"University of Turku Technical Reports, No.3 — August 2015"

3D MESH SIMPLIFICATION

A survey of algorithms and CAD model simplification tests

Jukka Arvo | Antti Euranto | Lauri Järvenpää | Teijo Lehtonen | Timo Knuutila

Jukka Arvo
University of Turku, Technology Research Center, 20014 Turun yliopisto, Finland
jukka.arvo@utu.fi

Antti Euranto
University of Turku, Technology Research Center, 20014 Turun yliopisto, Finland
antti.euranto@utu.fi

Lauri Järvenpää
University of Turku, Technology Research Center, 20014 Turun yliopisto, Finland
lauri.jarvenpaa@utu.fi

Teijo Lehtonen
University of Turku, Technology Research Center, 20014 Turun yliopisto, Finland
teijo.lehtonen@utu.fi

Timo Knuutila
University of Turku, Technology Research Center, 20014 Turun yliopisto, Finland
timo.knuutila@utu.fi

www.trc.utu.fi

ISSN 2341-8028 | ISBN: 978-951-29-6202-0

Technology Research Center

Turun yliopisto
University of Turku

Abstract

Simplification of highly detailed CAD models is an important step when CAD models are visualized or by other means utilized in augmented reality applications. Without simplification, CAD models may cause severe processing and storage issues especially in mobile devices. In addition, simplified models may have other advantages like better visual clarity or improved reliability when used for visual pose tracking. The geometry of CAD models is invariably presented in form of a 3D mesh. In this paper, we survey mesh simplification algorithms in general and focus especially to algorithms that can be used to simplify CAD models. We test some commonly known algorithms with real world CAD data and characterize some new CAD related simplification algorithms that have not been surveyed in previous mesh simplification reviews.

Keywords

3D mesh simplification, CAD models, computer graphics, Autodesk Maya, Blender, Autodesk 3dsMax, Simplygon, MeshLab

Contents

1 Introduction

Highly detailed polygonal CAD models are a common representation of 3D design data in many computer graphics and engineering applications. Commercial graphics hardware accelerators can render triangulated polygonal models efficiently and are already available on most mobile devices. Nevertheless, complex polygonal CAD models can still cause storage and performance issues on mobile devices due to their limited computational and storage resources. Therefore, complex meshes are typically simplified before they can be used in mobile applications.

Model complexity does not cause problems only to rendering, but similarly to many other applications that utilize 3D models. As an example, physics simulation plays an important role in product design process. The simulations are based on product CAD models. Model simplification has been reported to increase physical simulation performance by several magnitudes with negligible impact on the results [P95]. Further, the performance of collision detection of modelled objects (detecting intersections of objects) can be improved by simplification [P81, P107]. A specific issue in collision detection are small holes and other similar details that are frequent in mechanical devices. The holes are irrelevant from the collision detection point of view, but can increase the processing time remarkably.

Visual model-based tracking is still one example of those application areas, where model complexity causes problems. An essential part of the tracking is the detection of model features from a video camera image. Attempts to detect small model details that are below the image resolution waste processing power and may lead to false identifications.

Mesh compression is yet another application area that can utilize mesh simplification. The simplification alone is a lossy mesh compression technique. However, more often mesh simplification, especially making the mesh more regular, is used as a preparation step for the actual compression procedure, because mesh coding techniques benefit from a regular mesh structure.

Hierarchies of simplified mesh representations are useful in many applications. A level-of-detail (LOD) approach is frequently used in real-time rendering applications to save rendering time [P27]. In this approach, highly detailed representations are used when rendering the mesh from a short distance, whereas strongly simplified representations can be used in longer distances without compromising the image quality. A progressive mesh is a continuous mesh hierarchy that consists of a coarse base mesh and records that allow stepwise restoration of the original, highly detailed mesh. Progressive meshes can be used, for example, in interactive applications to

quickly show a sketch of the mesh, which is then gradually completed to a detailed view.

One interesting aspect of mesh simplification is its influence on the rendering quality of the mesh. In some cases, the simplification impairs the quality, but sometimes the situation is just the opposite. Namely, models may contain a large number of details that are not relevant for the purpose of the use. In these cases, the removal of the irrelevant details rather improves the visual appearance than dispairs it.

Model simplification has been a popular research topic for decades. One of the reasons to that is that simplification techniques are usually application specific, which provides nearly endless chances to vary the techniques and to adapt them to new use cases. Therefore, a huge number of articles have been written about simplification. Still, most of the methods can be classified to a few categories that are presented later in this paper.

Several earlier surveys have been published about mesh simplification, for example: [P11, P21, P23, P32, P47, P52, P75, P12, P45, P84, P95, P102, P103, P105]. Simplification continues to be an active research topic, hence it is worth to review the situation once again.

The contribution of this paper is twofold. First, the mesh simplification problem is formulated and the basic concepts and building blocks of mesh simplification are presented. Second, relevant simplification methods found from literature are surveyed, classified into a taxonomy, and then some results of example implementations are given in the context of real world CAD models. This paper is not a full overview of the large field of polygonal model simplification. In particular, the main focus of this paper is in basic mesh simplification methods as well as methods that are applicable to CAD models. Other mesh simplification related papers are not surveyed in this report.

2 Research question and research methods

2.1 Rationale and research question

This survey was done as part of the MARIN2 project, in which mesh simplification plays an important role. The objective of the survey was to achieve an understanding of the current state of art of mesh simplification research. The exact project specific simplification requirements were not known at the time of the report writing. Hence, it was essential to survey the methods extensively to have a solid base for the continuation work.

The main problem in the review was the big amount of the published mesh processing material. It was essential to strictly limit the survey only to mesh simplification methods and especially only to those methods that could potentially be utilized in CAD model simplification. Excluded topics included, for example, mesh compressing, mesh smoothing and parting line methods, only to mention some. Compression, meaning the representation of a mesh with as little storage space as possible, was somewhat problematic to delimit from the survey, because some methods that are primarily purposed for simplification can also be used for compression and vice versa.

2.2 Research method

The research was performed by searching for simplification publications from certain electronic scientific libraries. The searching was done using keywords that supposedly characterize simplification publications.

The research was started with some trial searches. They revealed some potential issues:

- Risk of excessive amount of search results
- Untypical terms used instead of e.g. "simplification" in some publications
- Limitations of search capabilities in certain electronic libraries

The first issue forced us to formulate the query string in a restrictive way, with the risk that some relevant publications could be missed. The second issue was solved by including several potential terms to the query string, but, of course, some rarely used terms might have been overlooked. The last issue forced us to use somewhat different search strategies in different libraries. In consequence, the search results from different libraries are not totally comparable.

The actual review was accomplished by searching the named scientific libraries in October and November 2014. The search strings were somewhat varied during the work as more experience was gained about the best searching strategies. Likewise, the differences between the capabilities of the libraries forced to vary the query strings in some cases. The search strings are listed in appendix B.

Next, all the found publications were cursory reviewed and those publications that were out of the scope of review, were eliminated. The remaining publications were evaluated according to their relevance to the review. The objective was to have publications from different sectors of mesh simplification area so that a reader would get an extensive overview of the area. Finally, detected shortages in the coverage of the publications were filled by including also some referenced publications and publications found from google-searches.

3 Basics

3.1 Problem statement

Formally, a mesh is a pair $M = (V, F)$, where V is a set of input coordinates, e.g., (x, y, z) positions, and F is a set of input faces, e.g., connectivity information of triangle vertices. Mesh simplification means one of the following two search problems [P23]:

$M = (V, F) \rightarrow M' = (V', F')$, where

1. $|V'| = n < |V|$ and $\| M - M' \|$ is minimal

2. $\| M - M' \| < \epsilon$ and $|V'|$ is minimal

The goal of search problem 1. in the above is to minimize the error between the original mesh and the simplified mesh when a size constraint n for simplified mesh is given. Correspondingly the search problem 2. uses an error bound ϵ to minimize the mesh size that no point in the simplified mesh is farther than ϵ distance from the original model.

3.2 Computational complexity

It has been shown in the computational geometry literature that computing a minimum complexity approximation is a NP-hard problem for convex polytopes [P24] and polyhedral terrains [P1]. This denotes that it is not possible to search for an exact solution to real mesh simplification problems. Instead, almost all mesh simplification algorithms search for polynomial-time approximations that are close to the optimal solution.

Even searching for an approximately solution by treating a mesh as a whole is not trivial. Therefore, many mesh simplification methods use relatively simple local heuristics. CAD models are typically very large and theoretical time complexity of mesh simplification algorithms should not exceed $O(NlogN)$ in practical applications, where N is the number of mesh vertices.

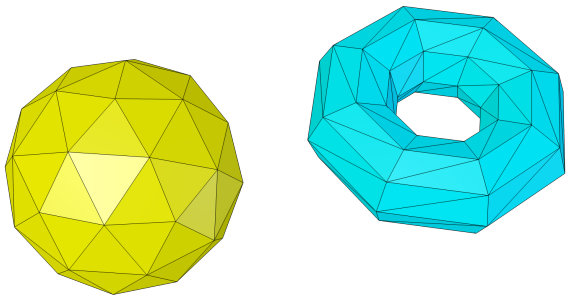


Figure 1: A mesh (sphere) with genus 0 and a mesh (donut) with genus 1

3.3 Topology

Mesh topology refers to the structure of the mesh and has two meanings. *Local topology* is the geometric structure surrounding a particular face or vertex, whereas *global topology* is the geometric structure of the whole polygonal mesh [P32].

Genus is one example of a global topology. The genus of a mesh is the number of holes in it. A cube has a genus of zero, while a donut has a genus of one, as shown in the figure 1.

A mesh has a manifold structure, if the local topology is everywhere locally equivalent to a disc; that is, the neighbourhood of every feature consists of a connected ring of polygons forming a single surface. In a triangulated manifold mesh, exactly two triangles share each edge and every triangle shares an edge with exactly three neighbouring triangles. If the mesh has borders, as shown in the figure 2, then the definition must be relaxed so that a border is a chain of edges that are adjacent only to a triangle, and a border vertex is surrounded by an incomplete cycle of triangles.

A mesh that does not fulfil the above properties, is said to have non-manifold topology (NMT). Examples about non-manifold meshes are shown in the figure 3. Non-manifold meshes are sometimes created on purpose, because they may allow a more compressed representation of an object. But more often they result from undesired behaviour of modeling or scanning applications.

Practically, all simplification algorithms are able to operate on manifold meshes. Sometimes, e.g. when a mesh originates from an unknown source, connectivity information related to every vertex needs to be checked to detect, whether the mesh has a manifold structure or not. There are also algorithms that can convert non-manifold meshes to manifold meshes.

Simplification algorithms can be topology-preserving or topology-modifying. Topology preserving algorithms require manifold input and preserve the manifold structure. Toplogy-modifying algorithms do not necessarily preserve the manifold topology, and such algorithms can close holes or join parts of the model and produce

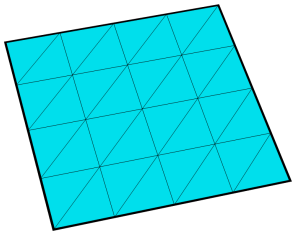


Figure 2: A manifold mesh with borders

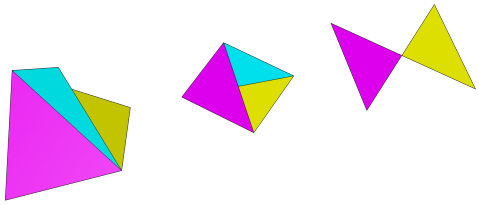


Figure 3: Non-manifold meshes. From left to right: An edge shared by three triangles, T-junction and a non-manifold point

obviously undesirable effects like face intersections.

One could think that the topology-preserving algorithms should always be preferred, but the question is actually two-sided. The topology-preserving algorithms usually produce better-looking simplification results than the topology-modifying algorithms. But on the other hand, topology-modifying algorithms have more opportunities for simplification and therefore they can reach better simplification ratios. For example, closing a hole may remarkably reduce the number of triangles that are needed in a mesh representation.

The importance of the topology preservation depends totally on the use case. For example, if the rendering speed is more important than visualization fidelity, then at least minor topology modifications are acceptable. Similarly, small holes have no role in collision detection. On the other hand, if the intention is to produce realistic visualisations, then topology preservation is important. In the same way, the topology modifications may be problematic in certain medical applications, where the shape features play an important role.

3.4 Approximation error and error metrics

A deviation between the original and the simplified meshes is called approximation error. Measuring the approximation error is a crucial part of most simplification methods. The measure is used, for example, for guiding the simplification process or for terminating the simplification process when the approximation error exceeds a given threshold value.

The approximation error can be managed in different manners. The following classification is from [P21].

- The error may be bounded locally, i.e. the approximation accuracy is known around each surface entity.
- The error may be bounder globally, i.e. the approximation accuracy is know only for the entire simplified mesh.
- Other criteria; usually curvature is taken into account to define a global bound on the surface.
- The approximation error is not evaluated; e.g. the simplification may be driven by a given simplification ratio.

