University of Groningen

Bachelor's thesis

Morphing planar curves using subdivision surfaces

Remi Brandt
s2509644

Primary supervisor
Dr. Jiří Kosinka

Secondary supervisor
Pieter Barendrecht

July 6, 2016

Abstract

An approach to morph between a pair of planar curves is to embed these curves in 3D at different z-axis positions, construct a connecting surface between the two curves, and interpret parallel slices of this surface perpendicular to the z-axis as the intermediate curves composing the morph. The connecting surface may be subdivided in an attempt to improve the quality of the morph. In this bachelor's thesis, the relationship between used subdivision scheme (Loop, sqrt(3), interpolating sqrt(3) or modified Butterfly subdivision) and morph quality is evaluated based on various quality measures.

An implementation is presented which is able to subdivide triangular meshes using the considered subdivision schemes and slice the resulting meshes at a position on the z-axis. Furthermore, it can compute various quality measures.

Test meshes have been collected which have been subdivided. The resulting subdivided meshes have been used to compute quality measures by the presented implementation. Based on data generated by the implementation, it is concluded that the considered approximating subdivision schemes create smoother meshes and preserve simpleness (the absence of self crossings) better than the considered interpolating subdivision schemes. It was however found that interpolating subdivision schemes preserve sharp features better and displace the original connecting surface less than approximating subdivision schemes.

Contents

1 Introduction **4**

2 Related Work **5**
2.1 Choice of technology . . . 5
2.2 OpenFlipper . . . 6
2.3 OpenMesh . . . 6
2.4 Subdivision . . . 7
2.4.1 Catmull-Clark subdivision . . . 7
2.4.2 Loop subdivision . . . 8
2.4.3 Modified Butterfly subdivision . . . 8
2.4.4 $\sqrt{3}$ subdivision . . . 9
2.5 Quality measures . . . 9
2.5.1 Quantitative measures . . . 10
2.5.2 Qualitative measures . . . 11

3 Implementation **12**
3.1 Subdivision . . . 12
3.2 Slicing . . . 12
3.2.1 Area preserved slicing . . . 14
3.3 Area . . . 14
3.4 Total positive curvature . . . 15
3.5 (Offset) non-overlapping area . . . 15
3.6 Self intersection test . . . 16
3.7 Derivative . . . 17

4 Results **18**
4.1 Methods . . . 18
4.1.1 Test meshes . . . 18
4.1.2 Metrics . . . 18
4.2 Results . . . 18
4.2.1 Smoothness . . . 21
4.2.2 Sharpness preservation . . . 23
4.2.3 Displacement . . . 25
4.2.4 Simpleness preservation . . . 25
4.2.5 Closedness preservation . . . 26

5 Conclusions **27**
5.1 Subdivision schemes . . . 27
5.2 Quality measures . . . 27

6 Future Work **29**

7 References **30**

8 List of Tables **31**

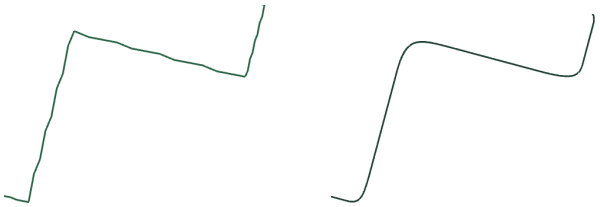
9 List of Figures **32**

10 Appendix **35**
10.1 Slice curves . . . 35
10.1.1 Original . . . 35
10.1.2 Loop Subdivision . . . 39
10.1.3 Sqrt3 Subdivision . . . 42
10.1.4 Interpolating Sqrt3 Subdivision . . . 45
10.1.5 Modified Butterfly Subdivision . . . 48
10.2 Area . . . 52
10.2.1 "-" to "+" . . . 52
10.2.2 "o" to "o" . . . 53
10.2.3 "U" to "V" . . . 55
10.2.4 Elephant . . . 56
10.2.5 Fertility . . . 57
10.2.6 Kitten . . . 59
10.2.7 Moai . . . 60
10.3 Total positive curvature . . . 61
10.3.1 "-" to "+" . . . 61
10.3.2 "o" to "o" . . . 63
10.3.3 "U" to "V" . . . 64
10.3.4 Elephant . . . 65
10.3.5 Fertility . . . 67
10.3.6 Kitten . . . 68
10.3.7 Moai . . . 69
10.4 Non-overlapping area . . . 71
10.4.1 "-" to "+" . . . 71
10.4.2 "o" to "o" . . . 72
10.4.3 "U" to "V" . . . 73
10.4.4 Elephant . . . 74
10.4.5 Fertility . . . 75
10.4.6 Kitten . . . 77
10.4.7 Moai . . . 78
10.5 Offset non-overlapping area . . . 79
10.5.1 Elephant . . . 79
10.5.2 Fertility . . . 81
10.5.3 Kitten . . . 82
10.5.4 Moai . . . 83
10.6 Reflection lines . . . 85
10.6.1 "-" to "+" . . . 85
10.6.2 Fertility . . . 86
10.7 Gaussian curvature . . . 87
10.7.1 Kitten . . . 87
10.8 Mean curvature . . . 88
10.8.1 Kitten . . . 88

1 Introduction

Morphing between two planar curves can be defined as follows; given two planar curves, commonly referred to as source and target curve, a sequence of interpolated curves are constructed such that blended curves adjacent to each other in the sequence are geometrically close [13]. The adopted approach to morphing uses subdivision surfaces to morph between planar curves. Two planar curves are represented in 3D space, the source curve is embedded in the $z = 0$ plane and the target curve in the $z = 1$ plane. These curves are then connected by a surface, the planar slices perpendicular to the z-axis represent intermediate curves. The connecting surface is represented as a triangular mesh. In an attempt to improve morphing quality, the connecting surface may be subdivided.

In the following example, a "+" curve (source curve) has been connected with a "+" curve which is rotated 45 degrees around the z-axis (target curve). Figure 1a illustrates part of a slice curve of the original mesh at position $z = 0.34$ and Figure 1b illustrates part of a slice curve of the mesh at position $z = 0.34$ after Loop [1] subdivision has been performed on the original mesh. In this example, Loop has increased the smoothness of the intermediate curve but has not fully preserved the sharp features which are present in the source and target curve.



(a) Slice of original mesh

(b) Slice of Loop subdivision surface

Figure 1: Slice at position $z = 0.34$ before and after Loop subdivision.

The implementation presented in this thesis can subdivide triangular meshes using Loop, $\sqrt{3}$ [2], interpolating $\sqrt{3}$ [3] and modified Butterfly [4] subdivision and slice the resulting meshes at a position on the z-axis. Optionally, the implementation is able to scale the slice such that the area enclosed by intermediate curves linearly transfer from the area enclosed by the source curve to the area enclosed by the target curve. Furthermore, the presented implementation can generate data which describes how subdivision has affected the morph quality. The implementation does this by determining the *area* and *total positive curvature* of slice curves, calculating the *non-overlapping area* of slices taken from the subdivided mesh and the original mesh at the same position on the z-axis, determining the *offset non-overlapping area* of meshes: the *non-overlapping area* of two slices taken at different positions on the z-axis from the same mesh, *testing for holes* in the mesh, *testing for self-intersections* and determining the *derivative* of meshes: the difference of face normals among adjacent faces. The developed implementation is discussed in Section 3.

Using results generated by the developed implementation, the relationship between used subdivision schemes and morph quality is evaluated based on various morph qualities in Section 4.

[1] An approximating subdivision scheme developed by Loop in 1987 for triangular meshes.
[2] An approximating subdivision scheme developed by Kobbelt in 2000 for triangular meshes
[3] An interpolating subdivision scheme developed by Labsik, and Greiner in 2000 for triangular meshes.
[4] An interpolating subdivision scheme developed by Zorin, Schröder, and Sweldens in 1996 for triangular meshes.

2 Related Work

Morphing (also known as metamorphosis or blending) is an active research area. Earlier research has been undertaken that is relevant to what is to be accomplished by this project. In this chapter, findings from those research projects, as well as adopted technology will be discussed.

2.1 Choice of technology

A number of frameworks were considered when a framework was chosen on top of which this project could be implemented. On the basis of the evidence currently available, it seems fair to suggest that OpenFlipper is the most suitable platform to build a plug-in on top of. The competitive advantage of each and the cost/effort to learn/use these are summarized below:

- **Geomagic[5], Maya[6], and Cinema 4D[7]** Each of these programs are proprietary software. The programs provide an SDK through which plug-ins can be developed. Yet, no access to, or modification of the source code is allowed. Furthermore, functionality of plug-ins is restricted by the SDK. Plug-ins for Maya, Geomagic and Cinema 4D can be written in C++. Extensive developer documentation is available. The programs are unnecessarily complex for this project as the scope of the programs are not just processing and visualization of geometric models.

- **MeshLab[8]** Open source program, written in C, C++ and supports Python scripting. The program is intended for the processing and editing of unstructured triangular meshes. MeshLab only supports Loop and Butterfly subdivision by default. Gaussian and mean curvature visualization are supported. MeshLab can be used free of charge.

- **Blender[9]** Open source program, written in C, C++ and supports Python scripting. Extensive developer documentation is available. The program is unnecessarily complex for this project as the scope of the program is not just processing and visualization of geometric models. Blender only supports Catmull-Clark and some interpolating subdivision schemes by default. Blender can be used free of charge.

- **Standalone application** Any programming language can be chosen, and there is complete freedom licensing-wise. However, basic functionality like subdivision would need to be be implemented.

- **OpenFlipper[10]** Open source program, programmed in C++ and supports Python scripting. Extensive developer documentation is available. Build-in subdivision and slicing functionality. Supports Loop, $\sqrt{3}$, interpolating $\sqrt{3}$ and modified Butterfly subdivision for triangular meshes, as well as Catmull-Clark subdivision for quadrilateral meshes by default. The program has a number of build-in visualization tools helpful for mesh quality analysis. OpenFlipper can be used free of charge.

[5] Geomagic: http://geomagic.com
[6] Maya: http://maya.com
[7] Cinema 4D: http://www.maxon.net/products/cinema-4d-studio
[8] MeshLab: http://meshlab.sourceforge.net
[9] Blender: https://www.blender.org
[10] OpenFlipper: http://www.openflipper.org

2.2 OpenFlipper

OpenFlipper is an open-source framework for processing and visualization of complex geometric models. OpenFlipper is developed on top of the Qt framework [11]. The Qt framework allows for easy cross-platform development. The OpenFlipper framework consists of two parts: The core application and a set of plug-ins [10]. For this project, a plug-in for OpenFlipper is developed.

The core of OpenFlipper has a number of responsibilities. Firstly, it deals with representing the currently loaded scene in a hierarchical scene graph data structure. This data structure is used to take care of transforming and rendering its nodes. These nodes correspond to objects, OpenGL states, as well as auxiliary information such as selections. Due to the scene graph's hierarchical structure, the states of a node are also applied to all of its attached child nodes. Secondly, the core creates a Qt and OpenGL context, the application window and some basic GUI elements. Thirdly, the core supplies a basic rendering system used as a fall-back solution on systems with outdated graphics cards or application setups that do not incorporate any rendering plug-in. Lastly, the core manages interaction and communication between plug-ins and organizes the user interface [10].

Additional functionality is added through dynamically linked plug-in libraries. OpenFlipper's API provides a variety of plug-in interfaces from which plug-ins may inherit in order to assess a set of specialized functions. These functions are used for the communication between plug-ins and the core application, as well as between different plug-ins [10].

OpenFlipper's API provides a variety of functions which is helpful for this project. Firstly, a number of subdivision schemes are implemented, namely: Loop, $\sqrt{3}$, interpolating $\sqrt{3}$ and modified Butterfly subdivision for triangular meshes, as well as Catmull-Clark subdivision for Quadrilateral meshes. Secondly, OpenFlipper's API provides visualization functionality which can be used to assess mesh quality [9]. Lastly, OpenFlipper's API enables the creation of a polyline where a plane intersects with a mesh.

OpenFlipper is licensed under the GNU Lesser General Public License version 3 as published by the Free Software Foundation [12].

2.3 OpenMesh

Triangular meshes can approximate arbitrary shapes to any approximation tolerance, and they can be processed efficiently by the current graphics hardware. A polygonal mesh consists of a set of vertices, edges, faces, and topological relations between them. Based on these relations, a data structure defines how each element is stored, and what references to its neighborhood it needs. Three prominent types of data structures to store polygonal meshes are face-based, edge-based, and half-edge based data structures. Face-based data structures store for each face pointers to its vertices and its neighboring faces. Edge-based data structures store for each edge pointers to both vertices and to the neighboring edges [1].

The adopted data structure to store triangular meshes for this project is OpenMesh. OpenMesh is a half-edge data structure for the static representation, and dynamic handling of arbitrary polygonal meshes [1]. OpenFlipper was built on top of OpenMesh.

Subdivision algorithms frequently need to navigate through the one-ring-neighborhood of a vertex. OpenMesh is suitable for such algorithms because it allows for constant–time access to the one–ring neighborhood of vertices. While face-based structures lack the explicit representation of edges, and edge-based structures lose efficiency because of their missing orientation of edges, halfedge-based structures overcome these disadvantages [1].

[11] Qt: https://www.qt.io
[12] GNU lesser general public license: https://www.gnu.org/licenses/lgpl-3.0.html

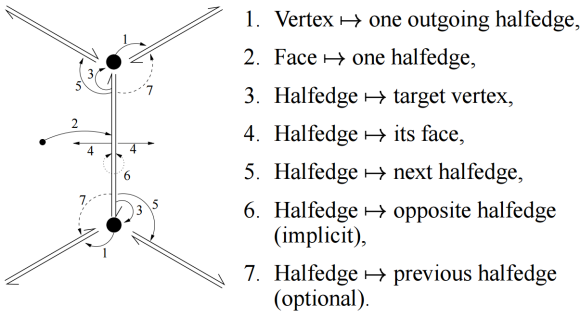


Figure 2: Topological relations stored by OpenMesh [1].

OpenMesh is licensed under the GNU Lesser General Public License version 3 as published by the Free Software Foundation [13].

2.4 Subdivision

A subdivision surface is the limit surface obtained by recursively applying a subdivision scheme to a given polygonal mesh G. The subdivision scheme creates new vertices, edges, and faces, resulting in a denser and smoother polygonal mesh. Depending on the type of subdivision scheme used, the location of initial vertices of mesh G may be updated as well. Interpolating schemes require original vertices to remain at their original position, where this is not the case for approximating schemes. The surfaces generated with interpolating schemes are sometimes not sufficiently smooth, and the subdivision rules have to be applied more times than would be needed with approximating schemes to obtain the same results. The approximating schemes, on the other hand, generally do not produce surfaces which interpolate the control-mesh, but they do result in smooth surfaces [14]. Subdivision schemes implemented by the Subdivision plug-in of OpenFlipper will be discussed in this section. It is considered beyond the scope of this project to extensively discuss the details of each subdivision scheme, therefore only a basic description of each algorithm is given.

2.4.1 Catmull-Clark subdivision

Approximating subdivision scheme developed by Catmull and Clark in 1978 for quadrilateral meshes [3, 14, 15]. The smoothness of the limit surface is G^2 everywhere except at extraordinary points, where it is G^1. The plug-in that is developed for this project accepts triangular meshes as input. Catmull-Clark subdivision could be used anyway as any triangle can be converted into three quads. It should be noted that the quality of meshes which have first been converted from a triangular mesh to quad mesh and have then been subdivided with Catmull-Clark may be relatively low [15]. In each subdivision step, the following refinements are performed [3]:

- For each face, add a face point which is the average of all original points defining the face.
- For each edge, add an edge point of which the location is determined by the average of the neighboring face points and the two original endpoints.

[13] GNU lesser general public license: https://www.gnu.org/licenses/lgpl-3.0.html

- Translate each original point to point p, according to

$$p = \frac{Q}{n} + \frac{2R}{n} + \frac{S(n-3)}{n} \tag{1}$$

where

Q: The average of the new face points of all faces adjacent to the old vertex.

R: The average of the midpoints of all old edges incident with the old vertex.

S: Old vertex.

- New edges are now formed by connecting each new face point to the new edge points of the edges defining the old face, and connecting each new vertex point to the new edge points of all old edges incident with the old vertex.
- New faces are then defined as those enclosed by new edges.

2.4.2 Loop subdivision

Approximating subdivision scheme developed by Loop in 1987 for triangular meshes [8, 14, 15]. The smoothness of the limit surface is G^2 everywhere except at extraordinary points where it is G^1. In each subdivision step, the following refinements are performed [8, 14]:

- Divide each triangle into 4 triangles by splitting each edge at its midpoint, and connecting new vertices.
- The locations of new vertices are determined by the weighted average of the locations of original vertices in its 1-sing neighborhood.
- The locations of old vertices are adjusted by considering the weighted average of the location of its neighboring vertices.

2.4.3 Modified Butterfly subdivision

An interpolating subdivision scheme developed by Zorin, Schröder, and Sweldens in 1996 for triangular meshes [17]. The smoothness of the limit surface is G^1 everywhere except for extraordinary points [12]. The scheme is based on the Butterfly scheme introduced by Nira Dyn, David Levine and John A. Gregory [5]. In each subdivision step, the following refinements are performed [17]:

- To each edge a new vertex is added at the edge's midpoint.
- The location of each new vertex is based on the weighted average of nearby vertices. Which vertices are considered in the average depends on the valence of the two vertices which are connected by the edge, whether or not the vertices are K-vertices, and whether or not they are extraordinary. Furthermore the location of each new vertex is determined by whether or not the edge is a boundary edge.

2.4.4 $\sqrt{3}$ subdivision

OpenFlipper supports both an interpolating and an approximating $\sqrt{3}$ subdivision scheme.

Interpolating $\sqrt{3}$ scheme
Scheme developed by Labsik, and Greiner in 2000 for triangular meshes. In each subdivision step, the following refinements are performed [7]:

- A new vertex is inserted for every face of the given mesh.
- These vertices are connected to the old vertices of the triangle. All old edges of the triangle are now flipped as illustrated in Figure 3.
- The location of the new vertices are determined based on the weighted average of near vertices.

Approximating $\sqrt{3}$ scheme
Scheme developed by Kobbelt in 2000 for triangular meshes. The smoothness of the limit surface is G^2 everywhere except for the extraordinary points where it is G^1 [6].

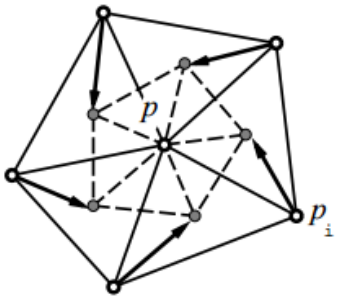


Figure 3: Flipping of edges in $\sqrt{3}$ subdivision [6].

2.5 Quality measures

No formal definition of a high-quality morph exists. It can, however, be stated that for a morph to be of high-quality, it should posses the following characteristics [4]:

- Smoothness: adjacent intermediate curves should transition smoothly and should each retain smooth features of the original curves.
- Sharpness preservation: intermediate curves should each retain sharp features present in the original curves.
- Displacement: subdivision should change meshes as little as possible because the original source and target curves should be as similar as possible to the slices at $z = 0$ and $z = 1$ respectively of connecting subdivision surfaces.
- Simpleness preservation: if source and target curves do not cross themselves, so should intermediate curves.

- Closedness preservation: intermediate curves should be defined for every position along the z-axis and if the source and target curves are closed, so should intermediate curves.
- Feature preservation: although beyond the scope of this project, features common to both source and target curves (for example head or legs) should be preserved during the morph.

Table 1 lists these morph qualities, stating for each of them how they can be measured.

Qualities	**Quantitative measures**	**Qualitative measures**
Smoothness	Offset non-overlapping area	Reflection lines
	Area	Gaussian curvature
	Total positive curvature	Mean curvature
Sharpness preservation		Reflection lines
		Derivative
Displacement	Non-overlapping area	
Simpleness preservation	Self intersection test	
Closedness preservation	Mesh hole detection	

Table 1: Morph quality measures

2.5.1 Quantitative measures

- **Non-overlapping area** Measures the difference between two slices. To compute the *non-overlapping area*, slice two meshes at position $z = n$, and compute the *non-overlapping area* between the two slices. Original mesh and subdivision surface should in the context of shape morphing be as close as possible to each other. Therefore small values of *non-overlapping area* are preferred.

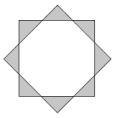


Figure 4: Indicated in gray is the *non-overlapping area* of two squares.

- **Offset non-overlapping area** Measures how smoothly the transition from source to target curve through intermediate curves occurs. To compute *offset non-overlapping area*, slice a mesh at position $z_0 = n$ and at position $z_1 = n - \epsilon$ where $\epsilon > 0$, then compute the *non-overlapping area* between the two slices. Smaller values indicate less difference between the slices at position $z_0 = n$ and $z_1 = n - \epsilon$.
- **Area** Area enclosed by slice curves. Area of intermediate curves should transition smoothly between that of source and target [4].
- **Mesh hole detection** No holes may be introduced by subdivision schemes. To find holes, slice the mesh at a number of positions and determine whether the resulting slices are closed.

- **Total positive curvature** *Total positive curvature* of a slice can be determined by adding up the absolute value of curvature at each point of the slice. *Total positive curvature* allows the assessment of the change in curvature of the slice as the slice position is changed. The variance of *total positive curvature* among adjacent slices should be small. The absolute value of curvature is taken, as this allows to assess the smoothness of slices. *Total positive curvature* of interpolated curves should vary smoothly between that of source and target. A lower number indicates a curve which is closer to being a convex curve.
- **Self intersection test** Determine for each face whether it intersects with another face. Reports self-intersection of a mesh.

2.5.2 Qualitative measures

- **Gaussian curvature** Indicates how locally elliptic, hyperbolic or parabolic a surface is. An area where *Gaussian curvature* has little variance indicates a smooth area.
- **Mean curvature** Reports areas where curvature changes abruptly. An area where *mean curvature* has little variance indicates a smooth area.
- **Derivative** Determine how much face normals differ between neighboring faces. *Derivative* can be determined in the x, y, z or all axes. Areas where there is much difference between neighboring faces suggest that sharp features are present.
- **Reflection lines** Conceptually, *reflection lines* are obtained by computing reflections of a set of long linear parallel light sources, aligned with a fixed direction. Visualizing such reflections makes many types of surface irregularities apparent [16].

3 Implementation

The presented implementation can subdivide triangular meshes using Loop, $\sqrt{3}$, interpolating $\sqrt{3}$ and modified Butterfly subdivision and slice the resulting meshes at a position on the z-axis. Optionally, the implementation is able to scale the slice such that the area enclosed by intermediate curves linearly transfer from the area enclosed by the source curve to the area enclosed by the target curve.

Furthermore, the presented implementation allows to analyze how subdivision has affected morph quality using the quality measures presented in Section 2.5.

OpenFlipper was chosen as the development platform because extensive developer documentation is available, it was programed in a programming language the author of this thesis was already familiar with, it has build-in subdivision (Loop, $\sqrt{3}$, interpolating $\sqrt{3}$ and modified Butterfly subdivision) and slicing functionality, the program has a number of build-in visualization tools helpful for mesh quality analysis and OpenFlipper is an open source program which can be used free of charge. Please consult Section 2.1 for a detailed comparison of alternatives.

3.1 Subdivision

The subdivision features of OpenMesh were used to facilitate the subdivision features of the plug-in.

3.2 Slicing

Slicing functionality has been developed for the plug-in. A morphing axis can be selected (the z-axis in this thesis) and a position between 0 and 1 can be selected which indicates where to slice all objects or only target objects between its highest and lowest position on the morph axis. Slices may consist of multiple components and components may be closed or open. After the slice has been computed, a polyline for each component of the slice is added to the scene-graph. If the slice consists of multiple components, they are grouped together. Slicing functionality has been implemented in the following way:

Do the following for each triangular object or for each of the target triangular objects in the scene-graph:

1. Determine the cut plane from the slice position and morph axis.
2. Mark all half edges and vertices of object as uncut.
3. **While** it is possible to find an edge E of the object that is cut by the plane, has not been cut before, and if E intersects with the cut plane at an endpoint, that endpoint has not been cut before either:
   (a) Find a face of which edge E is an edge. Set $current_face$ equal to this face.
   (b) $flip_dir$ = false.
   (c) $closed$ = true.
   (d) $expansionLevel = 0$.
   (e) $stop$ = false.
   (f) $linePoints$ is a polyline, currently without vertices.

(g) *startCandidates* is a vector of faces, currently without any faces.

(h) *expandable* is a vector of faces, currently only contains *current_face*.

(i) **While** *stop* == false:

i. *stop* = true

ii. **Loop** through each edge part of *current_face*, stop if *break*() is called:

If edge is already cut:

- Skip edge

Else:

- If there is no intersection of the cut plane and edge, skip edge.
- If both endpoints of the edge lie on cut plane, skip edge.
- If the intersection is at an endpoint of edge, mark that vertex as cut.
- Mark edge as cut.
- *stop* = true

If there is a face on the other side of the edge:

- *current_face* = The face on the other side of the edge.

Else:

- *stop* = true.

If *flip_dir* == false:

- Add intersection point to at the back *linePoints*.

Else:

- Add intersection point to at the front *linePoints*.
- *closed* = false

break()

iii. **If** *stop* **== true**

If did not cut an edge:

If *expansionLevel* > 3:

- Error: expanded rings but still nothing found.

Else:

- Add *expansionLevel* ring of the start-face to *startCandidates*.
- Increase *expansionLevel* by one.
- Remove all elements from *expandable*.

If start candidates is not empty:

- *current_face* = *startCandidates* last item
- Push *current_face* at back to *expandable*
- Pop at back from *startCandidates*
- *stop* = false

Else if *flip_dir* == false:

- *flip_dir* = true
- *stop* = false

(j) Add *linePoints* (represents a slice component) to scene-graph together with *closed* which indicates whether or not it is closed.

4. If slice consists of multiple components, add them to a group.

3.2.1 Area preserved slicing

Optionally, slice curves can be scaled by the plug-in, such that the area enclosed by the slice is linearly interpolated from that enclosed by the source to that enclosed by the target curve. The source, intermediate and target slice curves must each be closed and may each only consist of one component. Area preserved slicing has been implemented in the following way:

Do the following for each triangular object or for each of the target triangular objects in the scene-graph:

1. Slice object at position p (position between source (0) and target (1) curve), resulting in polyline L. Stop if slice consists of multiple components.
2. Slice object at position 0 and compute area of slice. Stop if slice consists of multiple components.
3. Slice object at position 1 and compute area of slice. Stop if slice consists of multiple components.
4. Determine centroid of L.
5. Compute area of L
6. Compute scale factor s:

$$s = \sqrt{\frac{a_0}{a_p} * (1 - p) + \frac{a_1}{a_p} * p} \tag{2}$$

   where

   p = Position between source (0) and target (1) curve
   a_0 = Area enclosed by slice at position 0
   a_1 = Area enclosed by slice at position 1
   a_p = Area enclosed by slice at position p

7. For each point of L:
   (a) Translate centroid to origin.
   (b) Multiply point by scale factor s.
   (c) Translate centroid back to original position.
8. Add L to scene-graph.

3.3 Area

The plug-in allows to compute the *area* enclosed by slice curves. Computing the *area* of slices has been implemented in the following way:

Do the following for each triangular object or for each of the target triangular objects in the scene-graph:

1. Slice the object at a position.

2. For each component of the slice, compute the area of the component using the Shoelace algorithm [2]. Return -1 if at least one of the components of the slice is not closed. Otherwise return the cumulative area.

3.4 Total positive curvature

The plug-in allows to compute the *total positive curvature* (also referred to as total absolute curvature) enclosed by slice curves. Computing the *total positive curvature* of slices has been implemented in the following way:

Do the following for each triangular object or for each of the target triangular objects in the scene-graph:

1. Slice the object, for each component (*polyline*) of the slice, do the following:
   (a) Filter points of the *polyline*, such that no points adjacent to each other in the *polyline*'s list of points are at the same position.
   (b) **If** *polyline* is open:

$$tpc = \sum_{i=0}^{n-3} \pi - \arccos(\widehat{(p_{i+1} - p_i)} \cdot \widehat{(p_{i+1} - p_{i+2})}) \tag{3}$$

   where

   n = Number of points part of *polyline*.
   p_i = Point of *polyline* with index i (zero based numbering).

   (c) **Else**:

$$tpc = \sum_{i=0}^{n-1} \pi - \arccos(\widehat{(p_{i+1} - p_i)} \cdot \widehat{(p_{i+1} - p_{i+2})}) \tag{4}$$

   where

   n = Number of points part of *polyline*.
   p_i = Point of *polyline* with index i (zero based numbering), where $p_n = p_0$ and $p_{n+1} = p_1$.

2. Return the cumulative *tpc* of all components.

3.5 (Offset) non-overlapping area

The plug-in allows to compute the *non-overlapping area* of a slice of Mesh A, and a slice of Mesh B. A number of steps can be selected: Slice the indicated number of times with equal distance between slices. For example: when equal to 2, slice at top and bottom. When equal to 3, slice at top, middle and bottom. Optionally, offset position can be selected: Slice Mesh A at step n and Mesh B at step $n - 1$. It is not supported to determine the *non-overlapping area* of two slices (S_A and S_B) which consist out of a different number of components or where one component of slice S_A overlaps multiple components of slice S_B. Computing the *non-overlapping area* of two slices has been implemented in the following way:

1. Slice both meshes at their respective position, store face normals of the meshes at slice curve points.
2. Determine the centroid of each slice component.
3. $nonOverlappingArea = 0$.
4. Make sure there are the same number of components of A as there are of B, else return -1.
5. For each component of slices A:
   (a) Find the closest component of slice B to the current one of A, based on its centroid.
   (b) Determine whether the two components are "One in the other", "One next to other" or "Intersecting".
   (c) **If** "One in the other":
      i. Add to *nonOverlappingArea* the absolute difference in area between both components.
   (d) **Else if** "One next to other":
      i. Add to *nonOverlappingArea* the sum of the areas of both components.
   (e) **Else if** "Intersecting":
      i. Determine the outline of both components by first determining the top right vertex, this vertex will be part of the outline. Then points are added to the outline, starting from the top right vertex, by traversing both slice lines and adding points whenever both lines intersect and adding the line points of the outer curve in between intersections. Which component is locally the outer one is determined from the face normal at the position where the slice point was taken from.
      ii. Add to *nonOverlappingArea*:

$$A_{area} - C + B_{area} - C \tag{5}$$

   where
   A_{area} = Area of line A.
   B_{area} = Area of line B.
   C = A_{area} + B_{area} - area of the outline.

6. Return *nonOverlappingArea*

3.6 Self intersection test

The plug-in allows to check for self-intersections in meshes. Intersecting faces are colored red, non-self-intersecting green. *Self intersection test* has been implemented in the following way:

Do the following for each triangular object or for each of the target triangular objects in the scene-graph:

1. For each face-face combination:

(a) Make sure the face is not a bordering face of the other, as this causes false intersections due to rounding errors.

(b) Determine whether the two faces intersect using "A Fast Triangle-Triangle Intersection Test" by Tomas Moller [11].

3.7 Derivative

The plug-in allows to determine how much the face normal differs between neighboring faces (derivative). *Derivative* can be determined in x, y, z or all axes. For each object average dx, dy and dz (average difference of normal between neighboring faces) are given, as well as the respective maximum values. Furthermore, faces are colored to indicate their *derivative*. *Derivative* has been implemented in the following way:

Do the following for each triangular object or for each of the target triangular objects in the scene-graph:

1. Add Vertex Property *acc* to each vertex of object.
2. Set *acc* equal to the corresponding vertex normal, with the range of each element between 0 and 1 where it was between -1 and 1 before.
3. Set the color of each face to equal its normal, with the range of each normal element between 0 and 1 where it was between -1 and 1 before.
4. Perform the following as often as the object is to be differentiated:
   (a) For each face:
      i. Compute average difference of *acc* among edges enclosing face, store in face color property.
   (b) For each vertex:
      i. Determine the average color of faces neighboring vertices, store in *acc*.
5. For each face:
   (a) Set face color to x, y, z or xyz value.

4 Results

4.1 Methods

In order to be able to evaluate the relationship between morph quality and used subdivision scheme, the quality measures linked to morph qualities in Section 2.5 have been used to generate data on the respective qualities. Test meshes have been collected which are used to compute quality measures on. The results of quantitative measures have been made comparable by computing the *average difference* or *mean* of *quality measure* values among slices at various z-axis positions. In this Section used test meshes and metrics will be discussed first. Later, the performance of the considered subdivision schemes will be discussed for each considered morph quality.

