaps, disconnected tooth, non-manifold edges and holes) are highlighted. It also includes the bottom view of the skull which is entirely deformed.

of our work. We solve for the first three Eigenmodes on a shrink-wrapped surface.

5.1.2 Geodesic distance (Heat method)

Geodesic distance has numerous applications in geometry processing and physics-driven simulations. We calculate the geodesic distance using the heat method[19] on our shrink-wrapped meshes. Geodesic distance contours at different point sources show a smooth transition of the heat contours across the surface. Furthermore, there is no noticeable noise anywhere in the scalar field, proving the algorithm's robustness in producing simulation capable meshes.

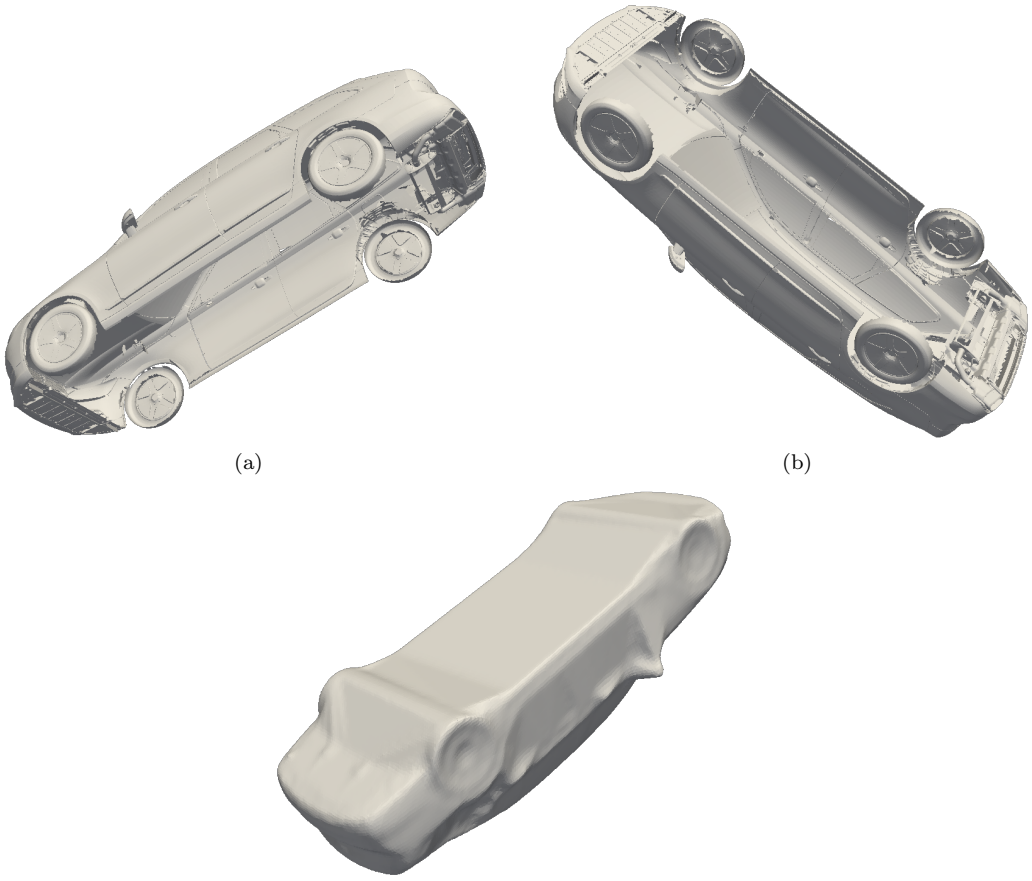
5.2 Selective Genus Closing

Almost all of the shrink wrapping algorithms are either used as remeshing (i.e. preserve the genus of the input mesh) or surface simplification algorithms (i.e. turn the input mesh into a topological sphere or Genus zero). However, there are scenarios where an indus-

(a) Wrapped (Front View)

(b) Wrapped (Bottom View)

Figure 15: Wrapped skull geometry (Watertight geometry with a genus zero)



(a)

(b)

(c)

Figure 16: Bad quality geometries shrink wrapped (Car with hole) - Input mesh along with wrapped output mesh. It can be observed that the bottom of the car is completely closed.

trial practitioner is only interested in closing selective holes. We could not find any other works that directly address this problem. We propose two ways to do this in our paper.