The approximation error can be measured using different error metrics. Some of them are listed below

- A sum of squared distances from sample points on the original mesh to the simplified mesh [P50].
- A sum of squared distances from a new vertex after an edge collapse operation to the adjacent faces of the original vertices [P86].

- A sum of squared tetrahedral volumes between the two mesh neighbourhoods [P71].
- Hausdorff distance [P7], explained in the appendixes C.
- An area-weighted sum of squared distances between the region modified by contraction and the original mesh [P80].

Quadric error metrics [P38] has gained popularity recently. It associates a 4 x 4 matrix to each mesh vertex, characterising the simplification error of that vertex. The benefit of the technique is that only rather simple arithmetic operations are required to update the matrices in the course of the simplification procedure.

There exist also error metrics that can take appearance attributes like textures into account.

4 Classification of mesh simplification methods

The numerous mesh simplification methods can be classified to the following categories.

- *Region merging and vertex clustering* methods reduce the amount of mesh features by merging them.
- *Decimation methods* reduce the amount of mesh features by removing them.
- *Re-tiling methods* re-build a mesh to optimize its structure according to some given criteria.
- *Multiresolutional wavelet methods* present a mesh using a base mesh and wavelets.
- *Image based methods* use the visual appearance of the mesh to guide the simplification procedure.
- *Neural network based methods* can be taught to simplify meshes.

The methods that belong to the same category use similar simplification operations. They usually differ from each other in the strategy that they use to select the mesh elements to be handled. Sometimes the methods also have some freedom to choose the operation details in different ways, for example, there might be several options to replace removed or merged mesh elements.

The end of this chapter presents a short review of the role of symmetry in mesh simplification. CAD models typically consist of parts that are at least approximately symmetric. If the symmetry is not taking into account in simplification, the result may be visually unsatisfactory, e.g. symmetric parts look asymmetric and circular holes misshapen. The symmetry awareness provides also opportunities for stronger mesh reduction and rendering speed advantages.

4.1 Co-planar region merging

Region merging algorithms reduce the number of mesh features by merging mesh regions together. Usually they first search for co-planar or nearly co-planar faces,

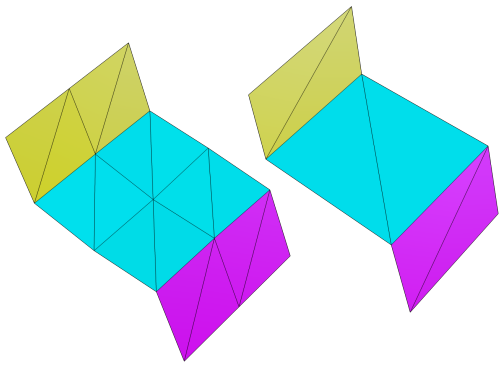


Figure 4: Merging almost co-planar faces

merge them into larger polygons and finally triangulate the polygon to generate a simplified representation of the region [P28, P58, P100] (figure 4).

The region merging algorithms are especially suitable for models that contain large, planar surfaces. Examples of such surfaces are walls in a building model and bulkheads in a ship construction model. Such surfaces may have been modelled using a big number of triangles because of the technical properties of the modeling application or because of the aim to re-tile a slightly curved surface accurately. Even in the latter case, the merging may have only a minor effect to the visual appearance of the surface.

In its basic form, merging is done by selecting two adjacent faces that are co-planar enough according to some criteria and then merging them. The procedure can be enhanced by continuing the merging until no faces that are adjacent to the already merged faces meet a planarity or a normal error criterion [P48, P57]. This process is repeated iteratively for the remaining faces. Then the borders of merged faces are simplified and the resulting regions are triangulated to get the final simplified mesh.

A remarkable advantage in merging is that it does not change the topology of the mesh. In addition, the error criteria is intuitive. A drawback in the approach is that the polygon triangulating operation may be time consuming, which limits the applicability of this approach to highly complex meshes. However, if the model consists of separate parts, then the simplification could be parallelized, thus allowing more complex meshes to be processed in a reasonable time.

4.2 Vertex clustering

Vertex clustering algorithms combine nearby mesh vertices into clusters and then replace each cluster with a single vertex. This causes many faces of the mesh to degenerate into points or edges. An example of 2D uniform vertex clustering is shown in the figure 5.

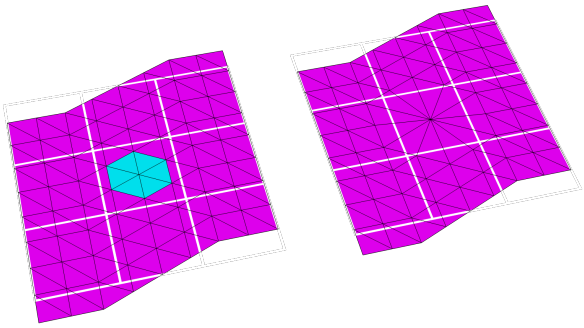


Figure 5: Example of two dimensional vertex clustering

Vertex clustering algorithms often operate by first covering the mesh with an axis-aligned 3D grid and then by replacing all the vertices inside each bounding box with a single vertex. Some vertex clustering methods use, e.g., octrees [P74, P88, P59], instead of a uniform grid, to adapt the clustering resolution more tightly to the geometric distribution of the original mesh.

There are different options to compute the location of the merged vertex. It can be one of the original vertices, an average of the vertices, the centre of the cluster, or its location can be selected by minimizing some error function.

An important property of the basic vertex cluster operation is that it can be applied to arbitrary sets of input vertices. However, the cluster operation does not always preserve topology. Deciding whether each clustered vertex lies inside or outside the object requires manifold topology when triangle information is regenerated.

It is inherently difficult to handle surface attributes, e.g. colour and texture, robustly in vertex clustering. For example, if there are multiple triangles with different colours, simplified geometry colours may look noticeably different than the original geometry.

Vertex clustering operations often process input triangles independently in a random order, which allows parallel implementation. Thus, GPUs that support programmable shader stages can run shader implementations of vertex cluster operators [P26, P60], and there are method variations for multi-core processor or computer cluster environments [P106, P14, P53].

Basic vertex clustering performs poorly on models with high-frequency details, because vertex cluster operators do not take local geometric features into consideration and try to preserve, e.g., sharp edges. This restricts the usefulness of basic vertex clustering on CAD models. One alternative to alleviate this issue is to use adjacency relationship of all points and to compute unit normals to each vertex. This information can then be used to guide an adaptive spatial subdivision until angles between the normals in one cluster are below a certain threshold [P82, P78].

Alternatively, per vertex weights can be computed to represent, e.g., a visual importance of each vertex and this information can be used to adjust the vertex clustering process [P73]. Quadric error metric (see section 3.4) can be used to position the representative vertex of a cluster so that the quadric error of the representative vertex is minimized. It has been shown that by using quadratic error metrics, positioning of the clustered vertices can be improved [P69]. Still different metrics to preserve topological features have been proposed in meshes [P17, P13].

4.3 Geometry decimation

Geometry decimation methods remove either edges, vertices or faces from a mesh. Usually they operate one element at a time until some ending criteria is fulfilled, e.g. the approximation error exceeds a user specified limit.

The different decimation operations are surveyed in the following subsections.

4.3.1 Vertex decimation

Vertex decimation methods simplify meshes by removing vertices from them. Usually they handle the vertices one by one and re-triangulate the resulting hole after each removal operation [P90]. Typically the removal of one vertex eliminates three edges and two faces as shown in the figure 6.

Vertex decimation algorithms differ from each other typically in two aspects: in the order in which the vertices are removed and in the re-triangulation strategy of the resulting holes. Often local surface properties, such as planarity, curvature, distance to the original mesh or distance from the average plane defined by the neighboring vertices guide the selection of the vertices to be removed and the re-triangulation of the resulting holes. If the vertices are selected in the order of increasing importance using e.g. one of the criteria above, then the selection can be speeded up by using hash-based bucketing [P35].

The re-triangulation problem of the holes is trivial as far as the holes are convex. However, if a hole is e.g. star-shaped, then the problem becomes non-trivial. A generalization of the tessalating holes of 2D polygons and 3D volumes has been introduced to alleviate these issues [P85].

An edge flip operation swaps the orientation of a quad diameter that is formed from two triangles. Series of edge flipping operations can be used to minimize the approximation error of vertex decimation, and to achieve better approximations with smaller amount of triangles that can be achieved without edge flipping [P20]. The approximation error here can be volumetric based on two tetrahedras formed from two neighboring triangles around the removed vertex, and where a lower volume sum denotes better approximation error. In addition, high triangle aspect ratio denotes good approximation error, and the mesh face area difference that the vertex removal generates is proprtional to the approximation error (a small error is better).

The vertex removal operation can be guided by one-sided Hausdorff distance between the faces of the original and simplified mesh [P64]. The vertices are removed

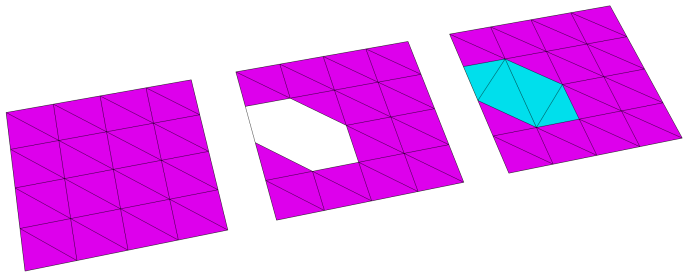


Figure 6: An example of vertex removal and hole re-triangulation

in an order of minimum contribution to the Hausdorff distance until the distance exceedes a predefined limit.

It has been shown that optimizing equiangularity of a surface triangulation can provide good overall approximations of the simplified mesh shapes [P91]. The re-triangulation error of the simplfied mesh can be minimized by iteratively removing vertices that optimize the equiangularity of the simplified mesh.

Vertices of the simplified mesh can be classified as hyperbolic if the vertex is indise a convex hull surface of the neighboring vertices. By removing non-hyperbolic vertices that have a small solid angle and volume with respect to local neighboring vertices (in analogy with spherical volume) the order of removed vertices can be enhanced [P19].

4.3.2 Edge decimation

Edge decimation aka edge collapse methods simplify meshes by removing edges from them. The removed edge is replaced by a single vertex as shown in the figure 7. Similarly as in vertex decimation, the edges are usually processed one by one.

The edge decimation methods differ from each other mainly in two ways, namely in the edge processing order and in the placing of the new vertex. The new vertex can be one of the original edge end points [P104, P86] the center of the collapsed edge [P2], arbitrary point along the collapsed edge [P50], or arbitrary point in the neighborhood of the collapsed edge [P89, P38].

Progressive mesh techniques can be built upon edge decimation. The base mesh is created with a series of edge collapse operations. The original mesh can be re-generated step by step by revoking the collapse operations with a series of edge split operations that are the dual operations for the collapse operations [P50]. Progressive meshes use an energy minimization approach to generate the base mesh. The energy function is based on three terms: a distance that measures the closeness of fit, a vertex count weighting that penalizes mesh with a large number of vertices, and a

regularizing term that conceptually places springs of length zero on the edges of the mesh. The spring term is added to the energy function to pull vertices towards neighbouring vertices as otherwise the closeness fit tems may not converge in all cases. The concept of progressive mesh has been extended to generate dynamic view-dependent simplifications according to the camera location [P51].

Quadric error metrics (see 3.4) provide a means to guide the simplification process with relatively small additional data footprint [P38]. The algorithm iteratively merges pairs of vertices, which need not necessarily be connected by an edge. The error caused by an edge decimation operation can be derived from the sum of the quadric error metrics of the vertices being merged. The algorithm maintains a set of all possible vertex pairs and their error terms. In each iteration, the algorithm removes the pair with the lowest error and then computes new quadric error terms for the neighbouring vertices. Runtime complexity of the algorithms is often relatively high due to number of searched candidate vertex pairs. An adaptive threshold selection method has been proposed to alleviate the issue [P33].

Extension to handle surface attributes of simplified models with quadric error metric [P37, P49] and linear mapping between levels of detail [P22] have additionally been proposed.

Local curvature around each vertex can be estimated and embedded into the original quadric error metric, so that the metric will not only measure distance error, but reflect geometric variations of local surface [P109].

Edge decimation based weighted quadric error metrics have been proposed for the user controlled creation of multi-resolution meshes [P83, P62]. The user can control the simplification of a mesh by interactively painting the importance of regions as weights onto the mesh and preserve important features of original models even after drastic simplification.

Error metrics based on the theory of local differential geometry [P79] can be utilized in a way that the first and the second order discrete differentials approximated locally on a discrete polygonal surface are integrated into the distance error metric [P63]. This ad-hoc metric can measure highly curved and thin regions, and often preserve the features of original models even after drastic simplification.

It has been shown that simplification error introduced in edge collapse operation can be estimated by a weighted sum of squared distances between the region around the collapsed edge and the original mesh [P80]. Approximated mesh quality can be enhanced by positioning the collapsed vertex in a way that it minimizes the volume embedded between the simplified mesh and the original one [P3].

Especially with extremely large 3D models, retaining a history of the geometry of the original model during the edge collapse process can be challenging in memory limited environments. However, edge collapse operations can be computed efficiently without retaining the geometric history, and without scarifying the simplification quality remarkably [P71, P70].