4.1.1 Test meshes

Each test mesh has some special characteristic. None of the test meshes contains self-intersections. In Section 10.1 slice curves at different positions on the z-axis of the test meshes are illustrated for both the original meshes, as well as subdivision surfaces. Table 2 shows a preview of each test mesh, the name it is referred to in this document, and a description and motivation for the test mesh.

4.1.2 Metrics

All quantitative quality measures provide an array of real numbers for each *test mesh-subdivision scheme* combination where each element of the array represents the *quality measure* value of the slice at a position on the z-axis. Metrics have been defined to compare these arrays.

- **Average difference (σ)**
  The quality measures *total positive curvature*, *offset non-overlapping area* and *area* have been made comparable by computing the *average difference* of subsequent array elements. Average difference is defined as follows:

$$\sigma = \frac{1}{n-1} \sum_{i=1}^{n-1} |V_{i-1} - V_i| \tag{6}$$

  where

  V = Array of real numbers.
  V_i = Value of element of V with index i (zero-based numbering).
  n = Number of elements of V.

- **Mean ($\bar{x}$)**
  The quality measure *non-overlapping area* has been made comparable by computing the mean of the elements of its array.

4.2 Results

Quality measure data has been computed for each *test mesh-subdivision scheme* combination. The generated data is reported in Section 10 and used to compute metrics on, the quantitative measures are reported in Figure 5.

Preview	Name	Description
	"-" to "+"	A mesh which has no curved features and does have flat and sharp features.
	"o" to "o"	A mesh which connects a source and target mesh which are identical circles. This mesh will be helpful to determine displacement in intermediate curves.
	"U" to "V"	A mesh which connects a curve which has round features at the bottom with a curve with sharp features at the bottom.
	Elephant	A mesh which translates from one component to multiple components. The area enclosed by slice curve at position $z = 0$ and $z = 1$ are 0.
	Fertility	A mesh which translates from one component to multiple components. The area enclosed by slice curve at position $z = 0$ and $z = 1$ are 0.
	Kitten	A mesh which translates from one component to multiple components. The area enclosed by slice curve at position $z = 0$ and $z = 1$ are 0.
	Moai	A mesh which connects a source and target curve. The area enclosed by slice curve at position $z = 0$ and $z = 1$ are aligned with the z-axis.

Table 2: Test meshes

In Figure 5, there are 5 rows (which each correspond to a quantitative *quality measure*) by 5 columns (which each correspond to a *subdivision scheme*). Each cell of this 5 by 5 matrix consists of 3 rows by 3 columns, which corresponds to the *test meshes* from left-top to right bottom: "-" to "+", "o" to "o", "U" to "V", Elephant, Fertility, Kitten, Moai.

The "-" character indicates that no data is available for the element of the matrix. The reason may be that the data is not computable due to self-intersections, that the data is not captivating because its value is equal among all subdivision schemes or because it can not be computed for the original mesh.

For each *test mesh-quality measure-subdivision scheme* combination, the *average difference* or *mean* has been computed of the *quality measure* among different slices of the (subdivided) mesh. For each *test mesh-quality measure* combination there is one value per *subdivision scheme* and depending on the *quality measure* also a number for the original mesh. The element with the lowest number among these 4 or 5 numbers is shaded black, the highest is shaded white and intermediate values are given a shade in between, in accordance with their distance to the highest and lowest value.

Example: the sixth row is one of the rows which corresponds to the quality measure *area*. The first, fourth, seventh, tenth and thirteenth cell of this row correspond to the *test mesh* Moai. Say the *average difference* of *area* for the original Moai mesh is 5, for Loop it is 6, for $\sqrt{3}$ it is 7, for interpolating $\sqrt{3}$ it is 8 and for butterfly it is 9. The first cell will then be shaded black, the fourth dark gray, the seventh gray, the tenth light gray and the thirteenth cell white.

In the self-intersection row, black indicates that the least self-intersections are in the *test mesh* and white indicates that the most self-intersections are in the *test mesh*.

In the *area* row, black indicates that the *average difference* of *area* among positions on the mesh are the least, which may indicate a smoother mesh. White indicates that the *average difference* of *area* among positions on the mesh are the highest, which may indicate a less smooth mesh.

In the *total positive curvature* row, black indicates that the *average difference* of *total positive curvature* among positions on the mesh are the least, which may indicate a smoother mesh. White indicates that the *average difference* of *total positive curvature* among positions on the mesh are the highest, which may indicate a less smooth mesh.

In the *non-overlapping area* row, black indicates that the *mean* of *non-overlapping area* among positions on the mesh are the least, which indicates that the mesh has been changed the least by subdivision. White indicates that the *mean* of *non-overlapping area* among positions on the mesh are the highest, which indicates that the mesh has been changed the the most by subdivision.

In the *offset non-overlapping area* row, black indicates that the *average difference* of *non-overlapping area* among positions on the mesh are the least, which may indicate a smoother mesh. White indicates that the *average difference* of *offset non-overlapping area* among positions on the mesh are the highest, which may indicate a less smooth mesh.

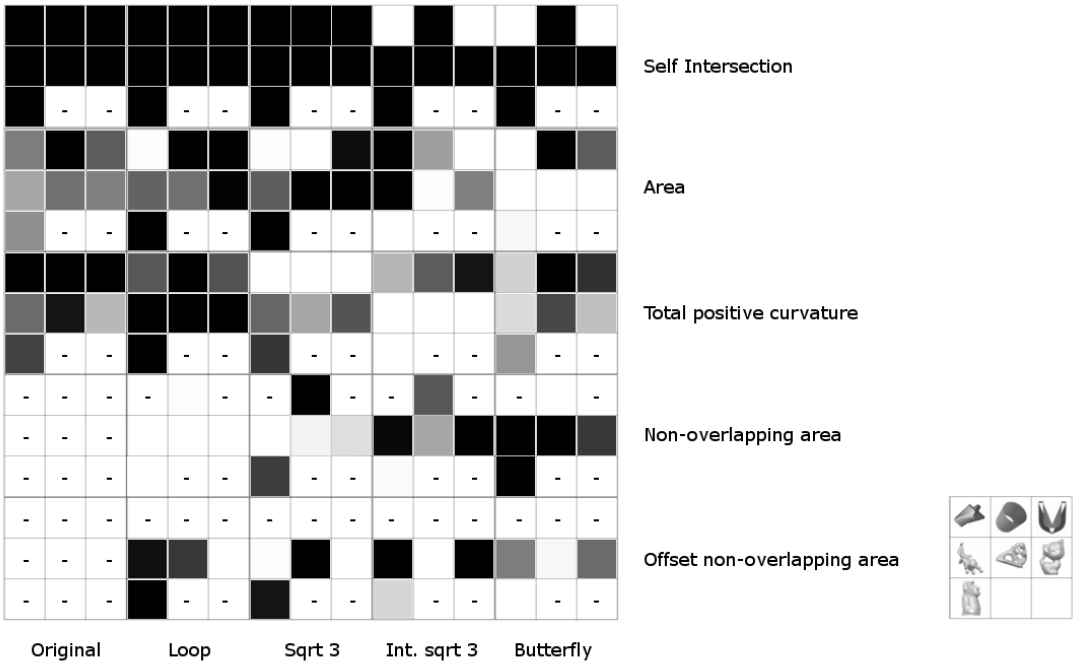


Figure 5: Matrix evaluating relationship between subdivision schemes and quality measures.

4.2.1 Smoothness

A number of quality measures have been chosen in Section 2.5 which measure the morph quality smoothness. The results of each of those quality measures is discussed in this section, followed by a general conclusion about the relation between used subdivision scheme and the morph quality smoothness.

Offset non-overlapping area
The gray values in the *offset non-overlapping area* rows have been averaged per *subdivision scheme*. Example: the average gray value of the *test meshes* which share that they have all been Loop-subdivided and are in the three *offset non-overlapping area* rows was computed, the average gray value of the *test meshes* which share that they where subdivided with Butterfly and are in the three *offset non-overlapping area* rows was computed, etc.

By comparing those averages, it is suggested that Loop creates the smoothest meshes, then interpolating $\sqrt{3}$, followed by $\sqrt{3}$ and Butterfly creates the least smooth meshes. It should be noted that the difference between interpolating $\sqrt{3}$ and $\sqrt{3}$'s performance is relatively small.

As could be expected, the difference between the performance of subdivision schemes is more significant when the original mesh is more sharply curved. In the plot, the difference between the subdivision schemes is the greatest between $z = 0.52$ and $z = 0.6$ which is a sharply curved part of the Moai mesh. Between $z = 0.6$ and $z = 0.68$ the original mesh has less sharply shaped features and the plot reflects this.

Please consult Section 10.5 to view the *offset non-overlapping area* plots for each *test mesh-subdivision scheme* combination.

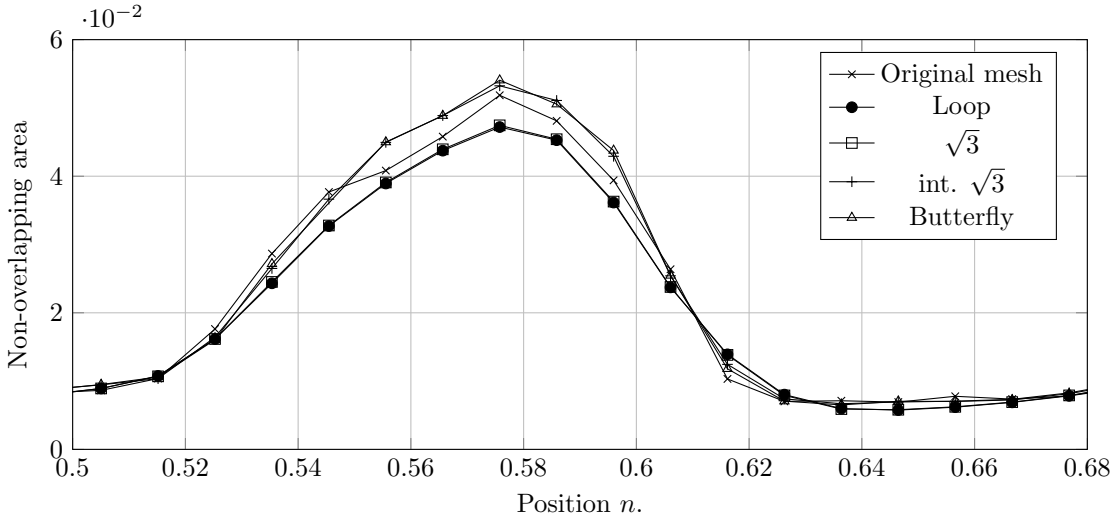


Figure 6: **Moai with Loop, $\sqrt{3}$, int. $\sqrt{3}$ and Butterfly.** *non-overlapping area* of slice at position n and slice at position $n-1$.

Area

The gray values in the *area* rows have been averaged per *subdivision scheme*. By comparing those averages, it is suggested that Loop creates the smoothest meshes, then $\sqrt{3}$, followed by interpolating $\sqrt{3}$ and Butterfly creates the least smooth meshes. It should be noted that the difference between average gray value of Loop and the average gray value of $\sqrt{3}$ differ relatively little. If the first three *test meshes* are not included in the average, $\sqrt{3}$ performs better than Loop. Please consult Section 10.2 to view the *area* plots for each *test mesh- subdivision scheme* combination.

Total positive curvature

The gray values in the *total positive curvature* rows have been averaged per *subdivision scheme*,by only considering the *test meshes* Elephant, Fertility, Kitten and Moai. The other three *test meshes* were not included as they distort the results due to self-intersections or small variation among subdivision schemes. By comparing the averages it is suggested that Loop creates the smoothest meshes, then $\sqrt{3}$, followed by Butterfly and interpolating $\sqrt{3}$ creates the least smooth meshes. Figure 7 confirms this observation. Loop and $\sqrt{3}$'s curves are most smooth, where interpolating $\sqrt{3}$ and Butterfly's curves are less smooth. Please consult Section 10.3 to view the *total positive curvature* plots for each *test mesh-subdivision scheme* combination.

Reflection lines

On average among the *test meshes*, it has become apparent by evaluating reflection lines that Loop creates the smoothest meshes, then $\sqrt{3}$, followed by interpolating $\sqrt{3}$ and Butterfly creates the least smooth meshes. Table 3 confirms this observation. Loop and Sqrt 3's reflection lines are most smooth, where interpolating $\sqrt{3}$ and Butterfly's reflection lines are less smooth. Please consult Section 10.6 to view all generated reflection line images.

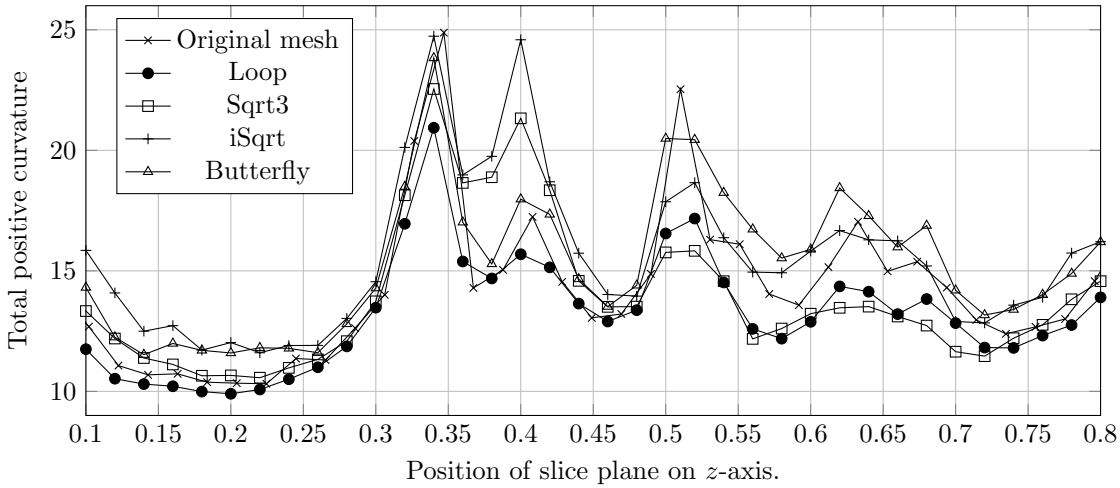


Figure 7: **Kitten.** Total positive curvature.

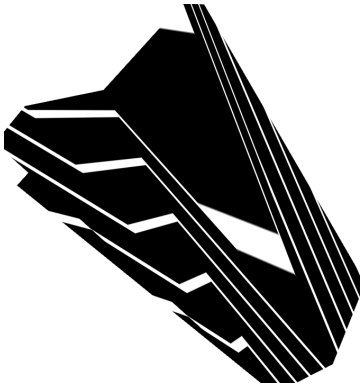
Gaussian curvature
On average among the *test meshes*, it has become apparent by evaluating *Gaussian curvature* that the approximating subdivision schemes Loop and $\sqrt{3}$ create smoother meshes than the interpolating schemes interpolating $\sqrt{3}$ and Butterfly. This was concluded based on the absence of small irregularities in approximating schemes where they are present in the interpolating schemes. These irregularities are visualized as small dots and lines in the interpolating $\sqrt{3}$ subdivision surface and Butterfly subdivision surface Figures in Section 10.7.

Mean curvature
On average among the *test meshes*, it has become apparent by evaluating *mean curvature* that the approximating subdivision schemes Loop and $\sqrt{3}$ create smoother meshes than the interpolating schemes interpolating $\sqrt{3}$ and Butterfly. This was concluded based on the absence of small irregularities in approximating schemes where they are present in the interpolating schemes. These irregularities are visualized as small dots and lines in the interpolating $\sqrt{3}$ subdivision surface and Butterfly subdivision surface Figures in Section 10.8.

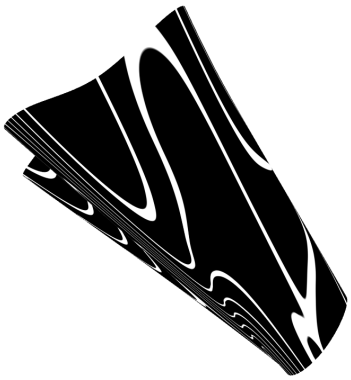
General conclusion
By averaging the results of the considered quality measures, it can be concluded that Loop creates the smoothest meshes, then $\sqrt{3}$, followed by interpolating $\sqrt{3}$ and Butterfly creates the least smooth meshes.

4.2.2 Sharpness preservation

A number of quality measures have been chosen in Section 2.5 which measure the quality sharpness preservation. The results of each of those quality measures are discussed in this section, followed by a general conclusion about the relation between used subdivision scheme and sharpness preservation.

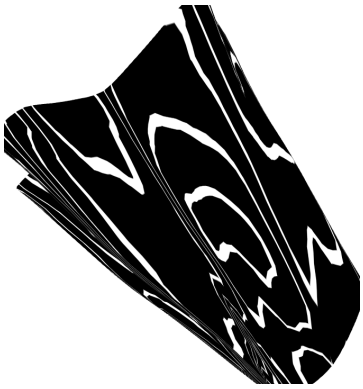


Original



Loop subdivision surface

$\sqrt{3}$ subdivision surface



Int. $\sqrt{3}$ subdivision surface

Butterfly subdivision surface

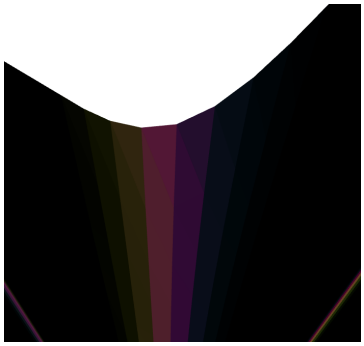
Table 3: **"-" to "+".** Reflection lines

Reflection lines
Table 3 suggests that interpolating $\sqrt{3}$ and Butterfly preserve sharpness better than Loop and $\sqrt{3}$. It is however hard to judge based on this *quality measure* whether or not this is true, therefore little weight will be given to this *quality measure* while evaluating the morph quality sharpness preservation. Please consult Section 10.6 to view all generated *reflection lines* images.

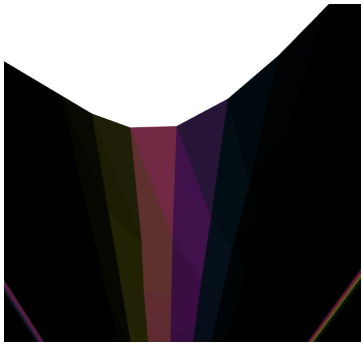
Derivative
In Table 4, the difference between neighboring face normals is computed once. Brighter colors indicate a bigger difference in face normal between neighboring faces and hence sharp features. On average among all test cases, it can be concluded that Loop preserves sharpness the least, then $\sqrt{3}$, followed by interpolating $\sqrt{3}$ and Butterfly preserves sharpness best. Table 4 illustrates this finding.

General conclusion
By averaging the results of the considered quality measures, it can be concluded that Loop preserves sharpness the least, then $\sqrt{3}$, followed by interpolating $\sqrt{3}$ and Butterfly preserves sharpness best.

Original mesh

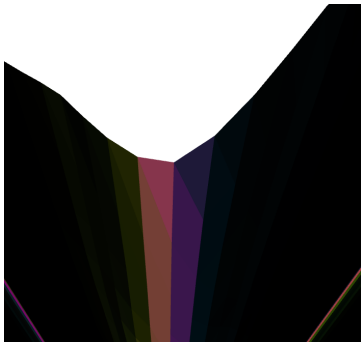


Loop subdivision surface



$\sqrt{3}$ subdivision surface

Int. $\sqrt{3}$ subdivision surface



Butterfly subdivision surface

Table 4: **"-" to "+".** Derivative

4.2.3 Displacement

The gray values in the *non-overlapping area* rows have been averaged per *subdivision scheme*. By comparing those averages, it is suggested that Loop-subdivision alters meshes the most, followed by $\sqrt{3}$, then interpolating $\sqrt{3}$ and butterfly subdivision alters meshes the least. From this it can be concluded that approximating schemes alter meshes more than interpolating schemes. Figure 8 confirms this finding: the curves of the approximating scenes are generally above those of the interpolating schemes. Interestingly, the results have shown that the difference between the displacement of approximating and interpolating schemes is more significant where the original mesh is more curved. This finding is confirmed by the graph: between position $z = 0.3$ and $z = 0.6$ the model elephant has his ears, which are sharply curved. The graph indeed shows the biggest difference between approximating and interpolating schemes in that position range. Other parts of the Elephant model are less sharply curved, which is reflected in the graph.

Please consult Section 10.4 to view the *non-overlapping area* plots for each *test mesh-subdivision scheme* combination.

4.2.4 Simpleness preservation

As can be observed from Figure 5, Loop-subdivision and $\sqrt{3}$-subdivision have not introduced self-intersections in *test meshes*. Interpolating $\sqrt{3}$-subdivision and butterfly subdivision have introduced self-intersections in *test meshes* where there were sharp features in the original meshes.

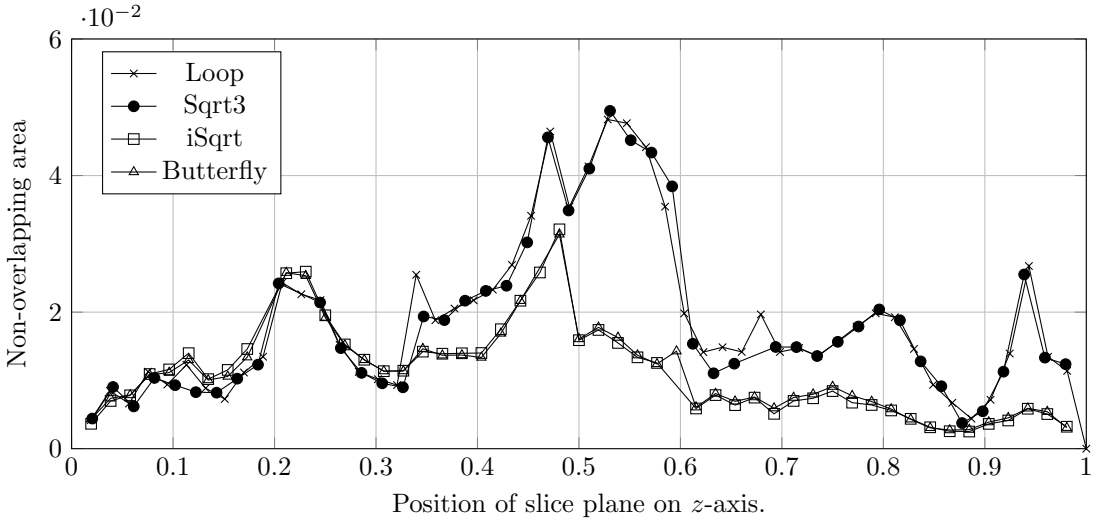


Figure 8: **Elephant with Loop, sqrt, isqrt and Butterfly.** Non-overlapping area slice.

4.2.5 Closedness preservation

No holes have been introduced in any of the meshes by any of the subdivision schemes, as was expected.

5 Conclusions

The goal of this bachelor's project is to research the relationship between used subdivision schemes and morph quality in the adopted approach to morphing. The presented implementation allows to subdivide and slice triangular meshes, as well as to generate data about how subdivision has affected morph quality. Using results generated by the developed implementation, the relationship between used subdivision schemes and morph quality is evaluated based on various quality measures. In this chapter, the research question is answered and the helpfulness of quality measures in answering the research question is assessed.

5.1 Subdivision schemes

Based on the results presented in Section 4, the considered subdivision schemes are rated based on qualities in Table 5. Table 5 was generated by considering the general conclusion about each morph quality, given in Section 4. In the general conclusion section of each morph quality in Section 4, it is explained how the ranking of subdivision schemes for the corresponding quality was determined. For each quality, the subdivision schemes are ranked from best to worst (++, +, -, - -) performing subdivision scheme.

	Loop	$\sqrt{3}$	**Interpolating** $\sqrt{3}$	**Modified Butterfly**
Smoothness	++	+	-	- -
Sharpness preservation	- -	-	+	++
Displacement	- -	-	+	++
Simpleness preservation	++	++	- -	- -
Closedness preservation	++	++	++	++

Table 5: Subdivision schemes rated based on qualities.

5.2 Quality measures

In this section, the helpfulness of quality measures in answering the research question is evaluated.

Offset non-overlapping area has proven to be a good measure of mesh smoothness. The quality measure *area* has the disadvantage that it is insensitive to the equal area shape changes and transformations of whole shapes in the $x - y$ plane between subsequent slices. *Offset non-overlapping area* does not have these disadvantages. An advantage of the quality measure *area* is that it is less computationally expensive than *offset non-overlapping area*.

As *total positive curvature* has given a quite unanimous indication of mesh smoothness resulting from subdivision schemes among test meshes, it seems fair to suggest that *total positive curvature* is a good measure of mesh smoothness.

Non-overlapping area is a very good indication of modification of original meshes by subdivision schemes as it has given quite unanimous results per subdivision scheme among test meshes.

Self-intersection tests has proven to be a good indicator of self-intersection. Thanks to this measure, it has become clear that interpolating subdivision schemes cause self-intersections in the presence of sharp features where this is not the case for approximating subdivision schemes. A disadvantage of this quality measure is that it is quite computationally expensive.

Mesh hole detection has not indicated any difference among subdivision schemes as none have introduced holes in test meshes.

Reflection lines, *Gaussian curvature* and *mean curvature* have shown to be unnecessary measures to assess the morph quality smoothness in the presence of the other metrics as they are not quantitative and therefore difficult to compare. *Reflection lines* have shown to be not fit to assess sharpness preservation as it was hard to find patterns in the results. *Derivative* has given a good indication of mesh sharpness but its results are hard to compare.

6 Future Work

There are various ways to improve the performance of the developed plug-in and to enhance the reliability of the drawn conclusions:

- **Enlarge range of test meshes** The results of this project were generated using a number of test meshes. It would be best if future study would include a larger range of test meshes to confirm that the conclusions drawn in this thesis are not only valid for the used test meshes but also in general.
- **Selective subdivision** The quality "sharpness preservation" could be enhanced by detecting sharp features or allowing to select sharp features and not subdividing or displacing the mesh less in those areas using modified subdivision rules.
- **Acceleration** To be able to better analyze the smoothness of meshes in a certain direction, the path a point travels from source to target curve could be determined. The acceleration of this point can then be determined as the position along the morph axis is changed from the z coordinate of the source curve to the z coordinate of the target curve.
- **Improve non-overlapping area computation** Allow to determine the *non-overlapping area* of two slices (A and B) which consist of a different number of components or where one component of slice A overlaps multiple components of slice B.
- **Preserve area** Allow to linearly interpolate between the area of a source and target curve, where the source, intermediate or target slice consists of multiple components.
- **Non-overlapping area tolerance** Obviate the need for a tolerance number to combat floating point number rounding errors.
- **Improve efficiency self-intersection test** Find a method to determine self-intersection of meshes which is more efficient than time complexity $O(n^2)$, where n is the number of faces in the mesh.

7 References

[1] Mario Botsch, Stefan Steinberg, Stefan Bischoff, and Leif Kobbelt. OpenMesh: A Generic and Efficient Polygon Mesh Data Structure. In *OpenSG Symposium 2002*, 2002.

[2] Bart Braden. The surveyor's *area* formula. *The College Mathematics Journal*, 17(4):326–337, 1986.

[3] Edwin Catmull and James Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-aided design*, 10(6):350–355, 1978.

[4] Daniel Cohen-Or, Amira Solomovic, and David Levin. Three-dimensional distance field metamorphosis. *ACM Transactions on Graphics (TOG)*, 17(2):116–141, 1998.

[5] Nira Dyn, David Levine, and John A Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM transactions on Graphics (TOG)*, 9(2):160–169, 1990.

[6] Leif Kobbelt. sqrt 3-subdivision. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 103–112. ACM Press/Addison-Wesley Publishing Co., 2000.

[7] Ulf Labsik and Günther Greiner. Interpolatory sqrt 3-subdivision. In *Computer Graphics Forum*, volume 19, pages 131–138. Wiley Online Library, 2000.

[8] Charles Loop. Smooth subdivision surfaces based on triangles. In *Department of Mathematics, The University of Utah, Masters Thesis*, 1987.

[9] Jan Möbius and Leif Kobbelt. Openflipper: An open source geometry processing and rendering framework. In *Curves and Surfaces*, pages 488–500. Springer, 2010.

[10] Jan Mobius, Michael Kremer, and Leif Kobbelt. Openflipper-a highly modular framework for processing and visualization of complex geometric models. In *Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), 2013 6th Workshop on*, pages 25–32. IEEE, 2013.

[11] Tomas Möller. A fast triangle-triangle intersection test. *Journal of graphics tools*, 2(2):25–30, 1997.

[12] Paola Novara, Lucia Romani, and Jungho Yoon. Improving smoothness and accuracy of modified butterfly subdivision scheme. *Applied Mathematics and Computation*, 272:64–79, 2016.

[13] Ryutarou Ohbuchi, Yoshiyuki Kokojima, and Shigeo Takahashi. Blending shapes by using subdivision surfaces. *Computers & Graphics*, 25(1):41–58, 2001.

[14] Per-Olof Persson, Michael J Aftosmis, and Robert Haimes. On the use of loop subdivision surfaces for surrogate geometry. *Proceedings of the 15th International Meshing Roundtable, Birmingham, AL, Oct*, 31, 2006.

[15] Jos Stam and Charles Loop. Quad/triangle subdivision. In *Computer Graphics Forum*, volume 22, pages 79–85. Wiley Online Library, 2003.

[16] Elif Tosun, Yotam I Gingold, Jason Reisman, and Denis Zorin. Shape optimization using reflection lines. In *Symposium on Geometry Processing*, pages 193–202. Citeseer, 2007.

[17] Denis Zorin, Peter Schröder, and Wim Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 189–192. ACM, 1996.