- Implicit Genus control

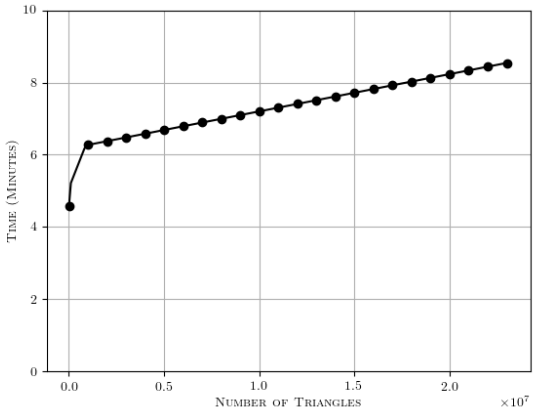


Figure 17: Run time comparison for increasing number of triangles

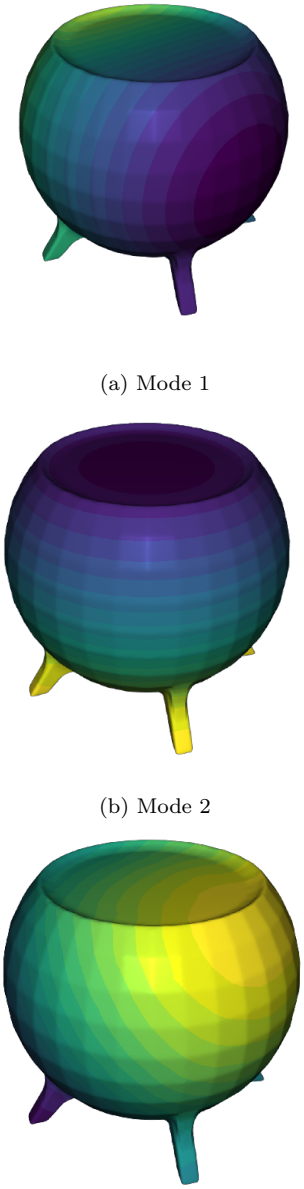
- Explicit Genus control

5.2.1 Implicit Genus control

In this case, no additional algorithms are required. However, to achieve the necessary genus, it can be an iterative process. We leverage the topological sphere level's ability to control the genus implicitly. A lower value for the topological sphere level usually leads to incomplete closing of the geometry. This can be a desirable side effect in the case of selective Genus control. We have an example geometry in figure 20 below with a huge topological hole in the top and a smaller one in the bottom. The effects of different topological sphere levels are shown in the figure 21. It can be observed that a value of 50 yields a Genus zero surface; however, at a level 10, only the smallest hole in the mesh is closed.

5.2.2 Explicit Genus control

Explicit Genus control requires prior information about the topological holes in the mesh. Therefore, we use our topological hole detection algorithm to accomplish the same. In this case, topological hole information is extracted from the hole detection algorithm, and this information is used as a boundary condition in the dilation stage. The hole detection algorithm would provide the centre, and the radius of the holes and the desired hole radius can be given as a criterion. The radius criterion is only suitable for circular holes. If the geometry also has non-circular holes, hole surface patches from the algorithm can be used to compute the surface area of individual holes. This can be used as a further filtering criterion. Then the faces which are part of the desired holes are not diffused into the volume, thus preserving the structure. The algorithm 2 would have to be modified for Explicit Genus control, and it can be seen in algorithm 7