4.3.3 Triangle decimation

Triangle decimation aka triangle collapse methods simplify meshes by removing whole triangles from them as illustrated in the figure 8. Triangle decimation can

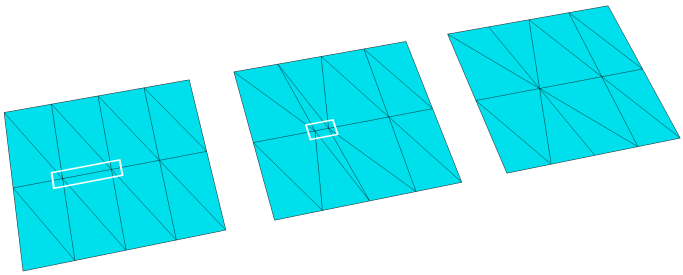


Figure 7: Edge decimation

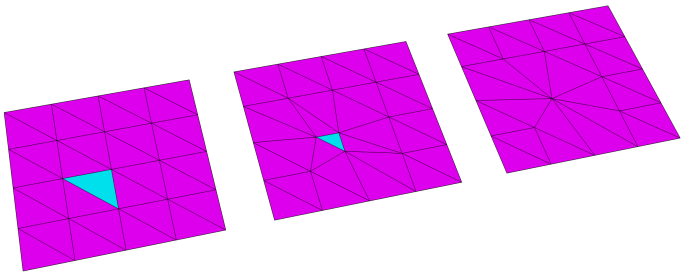


Figure 8: Triangle decimation

also be seen as a coarse version of edge decimation, because all the three edges of the removed triangle degenerate into a single vertex and hence the operation is equal to three simultaneous edge collapse operations, where one of the involved triangles is common to the collapse operations. As a consequence of that, three adjacent triangles degerate into edges and a total of four triangles are eliminated from the mesh. The benefit of triangle decimation is that the simplification progresses quickly, but unfortunately also the approximation error grows rapidly.

A triangle decimation method can be tuned to remove triangles that are nearly co-planar with their neighbouring triangles by calculating local curvatures and simultaneously preventing generation of long-narrow triangles. This is accomplished by weighting each triangle by the combination of the curvatures of the three edges and the equiangularity of the triangle. Then the least weighted triangle is removed, the affected region is re-triangulated and the triangle weights are updated [P44].

Triangle area can be used in the weighting instead of equiangularity [P40].

Triangle and edge collapse methods can be used simultaneously by classifying vertices based on their local geometric features [P54]. Then local regions that are nearly flat are collapsed using triangle decimation, while more challenging regions are simplified applying edge collapse operations. Additionally, the quadric error metrics has been used to evaluate the error in the triangle collapse process [P110]. Static level of detail mesh chains have been generated using triangle decimation [P39], and by storing information which vertices are removed in each triangle collapse operation, level of detail popping artefacts between two mesh levels can be reduced by linearly interpolating original and collapsed triangle vertices [P18].

4.4 Re-meshing and re-tiling

Re-meshing techniques are used to reconstruct meshes so that their appearance is preserved even though vertex locations, vertex density or connective information may significantly differ from the original mesh, see figure 9. Even the topology preservation is not usually guaranteed. Re-meshing can be used for example to to change the density of the mesh, to regularize the structure of the mesh or to re-shape the mesh triangles. An example of the last case is generation of nonobtuse triangle mesh that is composed of a set of triangles, in which every angle is less than or equal to 90 degrees [P68, P92].

Re-meshing can in some cases used also for mesh simplification, although that is not its primary application area. Therefore, the re-meshing methods are not surveyed in this paper. A good survey can be found from [P4].

Re-tiling algorithm [P96] processes a triangular input mesh by iteratively spreading a predefined number of new points to the input mesh to produce nearly uniform distribution of vertices with density weighted estimates of local curvatures. Then the original vertices are deleted iteratively to generate a new triangulation that has the same topology as the original surface. This re-tiling process replaces the input mesh with a coarser mesh. The method usually produces good results on smooth surfaces, while sharp features might be blurred.

4.5 Multiresolution wavelet and Fourier analysis

Wavelet decomposition methods use regular and hierarchical decompositon to build a wavelet representation of the mesh. Wavelet mesh representation is basically a coarse base mesh, and a set of progressive finer details that are encoded into a sequence of coefficients. Wavelet approaches have been proposed to manage regularly gridded meshes [P43, P46] and irregular meshes [P30, P16, P93, P25, P87, P98, P94].

Roughly, the wavelet decomposition is a three-phase process. The first phase partitions the input mesh into a small number of triangular regions and generates a simplified base mesh for each region. Second phase parametrizes each region using selected wavelet base. Third phase recursively subdivides each parametrized base mesh to a finer detail mesh based on a wavelet codebook. This process continues

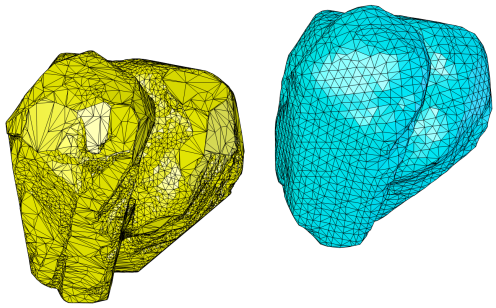


Figure 9: Re-meshing

until the deviation between the sub-divided mesh and the original mesh is under a given threshold.

Wavelet based mesh simplification methods can be enhanced to consider additional surface parameters such as texture, coordinates and colours in the simplification process [P94, P16].

Fourier analysis has also been used to simplify meshes in addition to wavelets [P67, P111], but such methods do not seem to have gained signigicant interest in the field of mesh simplification methods recently.

4.6 Image based simplification

Image based simplification algorithms use the perceptual quality of the rendered mesh to guide the simplification procedure, utilizing visual properties in a way that pure geometry based methods cannot do. The typical drawback of the algorithms is their high computational complexity, because they may have to render a model multiple times per each simplification step to evaluate different alternatives.

CAD models can remarkably benefit from image based simplification methods, because the models often have parts that are at least partially hidden, e.g. the supporting structures under a floor. The invisible parts of the structures can be simplified more aggressively than the visible parts.

The earliest image-based simplification algorithm [P72] utilized metrics that determines the cost of an edge collapse operation by rendering the simplified mesh from multiple viewpoints. The algorithm chooses the collapse operation that produces the least mean-square error for luminance color component.

Image based polygonal simplification can utilize perceptual metrics that are derived from measurements of perceptibility of visual stimuli [P101, P76]. Contrast sensitivity functions can indicate whether a simplification operation is visible, but cannot determine the best perceptible simplification. Methods using perceptual metrics cannot typically accomplish drastic simplification.

Visibility information can be leveraged on image based mesh simplification by defining a visibility function between the surfaces of a simplified model and cameras that are located on a surrounding sphere [P108]. The visibility function is defined for each surface of the mesh and the function describes the amount of visible surface pixels from currently rendered camera viewpoint. A table that indicates how many pixels from each triangle are visible from different camera position is computed, and the stored values give the end visibility value for each triangle. Once the visibility map is computed, the final visibility values are combined with quadric error measures [P38] to drive the actual mesh simplification process.

The importance of mesh regions can be used to guide the mesh simplification process [P66]. The motivation for this approach is to preserve regions that are visually interesting, no matter what the size of each important regions is. Preservation of interesting regions is achieved by generating a map of interesting regions and utilizing the map in the edge collapse operations [P108].

Image based mesh simplification can utilize viewpoint mutual information that is an information-theoretic measure [P99, P34]. The mutual information metric measures the correlation between a viewpoint and the polygons of the mesh. This information is achieved by rendering the mesh from multiple viewpoints. Mesh simplification is then based on the half-edge collapse and simplification error metrics based on viewpoint mutual information [P15].

4.7 Neural network based simplification

Mesh simplification based on neural networks is a quite unique approach to simplify 3D models. Simplification methods can train neural networks to replace local geometric shapes with pre-defined simplified shapes via example simplification results. The main advantage of neural network based methods is that they do not need to work with the whole object to reduce the mesh complexity.

Growing Neural Gas (GNG) algorithm [P36] is an unsupervised incremental clustering algorithm that can learn local triangle topology, and can be utilized to simplify meshes [P6, P5]. Mesh simplification based on the GNG algorithm has two steps: an optimization phase and a reconstruction phase. The optimization phase computes a simplified set of vertices that gives the best approximation of the input mesh based on selected error metric. Decision, which vertices are removed and how the vertex positions are approximated, is based on a set of rules that are being optimized during the iterative training process. The reconstruction phase uses the information about the new vertices provided by the optimization phase, the information about the original mesh vertices, and generates the faces of the simplified mesh.

4.8 Symmetry aware algorithms

Many CAD models or their parts have significant symmetry than can be utilized in the model simplification. However, the symmetry information is not explicitly

available in the models, because the CAD tools do not save information, for example, about copy or rotate operations. Instead, there is a need for a separate pre-processing step, where the global shape of the mesh is analysed and the possible symmetries are identified. The analysis is complicated by the fact that the symmetries may be only approximate, e.g. in case of non-rigid transformations like stretching.

Symmetry identification can be seen as a special case of shape correspondence problem, in which the aim is to find meaningful correspondences between two or more local shapes from triangle meshes. The term "meaningful" may range, depending on the problem in question, from geometric similarity to identical functionality. An example about the latter are handles in different dishes. In symmetry identification, the correspondences are either exact or approximate, perhaps reflected, geometric similarities between the segments of a single shape. Kaick et al. have made a survey of shape correspondence techniques [P56]. Liu et al. [P42] use identification of symmetry axis as a pre-processing step for finding correspondences between different shapes, but the identification of the symmetry axis might be a valuable result as such in symmetry identification. To extract the symmetry axis, they use a method that first generates a map to store symmetry information the surface on itself, then measures the distances from surface points to their correspondences in the map, and finally extracts a symmetry axis curve using the distance measures.

Golovinskiy et al. [P41] present a complete framework for symmetric mesh processing. Their solution consists of the following steps: symmetry analysis, geometric symmetrization, symmetric mapping, symmetric remeshing and restoring deformation. The framework can be used to enhance the symmetries of a mesh, to decompose a mesh into its symmetric parts and asymmetric residuals, and to establish correspondences between symmetric mesh features. The emphasis is on visibility aspects and compression, of which especially the first one is interesting from CAD model simplification point of view.

Leander [P65] presents in his master thesis two algorithms that are given a mesh and symmetry information about it as input. The first algorithm simplifies the symmetric part of the mesh with an arbitrary algorithm and then uses to result to represent the other part. The other algorithm uses the symmetry information to steer the simplification so that the resulting mesh is as symmetric as possible.

5 Practical simplification trials

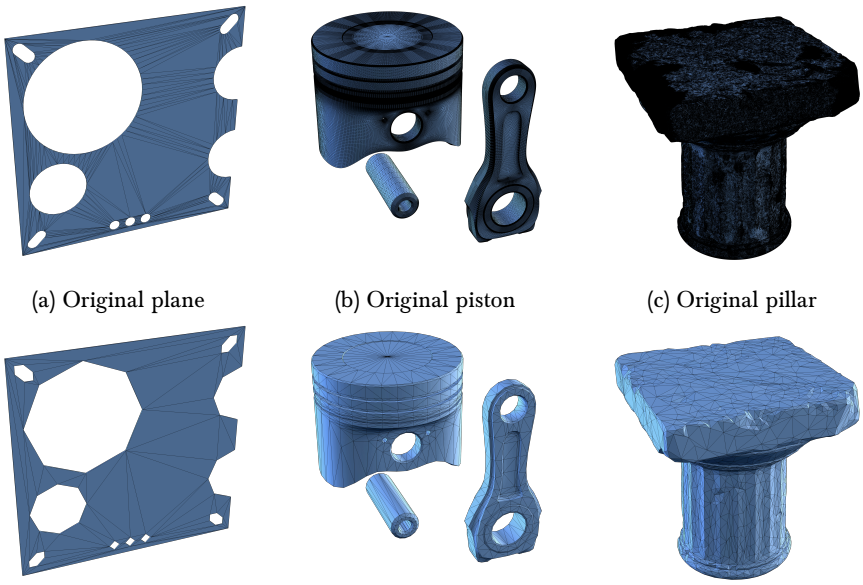
Some practical mesh simplification tests were made, using commonly available graphics tools. The main purpose of the testing was to get an overview of the capabilities of the existing tools as well as to get a better understanding about the feasibility and behaviour of some selected simplification methods.

5.1 Overview

The tests were carried out by applying edge collapse, different decimation and remeshing methods (see section 5.3 for detailed information) in addition to proprietary methods to three test objects. The first object, a thin plane with several holes, represents a tessellated CAD model composed of triangles. The second object, a piston, represents a higher-order surface that has been converted to a triangle mesh. The third object, a stone pillar, is an example of a dense and noisy mesh that has resulted from 3D-scanning or sculpting.