8 List of Tables

1 Morph quality measures . . . 10
2 Test meshes . . . 19
3 **"-" to "+".** Reflection lines . . . 24
4 **"-" to "+".** Derivative . . . 25
5 Subdivision schemes rated based on qualities. . . . 27
6 **"-" to "+".** Original curves . . . 35
7 **"o" to "o".** Original curves . . . 35
8 **"j" to "j".** Original curves . . . 35
9 **"O" to "box".** Original curves . . . 36
10 **"B" to "R".** Original curves . . . 36
11 **"U" to "V".** Original curves . . . 36
12 **Elephant.** Original curves . . . 37
13 **Fertility.** Original curves . . . 37
14 **Kitten.** Original curves . . . 38
15 **Moai.** Original curves . . . 39
16 **"-" to "+" with Loop.** Curves . . . 39
17 **"o" to "o" with Loop.** Curves . . . 40
18 **"U" to "V" with Loop.** Curves . . . 40
19 **Elephant with Loop.** Curves . . . 40
20 **Fertility with Loop.** Curves . . . 41
21 **Kitten with Loop.** Curves . . . 41
22 **Moai with Loop.** Curves . . . 42
23 **"-" to "+" with $\sqrt{3}$.** Curves . . . 42
24 **"o" to "o" with $\sqrt{3}$.** Curves . . . 43
25 **"U" to "V" with $\sqrt{3}$.** . . . 43
26 **Elephant with $\sqrt{3}$.** . . . 43
27 **Fertility with $\sqrt{3}$.** Curves . . . 44
28 **Kitten with $\sqrt{3}$.** Curves . . . 44
29 **Moai with $\sqrt{3}$.** Curves . . . 45
30 **"-" to "+" with int. $\sqrt{3}$.** Curves . . . 45
31 **"o" to "o" with int. $\sqrt{3}$.** Curves . . . 46
32 **"U" to "V" with int. $\sqrt{3}$.** Curves . . . 46
33 **Elephant with int. $\sqrt{3}$.** Curves . . . 46
34 **Fertility with int. $\sqrt{3}$.** Curves . . . 47
35 **Kitten with int. $\sqrt{3}$.** Curves . . . 47
36 **Moai with int. $\sqrt{3}$.** Curves . . . 48
37 **"-" to "+" with Butterfly.** Curves . . . 48

38 **“o” to “o” with Butterfly.** Curves . . . 49
39 **“U” to “V” with Butterfly.** Curves . . . 49
40 **Elephant with Butterfly.** Curves . . . 50
41 **Fertility with Butterfly.** Curves . . . 50
42 **Kitten with Butterfly.** Curves . . . 51
43 **Moai with Butterfly.** Curves . . . 52
44 **“-” to “+”.** Reflection lines . . . 85
45 **Fertility.** Reflection lines . . . 86
46 **Kitten.** Gaussian curvature . . . 87
47 **Kitten.** Mean curvature . . . 88

9 List of Figures

1 Slice at position $z = 0.34$ before and after Loop subdivision. . . . 4
2 Topological relations stored by OpenMesh [1]. . . . 7
3 Flipping of edges in $\sqrt{3}$ subdivision [6]. . . . 9
4 Indicated in gray is the *non-overlapping area* of two squares. . . . 10
5 Matrix evaluating relationship between subdivision schemes and quality measures. 21
6 **Moai with Loop, $\sqrt{3}$, int. $\sqrt{3}$ and Butterfly.** *non-overlapping area* of slice at position n and slice at position $n-1$. . . . 22
7 **Kitten.** Total positive curvature. . . . 23
8 **Elephant with Loop, sqrt, isqrt and Butterfly.** Non-overlapping area slice. 26
9 **“-” to “+” with Loop.** Area enclosed by slice. $\sigma = 0.001169530612$ . . . 52
10 **“-” to “+” with $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.001167679592$ . . . 53
11 **“-” to “+” with int. $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.0008011020408$ . . . 53
12 **“-” to “+” with Butterfly.** Area enclosed by slice. Not defined everywhere due to self-intersection. $\sigma = 0.00117154375$ . . . 53
13 **“o” to “o” with Loop.** Area enclosed by slice. $\sigma = 0$ . . . 54
14 **“o” to “o” with $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.0002465306122$ . . . 54
15 **“o” to “o” with int. $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.000153877551$ . . . 54
16 **“o” to “o” with Butterfly.** Area enclosed by slice. $\sigma = 0$ . . . 55
17 **“U” to “V” with Loop.** Area enclosed by slice. $\sigma = 0.0005839795918$ . . . 55
18 **“U” to “V” with $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.0006089183673$ . . . 55
19 **“U” to “V” with int. $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.001057836735$ . . . 56
20 **“U” to “V” with Butterfly.** Area enclosed by slice. $\sigma = 0.0007563877551$ . . . 56
21 **Elephant with Loop.** Area enclosed by slice. $\sigma = 0.05297383265$ . . . 56
22 **Elephant with $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.05295422857$ . . . 57
23 **Elephant with int. $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.052450352$ . . . 57
24 **Elephant with Butterfly.** Area enclosed by slice. $\sigma = 0.0537998$ . . . 57
25 **Fertility with Loop.** Area enclosed by slice. $\sigma = 0.1184813878$ . . . 58
26 **Fertility with $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.1150389796$ . . . 58
27 **Fertility with int. $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.1227161224$ . . . 58
28 **Fertility with Butterfly.** Area enclosed by slice. $\sigma = 0.1227924898$ . . . 59
29 **Kitten with Loop.** Area enclosed by slice. $\sigma = 0.1765239592$ . . . 59
30 **Kitten with $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.1764843265$ . . . 59
31 **Kitten with int. $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.178036898$ . . . 60

32 **Kitten with Butterfly.** Area enclosed by slice. $\sigma = 0.1795662041$ 60
33 **Moai with Loop.** Area enclosed by slice. $\sigma = 0.0147615102$ 60
34 **Moai with $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.01476202041$ 61
35 **Moai with int. $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.01579673469$ 61
36 **Moai with Butterfly.** Area enclosed by slice. $\sigma = 0.01576493878$ 61
37 **"-" to "+" with Loop.** Total positive curvature. $\sigma = 0.1744081633$ 62
38 **"-" to "+" with $\sqrt{3}$.** Total positive curvature. $\sigma = 0.5127541667$ 62
39 **"-" to "+" with int. $\sqrt{3}$.** Total positive curvature. $\sigma = 0.3625673469$ 62
40 **"-" to "+" with Butterfly.** Total positive curvature. $\sigma = 0.4212145833$ . . . 63
41 **"o" to "o" with Loop.** Total positive curvature. $\sigma = 0$ 63
42 **"o" to "o" with $\sqrt{3}$.** Total positive curvature. $\sigma = 0.3183489796$ 63
43 **"o" to "o" with int. $\sqrt{3}$.** Total positive curvature. $\sigma = 0.1142971429$ 64
44 **"o" to "o" with Butterfly.** Total positive curvature. $\sigma = 0$ 64
45 **"U" to "V" with Loop.** Total positive curvature. $\sigma = 7.053002041$ 64
46 **"U" to "V" with $\sqrt{3}$.** Total positive curvature. $\sigma = 21.76136531$ 65
47 **"U" to "V" with int. $\sqrt{3}$.** Total positive curvature. $\sigma = 1.673816327$ 65
48 **"U" to "V" with Butterfly.** Total positive curvature. $\sigma = 4.104079167$ . . . 65
49 **Elephant with Loop.** Total positive curvature. $\sigma = 3.3302168$ 66
50 **Elephant with $\sqrt{3}$.** Total positive curvature. $\sigma = 3.8786128$ 66
51 **Elephant with int. $\sqrt{3}$.** Total positive curvature. $\sigma = 4.7024868$ 66
52 **Elephant with Butterfly.** Total positive curvature. $\sigma = 4.5139824$ 67
53 **Fertility with Loop.** Total positive curvature. $\sigma = 2.7585062$ 67
54 **Fertility with $\sqrt{3}$.** Total positive curvature. $\sigma = 3.4516888$ 67
55 **Fertility with int. $\sqrt{3}$.** Total positive curvature. $\sigma = 3.81776$ 68
56 **Fertility with Butterfly.** Total positive curvature. $\sigma = 3.0568796$ 68
57 **Kitten with Loop.** Total positive curvature. $\sigma = 1.5852304$ 68
58 **Kitten with $\sqrt{3}$.** Total positive curvature. $\sigma = 1.858044$ 69
59 **Kitten with int. $\sqrt{3}$.** Total positive curvature. $\sigma = 2.4117896$ 69
60 **Kitten with Butterfly.** Total positive curvature. $\sigma = 2.20154$ 69
61 **Moai with Loop.** Total positive curvature. $\sigma = 0.5406688$ 70
62 **Moai with $\sqrt{3}$.** Total positive curvature. $\sigma = 0.7991141667$ 70
63 **Moai with int. $\sqrt{3}$.** Total positive curvature. $\sigma = 1.7513812$ 70
64 **Moai with Butterfly.** Total positive curvature. $\sigma = 1.2544252$ 71
65 **"-" to "+" with Loop.** Non-overlapping area slice. $\bar{x} = 0.01362278279$ 71
66 **"-" to "+" with $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.01109334047$ 71
67 **"-" to "+" with int. $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = n.a.$ 72
68 **"-" to "+" with Butterfly.** Non-overlapping area slice. $\bar{x} = n.a.$ 72
69 **"o" to "o" with Loop.** Non-overlapping area slice. $\bar{x} = 0.0173711$ 72
70 **"o" to "o" with $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.01280360361$ 72
71 **"o" to "o" with int. $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.01434957778$ . . . 73
72 **"o" to "o" with Butterfly.** Non-overlapping area slice. $\bar{x} = 0.017413$ 73
73 **"U" to "V" with Loop.** Non-overlapping area slice. $\bar{x} = 0.03745941087$ . . . 73
74 **"U" to "V" with $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.02758521304$ 74
75 **"U" to "V" with int. $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = n.a.$ 74
76 **"U" to "V" with Butterfly.** Non-overlapping area slice. $\bar{x} = n.a.$ 74
77 **Elephant with Loop.** Non-overlapping area slice. $\bar{x} = 0.01854460736$ 74
78 **Elephant with $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.01852367283$ 75
79 **Elephant with int. $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.01123209388$ 75
80 **Elephant with Butterfly.** Non-overlapping area slice. $\bar{x} = 0.01104177102$ . . 75

81 **Fertility with Loop.** Non-overlapping area slice. $\bar{x} = 0.06167092128$ 76
82 **Fertility with $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.05901141489$ 76
83 **Fertility with int. $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.042207975$ 76
84 **Fertility with Butterfly.** Non-overlapping area slice. $\bar{x} = 0.00587505102$ . . . 77
85 **Kitten with Loop.** Non-overlapping area slice. $\bar{x} = 0.0218482086$ 77
86 **Kitten with $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.0208735474$ 77
87 **Kitten with int. $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.01407313688$ 78
88 **Kitten with Butterfly.** Non-overlapping area slice. $\bar{x} = 0.01575873208$ 78
89 **Moai with Loop.** Non-overlapping area slice. $\bar{x} = 0.0072563662$ 78
90 **Moai with $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.006650298776$ 79
91 **Moai with int. $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.007241229286$ 79
92 **Moai with Butterfly.** Non-overlapping area slice. $\bar{x} = 0.006456916415$ 79
93 **Elephant with Loop.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.007322120921$ 80
94 **Elephant with $\sqrt{3}$.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.007560879211$ 80
95 **Elephant with int. $\sqrt{3}$.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.007306004231$ 80
96 **Elephant with Butterfly.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.00743124039$ 81
97 **Fertility with Loop.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.009798018391$ 81
98 **Fertility with $\sqrt{3}$.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.009311483529$ 81
99 **Fertility with int. $\sqrt{3}$.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.01150236706$ 82
100 **Fertility with Butterfly.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.01143622771$ 82
101 **Kitten with Loop.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.01508217079$ 82
102 **Kitten with $\sqrt{3}$.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.0150788382$ 83
103 **Kitten with int. $\sqrt{3}$.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.01451244556$ 83
104 **Kitten with Butterfly.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.01475344045$ 83
105 **Moai with Loop.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.00161646202$ 84
106 **Moai with $\sqrt{3}$.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.00164473596$ 84
107 **Moai with int. $\sqrt{3}$.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.001910749184$ 84
108 **Moai with Butterfly.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.001967704286$ 85

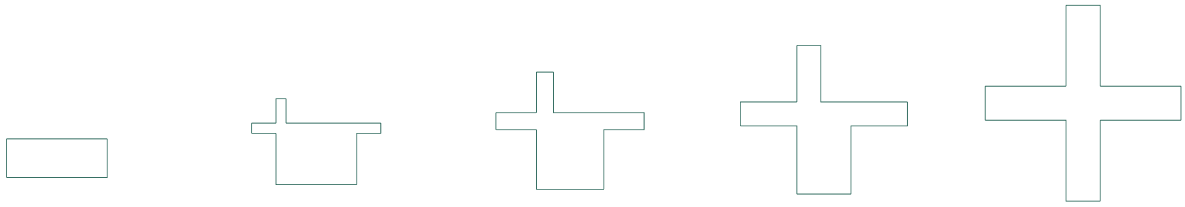
10 Appendix

10.1 Slice curves

Slice curves of *test meshes* are illustrated in this section. For each slice, the position of the slice plane on the z-axis is indicated.

10.1.1 Original

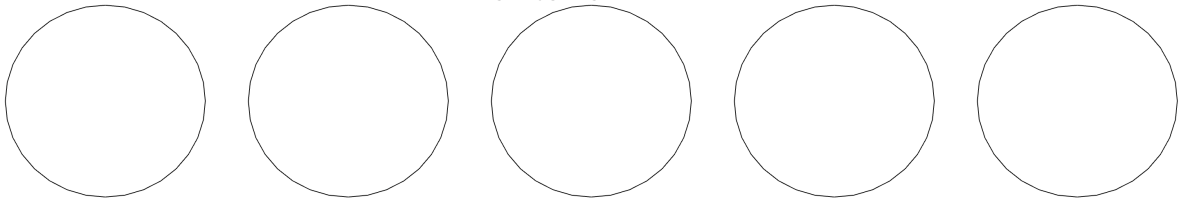
"-" to "+"



$z = 0$ $z = 0.25$ $z = 0.5$ $z = 0.75$ $z = 1$

Table 6: **"-" to "+".** Original curves

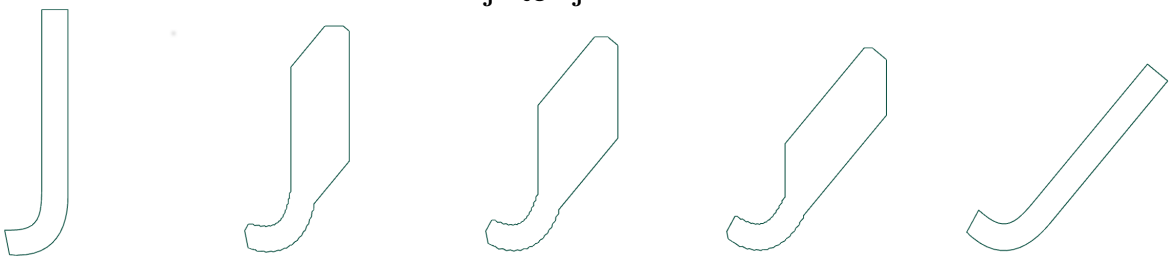
"o" to "o"



$z = 0$ $z = 0.25$ $z = 0.5$ $z = 0.75$ $z = 1$

Table 7: **"o" to "o".** Original curves

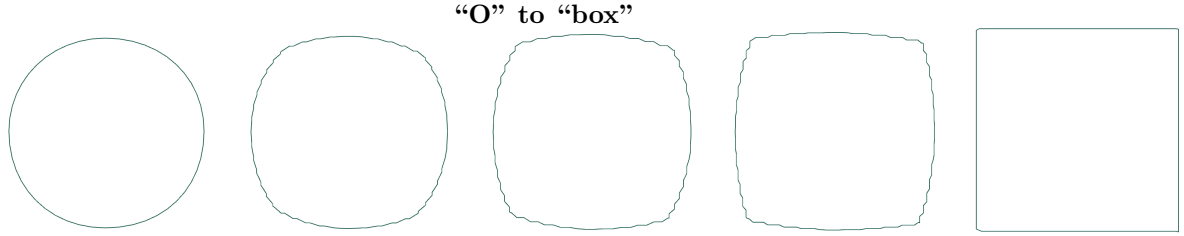
"j" to "j"



$z = 0$ $z = 0.25$ $z = 0.5$ $z = 0.75$ $z = 1$

Table 8: **"j" to "j".** Original curves

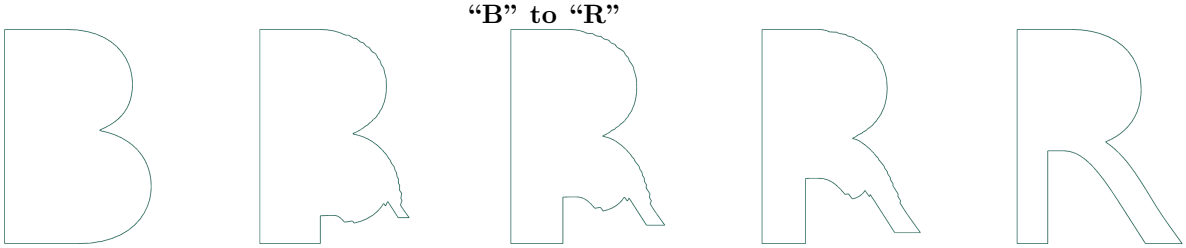
"O" to "box"



$z = 0$ $z = 0.25$ $z = 0.5$ $z = 0.75$ $z = 1$

Table 9: **"O" to "box".** Original curves

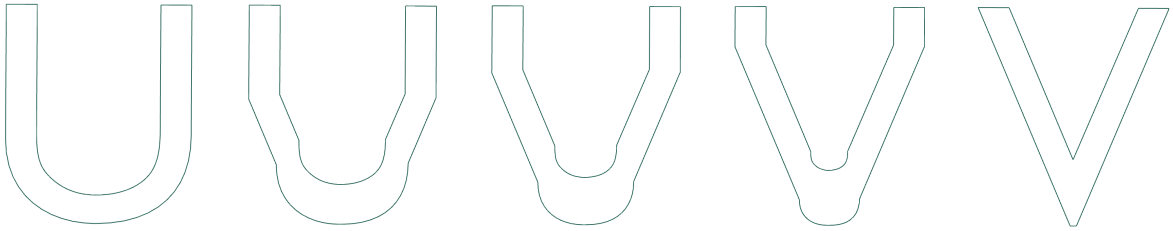
"B" to "R"



$z = 0$ $z = 0.25$ $z = 0.5$ $z = 0.75$ $z = 1$

Table 10: **"B" to "R".** Original curves

"U" to "V"



$z = 0$ $z = 0.25$ $z = 0.5$ $z = 0.75$ $z = 1$

Table 11: **"U" to "V".** Original curves

Elephant

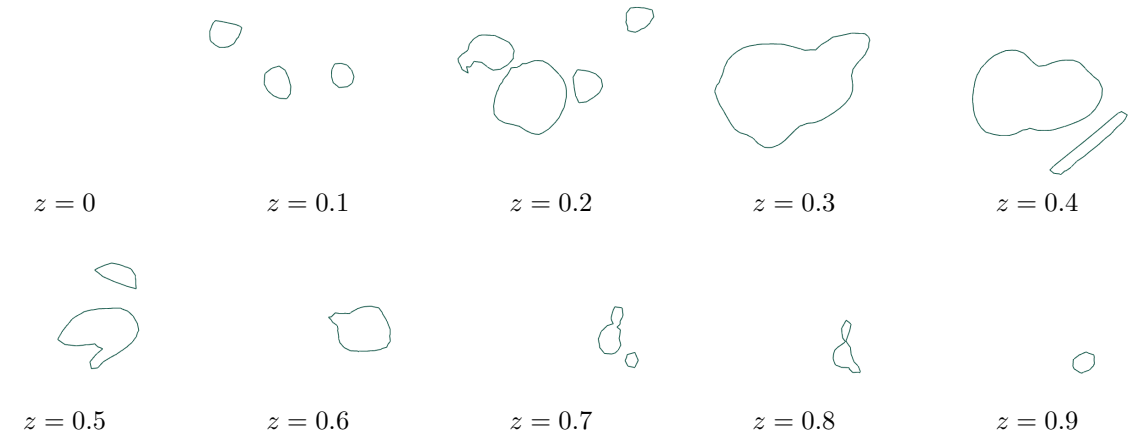


Table 12: **Elephant.** Original curves

Fertility

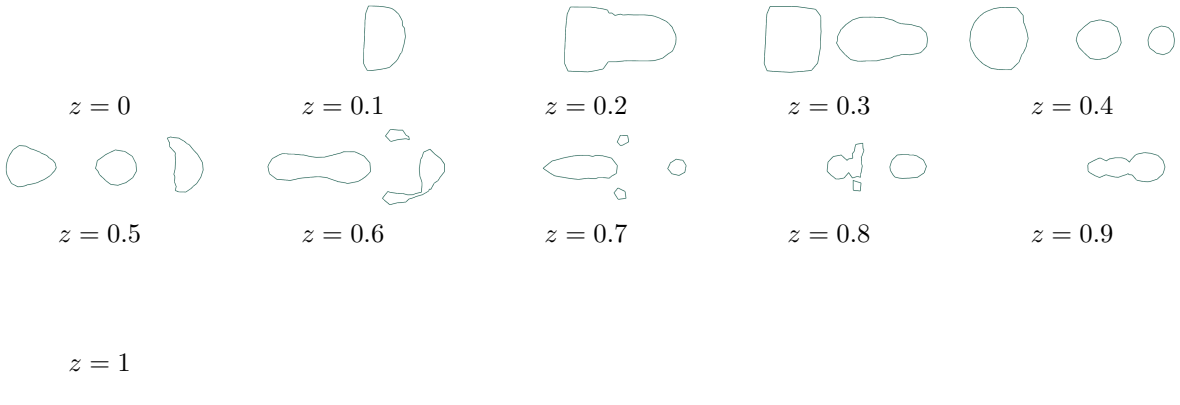


Table 13: **Fertility.** Original curves

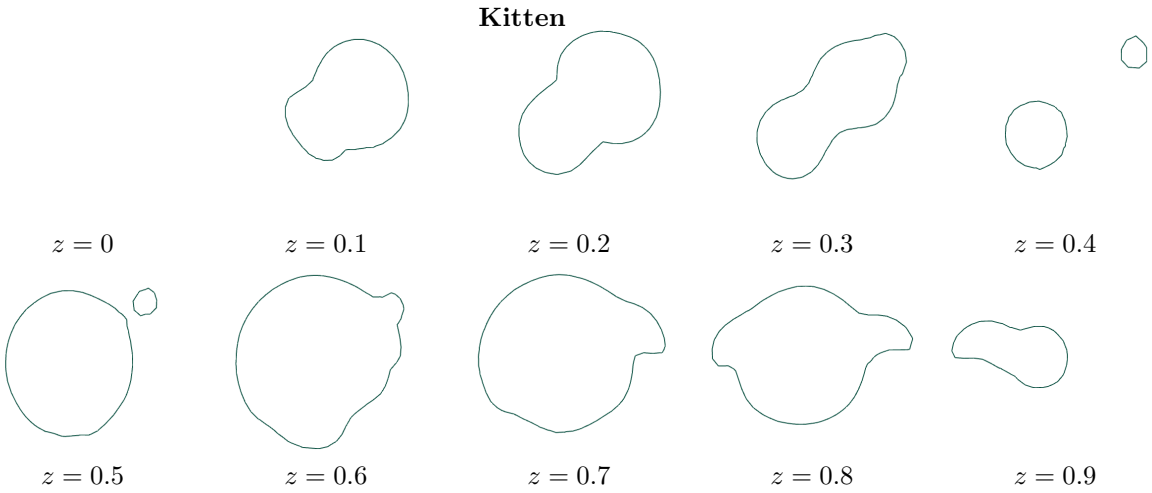


Table 14: **Kitten.** Original curves

Moai

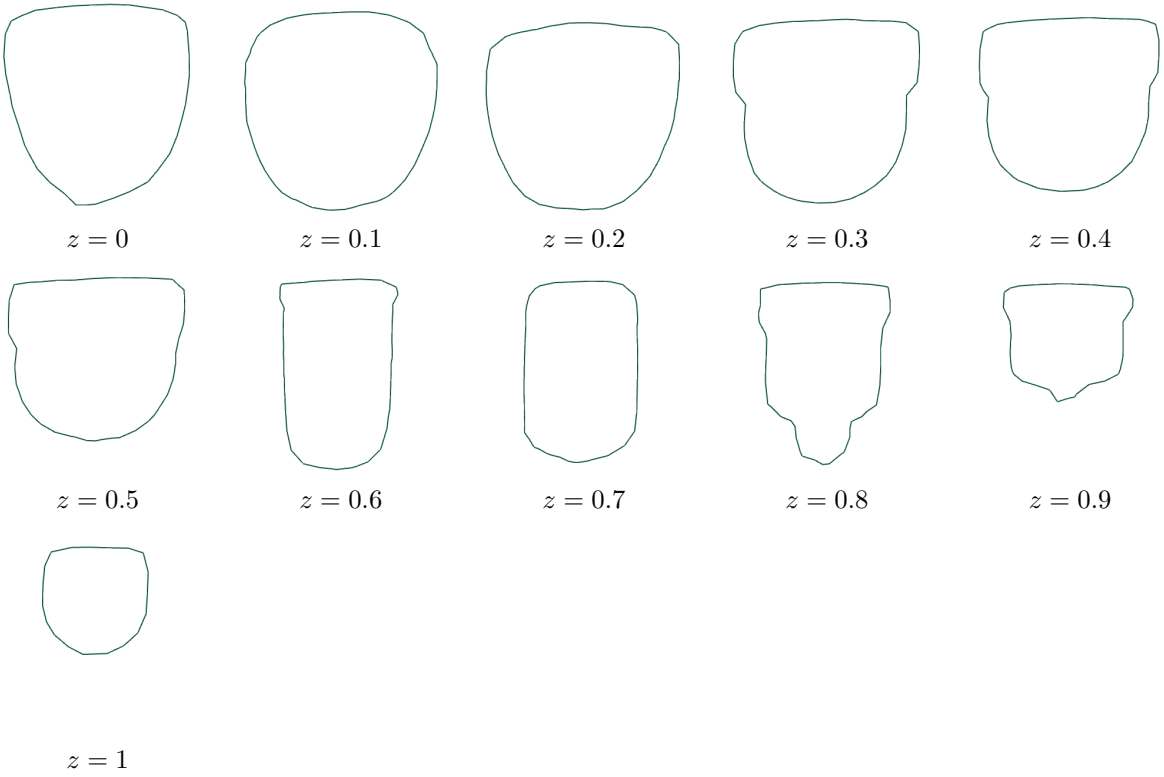


Table 15: **Moai.** Original curves

10.1.2 Loop Subdivision

"-" to "+"

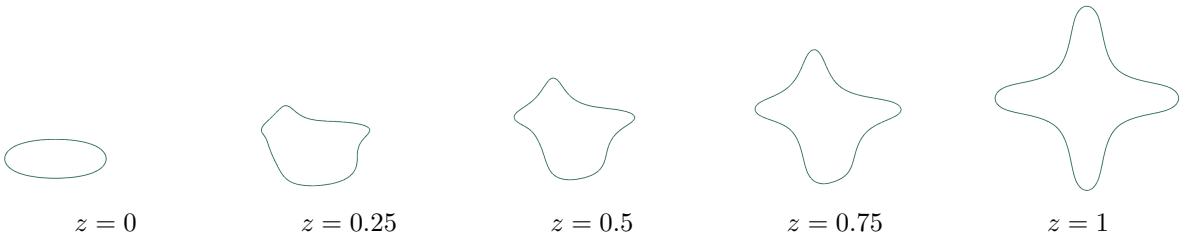
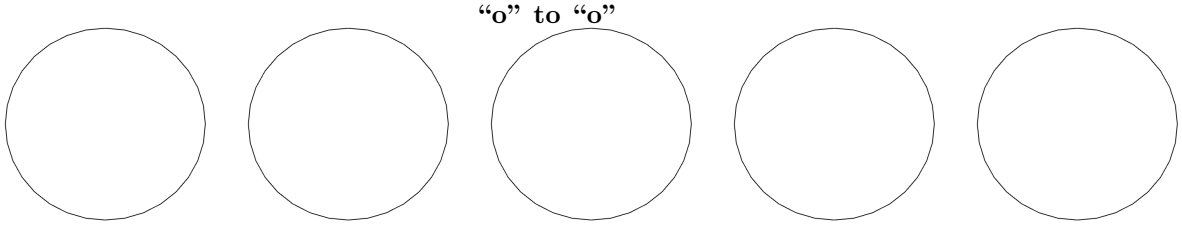


Table 16: **"-" to "+" with Loop.** Curves

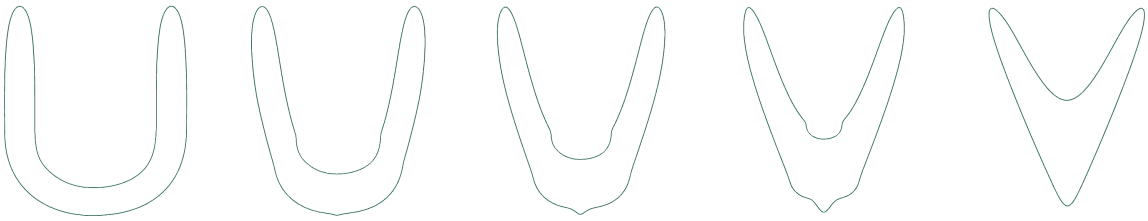
“o” to “o”



$z = 0$ $z = 0.25$ $z = 0.5$ $z = 0.75$ $z = 1$

Table 17: **“o” to “o” with Loop.** Curves

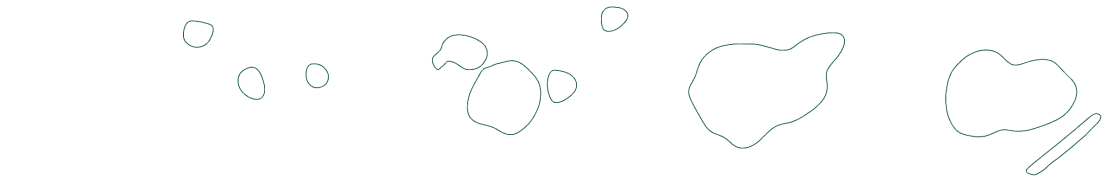
“U” to “V”



$z = 0$ $z = 0.25$ $z = 0.5$ $z = 0.75$ $z = 1$

Table 18: **“U” to “V” with Loop.** Curves

Elephant



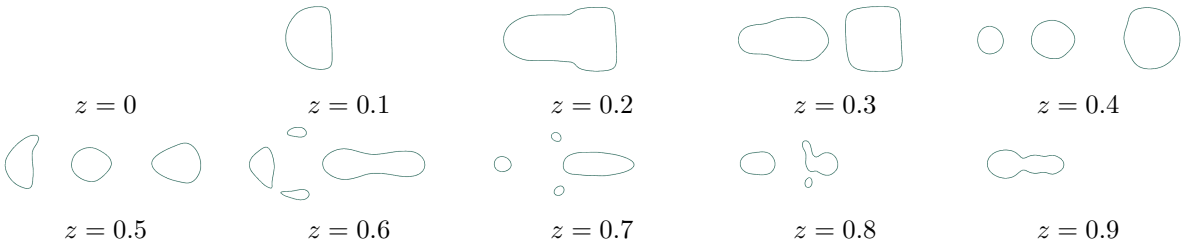
$z = 0$ $z = 0.1$ $z = 0.2$ $z = 0.3$ $z = 0.4$

$z = 0.5$ $z = 0.6$ $z = 0.7$ $z = 0.8$ $z = 0.9$

$z = 1$

Table 19: **Elephant with Loop.** Curves

Fertility



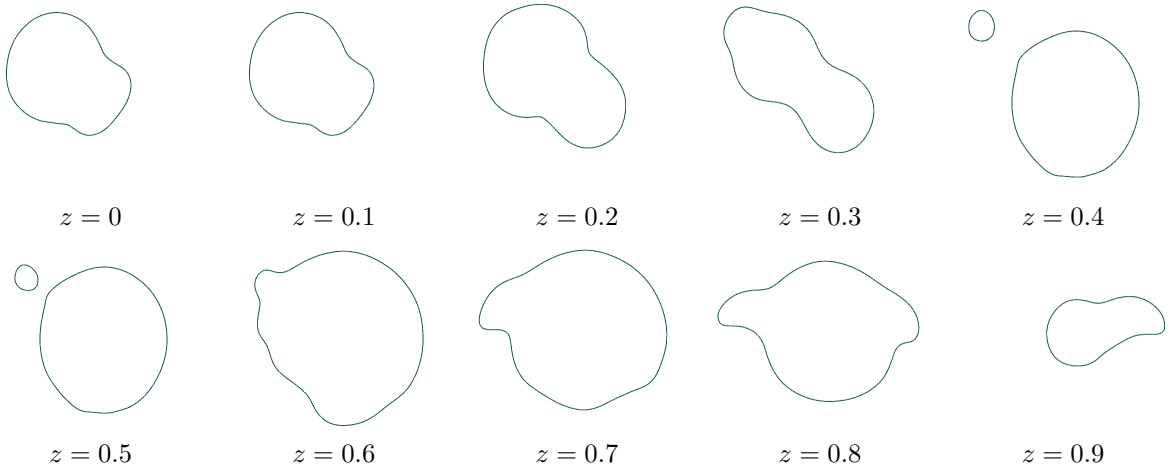
$z = 0$ $z = 0.1$ $z = 0.2$ $z = 0.3$ $z = 0.4$

$z = 0.5$ $z = 0.6$ $z = 0.7$ $z = 0.8$ $z = 0.9$

$z = 1$

Table 20: **Fertility with Loop.** Curves

Kitten



$z = 0$ $z = 0.1$ $z = 0.2$ $z = 0.3$ $z = 0.4$

$z = 0.5$ $z = 0.6$ $z = 0.7$ $z = 0.8$ $z = 0.9$

$z = 1$

Table 21: **Kitten with Loop.** Curves

Moai

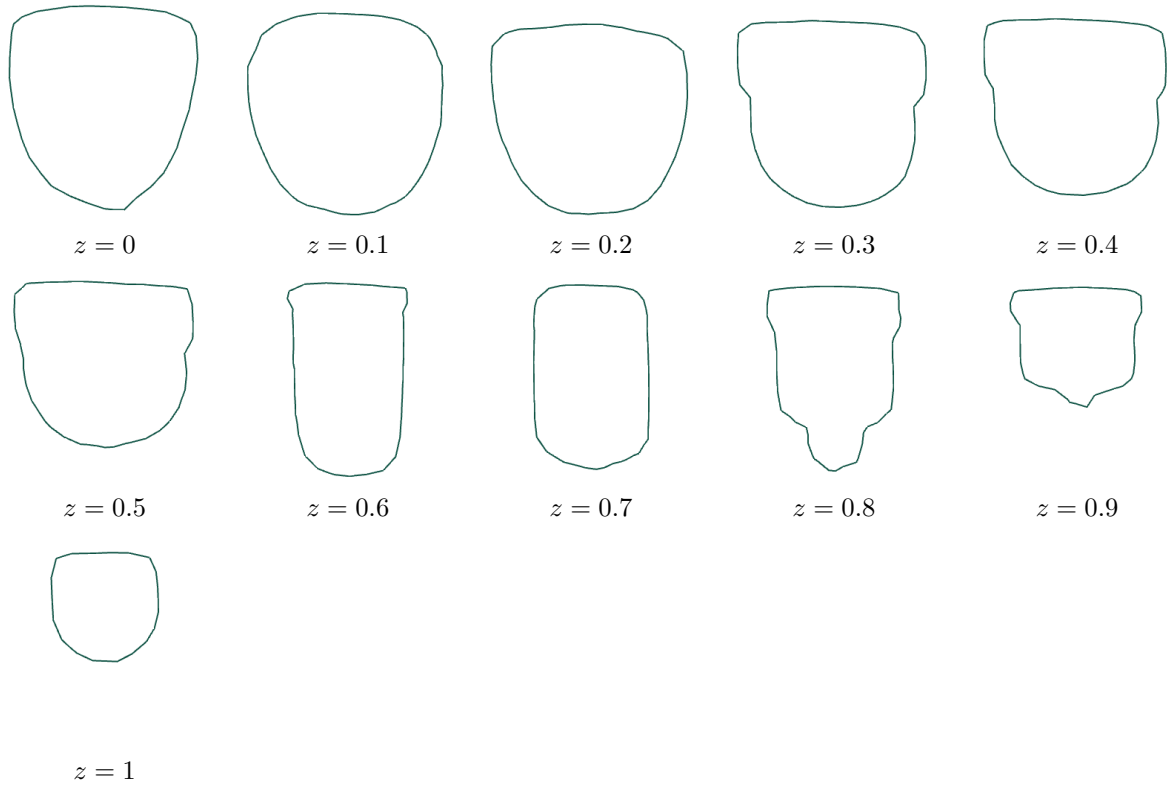


Table 22: **Moai with Loop.** Curves

10.1.3 Sqrt3 Subdivision

"-" to "+"