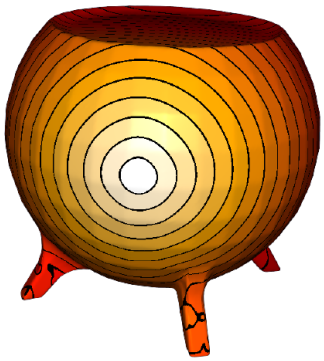
(a) Mode 1

(b) Mode 2

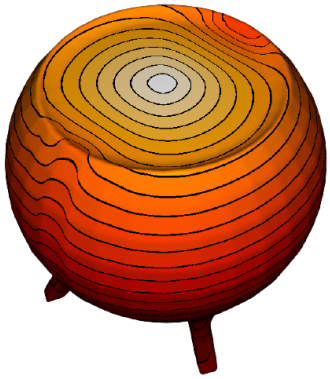
(c) Mode 3

Figure 18: First 3 Eigen modes of a shrink wrapped mesh

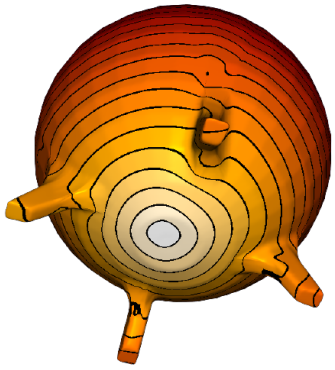
As explained earlier, the dilation process happens layer by layer. Therefore, to achieve selective genus control, one would have to ignore the blacklisted boundary cells for the dilation process.



(a) Side



(b) Top

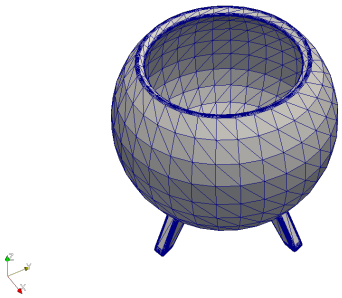


(c) Bottom

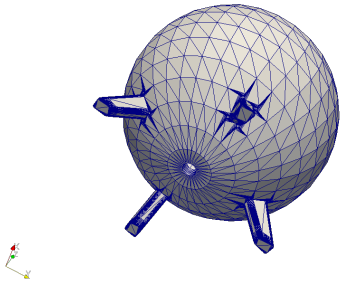
Figure 19: Geodesic distance computed on shrink wrapped meshes at different sources using Heat method

5.3 Fluid Volume Extraction

Fluid volume extraction is yet another excellent application of our shrink wrapping algorithm. If the goal is to generate the fluid volume or topological holes in a geometry, simple boolean operations help extract these volumes. There are practical difficulties in extracting topological holes in a geometry (multiple holes) since



(a) Bigger hole



(b) Smaller hole

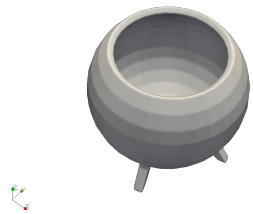
Figure 20: An example geometry with a big hole on top and a small hole at the bottom

there will be a lot of noise to sift through. However, suppose the industrial practitioner is interested in extracting a single fluid volume like a fluid volume of an interior of a car. In that case, it is possible to automate the process entirely. Once we have a Genus zero shrink-wrapped surface, a straightforward algorithm can be laid out with the following steps

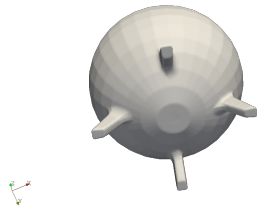
1. Subtract the genus zero shrink wrapped geometry from the input geometry
2. Split the result based on connectivity and compute the component volumes
3. Largest volume geometry is the fluid volume

We did not implement any of the boolean operations for this algorithm and used the existing functionalities from CGAL[16].

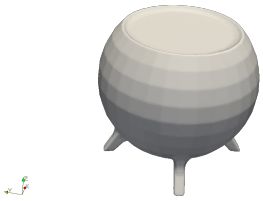
Smoothed particle hydrodynamics (SPH) is a meshless method requiring a volumetric point distribution for numerical simulation. In example 1, we show a partial car in figure 22 that has been shrink-wrapped, and its



(a) Level 10 (top)



(b) Level 10 (bottom)



(c) Level 50

Figure 21: Shrink wrapped geometry for different topological sphere levels

fluid volume has been extracted for a smoother particle hydrodynamic simulation.

In the second example, we show a case for raspberry pi, and its shrink-wrapped geometry and the subsequent fluid volume in figure 23. Again, it can be seen that our algorithm produces a very clean fluid volume. The results can be further de-noised to turn them into developable surfaces.