The three objects were first manually simplified by a 3D artist to get a baseline for the algorithmically simplified objects. In the case of the thin plane object, the vertices around the holes were collapsed in clockwise order and the triangulation was further improved by changing the edge-flow around the holes. Good edge-flow is an artistic term meaning that mesh edges are well positioned to express the curvature and sharp features of an object with the given complexity. With deforming meshes, edges are in addition positioned to allow deformations to happen smoothly and for geometry to collapse properly in order to hold the apparent volume of the mesh — in the case of realistic characters typically by following anatomically correct muscle group borders. Typically good edge flow uses edge loops extensively to ensure that deformation effects are kept local. Well-made edge flow also avoids long and thin triangles which use more processing power during GPU rendering.

The piston object was simplified by locating the vertices of the low polygon object to the vertices of the high polygon object (shrink-wrapping). The pillar was first simplified by applying a quadric error metrics based edge decimation method to it using Meshlab tool [P77], then shrink-wrapping the result to a higher polygon mesh and finally the decimation errors of the overlapping faces were fixed by hand. Original and optimized test objects are shown in the figure 10 and their sizes in the table 1.



(a) Original plane (b) Original piston (c) Original pillar

(d) Hand-optimized plane (e) Hand-optimized piston (f) Hand-optimized pillar

Figure 10: Original test meshes and the meshes after being simplified by an artist.

Test object	Original vertices	Original triangles	Simplified vertices	Simplified triangles
Thin plane	508	1048	26% (132)	28% (296)
Piston	222076	450176	0.95% (2105)	0.94% (4234)
Pillar	2261811	4523614	0.14% (3078)	0.14% (6152)

Table 1: Sizes of the original and simplified test meshes

5.2 Tools

This chapter gives an overview of the trialled tools: general digital content creation packages Autodesk Maya, Blender and Autodesk 3dsMax, and more specialized tools Simplygon and Meshlab. Tools were selected based on their availability and apparent popularity.

Autodesk Maya

Autodesk Maya [P9] is a 3D modeling and animation suite that can be used for modeling, animation and rendering. Maya can be extended with its own scripting language. It is frequently used to make 3D models and animations for feature films, television and game development.

Blender

Blender [P10] is an open source 3D modeling and animation suite that can be used for modeling, animation and rendering. In addition to more typical features, Blender supports compositing, motion tracking, video editing and game creation. Blender has Python API for scripting.

Blender has three built-in decimation methods: edge collapse, unsubdivide and planar. The unsubdivide is a reverse of the subdivide method that adds mesh density by dividing mesh faces. An unsubdivide operation works with quads, not with triangles and therefore needs a quad mesh input, from which it tries to remove face strips that subdivision created. The planar decimator apparently uses co-planar merging that has been presented in section 4.1.

In addition, Blender has a shrink wrap modifier and a re-meshing modifier. The shrink wrap modifier projects a mesh to the surface of an object or, in other words, wraps it around the object. The re-mesher, using Dual Contouring of Hermite Data [P55] method, creates a quad mesh of desired density from the input mesh. The two operations can be used together: a dense triangle mesh is first re-meshed and the resulting low-polygon mesh is then shrink wrapped to the original dense mesh.

Autodesk 3dsMax

Autodesk 3dsMax [P8] is a 3D modeling and animation suite that can be used for modeling, animation and rendering. The 3dsMax is frequently used by video game developers, commercial TV studios and architectural visualization companies. It can be extended and automated with scripts.

The 3dsMax tool offers optimizer and pro-optimizer modifiers for mesh simplification. These methods resemble typical decimation results with support for artist-guided simplification by marking edges.

Simplygon

Simplygon [P29] is a 3D computer graphics software for automatic 3D-optimization, based on proprietary methods for creating level-of-detail mesh hierarchies for use in games and other realtime content. Unlike the other applications, Simplygon is cloud-based and also has a SDK that can be connected with a plugin to other applications. Simplygon allows artists to guide the simplification process by using vertex weight data.

MeshLab

MeshLab [P77] is an open source tool for processing and editing of unstructured 3D triangular meshes. The Meshlab tool is widely used e.g. for pre-processing meshes for 3D-printing. It offers a wide variety of decimation and re-meshing algorithms and related mesh operations in different stages of development.

5.3 Trials

The following tests were conducted:

- **Autodesk Maya**:

  Maya's reduce modifier was used in the simplification trial. Reduce modifier is the only simplification modifier shipped with Maya and, by visual observation, it seems to be some variant of edge collapse decimator. Default simplification operation settings were used in the trials.

- **Blender**:

  The edge collapse method of decimate modifier was used in the simplification trial as other methods do not allow user to set triangle or vertex count targets.

- **Autodesk 3dsMax**:

  Only the pro-optimizer method was used in the simplification trial, as it is newer than the older optimizer method. The method settings were kept in their default values.

- **Simplygon**:

  Simplygon didn't have multiple selectable simplification methods and like other proprietary tools it doesn't state what simplification algorithm it uses. The reduction ratio was set to match the reference mesh size, while the adjustment sliders were kept at their default settings.

- **MeshLab**:

  The trial covered Meshlab filters for MC edge collapse, clustering decimation, quadric edge decimation, Poisson reconstruction [P61], VCG surface reconstruction and uniform mesh resampling.

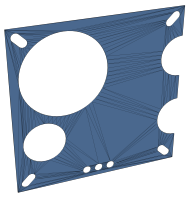
  NOTES:

  MC edge collapse: The current implementation showed some weird behaviour, because the simplification ratio was dependent on the mesh orientation. For that reason, MC edge collapse was left out of the final trials.

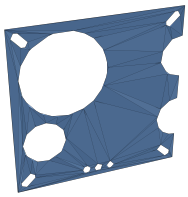
  Quadric edge decimation: The trial was run in mode that tries to preserve surface orientation and the genus of the mesh. An afterwards manual clean-up was performed for the mesh to fix broken parts of the mesh geometry.

  Poisson reconstruction: The test was done with octree depth set to 5 (smaller numbers result in lower polygon count). The solver divider was kept at 6.

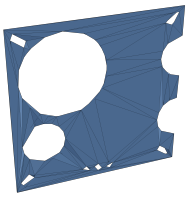
  All methods were run using 0.075 world unit parameter to match the polygon count of the artist simplified models.



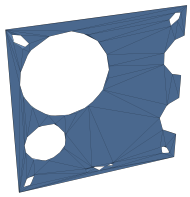
(a) Original plane



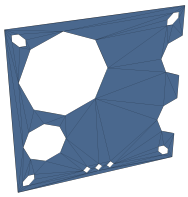
(b) Blender decimate modifier



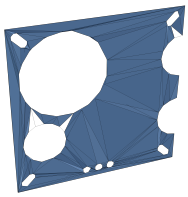
(c) Maya reduce



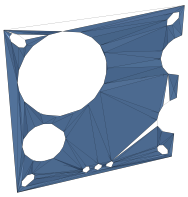
(d) Meshlab Quadric edge decimation 1



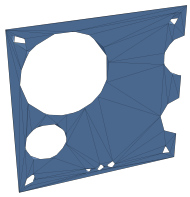
(e) Hand-optimized reference



(f) 3dsmax Prooptimizer



(g) Simplygon



(h) Meshlab Quadric edge decimation 2

Figure 11: Results of the plane mesh simplification tests. The problems in the shapes of the holes can be clearly seen. Lamina faces appear as additional holes in images f and g. The viewpoint of the object is such that the roughness of the exterior is not very clearly visible.

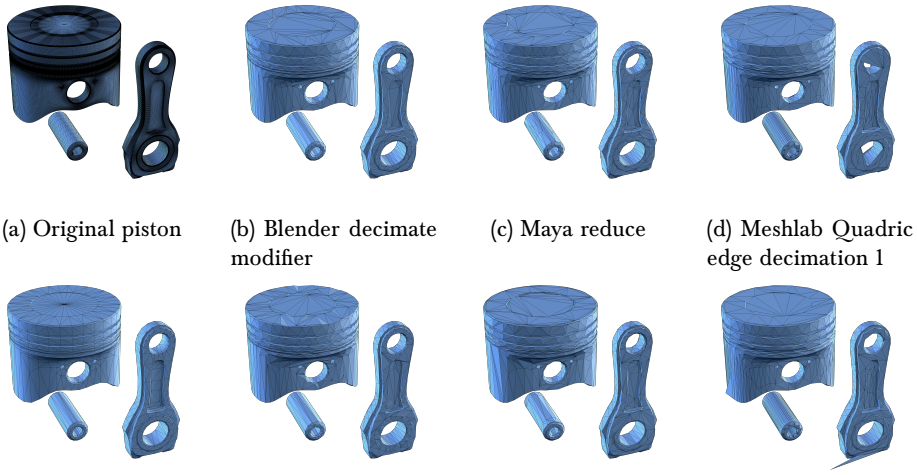
5.4 Results

5.4.1 Plane object

The plane object simplification results are shown in the figure 11. The plane object turned out to be in many ways problematic for the simplification algorithms. The algorithms seemed to prefer curvature preservation to thickness, making the flat areas of the plane rough and uneven. In addition, the algorithms broke the symmetry of the holes, causing the holes to look shapeless. Some algorithms failed to handle the thinness of the object properly. The ProOptimizer and Simplygon tools created triangle pairs that share their vertices, but have opposite orientation (these are sometimes called lamina faces). These triangles are supported by some tools (Maya, 3dsMax, Simplygon), but are unsupported in some other tools (Blender) and therefore appear as extra holes in some images.

5.4.2 Piston

The piston simplification results are shown in the figure 12. Like the thin plane, the piston object is an example of engineering model that has synthetic shapes. In the same way as in the plane case, the simplification algorithms violate the symmetries of the object, which can be seen most clearly around the holes and at the top of the piston. On the contrary, the piston is thicker than the plane and consists of a denser



(a) Original piston

(b) Blender decimate modifier

(c) Maya reduce

(d) Meshlab Quadric edge decimation 1

(e) Hand-optimized reference

(f) 3dsmax Proopti-mizer

(g) Simplygon

(h) Meshlab Quadric edge decimation 2

Figure 12: Results of the piston mesh simplifications. The undesired effects (lost symmetry, additional roughness) are most clearly visible around the holes and at the top of the piston.

triangle mesh, for what reason some of the undesired effects are not so pronounced.

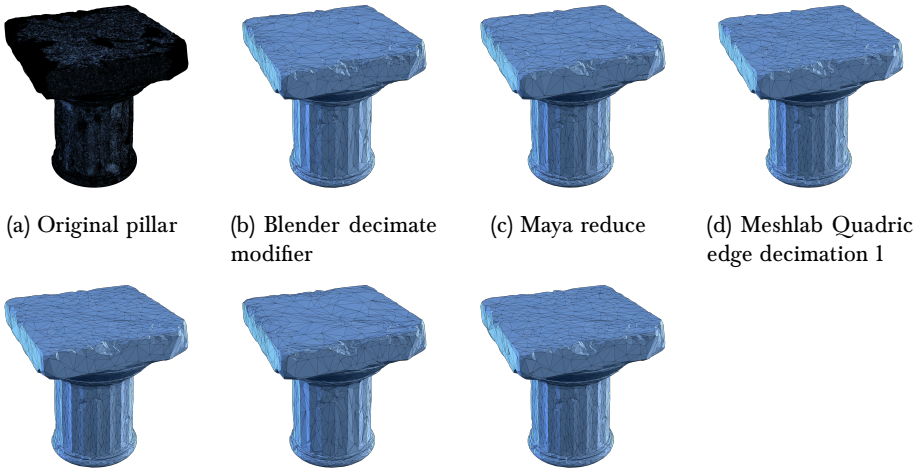
5.4.3 Pillar

The pillar simplification results are shown in the figure 13. All the algorithms behaved well in the case of the pillar when compared to the hand-optimized reference mesh. The pillar is an organic model without synthetic shapes or real symmetry, therefore the possible approximation errors are not clearly visible as they are lost in the noise. The pillar also has a dense triangular mesh and there are no thin parts in the object.

5.4.4 Additional experiments

Besides the actual simplification method comparisons, a few other simplification experiments were accomplished. In the first experiment, the pillar mesh was iteratively decimated by always halving the previous mesh density. The goal of the experiment was to visualize the effect of the different triangle reduction ratios to the object appearance. The results in the figure 14 show that sometimes even a remarkable decimation may only have negligible effect on the visual quality.

In the second experiment, the holes in the plane object were manually simplified by repeatedly collapsing every second vertex to its clockwise neighbour, while preserving the thickness and vertex symmetry of the opposite sides of the holes. In this case, the results in the figure 15 show that once the symmetry is preserved, an



(a) Original pillar (b) Blender decimate modifier (c) Maya reduce (d) Meshlab Quadric edge decimation 1

(e) Hand-optimized reference (f) 3dsmax Prooptimizer (g) Meshlab Quadric edge decimation 2

Figure 13: Results of the pillar mesh simplifications. There are no remarkable differences between the hand-optimized reference mesh and the algorithmically optimized meshes.