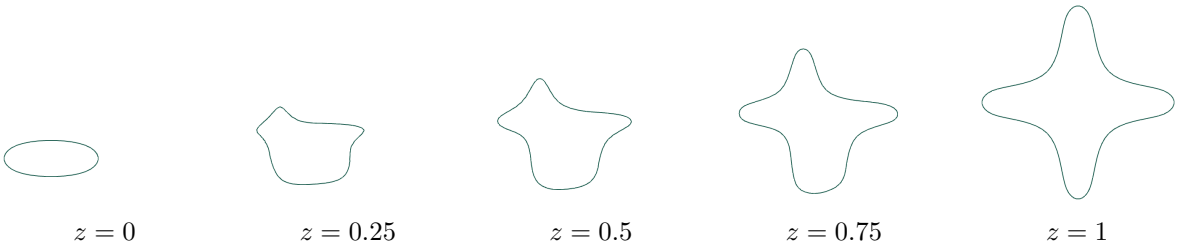
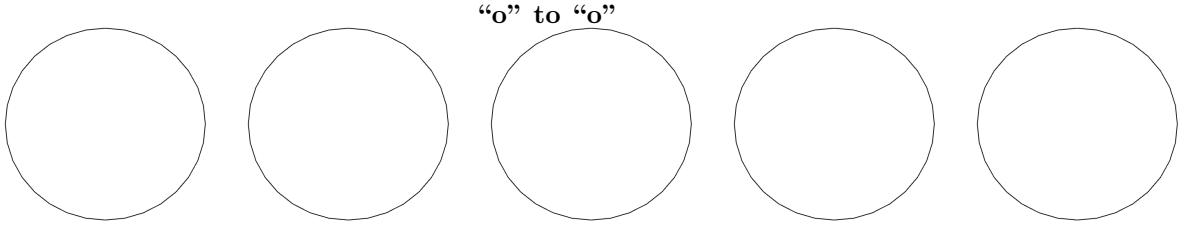


Table 23: **"-" to "+" with $\sqrt{3}$.** Curves

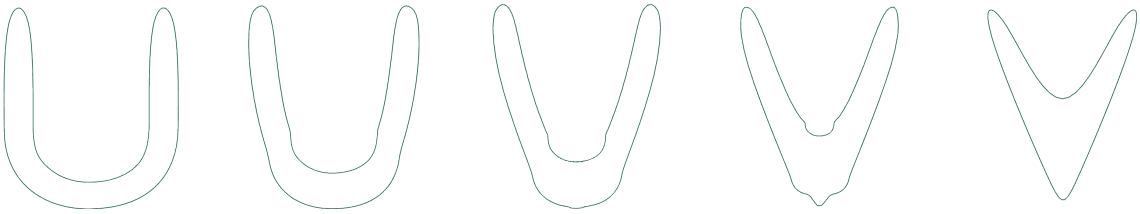
"o" to "o"



$z = 0$ $z = 0.25$ $z = 0.5$ $z = 0.75$ $z = 1$

Table 24: **"o" to "o" with $\sqrt{3}$.** Curves

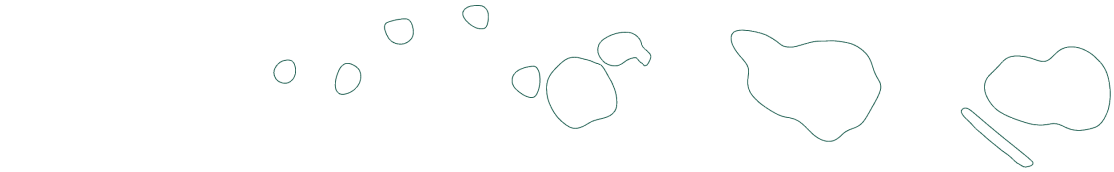
"U" to "V"



$z = 0$ $z = 0.25$ $z = 0.5$ $z = 0.75$ $z = 1$

Table 25: **"U" to "V" with $\sqrt{3}$.**

Elephant



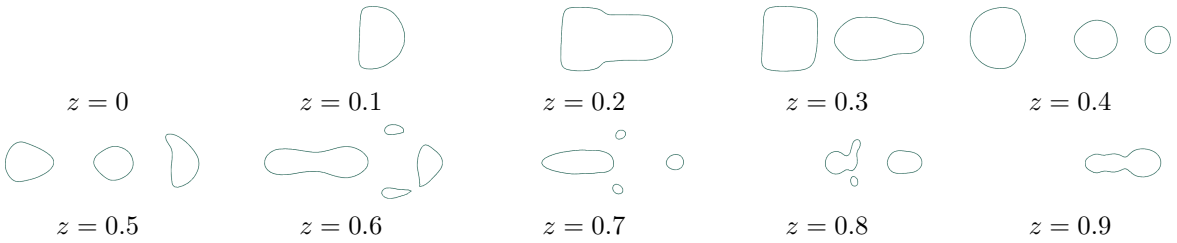
$z = 0$ $z = 0.1$ $z = 0.2$ $z = 0.3$ $z = 0.4$

$z = 0.5$ $z = 0.6$ $z = 0.7$ $z = 0.8$ $z = 0.9$

$z = 1$

Table 26: **Elephant with $\sqrt{3}$.**

Fertility



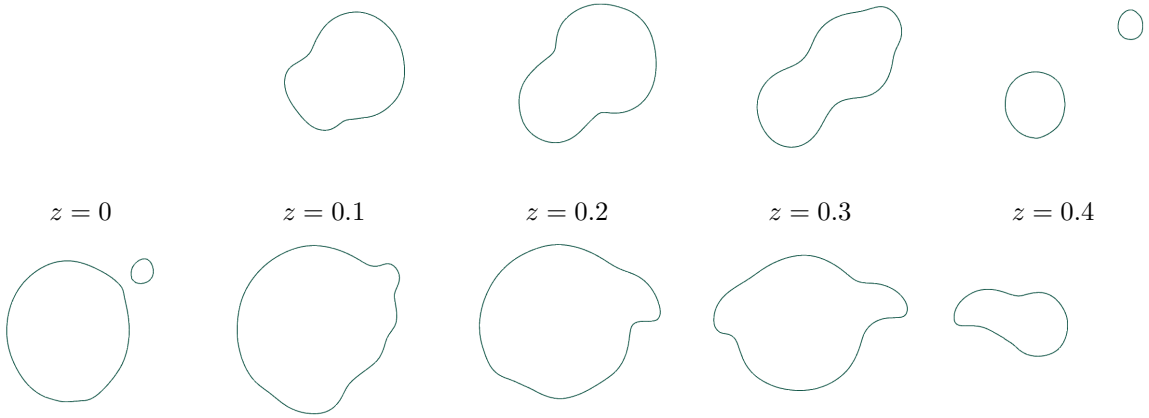
$z = 0$ $z = 0.1$ $z = 0.2$ $z = 0.3$ $z = 0.4$

$z = 0.5$ $z = 0.6$ $z = 0.7$ $z = 0.8$ $z = 0.9$

$z = 1$

Table 27: **Fertility with $\sqrt{3}$.** Curves

Kitten



$z = 0$ $z = 0.1$ $z = 0.2$ $z = 0.3$ $z = 0.4$

$z = 0.5$ $z = 0.6$ $z = 0.7$ $z = 0.8$ $z = 0.9$

$z = 1$

Table 28: **Kitten with $\sqrt{3}$.** Curves

Moai

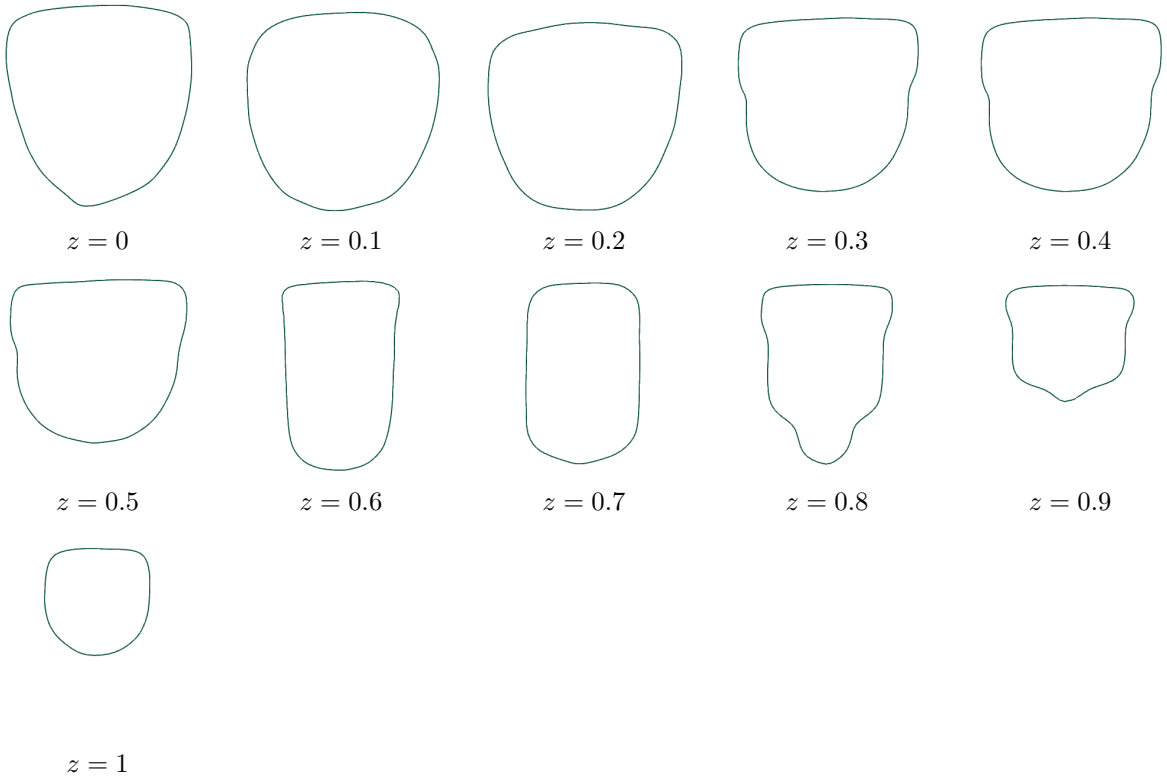


Table 29: **Moai with $\sqrt{3}$.** Curves

10.1.4 Interpolating Sqrt3 Subdivision

"-" to "+"

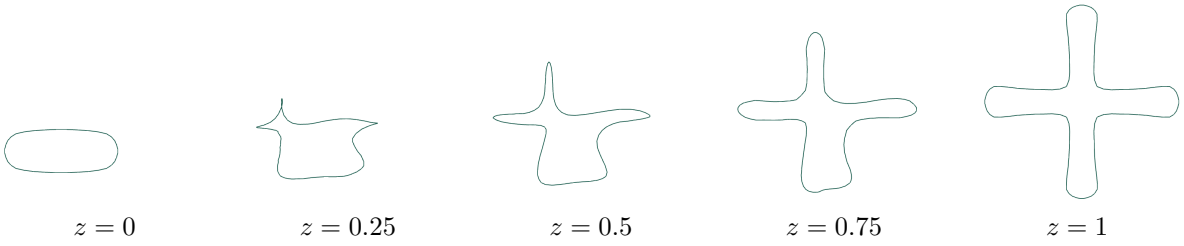
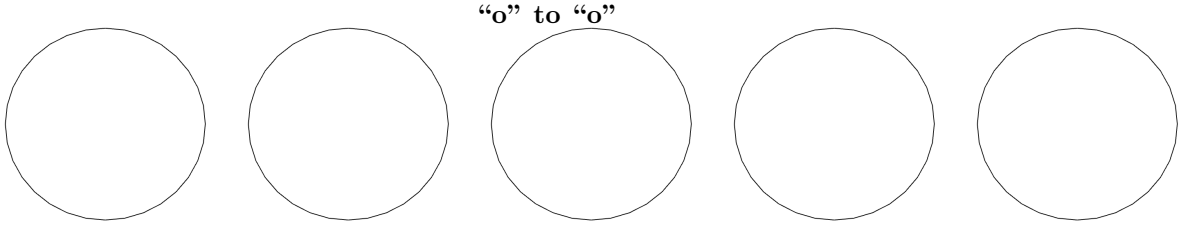


Table 30: **"-" to "+" with int. $\sqrt{3}$.** Curves

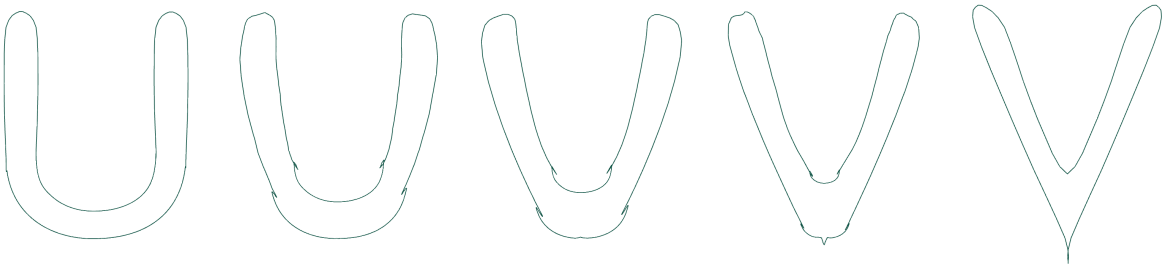
"o" to "o"



$z = 0$ $z = 0.25$ $z = 0.5$ $z = 0.75$ $z = 1$

Table 31: **"o" to "o" with int. $\sqrt{3}$.** Curves

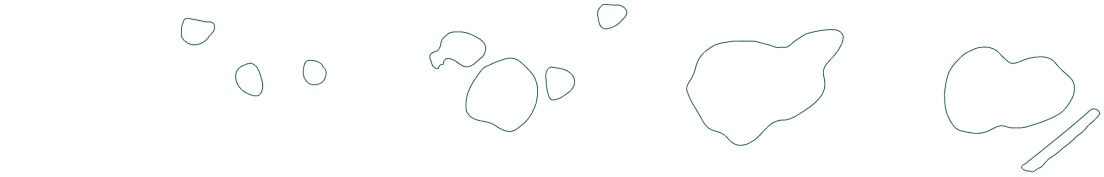
"U" to "V"



$z = 0$ $z = 0.25$ $z = 0.5$ $z = 0.75$ $z = 1$

Table 32: **"U" to "V" with int. $\sqrt{3}$.** Curves

Elephant



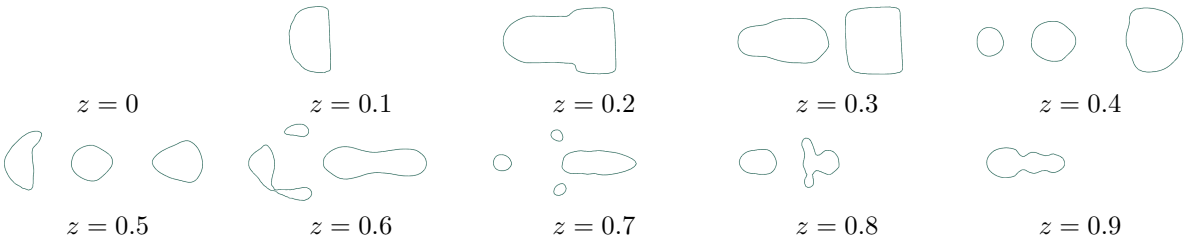
$z = 0$ $z = 0.1$ $z = 0.2$ $z = 0.3$ $z = 0.4$

$z = 0.5$ $z = 0.6$ $z = 0.7$ $z = 0.8$ $z = 0.9$

$z = 1$

Table 33: **Elephant with int. $\sqrt{3}$.** Curves

Fertility



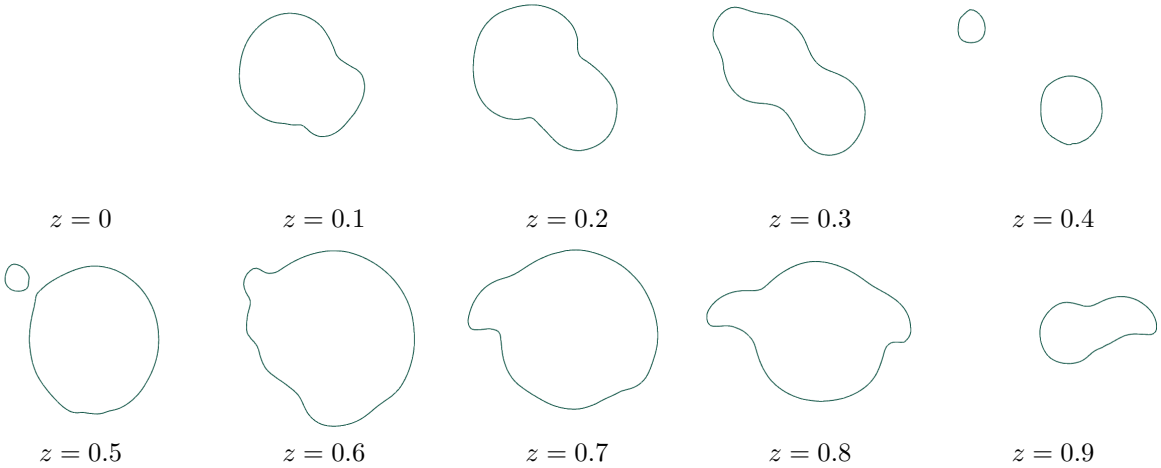
$z = 0$ $z = 0.1$ $z = 0.2$ $z = 0.3$ $z = 0.4$

$z = 0.5$ $z = 0.6$ $z = 0.7$ $z = 0.8$ $z = 0.9$

$z = 1$

Table 34: **Fertility with int. $\sqrt{3}$.** Curves

Kitten



$z = 0$ $z = 0.1$ $z = 0.2$ $z = 0.3$ $z = 0.4$

$z = 0.5$ $z = 0.6$ $z = 0.7$ $z = 0.8$ $z = 0.9$

$z = 1$

Table 35: **Kitten with int. $\sqrt{3}$.** Curves

Moai

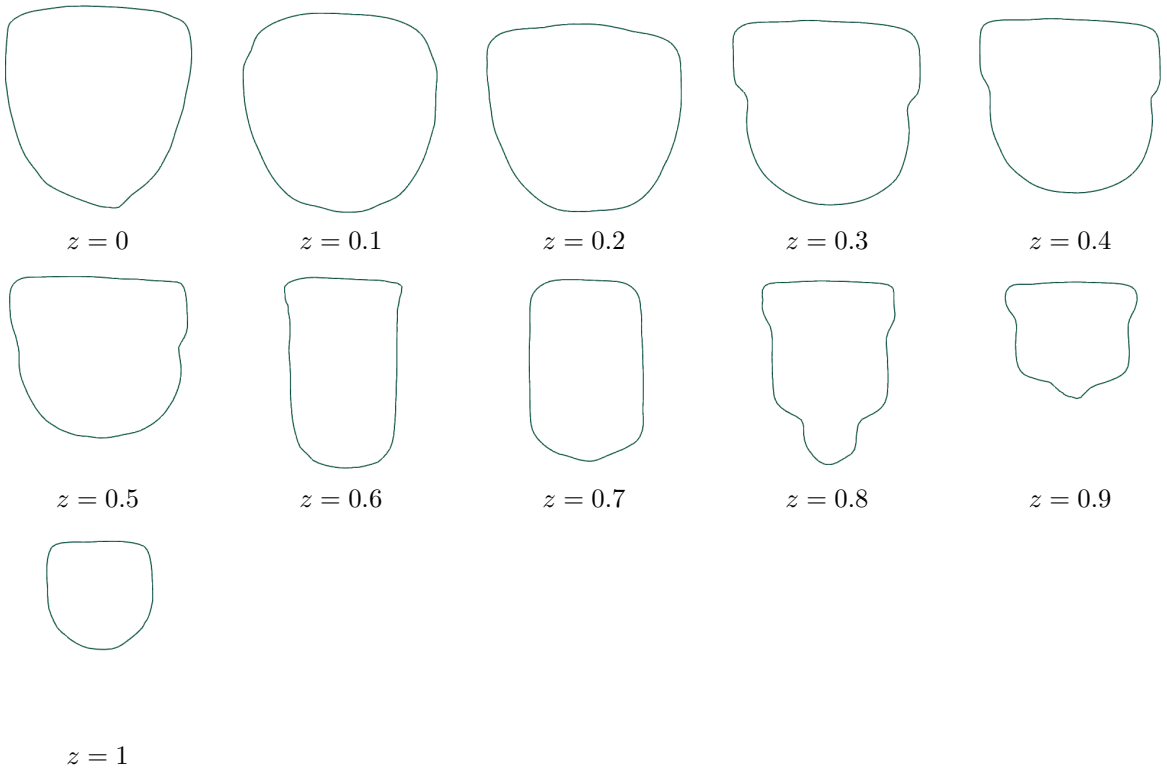


Table 36: **Moai with int. $\sqrt{3}$.** Curves

10.1.5 Modified Butterfly Subdivision

"-" to "+"

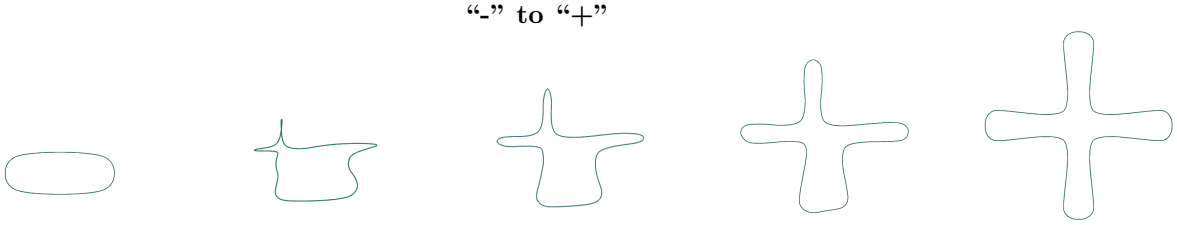
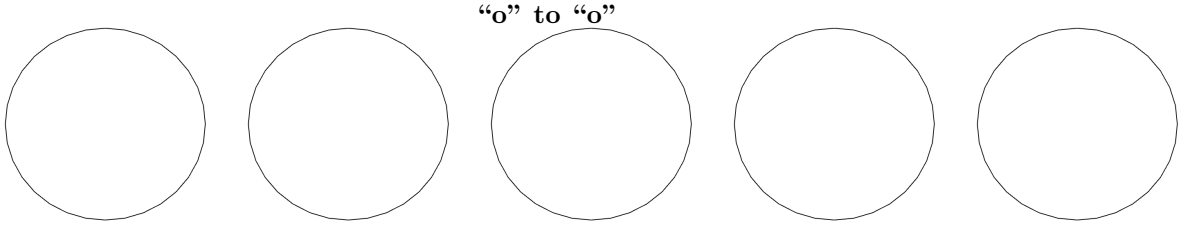


Table 37: **"-" to "+" with Butterfly.** Curves

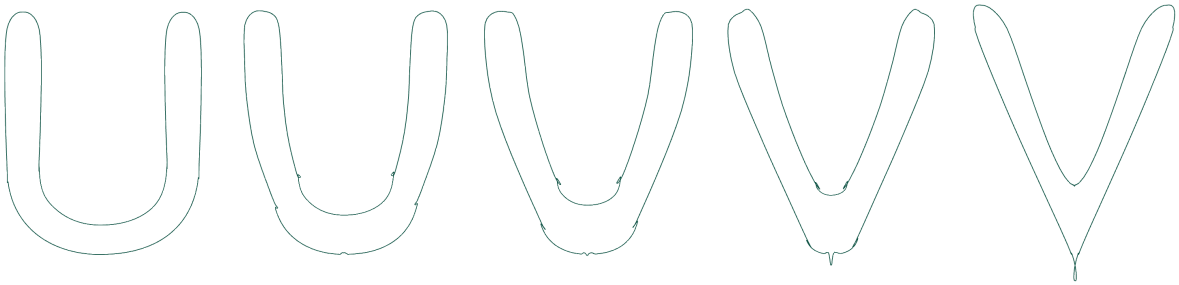
“o” to “o”



$z = 0$ $z = 0.25$ $z = 0.5$ $z = 0.75$ $z = 1$

Table 38: **“o” to “o” with Butterfly.** Curves

“U” to “V”



$z = 0$ $z = 0.25$ $z = 0.5$ $z = 0.75$ $z = 1$

Table 39: **“U” to “V” with Butterfly.** Curves

Elephant

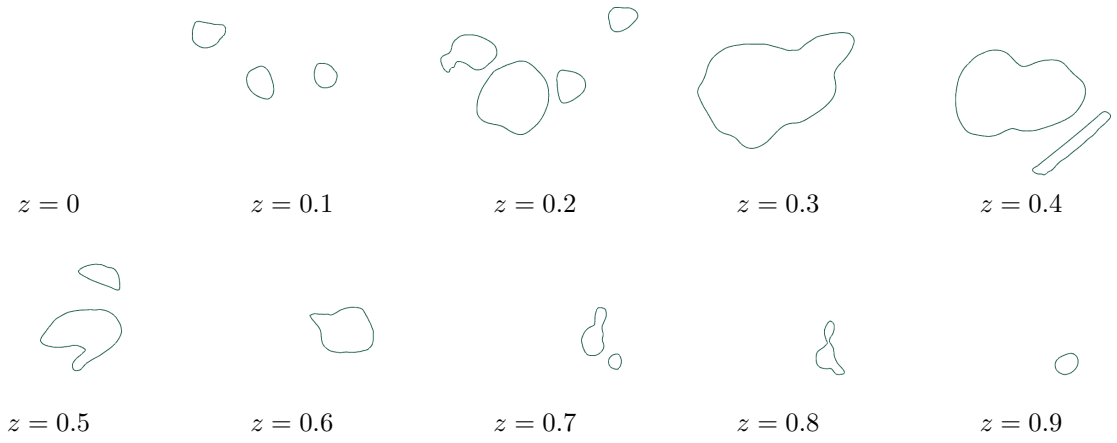


Table 40: **Elephant with Butterfly.** Curves

Fertility

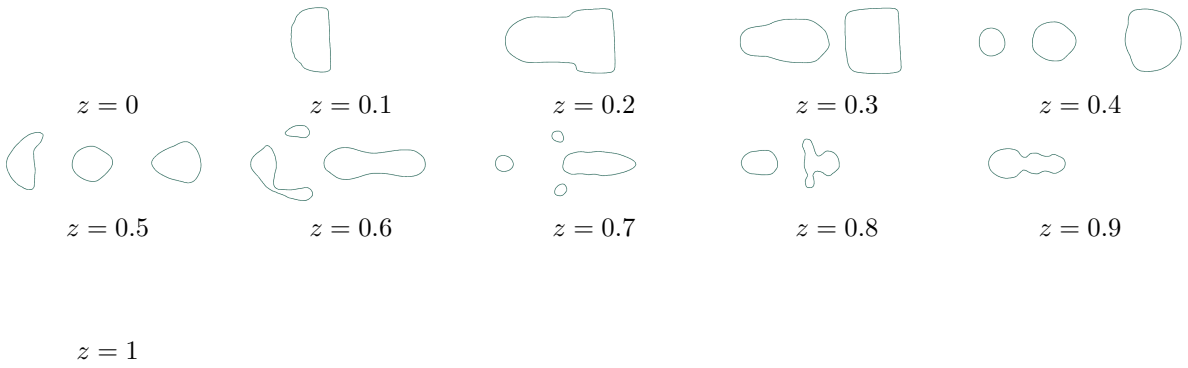
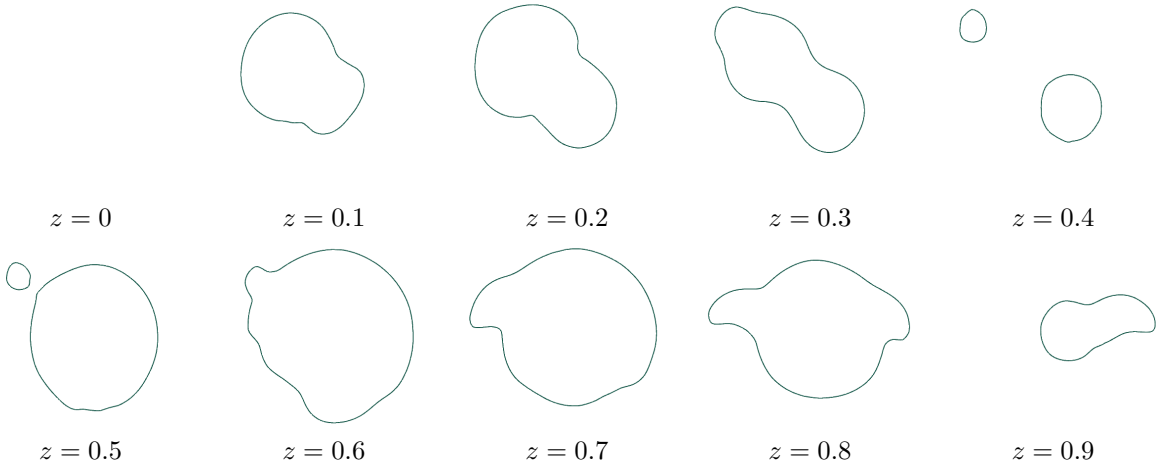


Table 41: **Fertility with Butterfly.** Curves

Kitten



$z = 1$

Table 42: **Kitten with Butterfly.** Curves

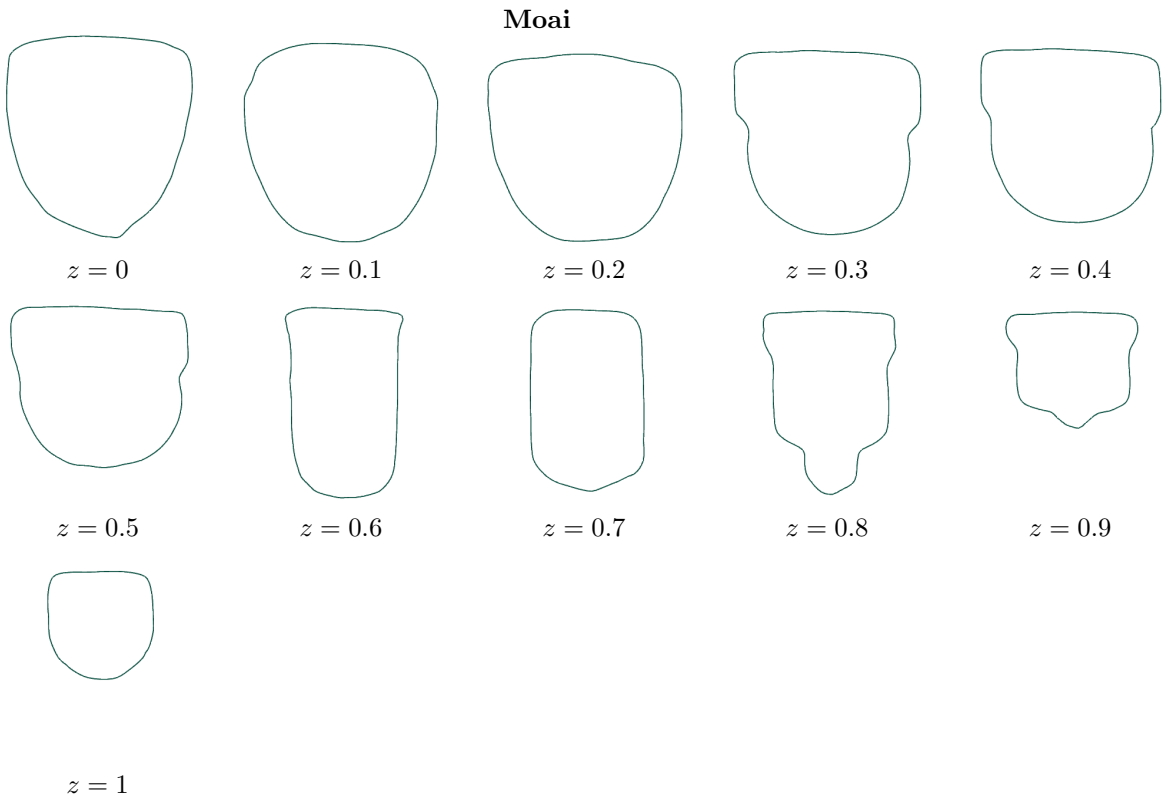


Table 43: **Moai with Butterfly.** Curves

10.2 Area

10.2.1 "-" to "+"