5.4 External aerodynamic simulation

One of the primary focuses of our investigation is external aerodynamic simulations. Since they do not require all the internal components of a geometry, a simplified geometry can be considered during early prototyping. We ran the RANS simulations on a generic shrink wrapped car geometry using OpenFoam[20]. The car geometry was meshed using the snappyHexMesh tool from OpenFoam with a base refinement of 10 cells in all three directions. We considered the entire car geometry for numerical simulation without using any symmetry boundary conditions. We used a steady-state incompressible SIMPLE solver for solving the Reynolds Averaged Navier Stokes equation with a k-ω SST turbulence model. A velocity inlet with a velocity of 20 ms was used as the boundary condition. The shrink-wrapped geometry produces physically consistent results, as shown in the figure 24. We have not made a rigorous mathematical analysis or experimental verification for these simulations. We only performed this simulation to show the suitability of shrink-wrapped geometries for computational fluid dynamic simulations.

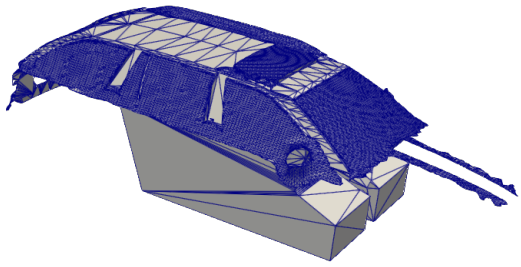
Algorithm 7: Dilation of the input surface (with Genus control)

Result: Dilated surface stored in the voxelized mesh

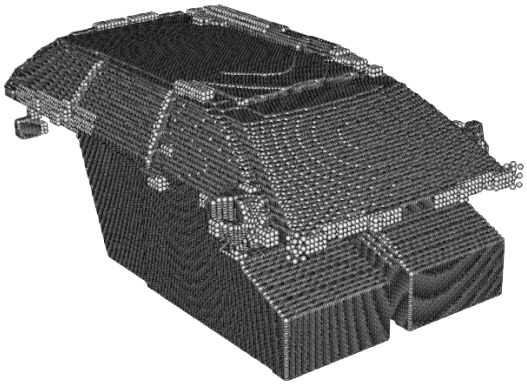
```
Detect holes using hole detection algorithm;
Mark the cells that intersect with the holes and
 mark them as blacklisted cells;
Initialize interior_cells as seed_cells;
Ensure that the blacklisted cells are removed
 from the initial seed_cells;
Initialize current_topological_sphere_level to 0;
while current_level ≤ topological_sphere_level
 do
    Initialize a newer_seeds_cells_id vector;
    forall cells in seed_cells do
        forall cell_neighbours in voxelized_mesh do
            if Neighbour is outside cell then
                Add neighbour to
                 newer_seeds_cells_id;
            end
        end
        Set newer_seeds_cells_id as seed_cells;
        Increment current_topological_sphere_level;
         if current_topological_sphere_level eq
         topological_sphere_level then
            Store these cells as
             topological_sphere_cells;
        end
    end
end
```

6. COMPARISON AGAINST SIMILAR APPROACHES

Many recent papers in geometry processing use deep learning-based approaches to solve geometric problems. One such recent article is *Point2Mesh*[21] where the authors shrink wrap an oriented point cloud based on self-similarity. Their algorithm is built on mesh-based convolutional neural networks and similar algorithms found in computer vision. We extended our



(a) Wrapped Car

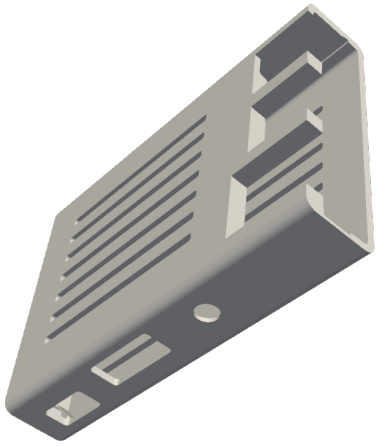


(b) Car fluid volume

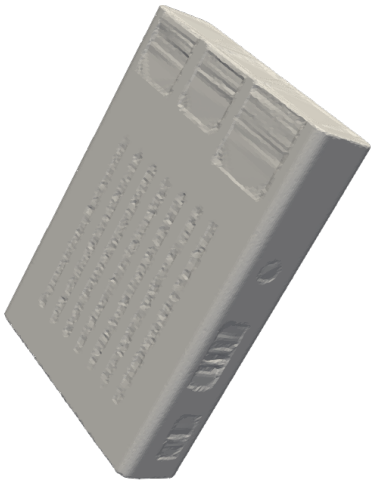
Figure 22: A Partial car geometry that has been shrink wrapped and its fluid volume extracted as a volumetric point cloud. Volumetric point clouds are otherwise considered as particle distributions for meshless methods like Smoothed particle hydrodynamics method.