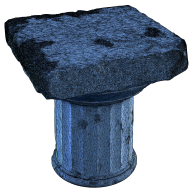
engineering model remains illustrative even if the number of triangles is strongly reduced. An automatic method like this could be ideal for CAD-model simplification.

In the third experiment, vertex clustering, Poisson reconstruction, VCG reconstruction and Uniform reconstruction were tested. First, the methods were applied to the plane object as shown in the figure 16. Like in the previous trials, the plane object turned out to be the most problematic for the mesh processing algorithms. The vertex clustering step produced adequate results, although the symmetry of the mesh was not preserved, but the VCG re-meshing distorted the shape of the mesh. Poisson reconstrucion and Uniform re-meshing did not produce feasible results at all. In case of the piston, all methods produced reasonable results as illustrated in the figure 17. The best results were achieved with pillar object as can be seen from the figure 18. While all the re-meshing methods loosed sharp and small details, the resulting pillar mesh has a nice topology that could be further improved manually.

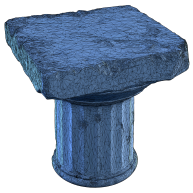
It should be noted that the re-meshing in these experiments was made with low mesh density. Using a higher mesh density would have produced better results, but then the triangle count of the meshes had rather increased than decreased.

In the fourth experiment, the Poisson reconstructed pillar was shrink wrapped onto the original mesh to improve the shape and the details of the simplified mesh. The results are shown in the figure 19.

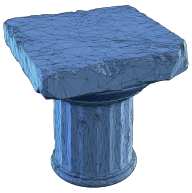
(a) Original pillar with 2.261.811 vertices (4.523.614 triangles)



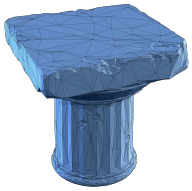
(b) Halved 4 times, 141.366 vertices (282.724 triangles)



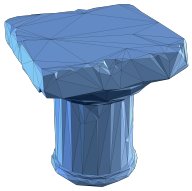
(c) Halved 6 times, 35.344 vertices (70.680 triangles)



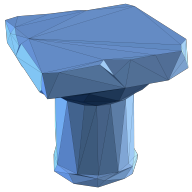
(d) Halved 8 times, 8.839 vertices (17.670 triangles)



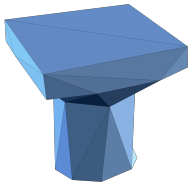
(e) Halved 10 times, 2.212 vertices (4.416 triangles)



(f) Halved 12 times, 556 vertices (1.104 triangles)

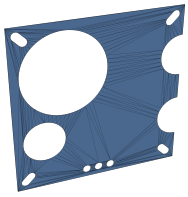


(g) Halved 14 times, 142 vertices (276 triangles)

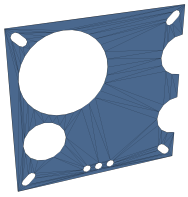


(h) Halved 16 times, 38 vertices (68 triangles)

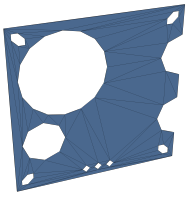
Figure 14: Halving the mesh density. There is almost no difference at level 4 even though the vertex count is only 1/16th of the originial mesh. The visual fidelity of the silhouette (object outline) starts to suffer after level ten.



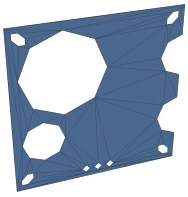
(a) Original plane



(b) Clockwise manual collapsing half of vertices

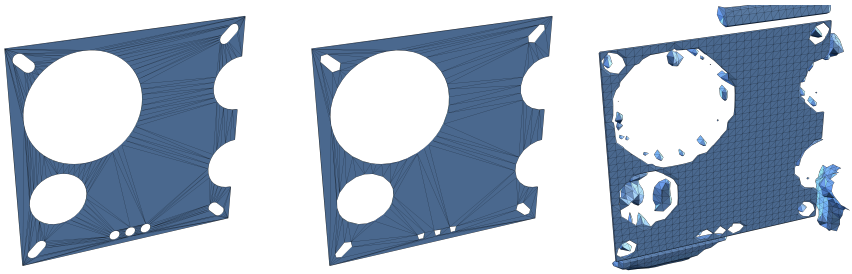


(c) Repeat



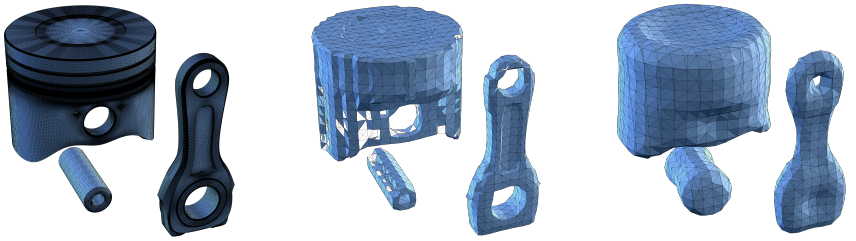
(d) Repeat

Figure 15: Manually and iteratively collapsing every second vertex on the hole edges to their clockwise neighbours. Thickness and symmetry is preserved and engineering model remains illustrative.



(a) Original plane (b) Clustering decimation (c) VCG reconstruction

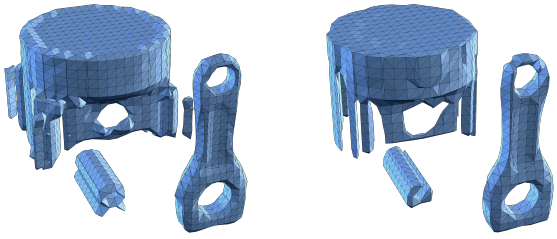
Figure 16: Clustering and re-meshing the plane object. Re-meshing increased the mesh density and in this case introduced defects due to thinness and high genus of the object. The results of Poisson reconstruction and Uniform re-meshing operations were unsatisfactory and are not shown in the figure.



(a) Original piston

(b) Vertex clustering loses curvature of the original object. This is quite typical for all grid-based approaches.

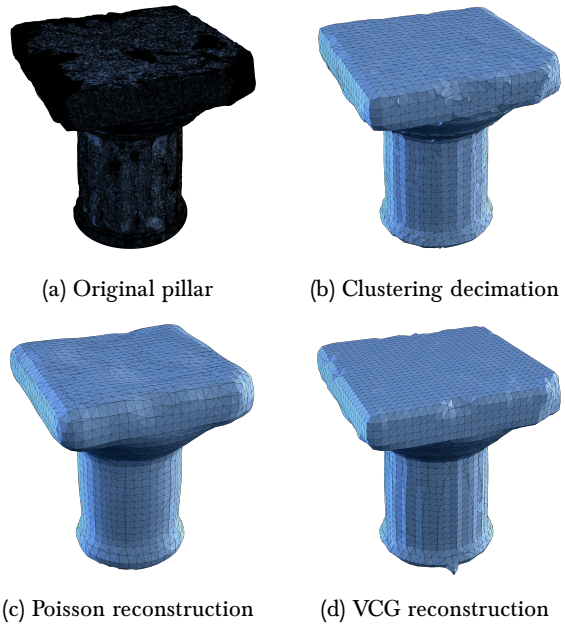
(c) Poisson reconstruction closes holes.



(d) VCG reconstruction has problems with holes and with some round features.

(e) Uniform remeshing has problems with round features.

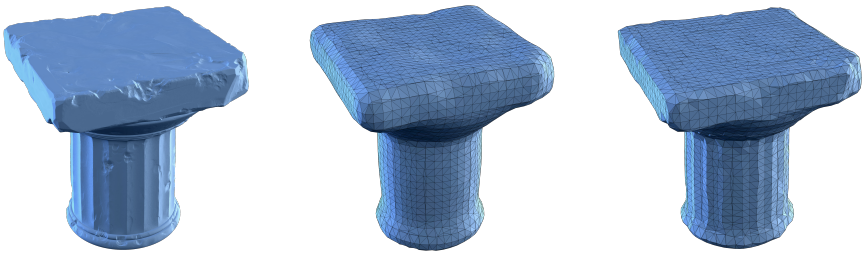
Figure 17: Clustering and remeshing the piston mesh. As the object is thicker and has dense triangle structure, all methods gave reasonable results.



(a) Original pillar (b) Clustering decimation

(c) Poisson reconstruction (d) VCG reconstruction

Figure 18: Clustering and re-meshing the pillar object. Dense, thick and organic mesh works well with the remeshers. The resulting meshes have "nice" topology and could be further improved manually.



(a) Original pillar (b) Poisson reconstruction (c) Poisson reconstruction shrinkwrapped onto original

Figure 19: Shrink wrapping the result of the Poisson reconstruction onto the original mesh in order to improve shape and details.

5.5 Discussion

The overall impression about the mesh simplification trials was that the tested simplification algorithms do not usually produce satisfactory results, if the models are thin or holed and if they are looked at closely. The simplified meshes would need a clean-up by an artist to be usable in production. In addition, the algorithms seem to violate nice edge flows that are perhaps designed by artists and designers or are remnants from higher-order surface conversion. In many cases, it would be faster for artists to do the simplification manually from the beginning than first using an automatic method, followed by a manual clean-up.

If the resulting object is static and doesn't have any deforming animations or its image on the screen is small, then the algorithms produce better results, because the topology does not play such an important role. However, the problems in the edge-flow and in the triangulation usually become more apparent if the object has deforming animation.

The tested simplification algorithms often produce feasible results, when they are used to simplify objects that do not contain a lot of symmetry or sharp features, which are typical in CAD models. Thus, if the simplified low polygon mesh is mainly used, e.g., as a target mesh for normal map bakers which transfer the original high polygon details into a texture for low polygon mesh for use in game engines [P97, P31], then approximating all the details of the high-poly object is not paramount as the mesh details are encoded into the baked normal maps. In such cases, a simple holding shape with a reasonably matching silhouette works fine as the simplified target mesh. Baking object details into normal maps is a good approximation also for level-of-detail models, which are viewed from the distance and for meshes, which have lot of shallow surface details (like engravings). Commonly in game productions, artists hand-optimize real-time meshes for the silhouette and add details via normal maps. Most of the tested mesh optimizations algorithms seem to produce results that are aimed for this.

For simplifying technical and engineering models for non-photo-realistic augmented reality use, where only the most basic shading is used without any normal maps, all the tested simplification methods are unsuitable. An ideal algorithm should keep the thickness of the object and only approximate curvature, not preserve it. For example, curvature around holes is visually not as important as keeping vertex symmetry at the opposite ends of the hole. In engineering applications, vertex positions around a hole should be symmetric and there should preferably be vertices at width and height extremes. In a real time graphic, a hole can be approximated even with four vertices that are placed symmetrically. More accurate curvature should be represented always with multiple of four vertices (8, 12, 16, ...) to keep vertex pairs at the opposite sides.

Finally, the experiments showed that while re-meshing is a suitable tool for noisy, organic mesh simplifications, it is not a good approach for engineering models. On the one hand, if the target mesh density is low enough for the number of the polygons to be reduced, then sharp features and small details are typically lost. On the other hand, if the target mesh density is high enough to preserve those features, then the number of polygons rather increases than decreases. The reason for this phenomena

is that it is neither feasible nor economic to represent both the fine details and the planar areas of the mesh with equal mesh density.

A wishlist for illustrative CAD model simplification:

- Vertices that are not removed during simplification process should have the same coordinates as the matching vertices have on the original object, in order to preserve the exact dimensions.
- Edge loops should be preserved to limit topology violations.
- Vertex symmetry on the opposite sides of holes should be preserved for engineering "look".
- Vertex counts for holes should be multiple of four to preserve the illustrative width and height extremes.
- Sharp features should be preserved to keep the appearance of the object (this might be a consequence of edge loop and vertex position preservation).

6 Conclusions

Mesh simplification has been a popular research topic over several decades and there exists a vast amount of different simplification approaches and methods that were reviewed in this survey. There are at least a couple of reasons that explain the big number of publications. First, the mesh simplification is an optimization problem, an typically different mesh simplification algorithms are tailored to preserve certain features in the simplified mesh. Second, the simplification is NP-hard problem. Therefore, it is not possible to solve any realistic simplification problem accurately in the worst case. Instead, numerous different heuristics have usually been developed for preserving use case specific features in the simplified meshes.

A consequence of the computational complexity of the mesh simplification problem is that the practical methods are almost always local by their nature, meaning that the simplification methods handle the mesh elements one by one in some suitable order. A popular strategy is to define a function that represents the approximation error that a specific simplification operation would cause. The simplification operations are then applied in the order of smallest approximation error.

The local simplification strategies are computationally efficient, but the lack of the overall understanding of the mesh topology prevents the local methods to take into account global mesh features. Therefore, simplified meshes may have undesirable effects as, e.g., sharp features of the original mesh may be removed during the simplification process. The sharp features may be tempting targets for the local simplification operators, because they contain numerous small details that can be reduced without an excessive approximation error. However, removing the sharp features may in reality change the appearance of the mesh considerably.