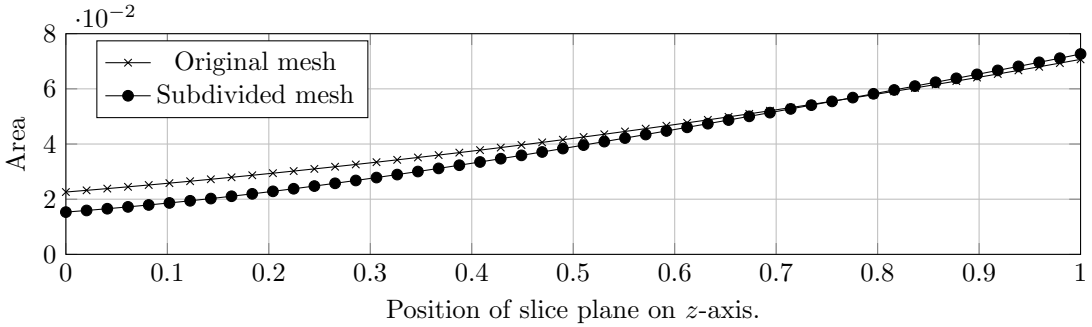


Figure 9: **"-" to "+" with Loop.** Area enclosed by slice. $\sigma = 0.001169530612$

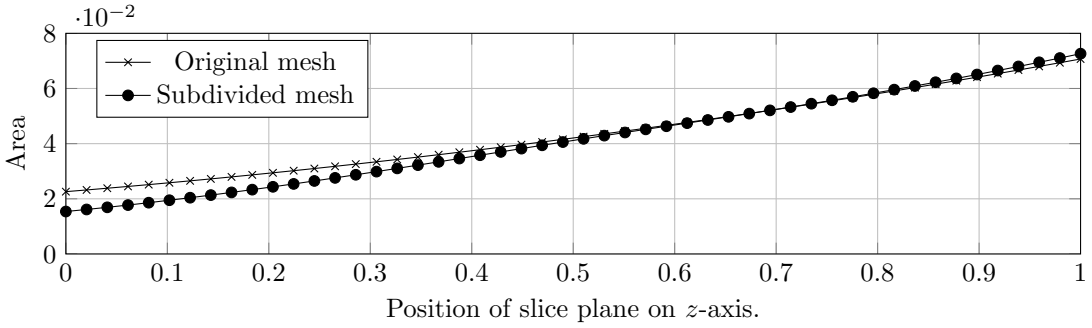


Figure 10: **"-" to "+" with $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.001167679592$

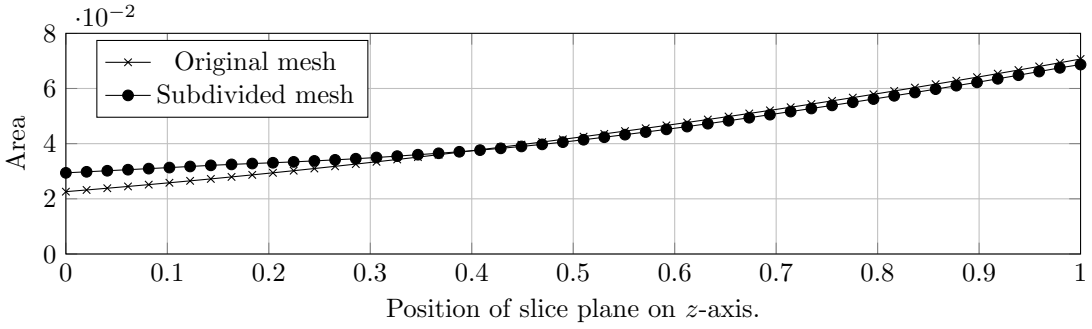


Figure 11: **"-" to "+" with int. $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.0008011020408$

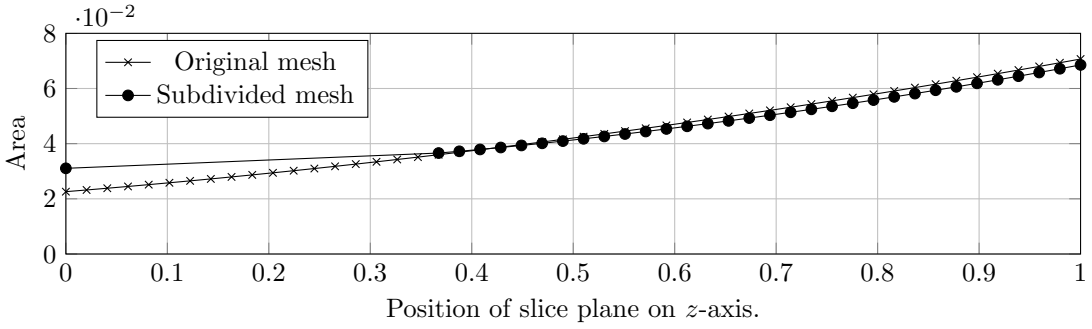


Figure 12: **"-" to "+" with Butterfly.** Area enclosed by slice. Not defined everywhere due to self-intersection. $\sigma = 0.00117154375$

10.2.2 "o" to "o"

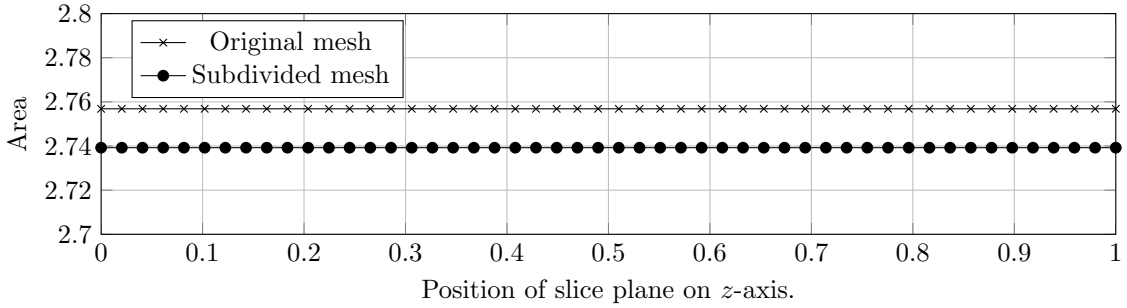


Figure 13: **"o" to "o" with Loop.** Area enclosed by slice. $\sigma = 0$

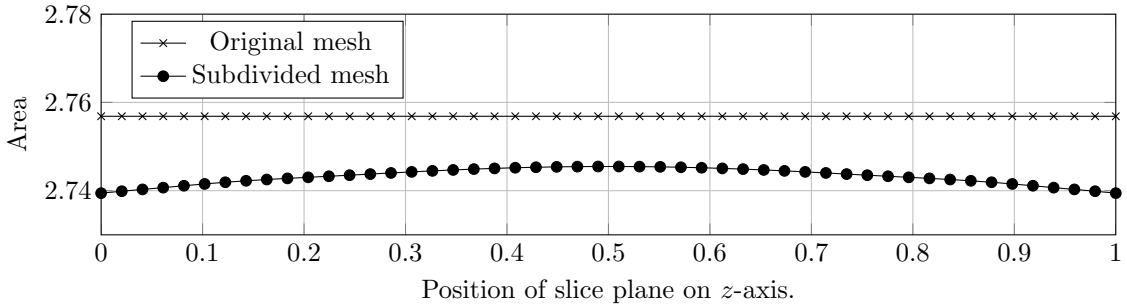


Figure 14: **"o" to "o" with $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.0002465306122$

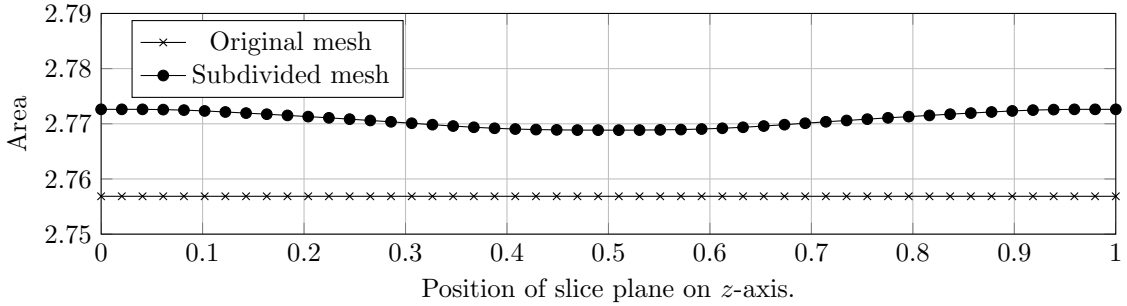


Figure 15: **"o" to "o" with int. $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.000153877551$

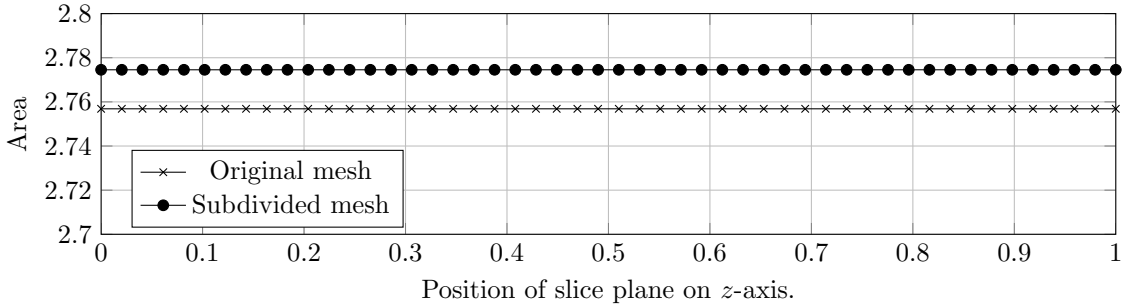


Figure 16: **"o" to "o" with Butterfly.** Area enclosed by slice. $\sigma = 0$

10.2.3 "U" to "V"

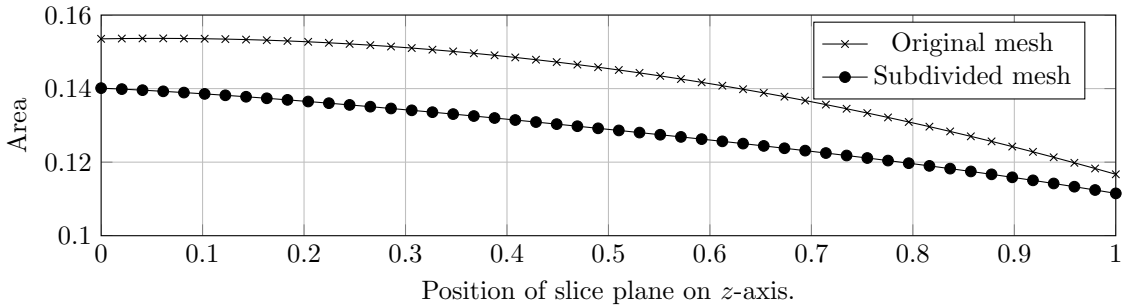


Figure 17: **"U" to "V" with Loop.** Area enclosed by slice. $\sigma = 0.0005839795918$

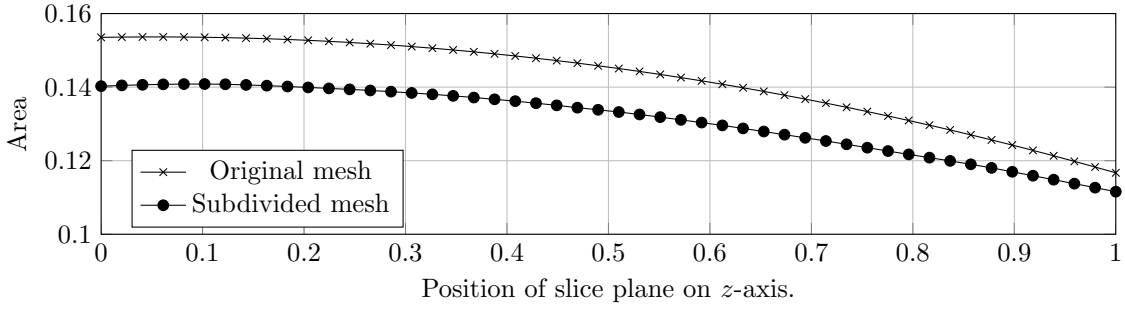


Figure 18: **"U" to "V" with $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.0006089183673$

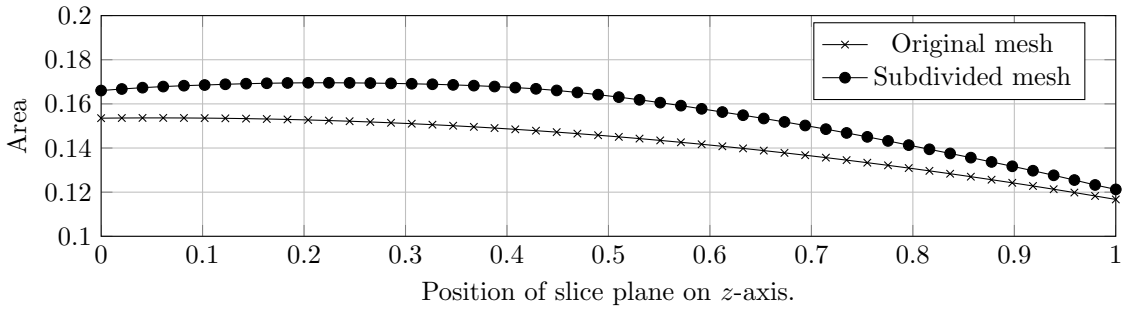


Figure 19: **"U" to "V" with int. $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.001057836735$

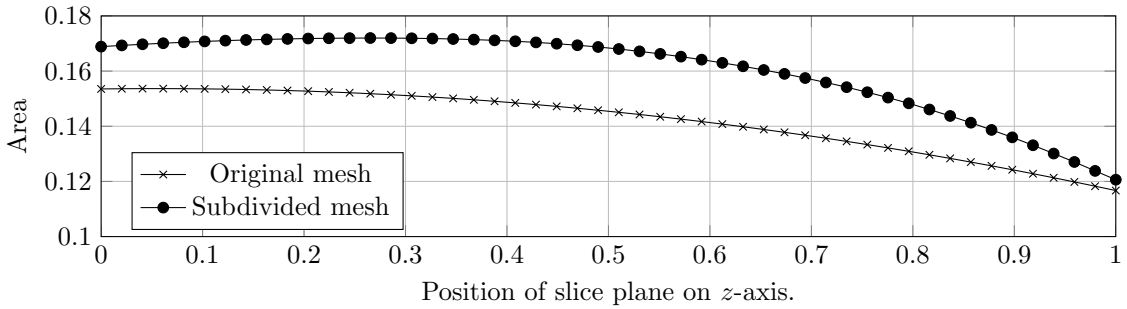


Figure 20: **"U" to "V" with Butterfly.** Area enclosed by slice. $\sigma = 0.0007563877551$

10.2.4 Elephant

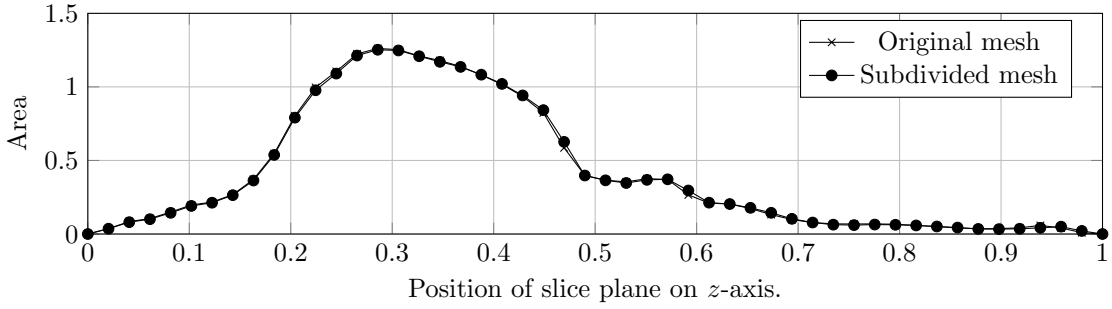


Figure 21: **Elephant with Loop.** Area enclosed by slice. $\sigma = 0.05297383265$

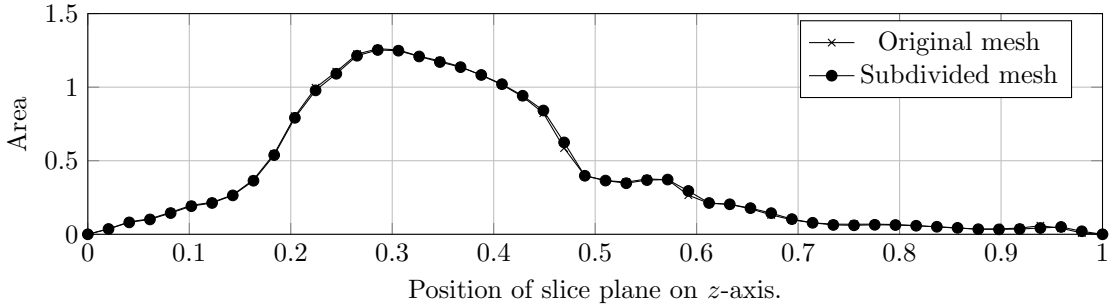


Figure 22: **Elephant with $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.05295422857$

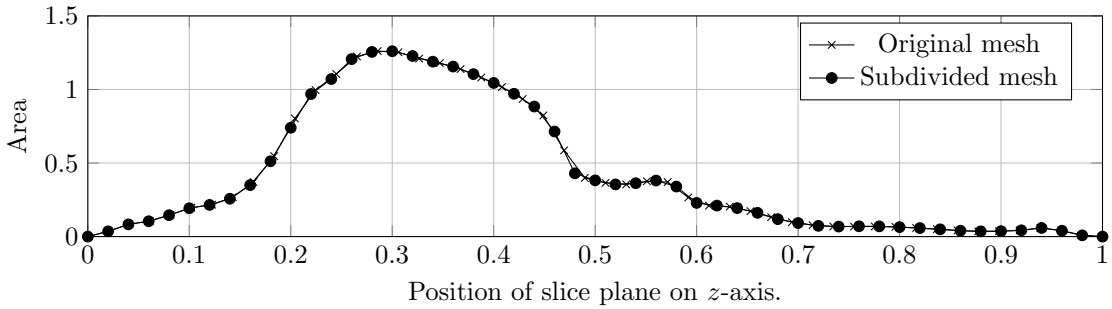


Figure 23: **Elephant with int. $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.052450352$

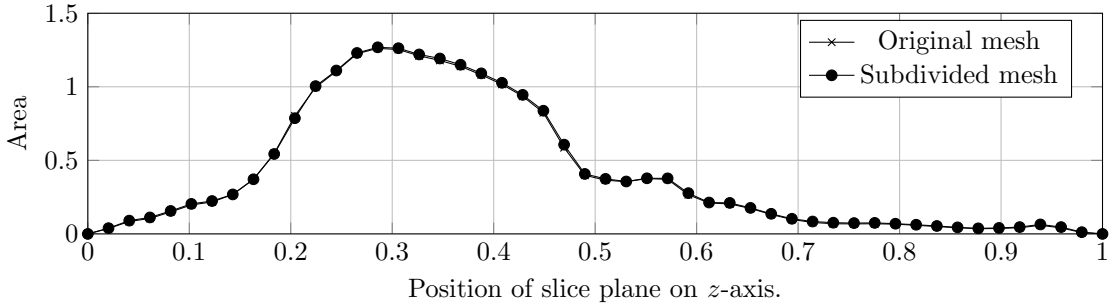


Figure 24: **Elephant with Butterfly.** Area enclosed by slice. $\sigma = 0.0537998$

10.2.5 Fertility

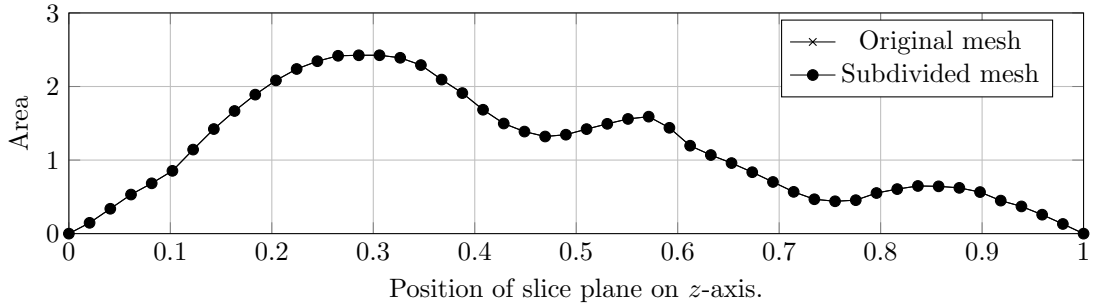


Figure 25: **Fertility with Loop.** Area enclosed by slice. $\sigma = 0.1184813878$

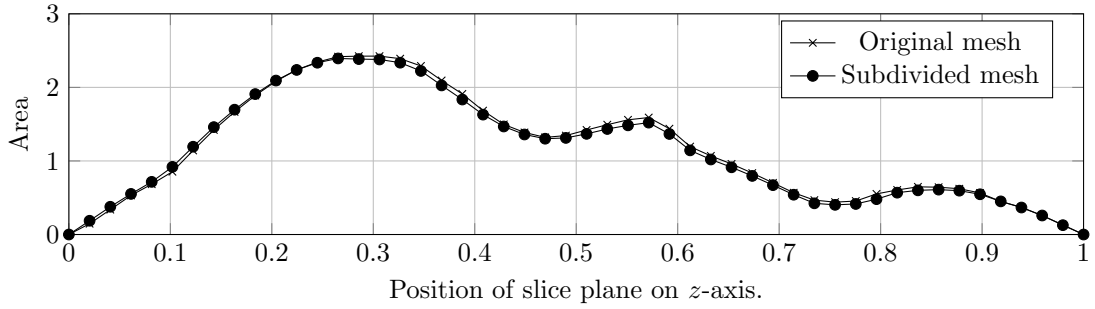


Figure 26: **Fertility with $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.1150389796$

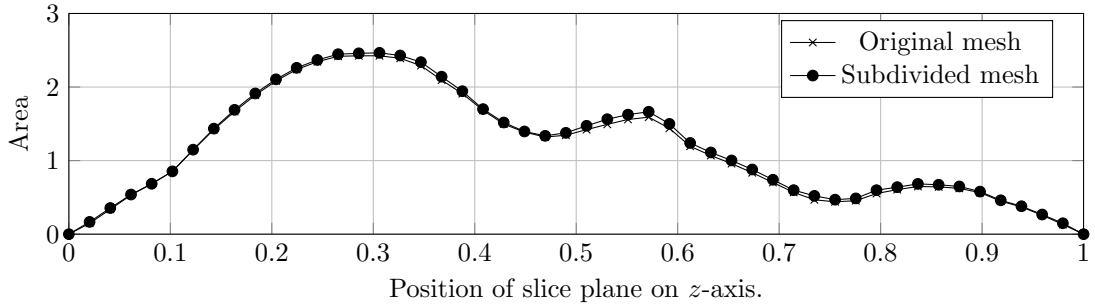


Figure 27: **Fertility with int. $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.1227161224$

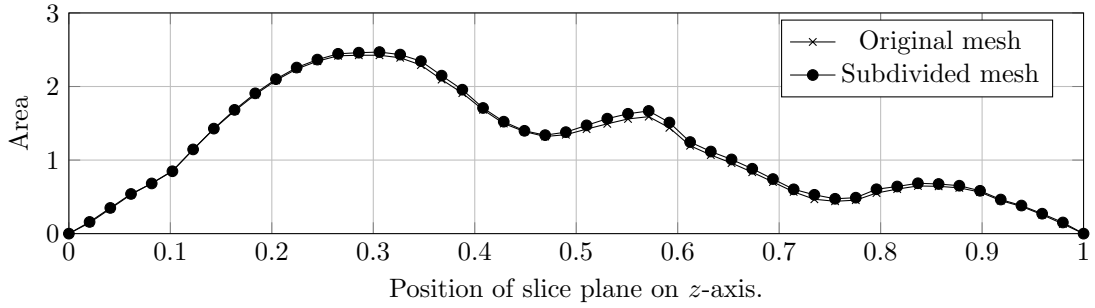


Figure 28: **Fertility with Butterfly.** Area enclosed by slice. $\sigma = 0.1227924898$

10.2.6 Kitten

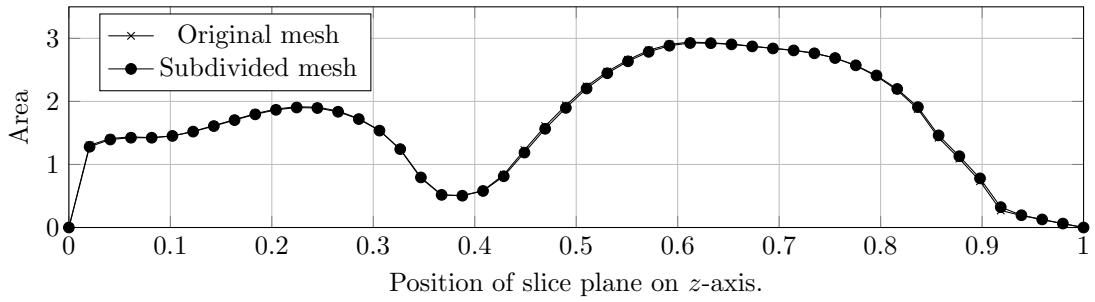


Figure 29: **Kitten with Loop.** Area enclosed by slice. $\sigma = 0.1765239592$

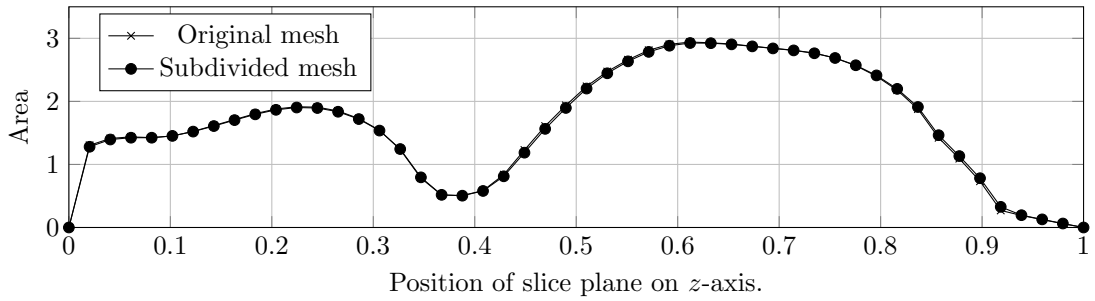


Figure 30: **Kitten with $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.1764843265$

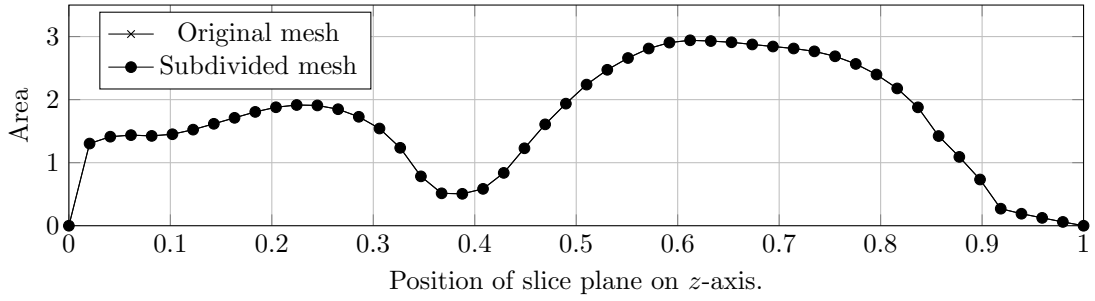


Figure 31: **Kitten with int. $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.178036898$

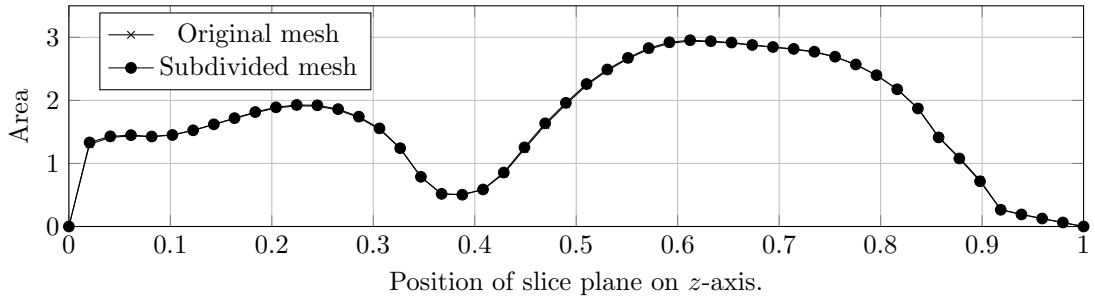


Figure 32: **Kitten with Butterfly.** Area enclosed by slice. $\sigma = 0.1795662041$

10.2.7 Moai

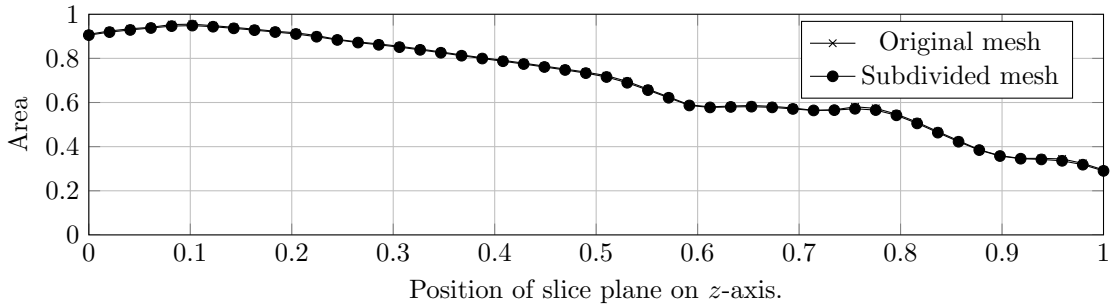


Figure 33: **Moai with Loop.** Area enclosed by slice. $\sigma = 0.0147615102$

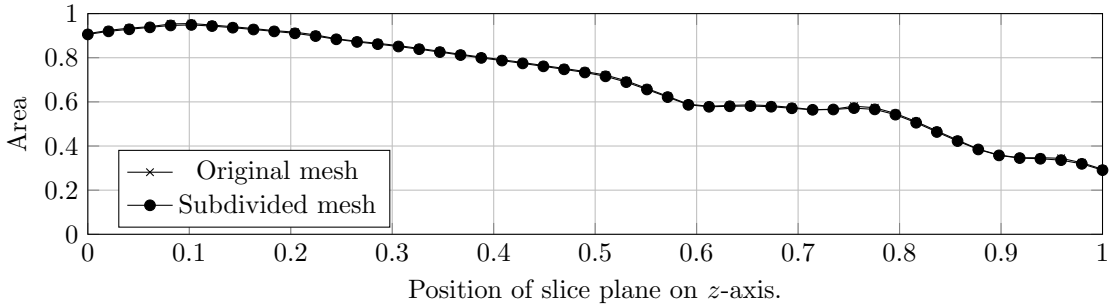


Figure 34: **Moai with $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.01476202041$