shrink wrapping algorithm for point clouds to make a fair comparison. Since we rely on generalized winding numbers for inside-outside segmentation, there is a straightforward extension to point clouds. Usually, this is done by computing point areas using a Voronoi diagram[12]. However, we found that such a complex approximation is not always required. We compute a series of local triangulations and consistently ensure their orientation using a greedy approach. Hence, our algorithm does not require an oriented point cloud. This modification ensures that we do not have to modify the rest of our shrink wrapping algorithm. Once the inside-outside segmentation is done in the octree, the rest of the algorithm remains the same. We chose the same geometries as the authors, and we found that we produce similar quality results in most cases. While we provided a variety of heuristics to avoid this in a surface mesh-based approach, we did not thoroughly investigate the same for point clouds since it was beyond the scope of our work.

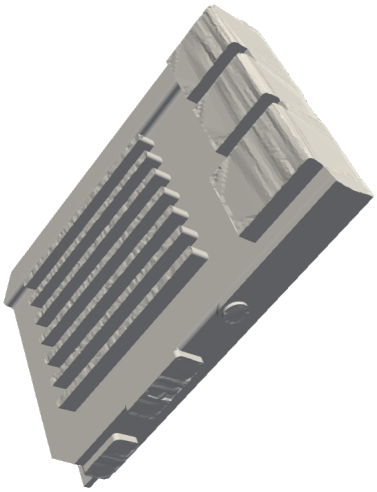
An observation can be made that our algorithm complements the authors' work nicely. If our algorithm is considered an initial priori, it improves the con-



(a) Raspberry pi case geometry



(b) Wrapped Geometry



(c) Fluid volume

Figure 23: A raspberry pi case and its fluid volume

vergence speed of *Point2Mesh* algorithm. For exam-

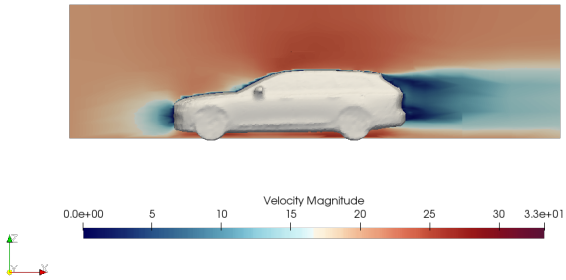


Figure 24: External aerodynamic simulation of a generic car model (shrink wrapped)

ple, their algorithm relies on an initial mesh computed based on a convex hull approach and converges to a ground truth based on self-similarity. However, our algorithm's mesh before the projection stage serves as a better priori. It also converges the *Point2Mesh*[21] algorithm in a fraction of the time. It can also be noticed that their approach is not meant for mechanical parts, and features found in CAD geometries cannot be preserved without significant modifications. Our goal is to produce genus-zero surfaces for aerodynamic simulations. We optionally provide variants that allow various levels of control over the genus of the wrapped surface. However, their approach is strictly a surface construction approach and does not consistently produce zero surfaces. It can only be achieved by stopping the algorithm halfway; the reconstruction might not be accurate globally in such cases, and a projection might be required.

7. LIMITATIONS

Since the proposed algorithms are built on top of morphological operators, they inherit the drawbacks of mathematical morphology. In concave regions, the erosion stops once it hits the closest triangle in the mesh. This leads to over closing of specific features in the mesh. However, since the primary application for our algorithms is external aerodynamic simulations, these do not make a massive difference in the macro scale. There are scenarios, however, where the algorithm can significantly alter the geometry. The techniques listed in section 4.3 can alleviate this problem to a large extent. However, no algorithmic guarantees can be made here.

8. CONCLUSION

We presented a practical algorithm that can perform Genus simplified shrink wrapping for polyhedral surfaces with the help of morphological operators. We also show that these algorithms extend easily for point clouds. One can implement the algorithms proposed in this paper in the same mesh used for numerical simulation, thereby avoiding another expensive volumetric mesh generation process. The algorithms also run at a linear runtime and are not heavily CPU bound. Furthermore, user-defined constraints and additional interactivity could lead to further improvements in the output quality of the algorithm. Finally, our fluid volume extraction variant of the algorithm can significantly benefit industrial fluid dynamic practitioners.