Symmetric meshes or symmetric mesh elements are another case, where the local methods may turn out to be problematic. For example, small perfectly circular holes appear very frequently in mechanical devices. If the holes are triangulated asymmetrically, their appearance may not be satisfactory. In the same way, if the opposite sides of a symmetric element are triangulated differently, the element may not appear symmetric any more. In these kind of cases, the use of the symmetry aware methods should be considered.

One of the key observations of this survey is that, despite the large variety of the mesh simplification techniques, it is still possible that dedicated simplification methods need to be implemented if there exists any application specific requirements. The general methods often fail to take into account the application specific features of the problem. At the same time, the diversity of the application areas causes that

it is hard to find a single existing method that would match well enough to the specific characteristics of a dedicated simplification problem. Anyway, there are good chances that the customized method could be a variant of some existing algorithm or it could be composed of the existing building blocks like the basic simplification operations and the different strategies that are used to guide the simplification process.

Another observation is that it is hard to design a fully automatic simplification method that would always produce satisfactory results. This is especially true regarding the visual appearance of the simplified mesh. Therefore, in many cases it would be beneficial to give the user a chance to somehow control the simplification process. It is, of course, totally use case specific, whether e.g. an interactive control is feasible or not.

Appendix A Glossary

Approximation error
The difference between the original and simplified meshes.

CAD
Computer Aided Design.

Delaunay triangulation
A triangulation of a vertex set with the property that no vertex falls in the interior of the circle that passes through all three vertices of any triangle in the triangulation.

Genus
Number of holes in a mesh.

GPU
Graphics Processing Unit.

Hausdorff distance
A measure for the distance of two metrics spaces, e.g. point sets.

High polygon (high poly)
A mesh that is compromised of relative large number of polygons.

Laplacian smoothing
A mesh smoothing algorithm that is based on vertex re-locating. Local neighbouring vertices are utilized in computing new locations for each vertex.

Level-of-Detail
An approach, in which the density of a mesh is reduced according to the viewing distance of the mesh. This approach is frequently used in real-time applications to improve mesh rendering performance. In such cases the reduced mesh representations are usually created in advance.

See also "Multi-resolution".

LOD
See "Level-of-Detail".

Low polygon (low poly)
A mesh that is compromised of relative small number of polygons.

Marching cubes
A computer graphics algorithm for extracting a polygonal mesh from a 3D grid of discrete surface samples.

Multi-resolution
A hierarchy of simplified mesh representations (for example for level-of-detail purposes).

NMT
Non-manifold topology (explained in the text).

Non-rigid transformation
An operation that, when performed on a geometric object, changes the size of the object, but not its shape: e.g. dilation and stretching.

NP-hard
Non-deterministic Polynomial-time Hard.

A class of problems that, in the light of present knowledge, have exponential running time compared to the size of the problem. In practice, these problems can be solved only approximately using heuristics methods that produce "good enough" results.

PM
See "Progressive mesh".

Progressive mesh
A continuous multi-resolution representation of a mesh, consisting of a coarse base mesh and a sequence of operations that incrementally refine the base mesh back to the original mesh.

QEM
See "Quadric Error Metric".

Quad mesh
A mesh constructed of quadrilaterals.

Quadric Error Metric
A quadric matrix represents the approximation error of a mesh vertex. The quadric matrix is initially constructed from triangle plane equations that share each vertex. During mesh simplification processes, the quadric error matrices sum the quadric matrices of removed vertices together. For more information, see 3.4.

Rigid transformation
Any operation that, when performed on a geometric object, does not change neither the shape nor the size of the object. The rigid transformations are: transfer, rotation and reflection.

Tessellation
The process of approximating a surface or an object with a mesh of polygons.

Triangular mesh
A mesh constructed of triangles.

Appendix B Search strings

The exact search strings and the approximate amounts of hits are listed below:

- ACM Digital Library (about 1000 hits):

  Abstract:(("cad model" OR (cad AND geometr*) OR mesh*) AND (optimi* OR reduc* OR compres* OR smoothing) AND NOT vlsi AND NOT "integrated circuit") OR Title:(("cad model" OR (cad AND geometr*) OR mesh*) AND (optimi* OR reduc* OR compres* OR smoothing) AND NOT vlsi AND NOT "integrated circuit")

- IEEExplore (about 8700 hits):

  ((("cad model") OR (cad AND geometr*) OR mesh*) AND (optimi* OR reduc* OR compres* OR smoothing) AND NOT vlsi AND NOT "integrated circuit")

- Elsevier Science Direct (about 100 hits):

  "mesh simplification" or "lod algorithm" or "cad model simplification" or "multiresolution model" or "3D triangulation" or "mesh approximation" or "multiresolution mesh"

  NOTE: Search covered "Abstract", "Title" and "Keywords" fields.

- CiteCeerX (about 200 hits):

  title((((("cad model") OR (cad AND geometr*) OR mesh*) AND (optimi* OR reduc* OR compres* OR smoothing) AND NOT vlsi AND NOT "integrated circuit")) AND abstract:(((((("cad model") OR (cad AND geometr*) OR mesh*) AND (optimi* OR reduc* OR compres* OR smoothing) AND NOT vlsi AND NOT "integrated circuit"))

- SpringerLink (about 1000 hits):

  "mesh simplification" or "lod algorithm" or "cad model simplification" or "multiresolution model" or "3D triangulation" or "mesh approximation" or "multiresolution mesh"

- WebOfScience (about 200 hits):

  TITLE: ("mesh simplification" OR "lod algorithm" OR "cad model simplification" OR "multiresolution model" OR "3D triangulation" OR "mesh approximation" OR "multiresolution mesh")

Appendix C Hausdorff distance

Let A and B be two non-empty sub sets of a metric space (M, d). The Hausdorff distance of the two sets is:

$$\mathrm{d_H}(A,B) = \max\left\{\sup_{a\in A}\inf_{b\in B}\mathrm{d}(a,b), \sup_{b\in B}\inf_{a\in A}\mathrm{d}(a,b)\right\}$$

The Hausdorff distance can be illustrated by taking a closer look at the definition in a case where A and B are compact 2D point sets as shown in figure 20. In this case, the expression

$$\inf_{b\in B}\mathrm{d}(a,b)$$

defines the distance from an arbitrary point $a \in A$ to B to be the distance from a to the closest point $b \in B$. There might be no closest point, for what reason infinum is used instead of minimum. Further,

$$\sup_{a\in A}\inf_{b\in B}\mathrm{d}(a,b)$$

defines the distance from A to B the be distance from the farthest point $a \in A$ to B. There might be no farthest point, for what reason the supremum is used instead

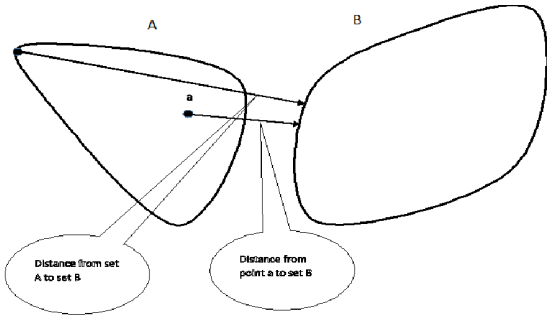


Figure 20: Hausdorff distance of two 2D point sets

of maximum.

Finally, the Hausdorff distance is the maximum of the distance from A to B and the distance from B to A.

Acknowledgements

The research has been carried out during the MARIN2 project (Mobile Mixed Reality Applications for Professional Use) funded by Tekes (The Finnish Funding Agency for Innovation) in collaboration with partners; Defour, Destia, Granlund, Infrakit, Integration House, Lloyd's Register, Nextfour Group, Meyer Turku, BuildingSMART Finland, Machine Technology Center Turku and Turku Science Park. The authors are from Technology Research Center, University of Turku, Finland.

Bibliography

Selected papers

[P1] Pankaj K. Agarwal and Subhash Suri. "Surface Approximation and Geometric Partitions". In: *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '94. Arlington, Virginia, USA: Society for Industrial and Applied Mathematics, 1994, pp. 24–33. isbn: 0-89871-329-3. url: http://dl.acm.org/citation.cfm?id=314464.314475.

[P2] Maria-Elena Algorri and Francis Schmitt. "Mesh Simplification". In: *Proceedings of the 1996 Eurographics, Tutorials*. 1996, pp. 77–86.

[P3] Pierre Alliez et al. "Mesh Approximation Using a Volume-Based Metric". In: *Proceedings of the 7th Pacific Conference on Computer Graphics and Applications*. PG '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 292–. isbn: 0-7695-0293-8. url: http://dl.acm.org/citation.cfm?id=826028.826485.

[P4] Pierre Alliez et al. "Recent Advances in Remeshing of Surfaces". In: *Shape Analysis and Structuring, Mathematics and Visualization*. Springer, 2008.

[P5] Rafael Álvarez et al. "A Mesh Optimization Algorithm Based on Neural Networks". In: *Inf. Sci.* 177.23 (Dec. 2007), pp. 5347–5364. issn: 0020-0255. doi: 10.1016/j.ins.2007.05.029. url: http://dx.doi.org/10.1016/j.ins.2007.05.029.

[P6] Rafael Álvarez et al. "GNG3D - A Software Tool for Mesh Optimization Based on Neural Networks". In: *International Joint Conference on Neural Networks*. IEEE Computer Society, 2006, pp. 4005–4012.

[P7] Nicolas Aspert, Diego Santa-cruz, and Touradj Ebrahimi. "MESH: Measuring Errors between Surfaces using the Hausdorff distance". In: 2002, pp. 705–708.

[P8] Autodesk Inc. *Autodesk 3ds Max: 3D modeling, animation, and rendering software*. http://www.autodesk.com/products/3ds-max. Accessed: March 2015.

[P9] Autodesk Inc. *Autodesk Maya: Comprehensive 3D animation software*. http://www.autodesk.com/products/maya. Accessed: March 2015.

[P10] Blender Foundation. *Blender: Free and open source 3D animation suite*. http://www.blender.org. Accessed: March 2015.

[P11] Mario Botsch et al. "Geometric Modeling Based on Polygonal Meshes". In: *ACM SIGGRAPH 2007 Courses*. SIGGRAPH '07. San Diego, California: ACM, 2007. isbn: 978-1-4503-1823-5. doi: 10.1145/1281500.1281640. url: http://doi.acm.org/10.1145/1281500.1281640.

[P12] Mario Botsch et al. *Polygon Mesh Processing*. AK Peters, 2010. isbn: 978-1-56881-426-1.

[P13] Tamy Boubekeur and Marc Alexa. "Technical Section: Mesh Simplification by Stochastic Sampling and Topological Clustering". In: *Comput. Graph.* 33.3 (June 2009), pp. 241–249. issn: 0097-8493. doi: 10.1016/j.cag.2009.03.025. url: http://dx.doi.org/10.1016/j.cag.2009.03.025.

[P14] Dmitry Brodsky and Jan Baekgaard Pedersen. "A Parallel Framework for Simplification of Massive Meshes". In: *Proceedings of the 2003 IEEE Symposium on Parallel and Large-Data Visualization and Graphics*. PVG '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 4–. isbn: 0-7695-2091-X. doi: 10.1109/PVGS.2003.1249038. url: http://dx.doi.org/10.1109/PVGS.2003.1249038.

[P15] P. Castelló et al. "Technical Section: Viewpoint-driven Simplification Using Mutual Information". In: *Comput. Graph.* 32.4 (Aug. 2008), pp. 451–463. issn: 0097-8493. doi: 10.1016/j.cag.2008.05.005. url: http://dx.doi.org/10.1016/j.cag.2008.05.005.

[P16] Andrew Certain et al. "Interactive Multiresolution Surface Viewing". In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '96. New York, NY, USA: ACM, 1996, pp. 91–98. isbn: 0-89791-746-4. doi: 10.1145/237170.237213. url: http://doi.acm.org/10.1145/237170.237213.

[P17] Kam-Fai Chan and Chi-Wah Kok. "Mesh Simplification by Vertex Cluster Contraction". In: *Proceedings of the Second IEEE Pacific Rim Conference on Multimedia: Advances in Multimedia Information Processing*. PCM '01. London, UK, UK: Springer-Verlag, 2001, pp. 1114–1119. isbn: 3-540-42680-9. url: http://dl.acm.org/citation.cfm?id=648109.747839.

[P18] Jun Chen and Xiong Shi. "Real-time LOD Algorithm Based on Triangle Collapse Optimization". In: *Proceedings of the 2011 Seventh International Conference on Computational Intelligence and Security*. CIS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 312–315. isbn: 978-0-7695-4584-4. doi: 10.1109/CIS.2011.76. url: http://dx.doi.org/10.1109/CIS.2011.76.

[P19] Chansophea Chuon and Sumanta Guha. "Volume Cost Based Mesh Simplification". In: *Proceedings of the 2009 Sixth International Conference on Computer Graphics, Imaging and Visualization*. CGIV '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 164–169. isbn: 978-0-7695-3789-4. doi:

10.1109/CGIV.2009.16. url: http://dx.doi.org/10.1109/CGIV.2009.16.

[P20] A. Ciampalini et al. "Multiresolution Decimation based on Global Error". In: *The Visual Computer* 13 (1997), pp. 228–246.