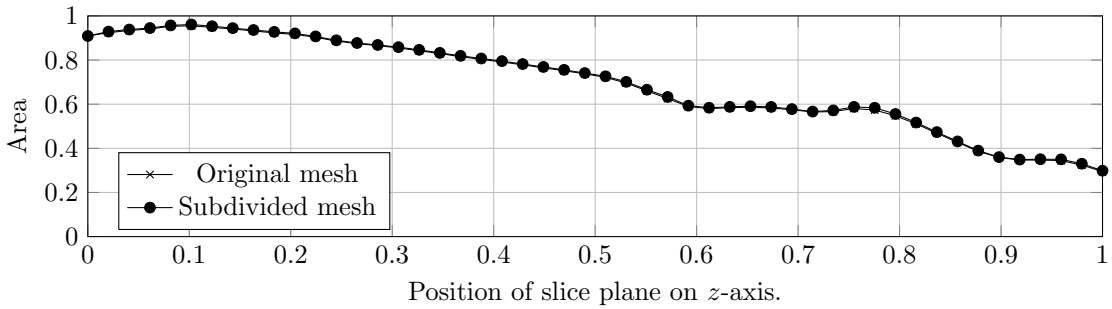


Figure 35: **Moai with int. $\sqrt{3}$.** Area enclosed by slice. $\sigma = 0.01579673469$

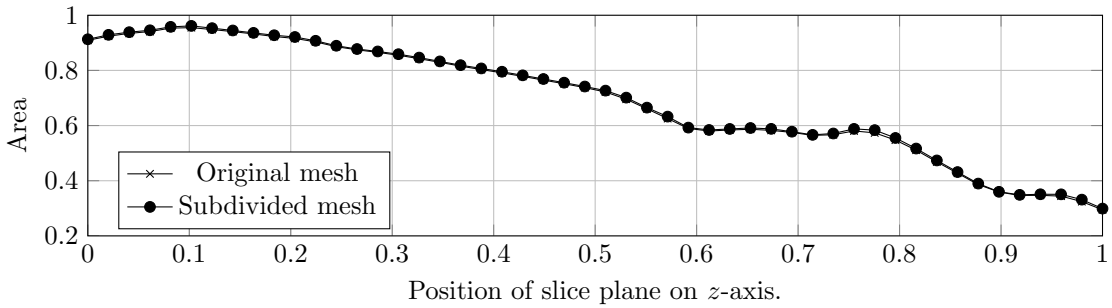


Figure 36: **Moai with Butterfly.** Area enclosed by slice. $\sigma = 0.01576493878$

10.3 Total positive curvature

10.3.1 "-" to "+"

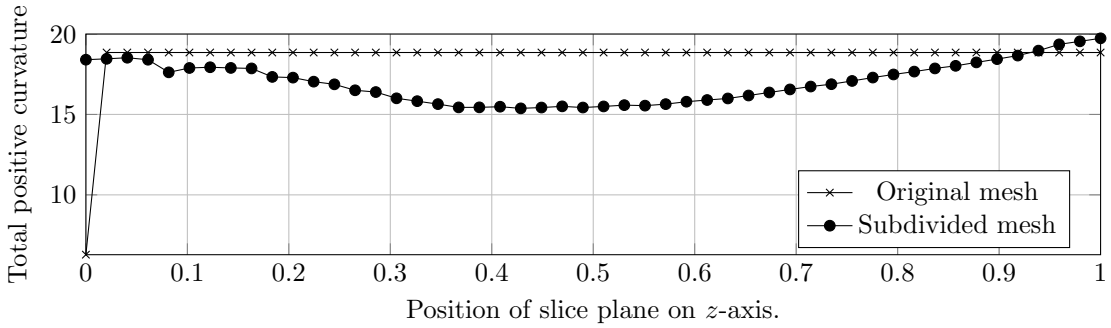


Figure 37: **"-" to "+" with Loop.** Total positive curvature. $\sigma = 0.1744081633$

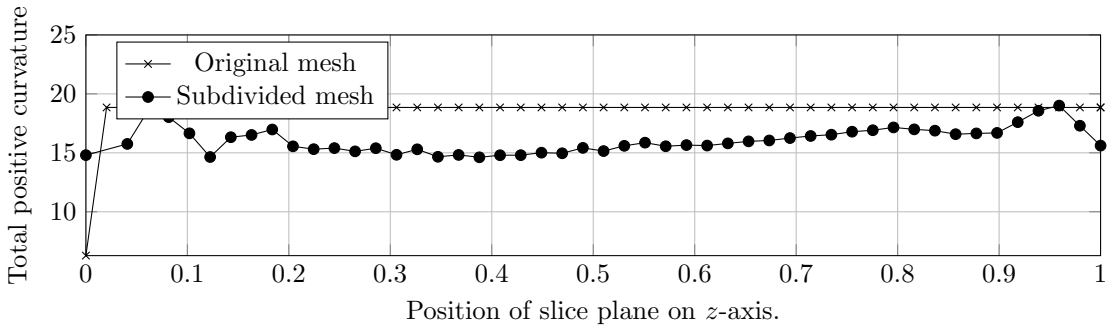


Figure 38: **"-" to "+" with $\sqrt{3}$.** Total positive curvature. $\sigma = 0.5127541667$

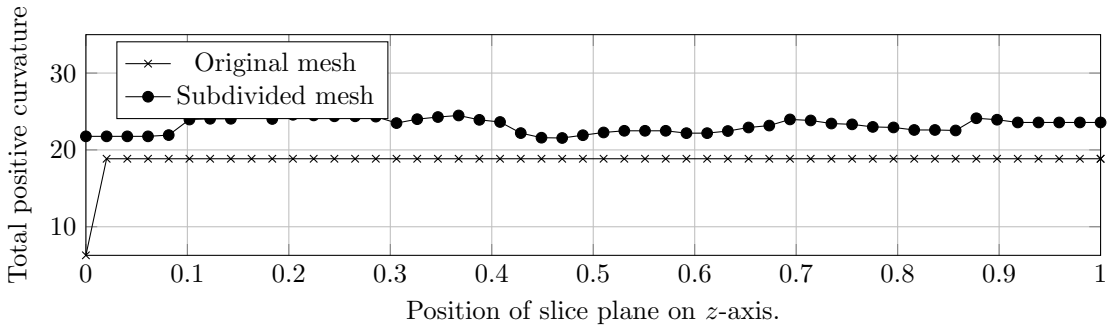


Figure 39: **"-" to "+" with int. $\sqrt{3}$.** Total positive curvature. $\sigma = 0.3625673469$

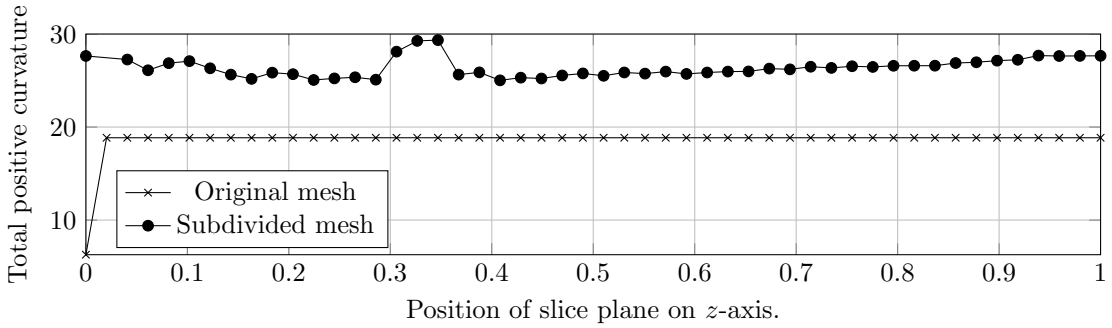


Figure 40: **"-" to "+" with Butterfly.** Total positive curvature. $\sigma = 0.4212145833$

10.3.2 "o" to "o"

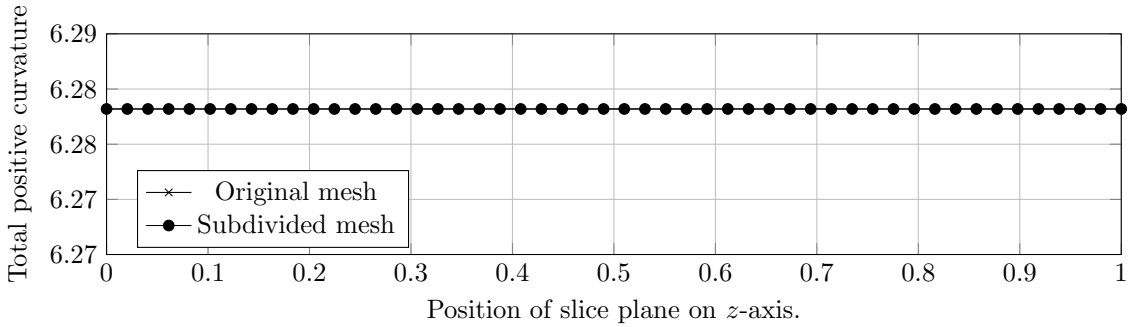


Figure 41: **"o" to "o" with Loop.** Total positive curvature. $\sigma = 0$

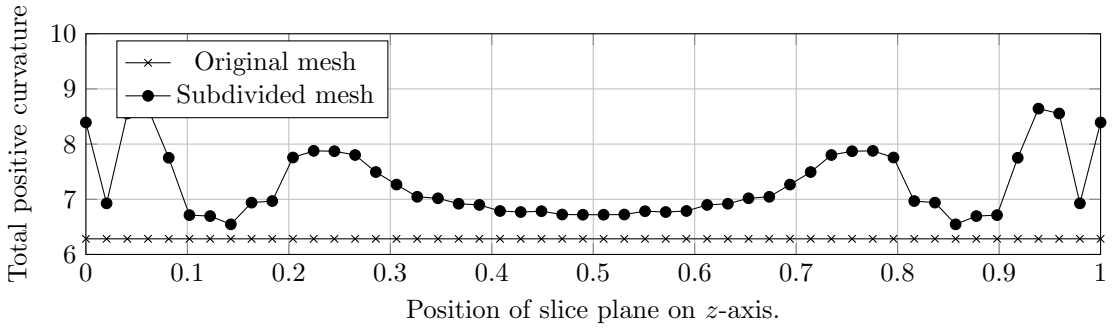


Figure 42: **"o" to "o" with $\sqrt{3}$.** Total positive curvature. $\sigma = 0.3183489796$

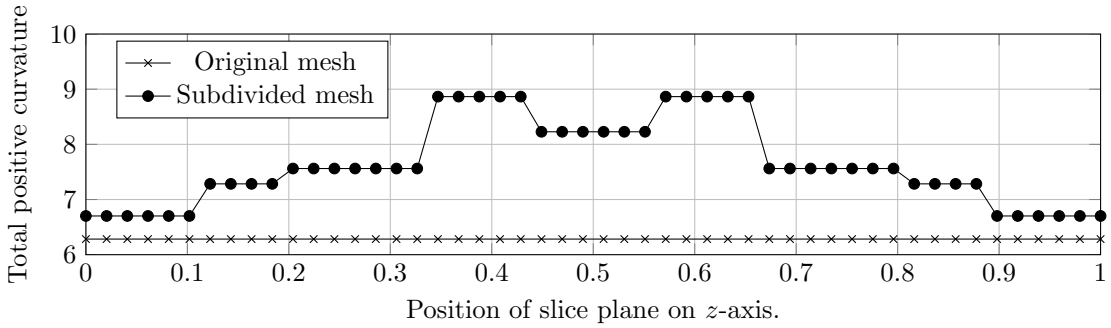


Figure 43: **"o" to "o" with int. $\sqrt{3}$.** Total positive curvature. $\sigma = 0.1142971429$

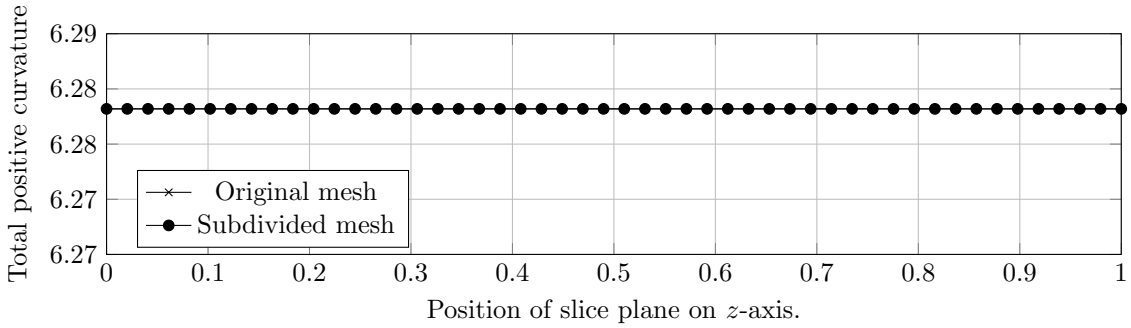


Figure 44: **"o" to "o" with Butterfly.** Total positive curvature. $\sigma = 0$

10.3.3 "U" to "V"

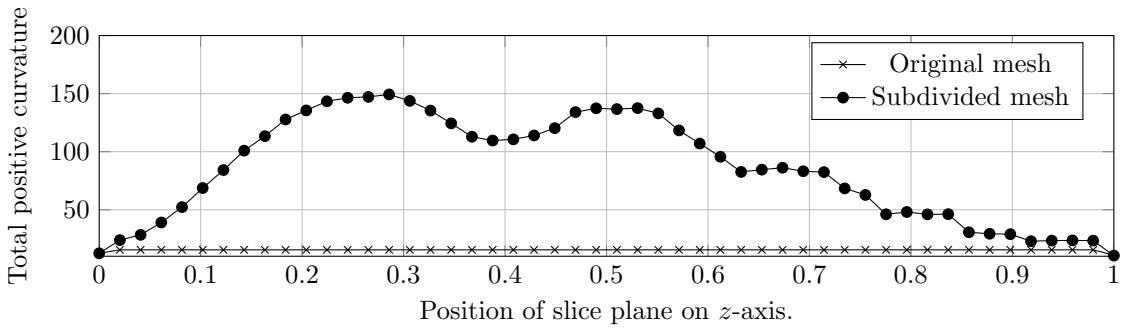


Figure 45: **"U" to "V" with Loop.** Total positive curvature. $\sigma = 7.053002041$

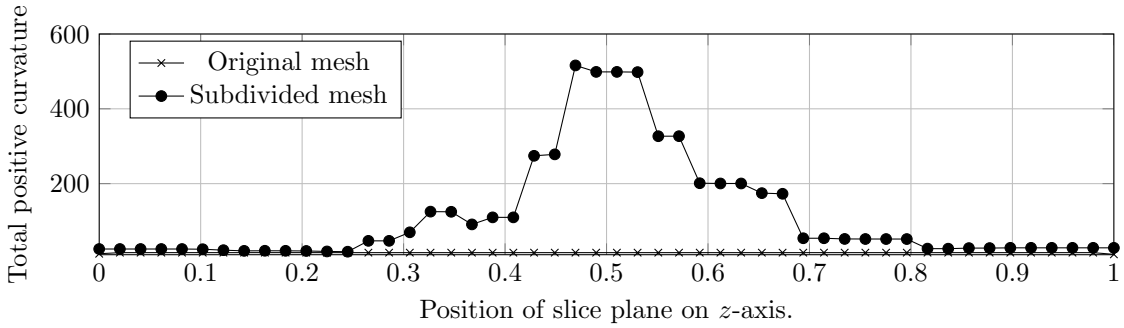


Figure 46: **"U" to "V" with $\sqrt{3}$.** Total positive curvature. $\sigma = 21.76136531$

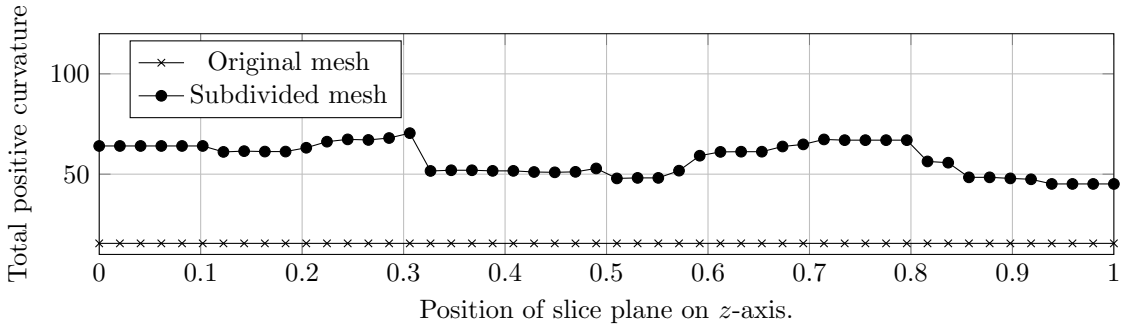


Figure 47: **"U" to "V" with int. $\sqrt{3}$.** Total positive curvature. $\sigma = 1.673816327$

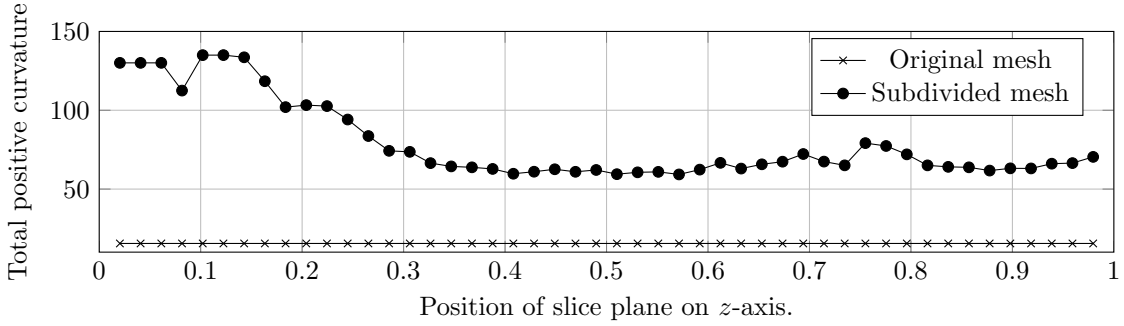


Figure 48: **"U" to "V" with Butterfly.** Total positive curvature. $\sigma = 4.104079167$

10.3.4 Elephant

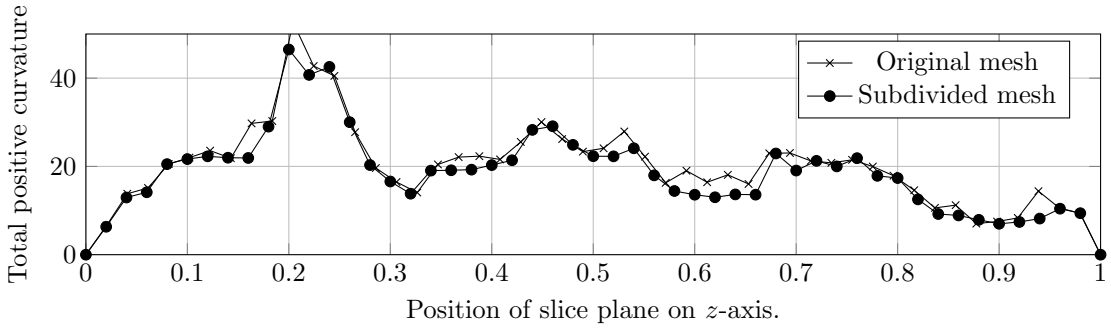


Figure 49: **Elephant with Loop.** Total positive curvature. $\sigma = 3.3302168$

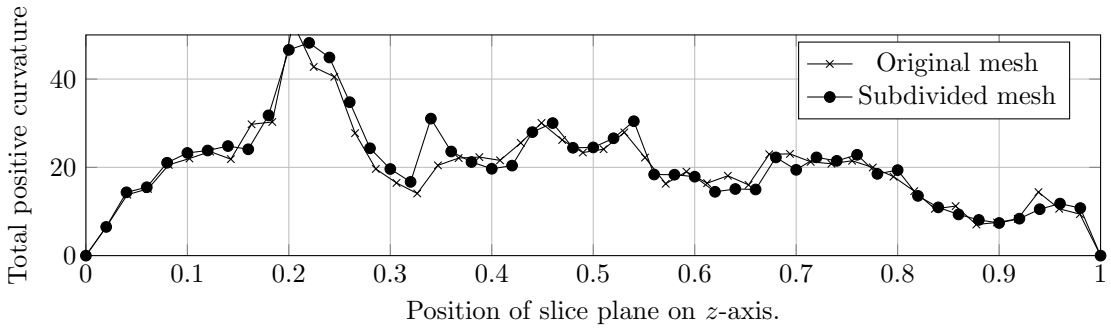


Figure 50: **Elephant with $\sqrt{3}$.** Total positive curvature. $\sigma = 3.8786128$

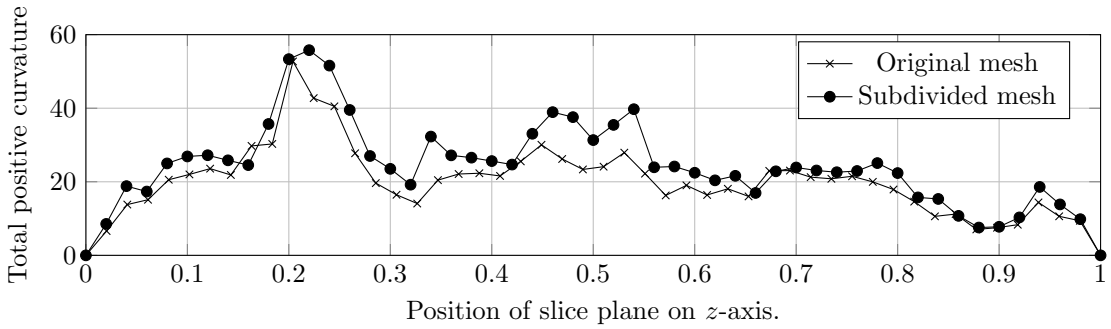


Figure 51: **Elephant with int. $\sqrt{3}$.** Total positive curvature. $\sigma = 4.7024868$

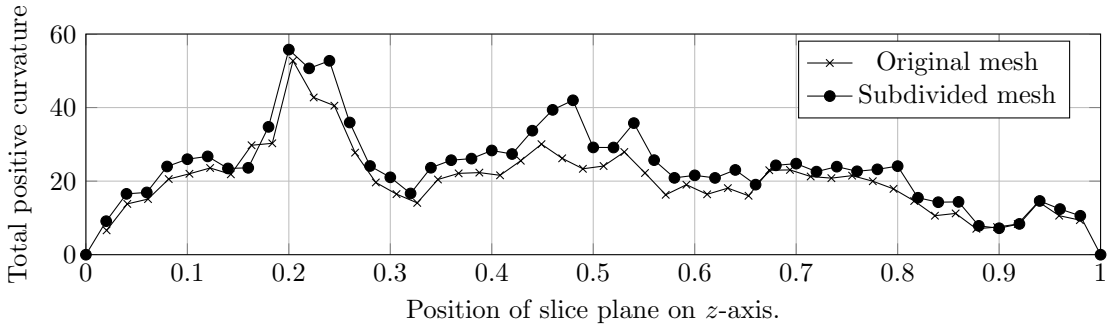


Figure 52: **Elephant with Butterfly.** Total positive curvature. $\sigma = 4.5139824$

10.3.5 Fertility

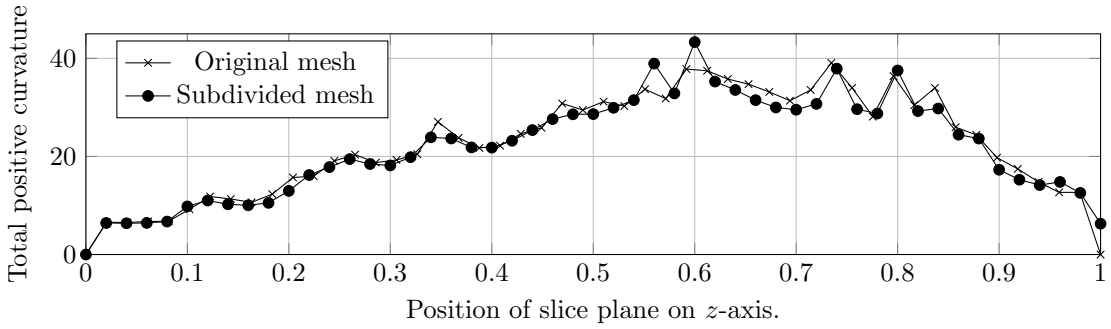


Figure 53: **Fertility with Loop.** Total positive curvature. $\sigma = 2.7585062$

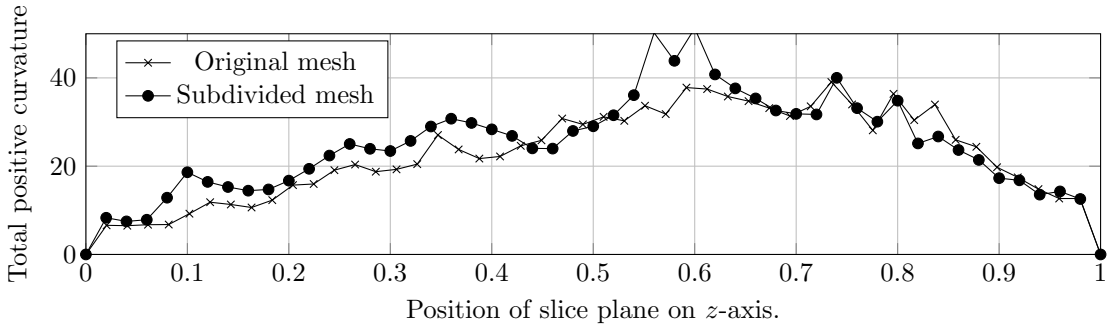


Figure 54: **Fertility with $\sqrt{3}$.** Total positive curvature. $\sigma = 3.4516888$

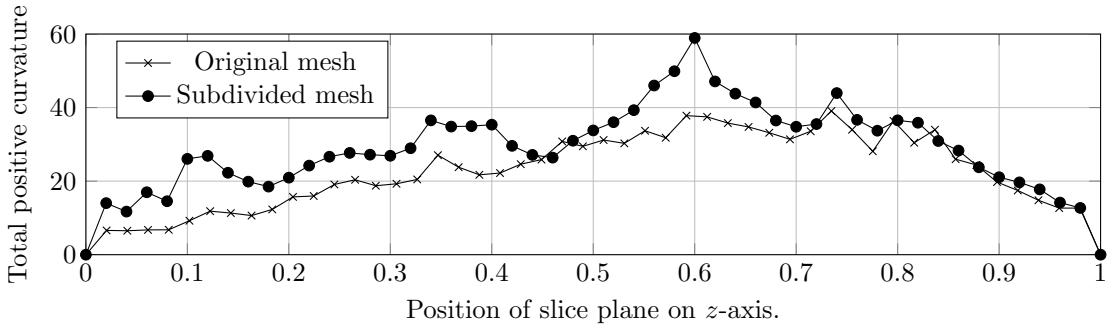


Figure 55: **Fertility with int. $\sqrt{3}$.** Total positive curvature. $\sigma = 3.81776$

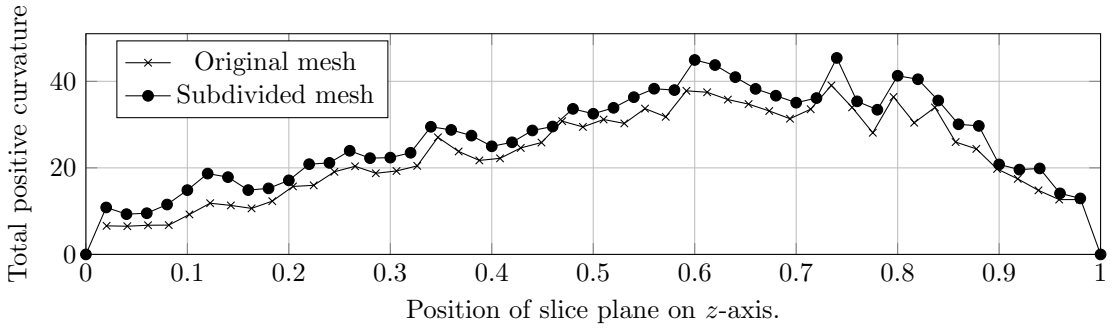


Figure 56: **Fertility with Butterfly.** Total positive curvature. $\sigma = 3.0568796$

10.3.6 Kitten

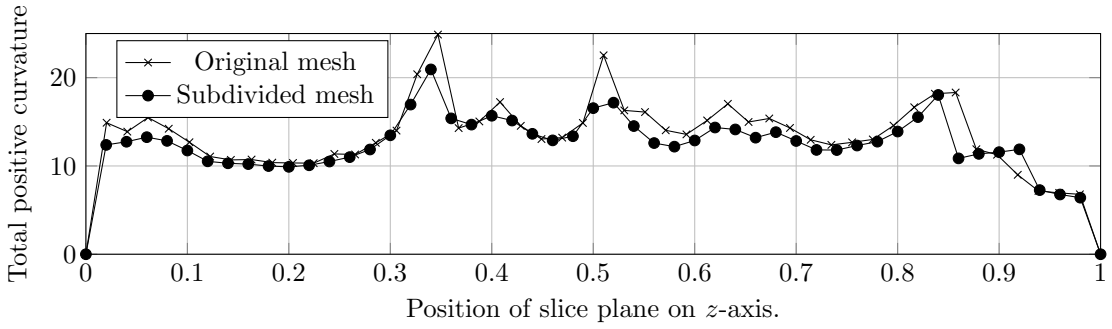


Figure 57: **Kitten with Loop.** Total positive curvature. $\sigma = 1.5852304$

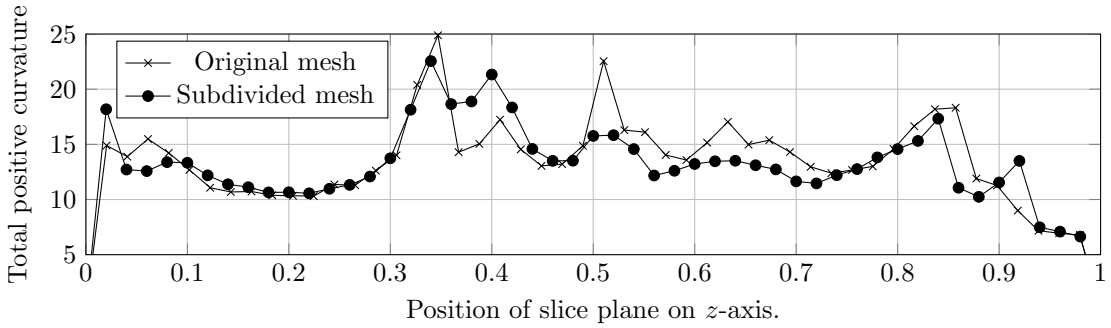


Figure 58: **Kitten with $\sqrt{3}$.** Total positive curvature. $\sigma = 1.858044$

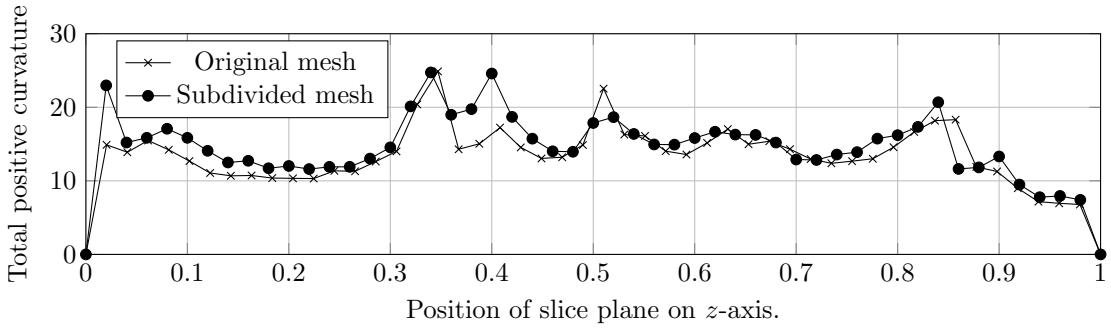


Figure 59: **Kitten with int. $\sqrt{3}$.** Total positive curvature. $\sigma = 2.4117896$

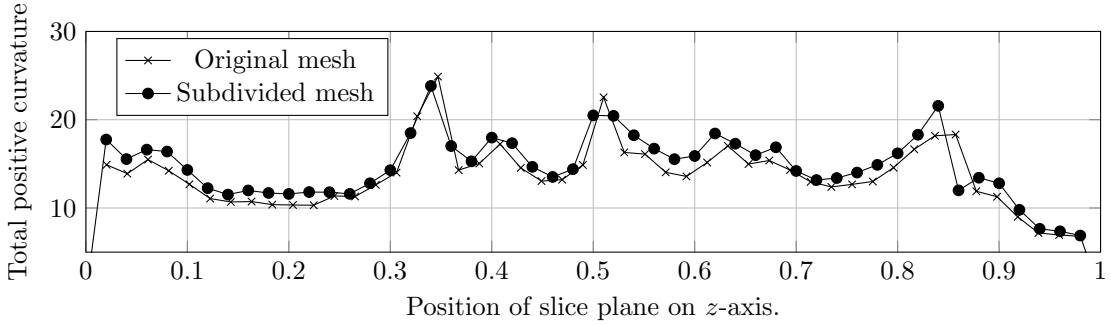


Figure 60: **Kitten with Butterfly.** Total positive curvature. $\sigma = 2.20154$