ACKNOWLEDGEMENTS

Österreichische Forschungsförderungsgesellschaft has funded this research under an industrial Ph.D. grant titled **"HIOMESH"**.

References

[1] Attene M., Campen M., Kobbelt L. "Polygon Mesh Repairing: An Application Perspective." *ACM Comput. Surv.*, vol. 45, no. 2, Mar. 2013. URL `https://doi.org/10.1145/2431211.2431214`

[2] Esteve J., Brunet P., Vinacau A. "Approximation of a Variable Density Cloud of Points by Shrinking a Discrete Membrane." *Computer Graphics Forum*, vol. 24, no. 4, 791–807, 2005

[3] Nooruddin F., Turk G. "Simplification and repair of polygonal models using volumetric techniques." *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 2, 191–205, 2003

[4] Lee Y.K., Lim C.K., Ghazialam H., Vardhan H., Eklund E. "Surface Mesh Generation for Dirty Geometries by Shrink Wrapping using Cartesian Grid Approach." P.P. Pébay, editor, *Proceedings of the 15th International Meshing Roundtable*, pp. 393–410. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006

[5] Wang Z.J., Srinivasan K. "An adaptive Cartesian grid generation method for 'Dirty' geometry." *International Journal for Numerical Methods in Fluids*, vol. 39, no. 8, 703–717, 2002

[6] Vijai Kumar S., Vuik C. "A Simple and Fast Hole Detection Algorithm for Triangulated Surfaces." *Journal of Computing and Information Science in Engineering*, vol. 21, no. 4, 02 2021. URL `https://doi.org/10.1115/1.4049030`. 044502

[7] Najman L., Talbot H. *Introduction to Mathematical Morphology*, chap. 1, pp. 1–33. John Wiley & Sons, Ltd, 2013. URL `https://doi.org/10.1002/9781118600788.ch1`

Figure 25: Few point cloud geometries from *Point2Mesh*[21] along with its output. It can be noticed that our wrap algorithm produces equally smooth results except for a few artefacts created as a result of morphological operators.

[8] Jeulin D. *Analysis and Modeling of 3D Microstructures*, chap. 19, pp. 421–444. John Wiley & Sons, Ltd, 2013. URL https://doi.org/10.1002/9781118600788.ch19

[9] Chen Z., Panozzo D., Dumas J. "Half-Space Power Diagrams and Discrete Surface Offsets." *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 10, 2970–2981, 2020

[10] Sellán S., Kesten J., Sheng A.Y., Jacobson A. "Opening and Closing Surfaces." *ACM Trans. Graph.*, vol. 39, no. 6, Nov. 2020. URL https://doi.org/10.1145/3414685.3417778

[11] Gärtner B. "Fast and Robust Smallest Enclosing Balls." *Proceedings of the 7th Annual European Symposium on Algorithms*, ESA '99, pp. 325–338. Springer-Verlag, London, UK, UK, 1999

[12] Barill G., Dickson N., Schmidt R., Levin D.I., Jacobson A. "Fast Winding Numbers for Soups and Clouds." *ACM Transactions on Graphics*, 2018

[13] Liu S., Wang C.C.L. "Fast Intersection-Free Offset Surface Generation From Freeform Models With Triangular Meshes." *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 2, 347–360, 2011

[14] Lorensen W.E., Cline H.E. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm." *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, p. 163–169. Association for Computing Machinery, New York, NY, USA, 1987

[15] Blanco J.L., Rai P.K. "nanoflann: a C++ header-only fork of FLANN, a library for Nearest Neighbor (NN) with KD-trees." https://github.com/jlblancoc/nanoflann, 2014

[16] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 5.3 edn., 2021. URL https://doc.cgal.org/5.3/Manual/packages.html

[17] Alexa M., Wardetzky M. "Discrete Laplacians on General Polygonal Meshes." *ACM Trans. Graph.*, vol. 30, no. 4, Jul. 2011. URL https://doi.org/10.1145/2010324.1964997

[18] Vallet B., Lévy B. "Spectral Geometry Processing with Manifold Harmonics." *Computer Graphics Forum*, vol. 27, no. 2, 251–260, 2008. URL https://doi.org/b24vx4