[P21] P. Cignoni, C. Montani, and R. Scopigno. "A Comparison of Mesh Simplification Algorithms". In: *Computers and Graphics* 22 (1997), pp. 37–54.

[P22] Jonathan Cohen, Dinesh Manocha, and Marc Olano. "Simplifying Polygonal Models Using Successive Mappings". In: *Proceedings of the 8th Conference on Visualization '97*. VIS '97. Phoenix, Arizona, USA: IEEE Computer Society Press, 1997, 395–ff. isbn: 1-58113-011-2. url: http://dl.acm.org/citation.cfm?id=266989.267108.

[P23] Jonathan Cohen et al. "Simplification Envelopes". In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '96. New York, NY, USA: ACM, 1996, pp. 119–128. isbn: 0-89791-746-4. doi: 10.1145/237170.237220. url: http://doi.acm.org/10.1145/237170.237220.

[P24] Gautam Das and Michael T. Goodrich. "On the Complexity of Optimization Problems for 3-dimensional Convex Polyhedra and Decision Trees". In: *Comput. Geom. Theory Appl.* 8.3 (Aug. 1997), pp. 123–137. issn: 0925-7721. doi: 10.1016/S0925-7721(97)00006-0. url: http://dx.doi.org/10.1016/S0925-7721(97)00006-0.

[P25] Ingrid Daubechies et al. "Wavelets on Irregular Point Sets". In: *Phil. Trans. R. Soc. Lond. A* 357 (1999), pp. 2397–2413.

[P26] Christopher DeCoro and Natalya Tatarchuk. "Real-time Mesh Simplification Using the GPU". In: *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*. I3D '07. Seattle, Washington: ACM, 2007, pp. 161–166. isbn: 978-1-59593-628-8. doi: 10.1145/1230100.1230128. url: http://doi.acm.org/10.1145/1230100.1230128.

[P27] L. DeFlorani, E. Puppo, and R. Scopigno. "Level-of-Detail in Surface and Volume Modeling". In: *IEEE Visualization 98 Tutorials*. Vol. 6. IEEE, 1998.

[P28] M.J. DeHaemer and M.J. Zyda. "Simplification of objects rendered by polygonal approximations". In: *Computers and Graphics* 15.2 (1991), 175––184.

[P29] Donya Labs AB. *Simplygon: Automagic 3D Optimization*. https://www.simplygon.com. Accessed: March 2015.

[P30] Matthias Eck et al. "Multiresolution Analysis of Arbitrary Meshes". In: *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '95. New York, NY, USA: ACM, 1995, pp. 173–182. isbn: 0-89791-701-4. doi: 10.1145/218380.218440. url: http://doi.acm.org/10.1145/218380.218440.

[P31] Epic Games, Inc. *Unreal: a suite of game development tools*. http://unreal.com. Accessed: April 2015.

[P32] Carl Erikson. *Polygonal Simplification: An Overview*. Tech. rep. TR 96-016. Dept. of CS, U. of North Carolina, Chapel Hill, 1996.

[P33] Carl Erikson and Dinesh Manocha. "GAPS: General and Automatic Polygonal Simplification". In: *Proceedings of the 1999 Symposium on Interactive 3D Graphics*. I3D '99. Atlanta, Georgia, USA: ACM, 1999, pp. 79–88. isbn: 1-58113-082-1. doi: 10.1145/300523.300532. url: http://doi.acm.org/10.1145/300523.300532.

[P34] Miquel Feixas, Mateu Sbert, and Francisco González. "A Unified Information-Theoretic Framework for Viewpoint Selection and Mesh Saliency". In: *ACM Trans. Appl. Percept.* 6.1 (Feb. 2009), 1:1–1:23. issn: 1544-3558. doi: 10.1145/1462055.1462056. url: http://doi.acm.org/10.1145/1462055.1462056.

[P35] Martin Franc and Václav Skala. "Fast Algorithm for Triangular Mesh Simplification Based on Vertex Decimation". In: *Proceedings of the International Conference on Computational Science-Part II*. ICCS '02. London, UK, UK: Springer-Verlag, 2002, pp. 42–51. isbn: 3-540-43593-X. url: http://dl.acm.org/citation.cfm?id=645458.655622.

[P36] B. Fritzke. "A Growing Neural Gas Network Learns Topologies". In: *Advances in Neural Information Processing Systems 7*. MIT Press, 1995, pp. 625–632.

[P37] Michael Garland and Paul S. Heckbert. "Simplifying Surfaces with Color and Texture Using Quadric Error Metrics". In: *Proceedings of the Conference on Visualization '98*. VIS '98. Research Triangle Park, North Carolina, USA: IEEE Computer Society Press, 1998, pp. 263–269. isbn: 1-58113-106-2. url: http://dl.acm.org/citation.cfm?id=288216.288280.

[P38] Michael Garland and Paul S. Heckbert. "Surface Simplification Using Quadric Error Metrics". In: *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '97. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997, pp. 209–216. isbn: 0-89791-896-7. doi: 10.1145/258734.258849. url: http://dx.doi.org/10.1145/258734.258849.

[P39] Tran S. Gieng et al. "Constructing Hierarchies for Triangle Meshes". In: *IEEE Transactions on Visualization and Computer Graphics* 4.2 (Apr. 1998), pp. 145–161. issn: 1077-2626. doi: 10.1109/2945.694956. url: http://dx.doi.org/10.1109/2945.694956.

[P40] Tran S. Gieng et al. "Smooth Hierarchical Surface Triangulations". In: *Proceedings of the 8th Conference on Visualization '97*. VIS '97. Phoenix, Arizona, USA: IEEE Computer Society Press, 1997, pp. 379–386. isbn: 1-58113-011-2. url: http://dl.acm.org/citation.cfm?id=266989.267102.

[P41] A. Golovinskiy, J. Podolak, and T. Funkhouser. "Symmetry-Aware Mesh Processing". English. In: *Mathematics of Surfaces XIII*. Ed. by EdwinR. Hancock, RalphR. Martin, and MalcolmA. Sabin. Vol. 5654. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 170–188. isbn: 978-

3-642-03595-1. doi: 10.1007/978-3-642-03596-8_10. url: http://dx.doi.org/10.1007/978-3-642-03596-8_10.

[P42] Eitan Grinspun et al. *Finding Surface Correspondences Using Symmetry Axis Curves.*

[P43] Markus H. Gross, Oliver G. Staadt, and Roger Gatti. "Efficient Triangular Surface Approximations Using Wavelets and Quadtree Data Structures". In: *IEEE Transactions on Visualization and Computer Graphics* 2.2 (June 1996), pp. 130–143. issn: 1077-2626. doi: 10.1109/2945.506225. url: http://dx.doi.org/10.1109/2945.506225.

[P44] Bernd Hamann. "A Data Reduction Scheme for Triangulated Surfaces". In: *Computer Aided Geometric Design* 11.2 (1994), pp. 197–214.

[P45] Mingyi He and Long Li. "Advance in triangular mesh simplification study". In: *Control Automation Robotics and Vision (ICARCV), 2010 11th International Conference on.* IEEE, 2010, pp. 1638–1643.

[P46] D. J. Hebert and HyungJun Kim. "Image Encoding with Triangulation Wavelets". In: *Proc. SPIE, Wavelet Applications in Signal and Image Processing* 2569 (1995), pp. 381–392.

[P47] P. Heckbert and M. Garland. "Survey of Polygonal Surface Simplification Algorithms". In: *ACM SIGGRAPH 1997 Courses.* SIGGRAPH '97. New York, NY, USA: ACM, 1997.

[P48] Paul Hinker and Charles Hansen. "Geometric Optimization". In: *Proceedings of the 4th Conference on Visualization '93.* VIS '93. San Jose, California: IEEE Computer Society, 1993, pp. 189–195. isbn: 0-8186-3940-7. url: http://dl.acm.org/citation.cfm?id=949845.949882.

[P49] Hugues Hoppe. "New Quadric Metric for Simplifying Meshes with Appearance Attributes". In: *Proceedings of the 10th IEEE Visualization 1999 Conference (VIS '99).* VISUALIZATION '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. –. isbn: 0-7803-5897-X. url: http://dl.acm.org/citation.cfm?id=832273.834119.

[P50] Hugues Hoppe. "Progressive Meshes". In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques.* SIGGRAPH '96. New York, NY, USA: ACM, 1996, pp. 99–108. isbn: 0-89791-746-4. doi: 10.1145/237170.237216. url: http://doi.acm.org/10.1145/237170.237216.

[P51] Hugues Hoppe. "View-Dependent Refinement of Progressive Meshes". In: *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques.* SIGGRAPH '97. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997, pp. 189–198. isbn: 0-89791-896-7. doi: 10.1145/258734.258843. url: http://dx.doi.org/10.1145/258734.258843.

[P52] Jerry O. Talton Iii. *A Short Survey of Mesh Simplification Algorithms.* Tech. rep. University of Illinois at Urbana-Champaign, Course Notes. University of Illinois at Urbana-Champaign, 2004.

[P53] Martin Isenburg et al. "Large Mesh Simplification Using Processing Sequences". In: *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*. VIS '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 61–. isbn: 0-7695-2030-8. doi: 10.1109/VISUAL.2003.1250408. url: http://dx.doi.org/10.1109/VISUAL.2003.1250408.

[P54] Veysi Isler et al. "Real-Time Multi-Resolution Modeling for Complex Virtual Environments". In: *In Proceedings of ACM Symposium on Virtual Reality Software and Technology*. 1996, pp. 11–20.

[P55] Tao Ju et al. "Dual Contouring of Hermite Data". In: ACM Press, 2002, pp. 339–346.

[P56] Oliver Van Kaick et al. *A Survey on Shape Correspondence*. 2011.

[P57] Alan D. Kalvin and Russell H. Taylor. "Superfaces: Polygonal Mesh Simplification with Bounded Error". In: *IEEE Comput. Graph. Appl.* 16.3 (May 1996), pp. 64–77. issn: 0272-1716. doi: 10.1109/38.491187. url: http://dx.doi.org/10.1109/38.491187.

[P58] Alan. D. Kalvin et al. "Constructing topologically connected surfaces for the comprehensive analysis of 3D medical structures". In: *In SPIE Medical Imaging V, Image Processing, Vol 1445*. SPIE, 1991, pp. 247–259.

[P59] Takayuki Kanaya and Ken-ichi Kobori. "A Method of Model Simplification Using Spatial Partitioning". In: *The Journal of the Institute of Image Information and Television Engineers* 56.4 (2002), pp. 636–642. doi: 10.3169/itej.56.636.

[P60] Takayuki Kanaya et al. "A Method for Storing Clustering Information of Model Simplification in GPUs". In: *Proceedings of Graphics, Visualization and Computer Vision, WSCG'2011*. 2011, pp. 121–126.

[P61] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. "Poisson Surface Reconstruction". In: *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*. SGP '06. Cagliari, Sardinia, Italy: Eurographics Association, 2006, pp. 61–70. isbn: 3-905673-36-3. url: http://dl.acm.org/citation.cfm?id=1281957.1281965.

[P62] Youngihn Kho and Michael Garland. "User-guided Simplification". In: *Proceedings of the 2003 Symposium on Interactive 3D Graphics*. I3D '03. Monterey, California: ACM, 2003, pp. 123–126. isbn: 1-58113-645-5. doi: 10.1145/641480.641504. url: http://doi.acm.org/10.1145/641480.641504.

[P63] Sun-Jeong Kim, Soo-Kyun Kim, and Chang-Hun Kim. "Discrete Differential Error Metric for Surface Simplification". In: *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*. PG '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 276–. isbn: 0-7695-1784-6. url: http://dl.acm.org/citation.cfm?id=826030.826623.

[P64] Reinhard Klein, Gunther Liebich, and Wolfgang Straßer. "Mesh Reduction with Error Control". In: *Proceedings of the 7th Conference on Visualization '96*. VIS '96. San Francisco, California, USA: IEEE Computer Society Press, 1996, pp. 311–318. isbn: 0-89791-864-9. url: http://dl.acm.org/citation.cfm?id=244979.245624.

[P65] Jonas Leander. *Development of Polygon Reduction Algorithms for Symmetric 3D Models*. 2005.

[P66] Chang Ha Lee, Amitabh Varshney, and David W. Jacobs. "Mesh Saliency". In: *ACM SIGGRAPH 2005 Papers*. SIGGRAPH '05. Los Angeles, California: ACM, 2005, pp. 659–666. doi: 10.1145/1186822.1073244. url: http://doi.acm.org/10.1145/1186822.1073244.

[P67] Yong-Gu Leea and Kunwoo Lee. "Geometric detail suppression by the Fourier transform". In: *Computer-Aided Design* 30.8 (1998), 677––693.

[P68] J. Y. S. Li and H. Zhang. "Nonobtuse Remeshing and Mesh Decimation". In: *Proceedings of the Eurographics Symposium on Geometry Processing*. 2006, pp. 235–238.

[P69] Peter Lindstrom. "Out-of-core Simplification of Large Polygonal Models". In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '00. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 259–262. isbn: 1-58113-208-5. doi: 10.1145/344779.344912. url: http://dx.doi.org/10.1145/344779.344912.