10.3.7 Moai

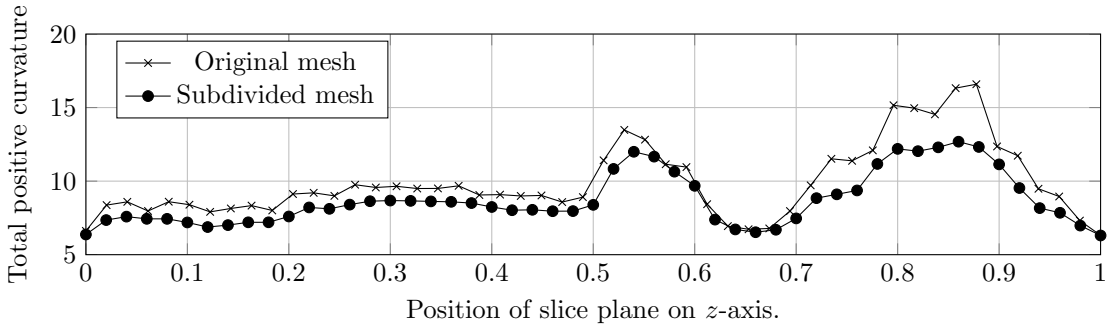


Figure 61: **Moai with Loop.** Total positive curvature. $\sigma = 0.5406688$

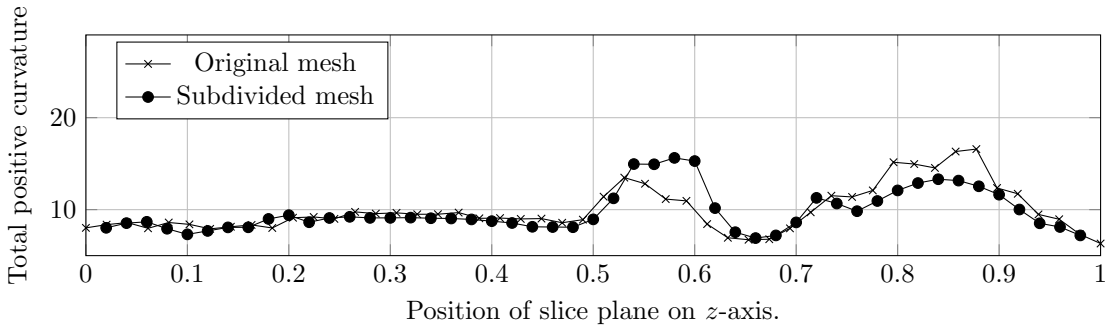


Figure 62: **Moai with $\sqrt{3}$.** Total positive curvature. $\sigma = 0.7991141667$

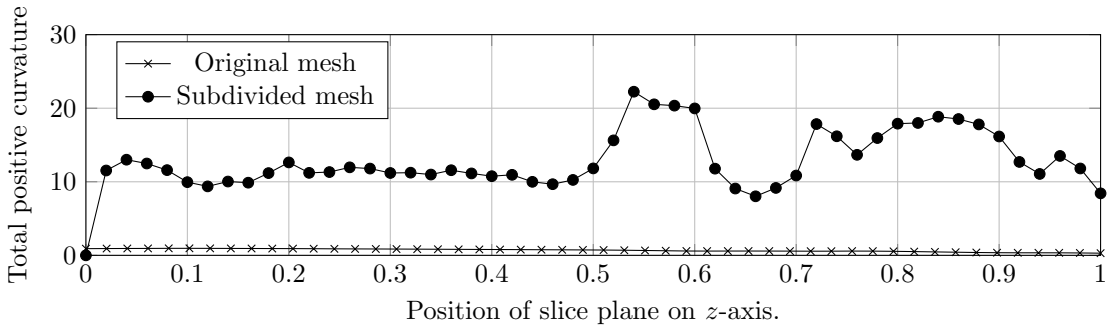


Figure 63: **Moai with int. $\sqrt{3}$.** Total positive curvature. $\sigma = 1.7513812$

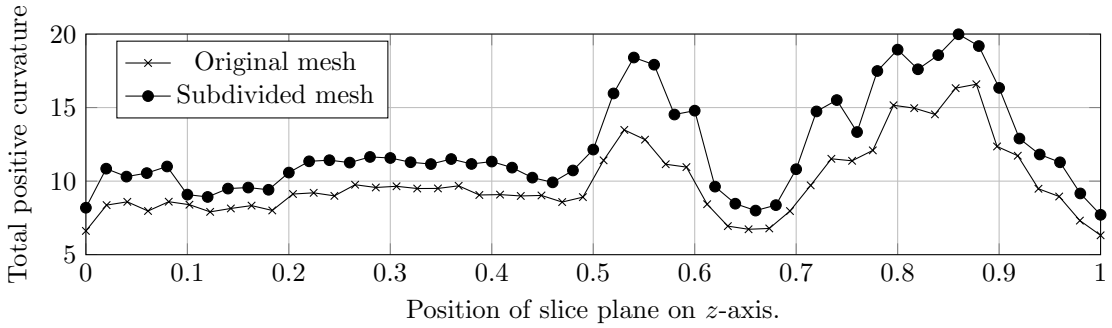


Figure 64: **Moai with Butterfly.** Total positive curvature. $\sigma = 1.2544252$

10.4 Non-overlapping area

10.4.1 "-" to "+"

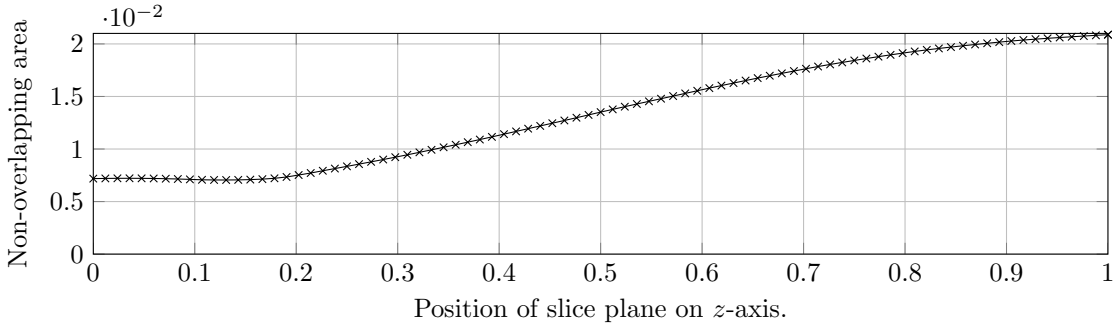


Figure 65: **"-" to "+" with Loop.** Non-overlapping area slice. $\bar{x} = 0.01362278279$

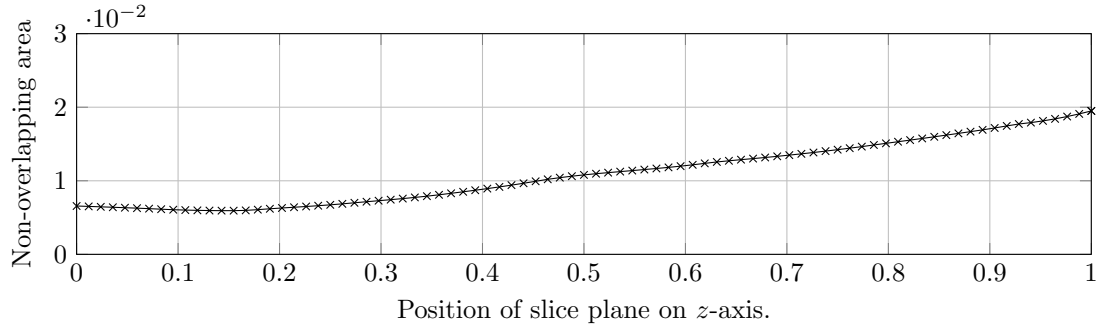


Figure 66: **"-" to "+" with $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.01109334047$

Mesh is self intersecting after subdivision.

Figure 67: **"-" to "+" with int. $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = n.a.$

Mesh is self intersecting after subdivision.

Figure 68: **"-" to "+" with Butterfly.** Non-overlapping area slice. $\bar{x} = n.a.$

10.4.2 "o" to "o"

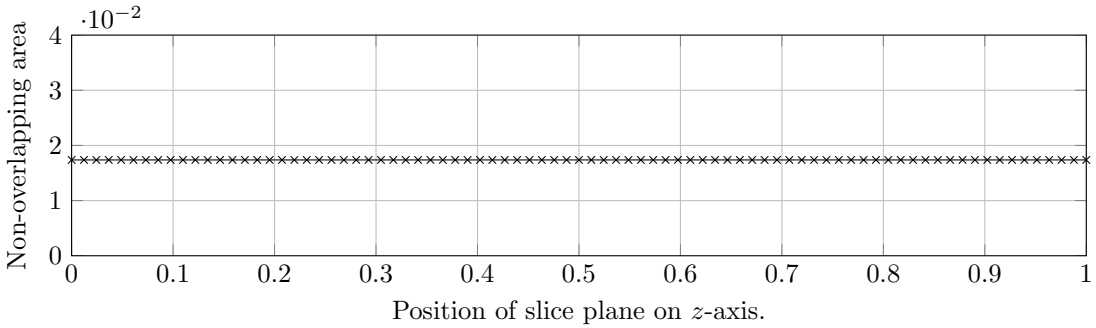


Figure 69: **"o" to "o" with Loop.** Non-overlapping area slice. $\bar{x} = 0.0173711$

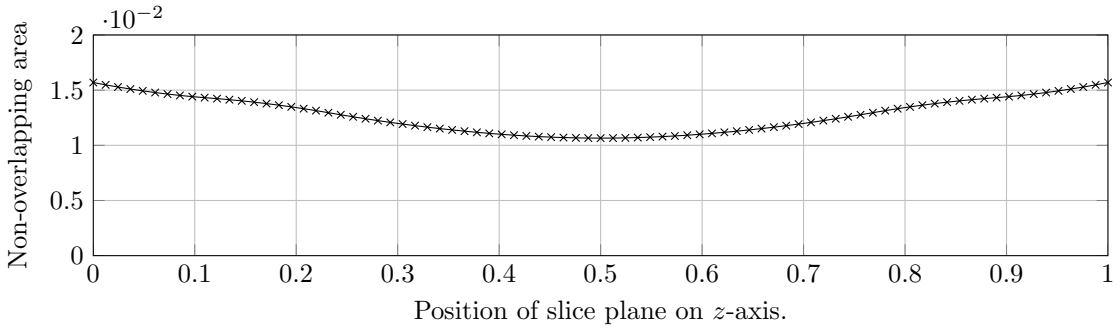


Figure 70: **"o" to "o" with $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.01280360361$

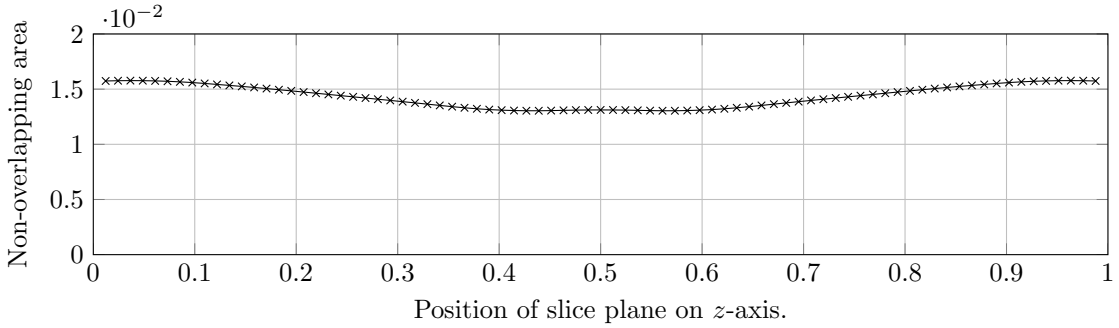


Figure 71: **"o" to "o" with int. $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.01434957778$

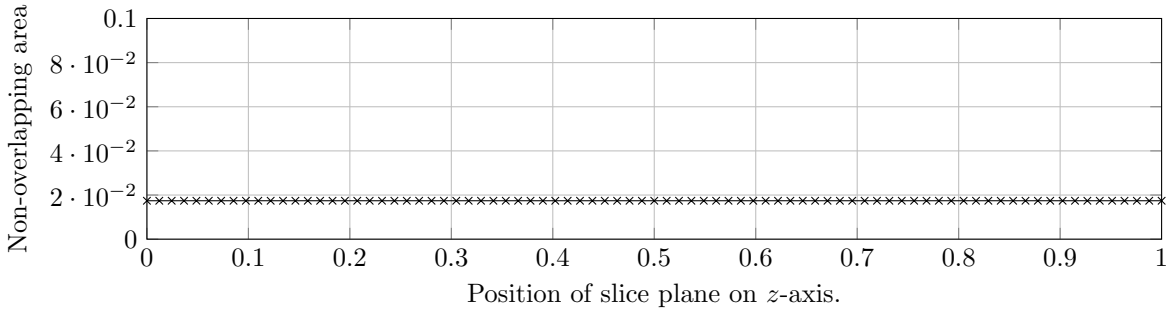


Figure 72: **"o" to "o" with Butterfly.** Non-overlapping area slice. $\bar{x} = 0.017413$

10.4.3 "U" to "V"

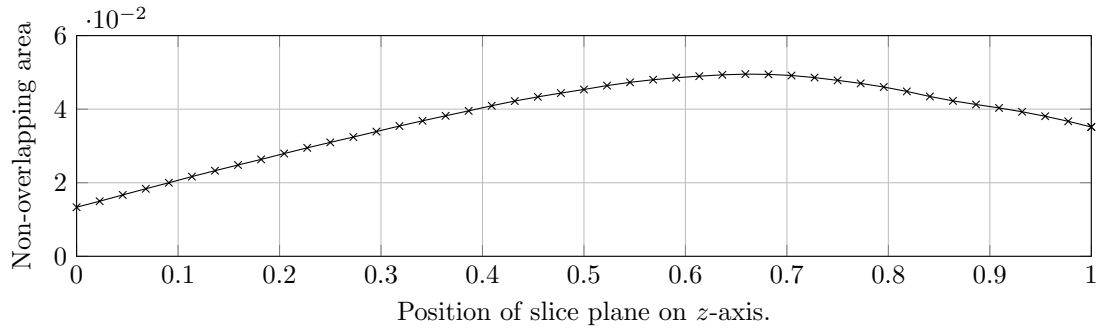


Figure 73: **"U" to "V" with Loop.** Non-overlapping area slice. $\bar{x} = 0.03745941087$

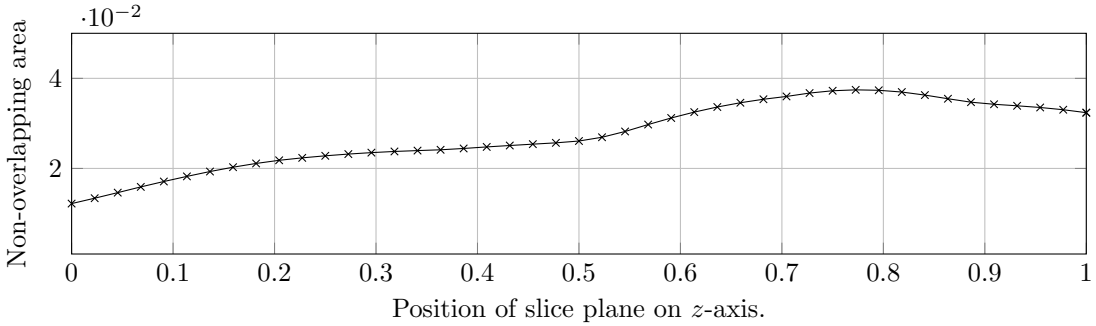


Figure 74: **"U" to "V" with $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.02758521304$

Mesh is self intersecting after subdivision.

Figure 75: **"U" to "V" with int. $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = n.a.$

Mesh is self intersecting after subdivision.

Figure 76: **"U" to "V" with Butterfly.** Non-overlapping area slice. $\bar{x} = n.a.$

10.4.4 Elephant

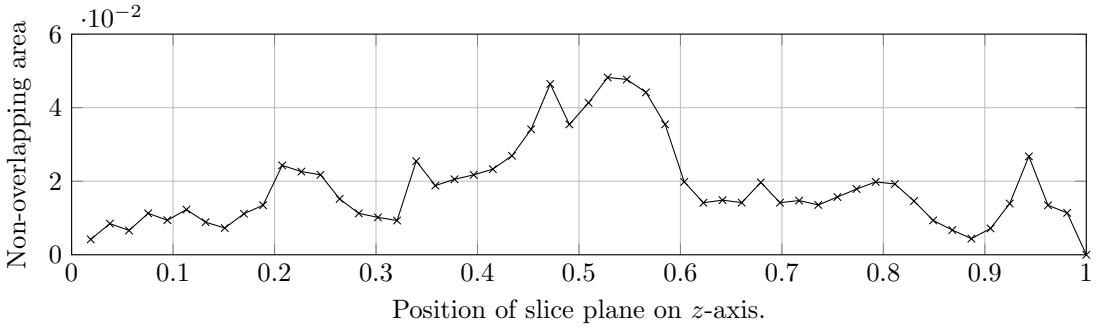


Figure 77: **Elephant with Loop.** Non-overlapping area slice. $\bar{x} = 0.01854460736$

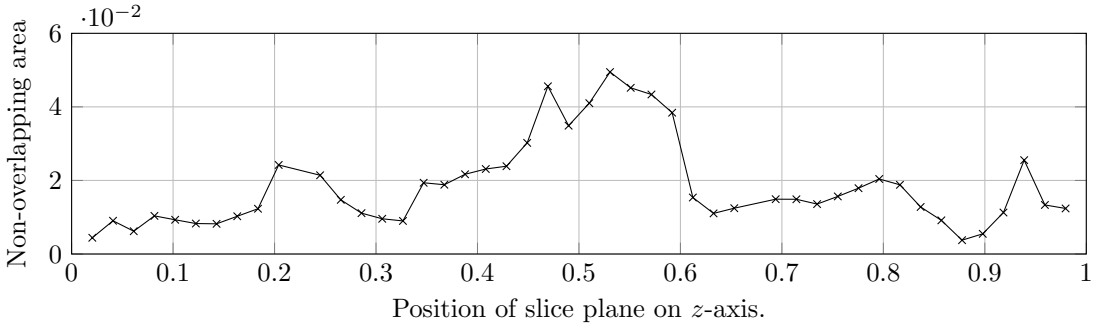


Figure 78: **Elephant with $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.01852367283$

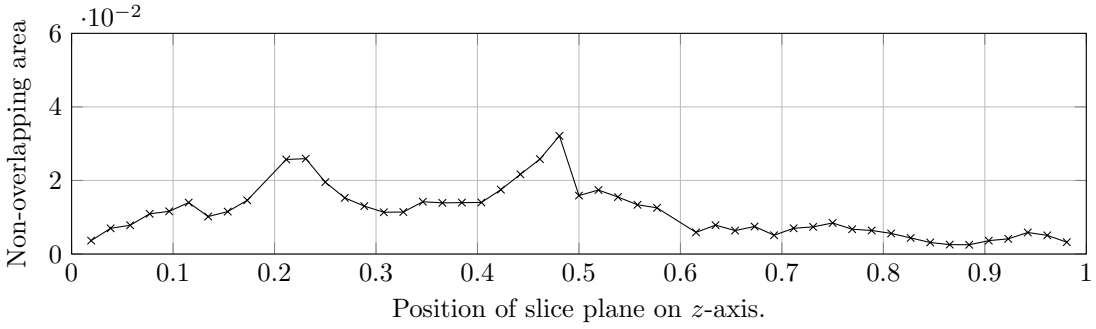


Figure 79: **Elephant with int. $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.01123209388$

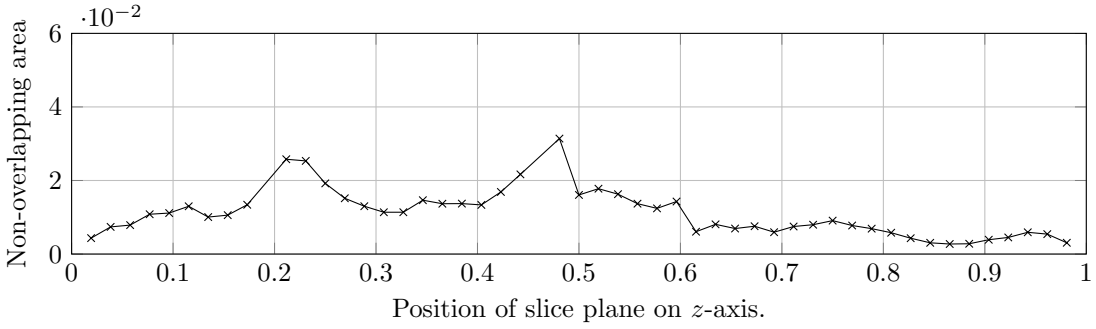


Figure 80: **Elephant with Butterfly.** Non-overlapping area slice. $\bar{x} = 0.01104177102$

10.4.5 Fertility

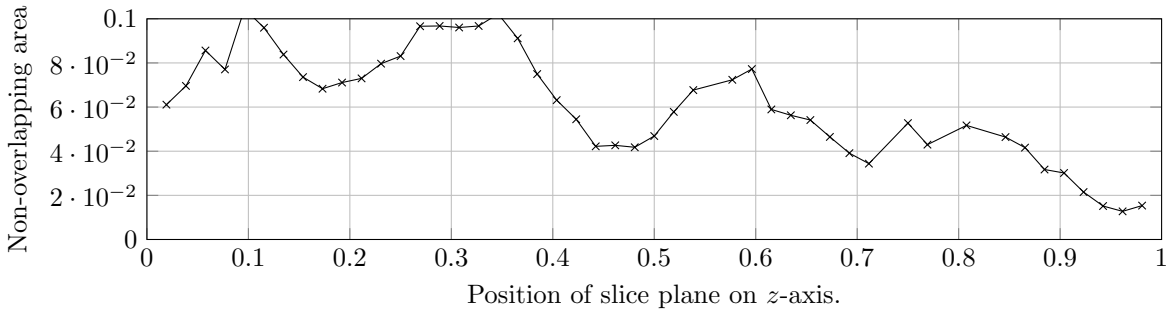


Figure 81: **Fertility with Loop.** Non-overlapping area slice. $\bar{x} = 0.06167092128$

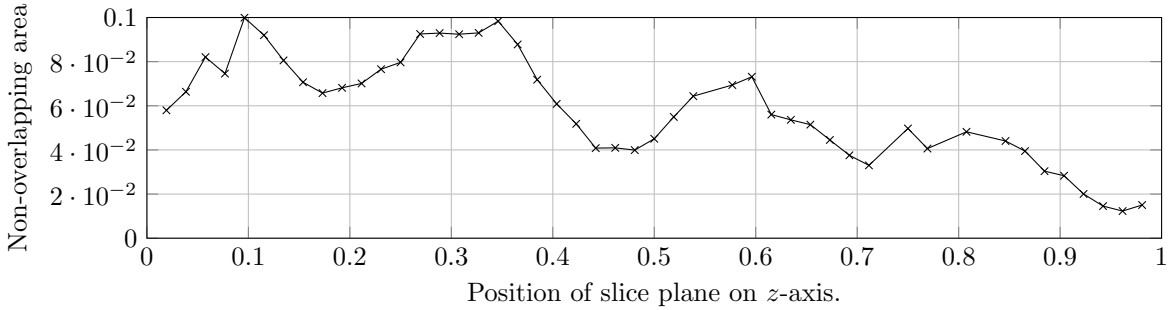


Figure 82: **Fertility with $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.05901141489$

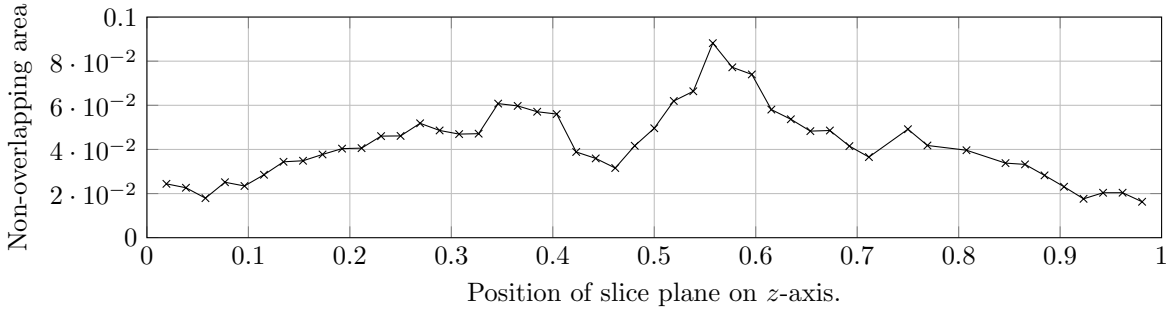


Figure 83: **Fertility with int. $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.042207975$

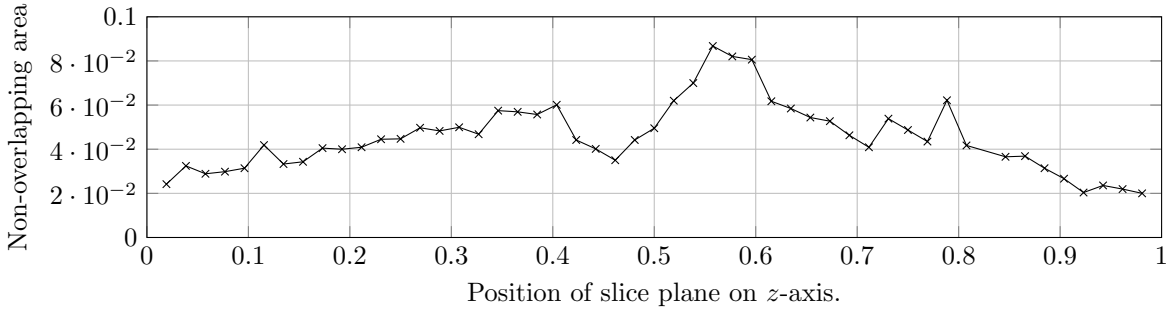


Figure 84: **Fertility with Butterfly.** Non-overlapping area slice. $\bar{x} = 0.00587505102$

10.4.6 Kitten

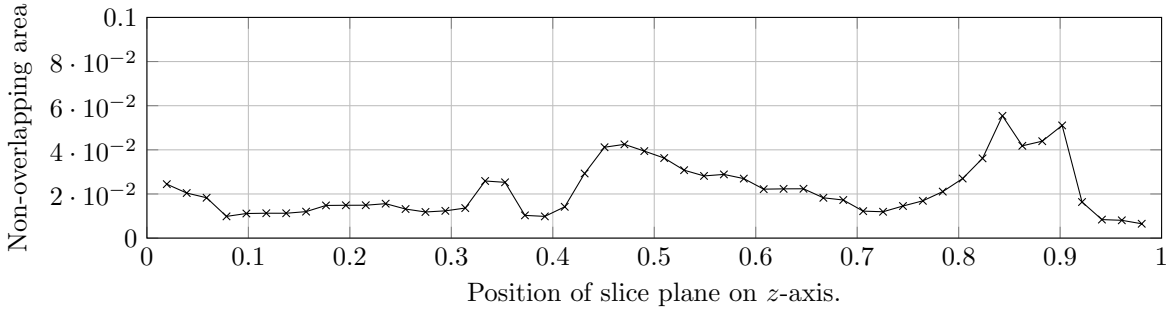


Figure 85: **Kitten with Loop.** Non-overlapping area slice. $\bar{x} = 0.0218482086$

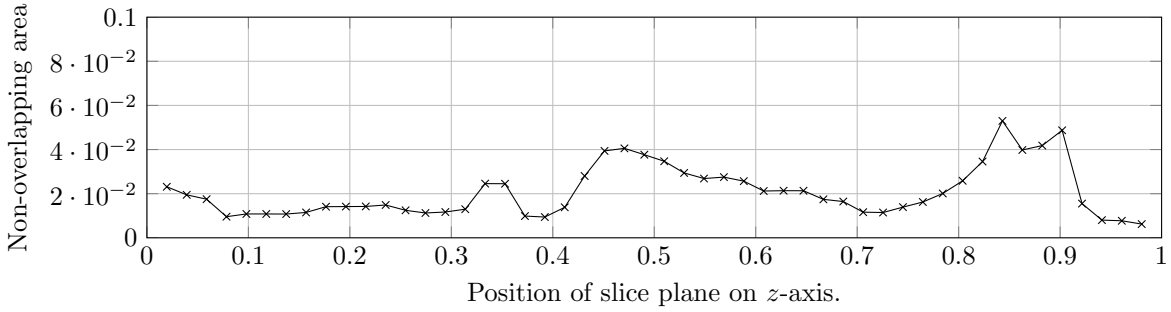


Figure 86: **Kitten with $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.0208735474$

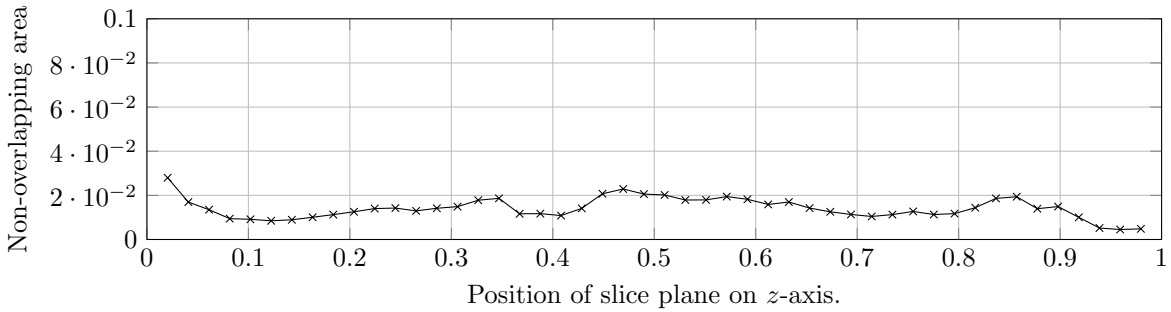


Figure 87: **Kitten with int. $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.01407313688$

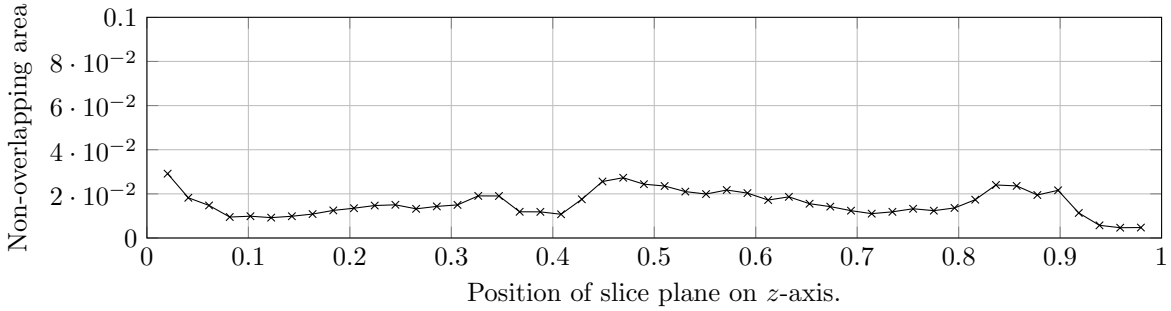


Figure 88: **Kitten with Butterfly.** Non-overlapping area slice. $\bar{x} = 0.01575873208$

10.4.7 Moai

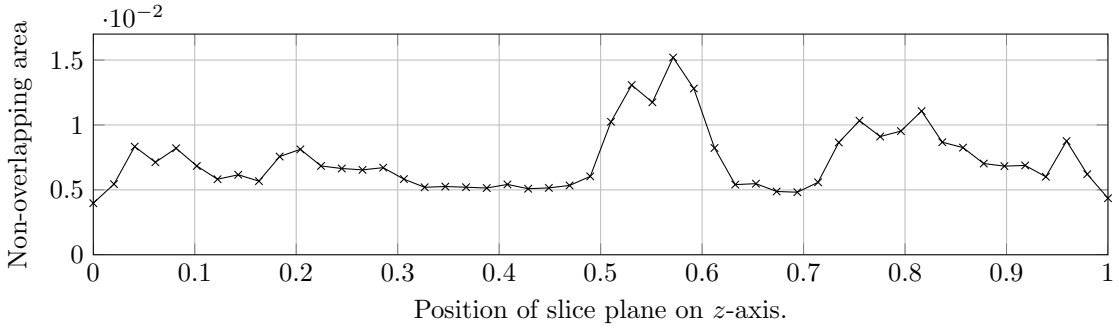


Figure 89: **Moai with Loop.** Non-overlapping area slice. $\bar{x} = 0.0072563662$