[19] Crane K., Weischedel C., Wardetzky M. "The Heat Method for Distance Computation." *Commun. ACM*, vol. 60, no. 11, 90–99, Oct. 2017. URL http://doi.acm.org/10.1145/3131280

[20] Foundation T.O. "OpenFOAM v8 User Guide.", 2021. URL https://cfd.direct/openfoam/user-guide

[21] Hanocka R., Metzer G., Giryes R., Cohen-Or D. "Point2Mesh: A Self-Prior for Deformable Meshes." *ACM Trans. Graph.*, vol. 39, no. 4, jul 2020. URL https://doi.org/10.1145/3386569.3392415

[22] Jacobson A., Kavan L., Sorkine-Hornung O. "Robust Inside-Outside Segmentation Using Generalized Winding Numbers." *ACM Trans. Graph.*, vol. 32, no. 4, Jul. 2013

[23] Carrier J., Greengard L., Rokhlin V. "A Fast Adaptive Multipole Algorithm for Particle Simulations." *SIAM J. Sci. Stat. Comput.*, vol. 9, no. 4, 669–686, Jul. 1988. URL https://doi.org/10.1137/0909044

APPENDIX A: GENERALIZED WINDING NUMBER BASED SOLID ANGLE FOR SURFACE SEGMENTATION

We rely on a classical differential geometry idea called winding numbers, which uses solid angles for surface segmentation. For a given surface S, for a query point p, the solid angle is the signed surface area of the projection of S onto the unit sphere centred at p as shown in figure 26. We rely on its definition in discrete setting[22].

$$\omega(\mathbf{p}) = 2 * \tan^{-1}\left(\frac{det([\mathbf{abc}])}{abc + (\mathbf{a}.\mathbf{b})c + (\mathbf{b}.\mathbf{c})a + (\mathbf{c}.\mathbf{a})b}\right)$$
$$Triangle = \{v_i, v_j, v_k\}$$
$$\mathbf{a} = v_i - p, \mathbf{b} = v_j - p, \mathbf{c} = v_k - p$$
$$a = \|\mathbf{a}\|$$
$$b = \|\mathbf{b}\|$$
$$c = \|\mathbf{c}\| \tag{4}$$

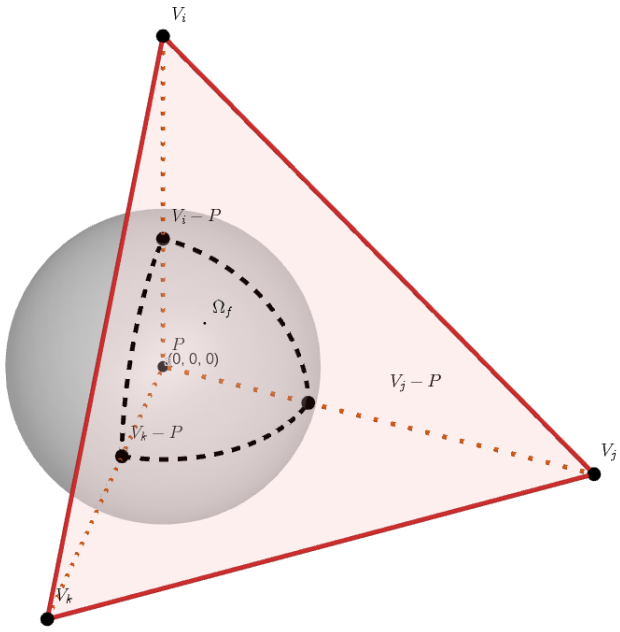


Figure 26: Solid angle of a query point

Given this relation for solid angle given by $\omega(\mathbf{p})$, we can compute winding number as follows

$$w(\mathbf{p}) = \sum_{n=1}^{nTriangles} \frac{1}{4\pi}\omega_f(\mathbf{p})$$

For every query point, the direct implementation $w(\mathbf{p})$ would require the contribution of all triangles in the surface mesh. Since this would yield a solution with time complexity of $O(n^2)$, we rely on the work of Gavin Barill et al[12]. They proposed a fast multipole method[23] style implementation that uses direct computation for triangles near the query point and approximates the result everywhere else, making it a $O(log(n))$ algorithm.