[P70] Peter Lindstrom and Greg Turk. "Evaluation of Memoryless Simplification". In: *IEEE Trans. Vis. Comput. Graph.* 5.2 (1999), pp. 98–115. doi: 10.1109/2945.773803. url: http://doi.ieeecomputersociety.org/10.1109/2945.773803.

[P71] Peter Lindstrom and Greg Turk. "Fast and Memory Efficient Polygonal Simplification". In: *Proceedings of the Conference on Visualization '98*. VIS '98. Research Triangle Park, North Carolina, USA: IEEE Computer Society Press, 1998, pp. 279–286. isbn: 1-58113-106-2. url: http://dl.acm.org/citation.cfm?id=288216.288288.

[P72] Peter Lindstrom and Greg Turk. "Image-driven Simplification". In: *ACM Trans. Graph.* 19.3 (July 2000), pp. 204–241. issn: 0730-0301. doi: 10.1145/353981.353995. url: http://doi.acm.org/10.1145/353981.353995.

[P73] Kok-Lim Low and Tiow-Seng Tan. "Model Simplification Using Vertex-clustering". In: *Proceedings of the 1997 Symposium on Interactive 3D Graphics*. I3D '97. Providence, Rhode Island, USA: ACM, 1997, 75–ff. isbn: 0-89791-884-3. doi: 10.1145/253284.253310. url: http://doi.acm.org/10.1145/253284.253310.

[P74] David Luebke and Carl Erikson. "View-dependent Simplification of Arbitrary Polygonal Environments". In: *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '97. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997, pp. 199–208. isbn: 0-89791-896-7. doi: 10.1145/258734.258847. url: http://dx.doi.org/10.1145/258734.258847.

[P75] David P. Luebke. "A Developer's Survey of Polygonal Simplification Algorithms". In: *IEEE Comput. Graph. Appl.* 21.3 (May 2001), pp. 24–35. issn: 0272-1716. doi: 10.1109/38.920624. url: http://dx.doi.org/10.1109/38.920624.

[P76] David P. Luebke and Benjamin Hallen. "Perceptually Driven Simplification for Interactive Rendering". In: *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*. London, UK, UK: Springer-Verlag, 2001, pp. 223–234. isbn: 3-211-83709-4. url: http://dl.acm.org/citation.cfm?id=647653.732291.

[P77] MeshLab (Istituto di Scienza e Tecnologie dell'Informazione). *MeshLab: An open source, portable, and extensible system for the processing and editing of unstructured 3D triangular meshes*. meshlab.sourceforge.net. Accessed: March 2015.

[P78] Li Nan et al. "A New Adaptive Mesh Simplification Method Using Vertex Clustering with Topology-and-Detail Preserving". In: *Proceedings of the 2008 International Symposium on Information Science and Engieering - Volume 01*. ISISE '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 150–153. isbn: 978-0-7695-3494-7. doi: 10.1109/ISISE.2008.131. url: http://dx.doi.org/10.1109/ISISE.2008.131.

[P79] Dyn Nira et al. "Mathematical Methods for Curves and Surfaces". In: ed. by Tom Lyche and Larry L. Schumaker. Nashville, TN, USA: Vanderbilt University, 2001. Chap. Optimizing 3D Triangulations Using Discrete Curvature Analysis, pp. 135–146. isbn: 0-8265-1378-6. url: http://dl.acm.org/citation.cfm?id=570828.570841.

[P80] Elena Ovreiu et al. "Mesh simplification using an accurate measured quadratic error". In: *10th International Symposium on Signals, Circuits and Systems (ISSCS)*. IEEE Computer Society, 2011, pp. 1–4.

[P81] Chao Peng and Yong Cao. "Integrating Occlusion Culling with Parallel LOD for Rendering Complex 3D Environments on GPU". In: *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '13. Orlando, Florida: ACM, 2013, pp. 187–187. isbn: 978-1-4503-1956-0. doi: 10.1145/2448196.2448235. url: http://doi.acm.org/10.1145/2448196.2448235.

[P82] Gao Pengdong et al. "Adaptive Mesh Simplification Using Vertex Clustering with Topology Preserving". In: *Computer Science and Software Engineering, 2008 International Conference on*. Wuhan, Hubei: IEEE, 2008, pp. 971 –974.

[P83] Erik Pojar and Dieter Schmalstieg. "User-controlled Creation of Multiresolution Meshes". In: *Proceedings of the 2003 Symposium on Interactive 3D Graphics*. I3D '03. Monterey, California: ACM, 2003, pp. 127–130. isbn: 1-58113-645-5. doi: 10.1145/641480.641505. url: http://doi.acm.org/10.1145/641480.641505.

[P84] Enrico Puppo and Roberto Scopigno. "Simplification, LOD and Multiresolution - Principles and Applications". In: *Proceedings of the 1997 Eurographics, Tutorials*. 1997.

[P85] Kevin J. Renze and James H. Oliver. "Generalized Unstructured Decimation". In: *IEEE Comput. Graph. Appl.* 16.6 (Nov. 1996), pp. 24–32. issn: 0272-1716. doi: 10.1109/38.544069. url: http://dx.doi.org/10.1109/38.544069.

[P86] Rémi Ronfard and Jarek Rossignac. "Full-range Approximation of Triangulated Polyhedra". In: *Computer Graphics Forum* 15 (3 1996), pp. 67–76. doi: 10.1111/1467-8659.1530067.

[P87] M. Roy et al. "Multiresolution analysis for meshes with appearance attributes". In: *Image Processing, 2005. ICIP 2005. IEEE International Conference*. IEEE, 2005, pp. 816–819.

[P88] Scott Schaefer and Joe Warren. "Adaptive Vertex Clustering Using Octrees". In: *SIAM Geometric Design and Computing*. 2003, pp. 491–500.

[P89] William J. Schroeder. "A Topology Modifying Progressive Decimation Algorithm". In: *Proceedings of the 8th Conference on Visualization '97*. VIS '97. Phoenix, Arizona, USA: IEEE Computer Society Press, 1997, 205–ff. isbn: 1-58113-011-2. url: http://dl.acm.org/citation.cfm?id=266989.267059.

[P90] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. "Decimation of Triangle Meshes". In: *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '92. New York, NY, USA: ACM, 1992, pp. 65–70. isbn: 0-89791-479-1. doi: 10.1145/133994.134010. url: http://doi.acm.org/10.1145/133994.134010.

[P91] Marc Soucy and Denis Laurendeau. "Multiresolution Surface Modeling Based on Hierarchical Triangulation". In: *Comput. Vis. Image Underst.* 63.1 (Jan. 1996), pp. 1–14. issn: 1077-3142. doi: 10.1006/cviu.1996.0002. url: http://dx.doi.org/10.1006/cviu.1996.0002.

[P92] Feng Sun et al. "Obtuse Triangle Suppression in Anisotropic Meshes". In: *Comput. Aided Geom. Des.* 28.9 (Dec. 2011), pp. 537–548. issn: 0167-8396. doi: 10.1016/j.cagd.2011.09.007. url: http://dx.doi.org/10.1016/j.cagd.2011.09.007.

[P93] W. Sweldens. "The Lifting Scheme: A New Philosophy in Biorthogonal Wavelet Constructions". In: *Wavelet Applications in Signal and Image Processing III*. Ed. by A. F. Laine and M. Unser. Proc. SPIE 2569, 1995, pp. 68–79.

[P94] Agnieszka Szczesna. "The Lifting Scheme for Multiresolution Wavelet-Based Transformation of Surface Meshes with Additional Attributes". In: *Proceedings of the International Conference on Computer Vision and Graphics: Revised Papers*. ICCVG 2008. Warsaw, Poland: Springer-Verlag, 2009, pp. 487–495. isbn: 978-3-642-02344-6. doi: 10.1007/978-3-642-02345-3_48. url: http://dx.doi.org/10.1007/978-3-642-02345-3_48.

[P95] Atul Thakur, Ashis Gopal Banerjee, and Satyandra K. Gupta. "A Survey of CAD Model Simplification Techniques for Physics-based Simulation Applications". In: *Comput. Aided Des.* 41.2 (Feb. 2009), pp. 65–80. issn: 0010-4485. doi: 10.1016/j.cad.2008.11.009. url: http://dx.doi.org/10.1016/j.cad.2008.11.009.

[P96] Greg Turk. "Re-tiling Polygonal Surfaces". In: *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '92. New York, NY, USA: ACM, 1992, pp. 55–64. isbn: 0-89791-479-1. doi: 10.1145/133994.134008. url: http://doi.acm.org/10.1145/133994.134008.

[P97] Unity Technologies. *Unity: development platform for creating 2D/3D games and interactive experiences*. http://unity3d.com. Accessed: April 2015.

[P98] Sébastien Valette and Rémy Prost. "Wavelet-Based Multiresolution Analysis of Irregular Surface Meshes". In: *IEEE Transactions on Visualization and Computer Graphics* 10.2 (Mar. 2004), pp. 113–122. issn: 1077-2626. doi: 10.1109/TVCG.2004.1260763. url: http://dx.doi.org/10.1109/TVCG.2004.1260763.

[P99] Ivan Viola et al. "Importance-Driven Focus of Attention". In: *IEEE Transactions on Visualization and Computer Graphics* 12.5 (Oct. 2006), pp. 933–940. url: http://www.cg.tuwien.ac.at/research/publications/2006/vis-foa/.

[P100] Cao Weiqun, Bao Hujun, and Peng Qunsheng. "An Algorithm for LOD by Merging Near Coplanar Faces Based on Gauss Sphere". In: *J. Comput. Sci. Technol.* 16.5 (Sept. 2001), pp. 450–457. issn: 1000-9000. doi: 10.1007/BF02948963. url: http://dx.doi.org/10.1007/BF02948963.

[P101] Nathaniel Williams et al. "Perceptually Guided Simplification of Lit, Textured Meshes". In: *Proceedings of the 2003 Symposium on Interactive 3D Graphics*. I3D '03. Monterey, California: ACM, 2003, pp. 113–121. isbn: 1-58113-645-5. doi: 10.1145/641480.641503. url: http://doi.acm.org/10.1145/641480.641503.

[P102] Burkhard Wünsche. "A Survey and Evaluation of Mesh Reduction Techniques". In: *Image and Vision Computing New Zealand, International Conference (IVCNZ)*. 1998, pp. 393–398.

[P103] Burkhard C. Wünsche. "A survey and analysis of common polygonization methods and optimization". In: *Machine Graphics and Vision* 6.4 (1997), pp. 451–486.

[P104] Julie C. Xia, Jihad El-Sana, and Amitabh Varshney. "Adaptive Real-Time Level-of-Detail-Based Rendering for Polygonal Models". In: *IEEE Transactions on Visualization and Computer Graphics* 3.2 (Apr. 1997), pp. 171–183. issn: 1077-2626. doi: 10.1109/2945.597799. url: http://dx.doi.org/10.1109/2945.597799.

[P105] M. Yirci and I. Ulusoy. "A comparative study on polygonal mesh simplification algorithms". In: *Signal Processing and Communications Applications Conference, IEEE 17th*. IEEE, 2009, pp. 736–739.

[P106] Lu Yongquan et al. "Parallel Implementation of Mesh Simplification on a Beowulf Cluster". In: *Proceedings of the 2010 Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science*. DCABES '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 160–164. isbn: 978-0-7695-4110-5. doi: 10.1109/DCABES.2010.37. url: http://dx.doi.org/10.1109/DCABES.2010.37.

[P107] Sung-Eui Yoon et al. "Fast Collision Detection Between Massive Models Using Dynamic Simplification". In: *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. SGP '04. Nice, France: ACM, 2004, pp. 136–146. isbn: 3-905673-13-4. doi: 10.1145/1057432.1057450. url: http://doi.acm.org/10.1145/1057432.1057450.

[P108] Eugene Zhang and Greg Turk. "Visibility-guided Simplification". In: *Proceedings of the Conference on Visualization '02*. VIS '02. Boston, Massachusetts: IEEE Computer Society, 2002, pp. 267–274. isbn: 0-7803-7498-3. url: http://dl.acm.org/citation.cfm?id=602099.602140.

[P109] Tang Zhanhong and Yan Shutian. "A Mesh Simplification Algorithm Based On Curvature Factor Of Collapsing Edge". In: *2nd International Workshop on Database Technology and Applications (DBTA)*. IEEE Computer Society, 2010, pp. 1–3.

[P110] Pan Zhigeng, Shi Jiaoying, and Zhou Kun. "A New Mesh Simplification Algorithm Based on Triangle Collapses". In: *J. Comput. Sci. Technol.* 16.1 (Jan. 2001), pp. 57–63. issn: 1000-9000. doi: 10.1007/BF02948853. url: http://dx.doi.org/10.1007/BF02948853.

[P111] 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*. SIGGRAPH '96. New York, NY, USA: ACM, 1996, pp. 189–192. isbn: 0-89791-746-4. doi: 10.1145/237170.237254. url: http://doi.acm.org/10.1145/237170.237254.