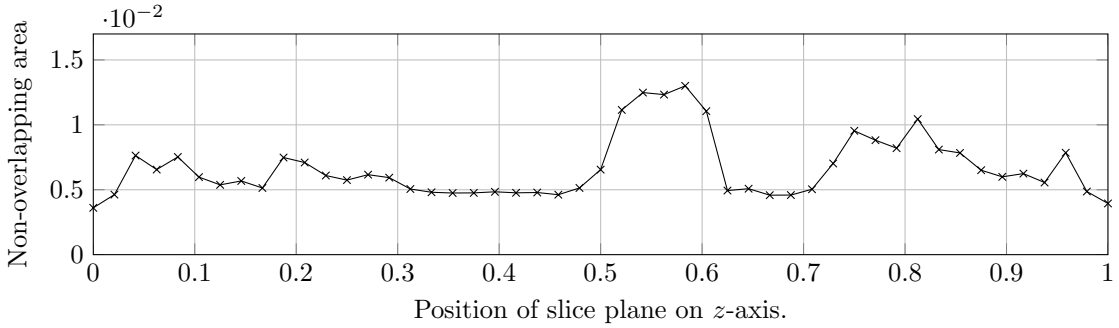


Figure 90: **Moai with $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.006650298776$

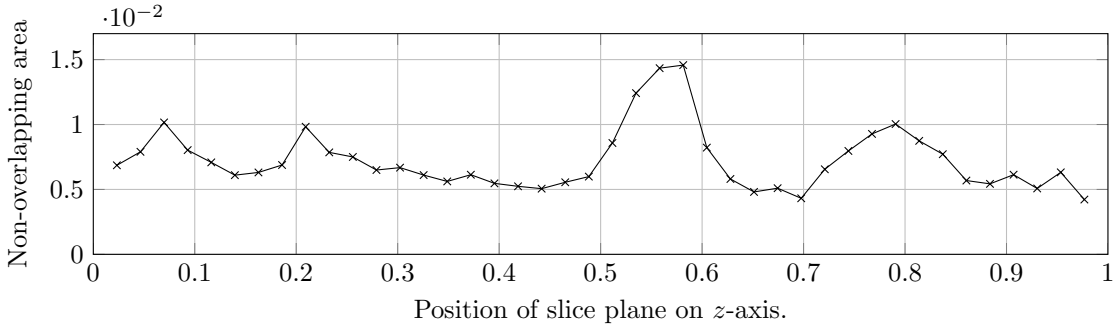


Figure 91: **Moai with int. $\sqrt{3}$.** Non-overlapping area slice. $\bar{x} = 0.007241229286$

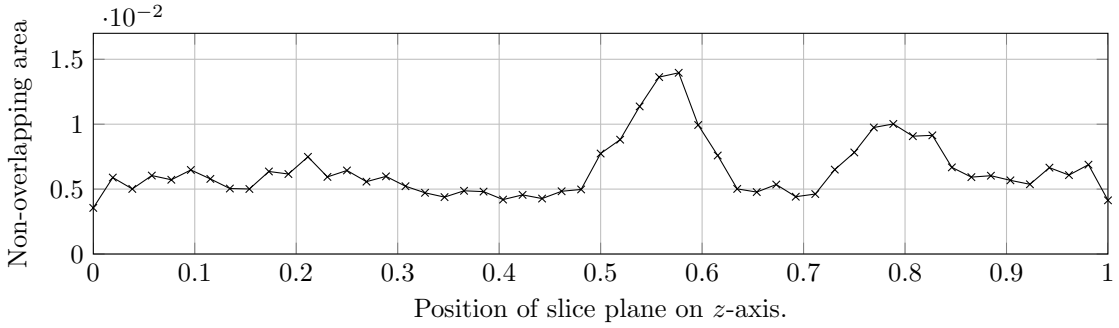


Figure 92: **Moai with Butterfly.** Non-overlapping area slice. $\bar{x} = 0.006456916415$

10.5 Offset non-overlapping area

10.5.1 Elephant

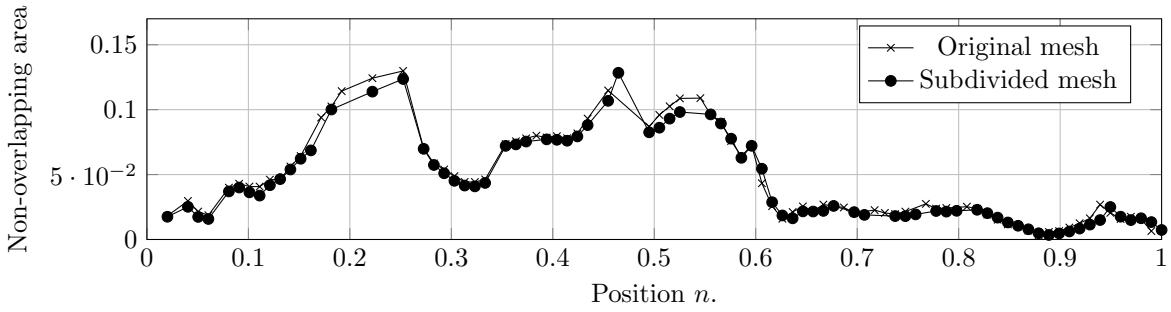


Figure 93: **Elephant with Loop.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.007322120921$

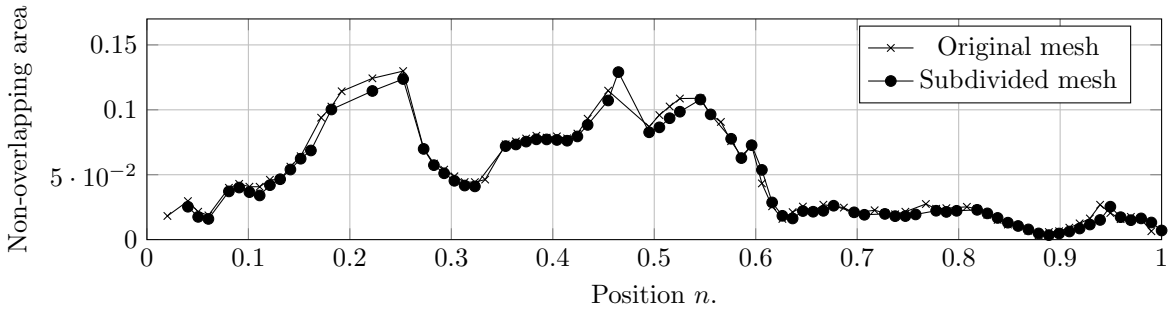


Figure 94: **Elephant with $\sqrt{3}$.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.007560879211$

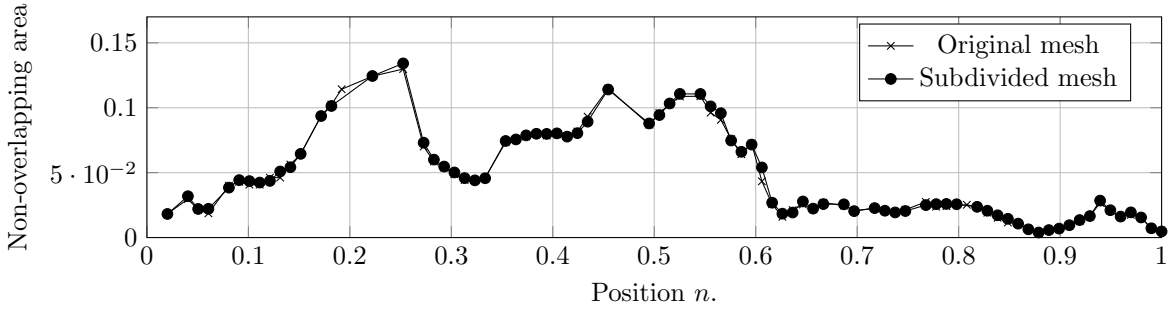


Figure 95: **Elephant with int. $\sqrt{3}$.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.007306004231$

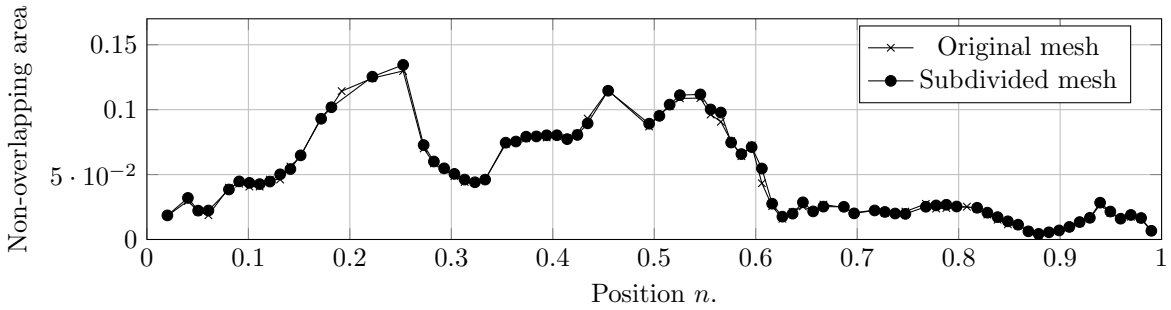


Figure 96: **Elephant with Butterfly.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.00743124039$

10.5.2 Fertility

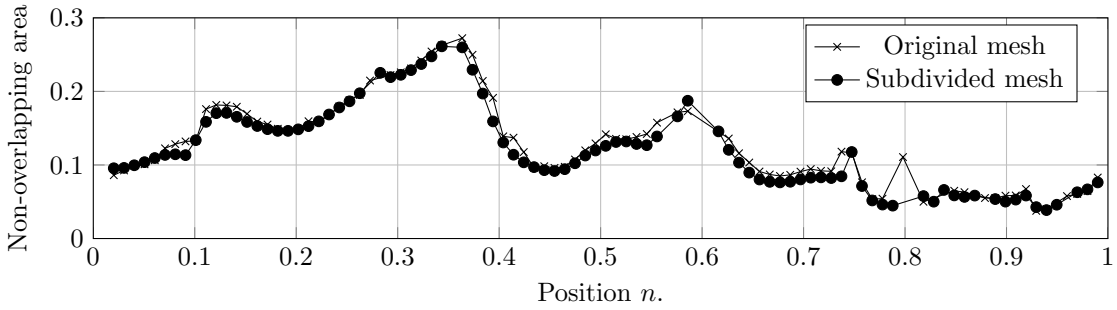


Figure 97: **Fertility with Loop.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.009798018391$

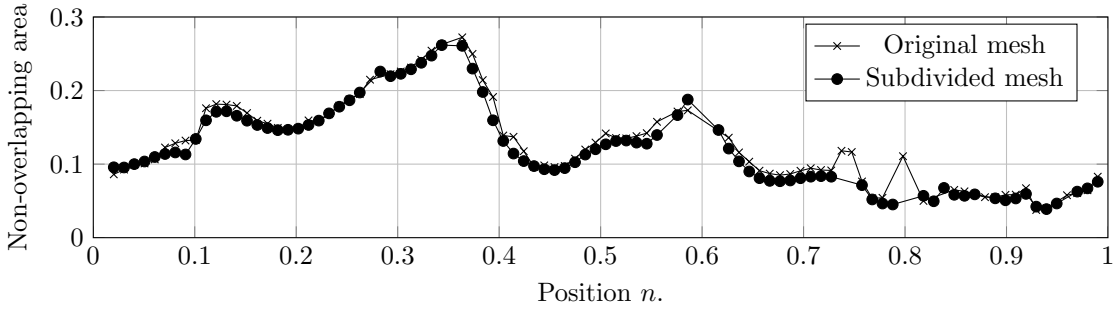


Figure 98: **Fertility with $\sqrt{3}$.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.009311483529$

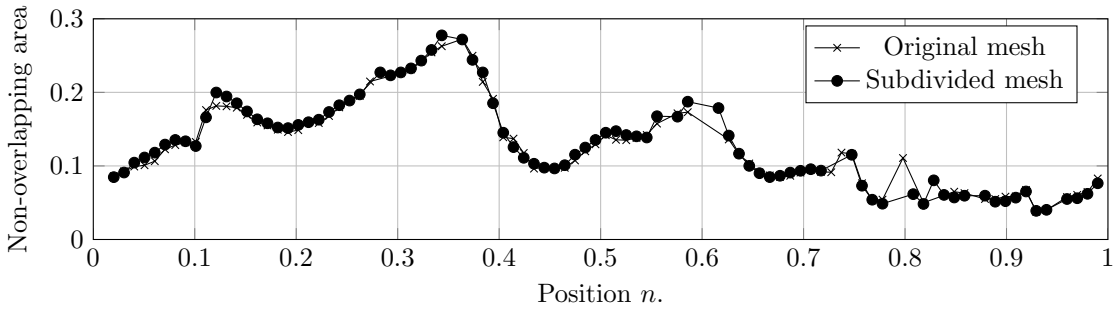


Figure 99: **Fertility with int. $\sqrt{3}$.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.01150236706$

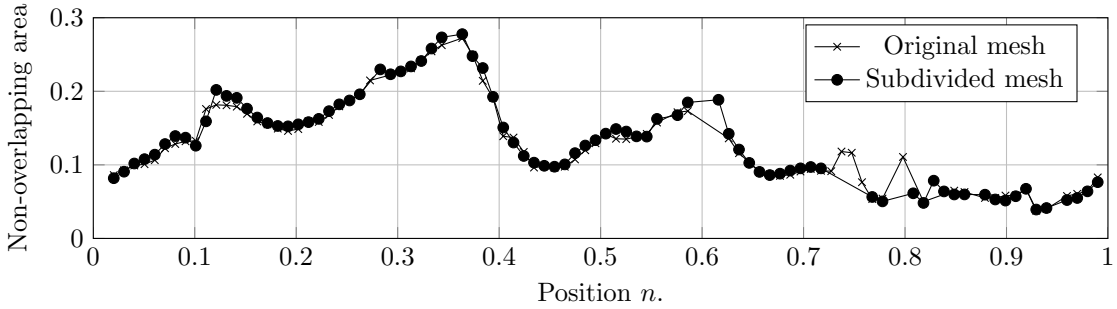


Figure 100: **Fertility with Butterfly.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.01143622771$

10.5.3 Kitten

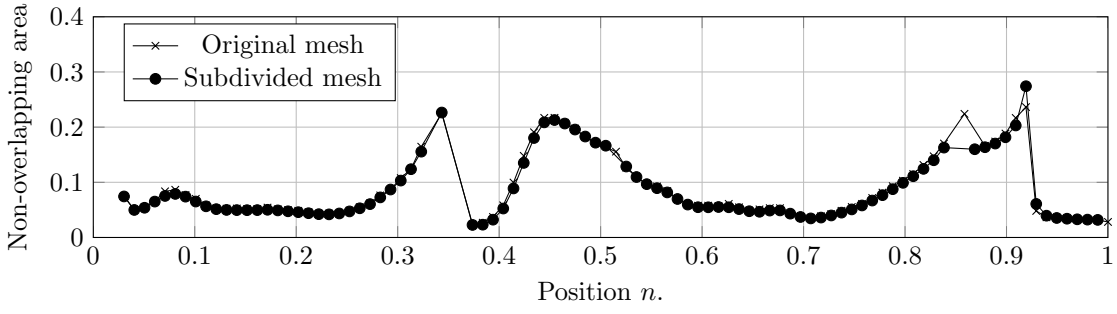


Figure 101: **Kitten with Loop.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.01508217079$

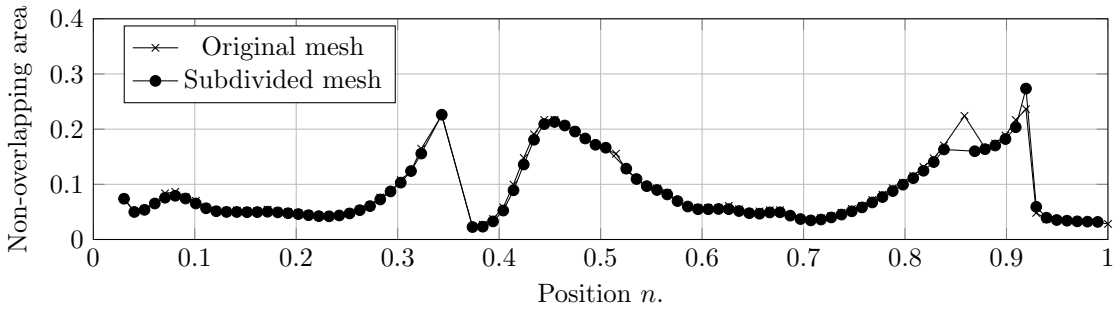


Figure 102: **Kitten with $\sqrt{3}$.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.0150788382$

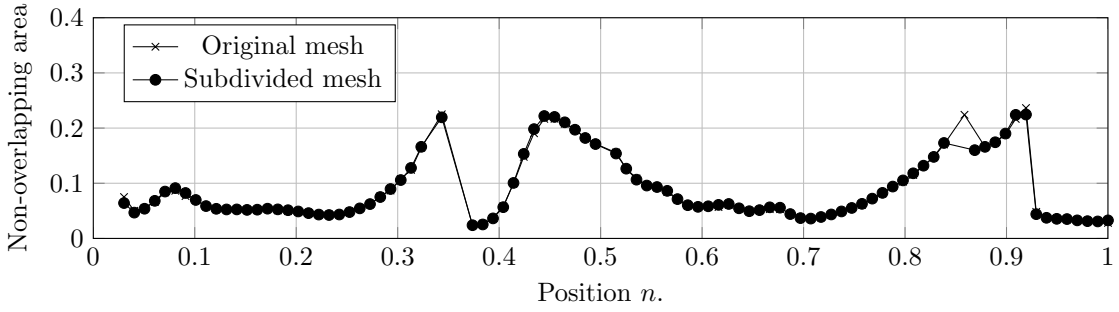


Figure 103: **Kitten with int. $\sqrt{3}$.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.01451244556$

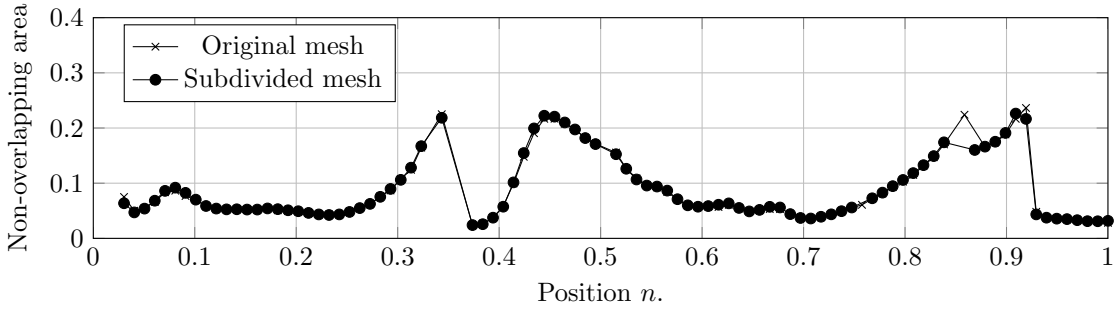


Figure 104: **Kitten with Butterfly.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.01475344045$

10.5.4 Moai

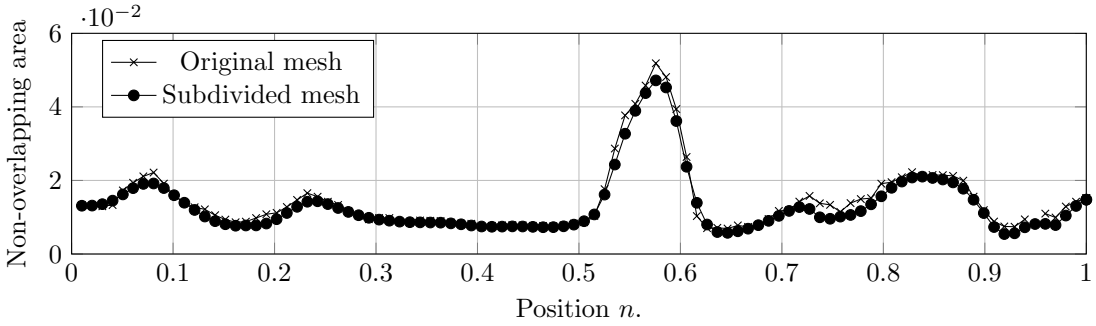


Figure 105: **Moai with Loop.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.00161646202$

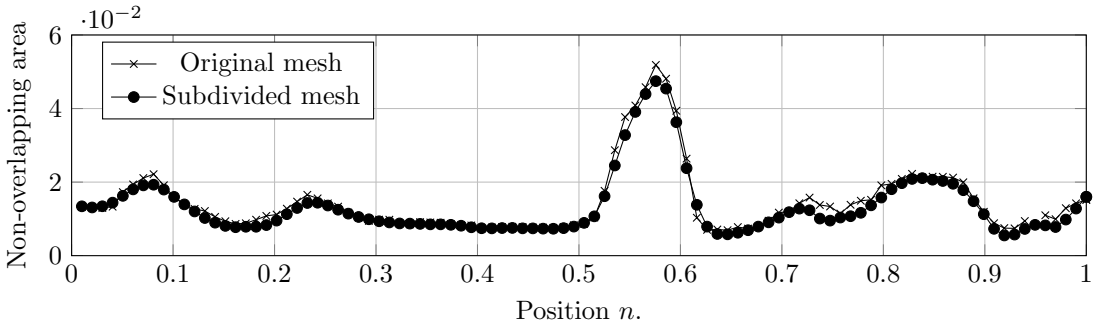


Figure 106: **Moai with $\sqrt{3}$.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.00164473596$

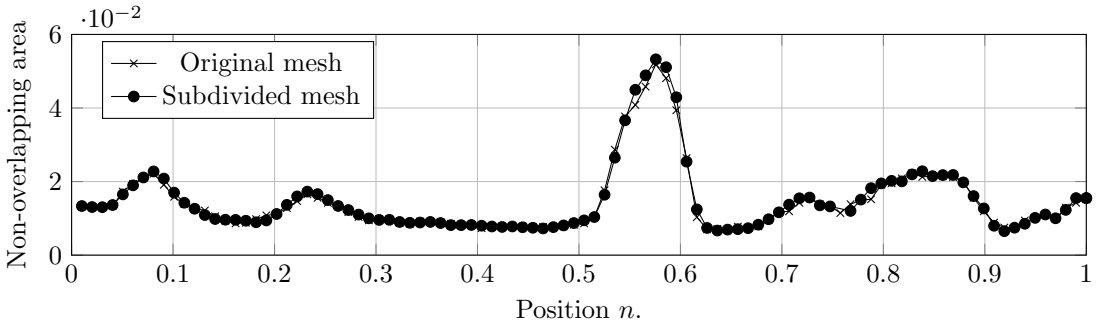


Figure 107: **Moai with int. $\sqrt{3}$.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.001910749184$

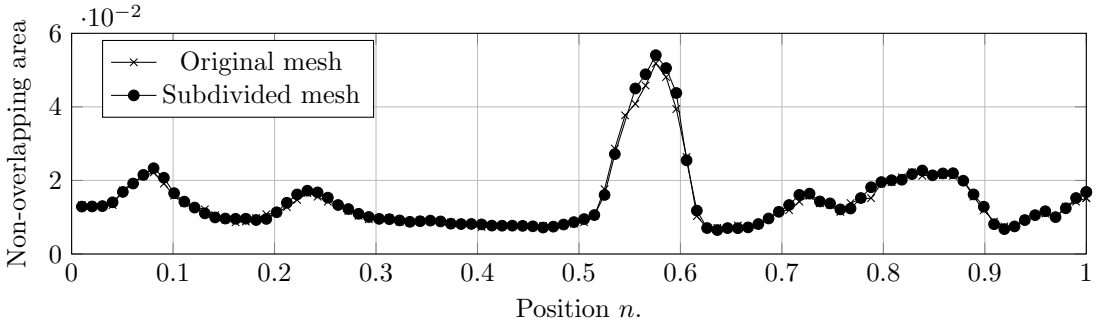
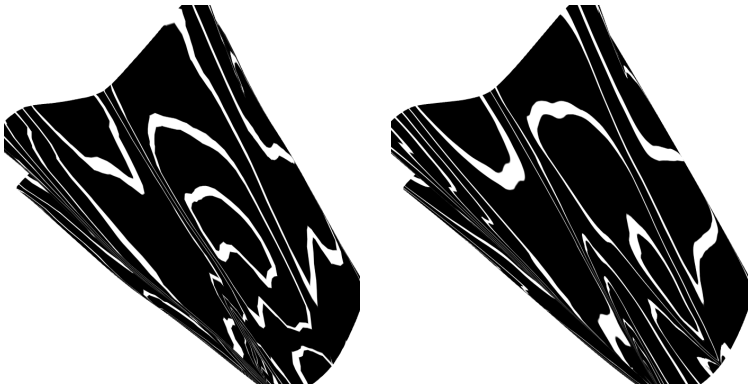


Figure 108: **Moai with Butterfly.** *non-overlapping area* of slice at position n and slice at position $n-1$. $\sigma = 0.001967704286$

10.6 Reflection lines

10.6.1 "-" to "+"

Original | Loop subdivision surface | $\sqrt{3}$ subdivision surface



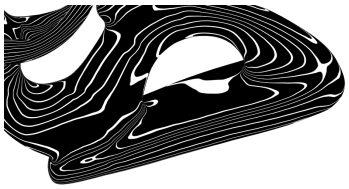
Int. $\sqrt{3}$ subdivision surface | Butterfly subdivision surface

Table 44: **"-" to "+".** Reflection lines

10.6.2 Fertility

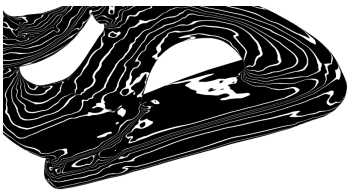
Original

Loop subdivision surface



$\sqrt{3}$ subdivision surface

Int. $\sqrt{3}$ subdivision surface

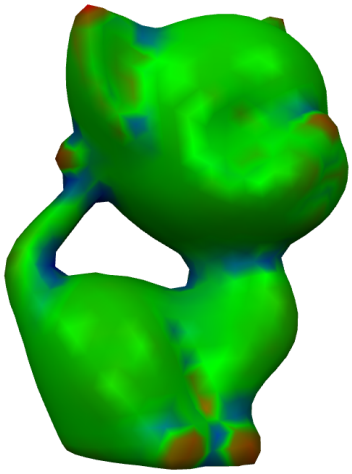


Butterfly subdivision surface

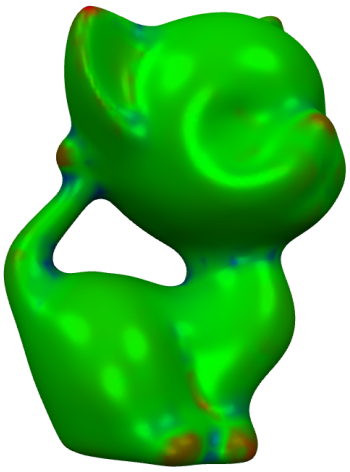
Table 45: **Fertility.** Reflection lines

10.7 Gaussian curvature

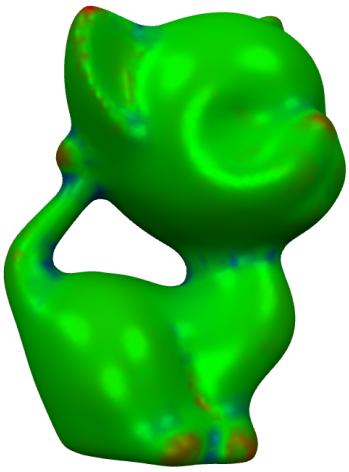
10.7.1 Kitten



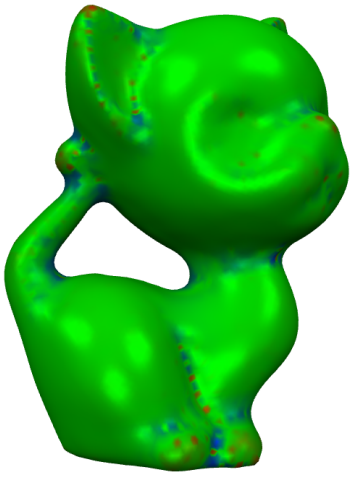
Original mesh
Range: -0.7956 0.9004



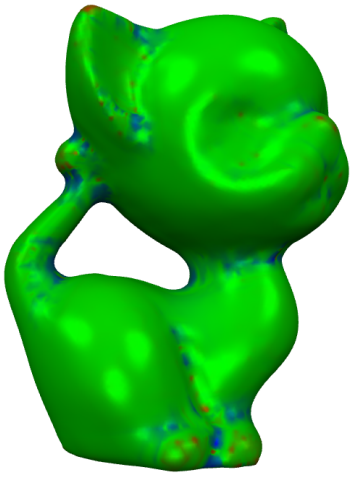
Loop subdivision surface
Range: -0.1170 0.0784



$\sqrt{3}$ subdivision surface
Range: -0.1996 0.1610



Int. $\sqrt{3}$ subdivision surface
Range: -0.2208 0.4234

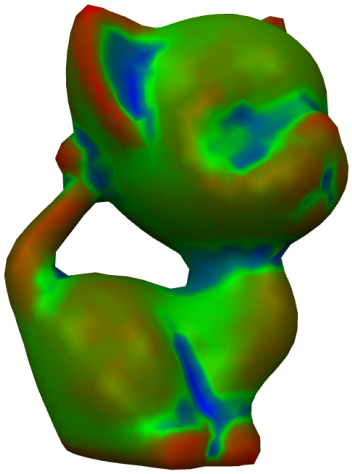


Butterfly subdivision surface
Range: -0.1123 0.2774

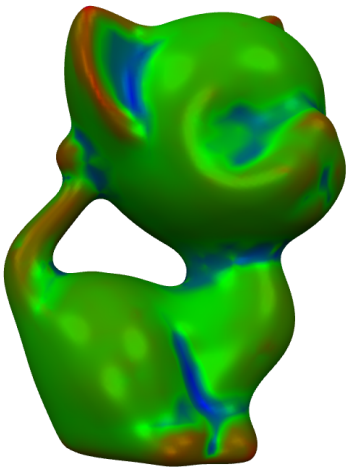
Table 46: **Kitten.** Gaussian curvature

10.8 Mean curvature

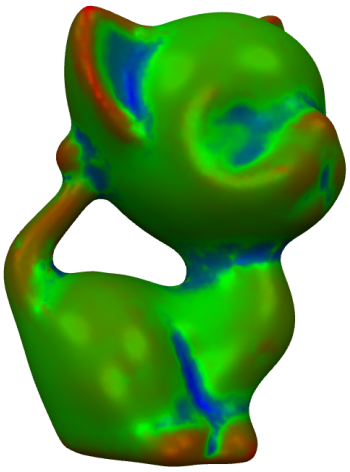
10.8.1 Kitten



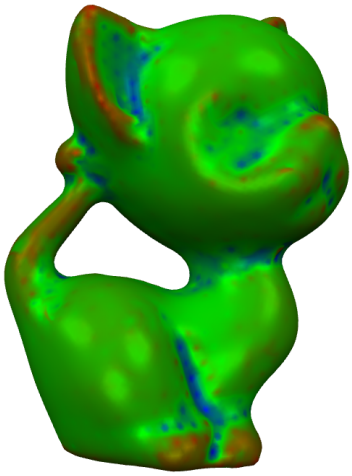
Original mesh
Range: -5.5881 8.1097



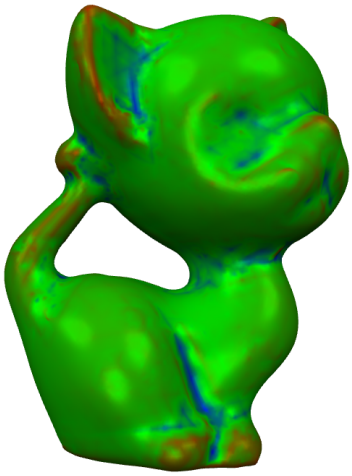
Loop subdivision surface
Range: -7.6961 14.8876



$\sqrt{3}$ subdivision surface
Range: -6.9873 12.2230



Int. $\sqrt{3}$ subdivision surface
Range: -15.2711 18.6489



Butterfly subdivision surface
Range: -18.2241 23.7869

Table 47: **Kitten.** Mean curvature