APPENDIX B: VARIOUS INPUT GEOMETRIES AND THEIR SHRINK WRAPPED RESULTS

We show a few geometries and their shrink-wrapped results. The results shown below are shrink-wrapped with a topological sphere level of 15. In all the cases shown below, our algorithm could produce a genus zero surface consistently. It also creates a two-manifold mesh suitable for external aerodynamics or finite element analysis simulations.

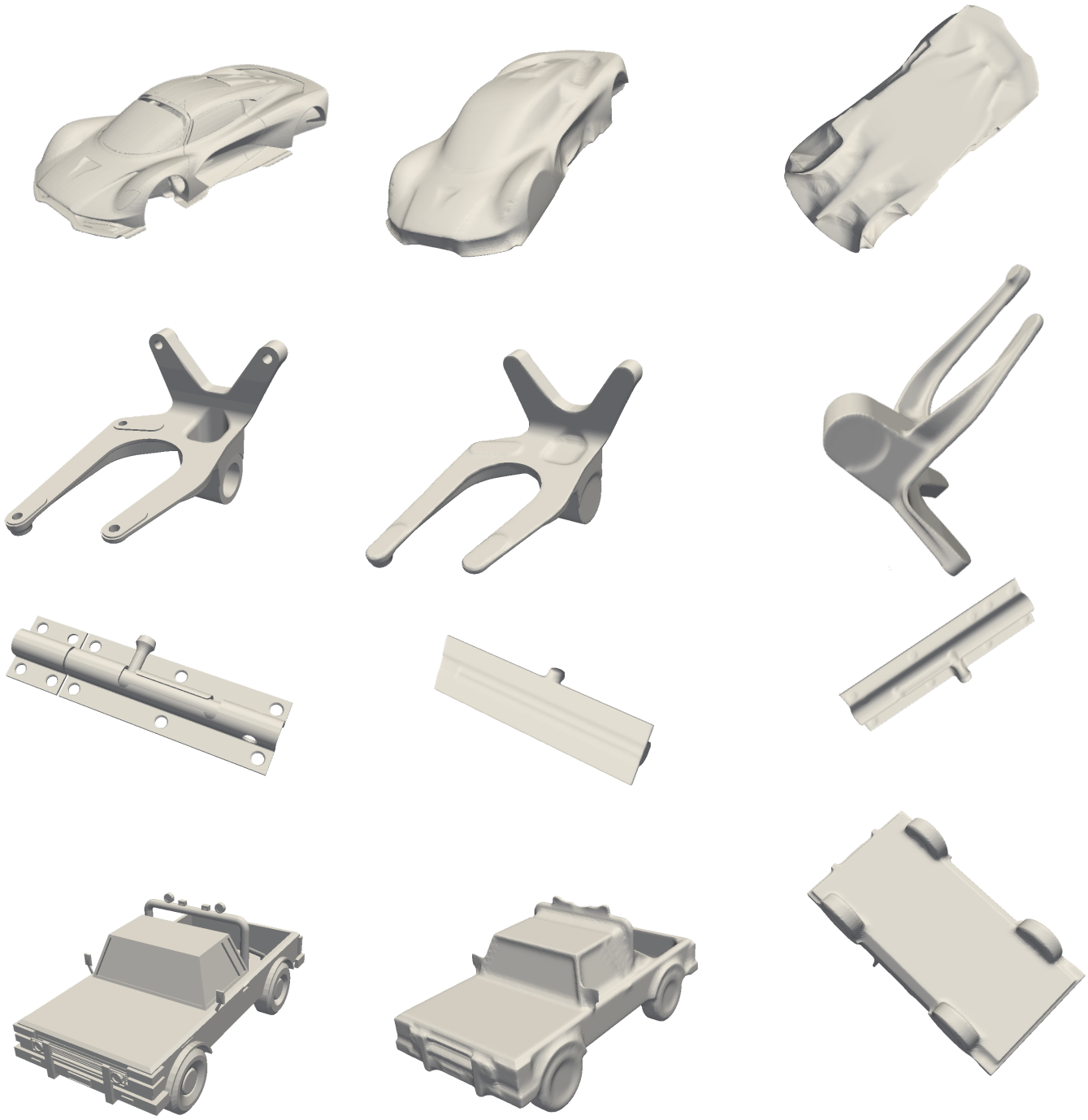


Figure 27: A variety of geometries and their shrink wrapped results.