Angle-Bounded 2D Mesh Simplification

Steffen Hinderink*, Manish Mandad, Marcel Campen

Osnabrück University, Germany

Abstract

We describe a method to simplify a 2D triangle mesh through decimation while preserving its quality by means of maintaining a strict lower bound on inner angles. Conformance to boundaries, interfaces, and feature constraints is preserved as well. Multiple options and strategies for the choice of local decimation operators and for the settling of geometric degrees of freedom are proposed and explored. A systematic evaluation leads to a final recommendation for the most beneficial approach. The resulting method enables the efficient generation of more parsimonious meshes than those obtained from existing methods, while exhibiting the same quality in terms of worst element shape, as is relevant, for instance, in finite element analysis and numerical simulation.

Keywords: Mesh decimation, Guaranteed-quality triangulation, Angle bound, Triangle collapse

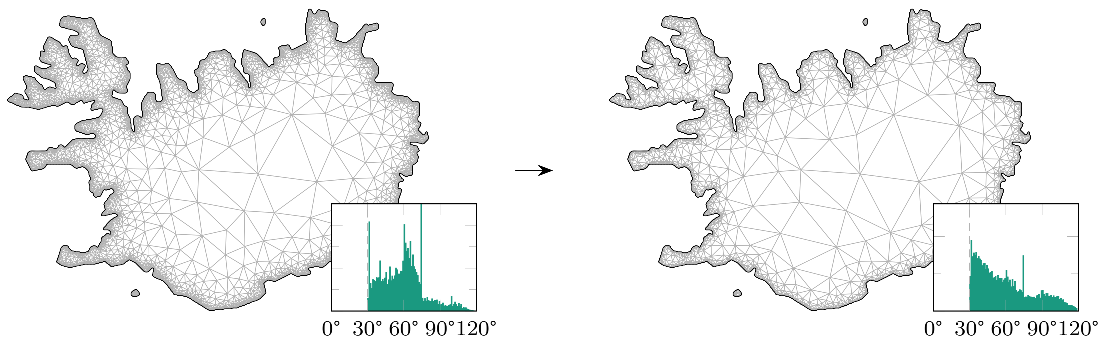


Figure 1: Left: Mesh generated using Delaunay refinement (Üngör, 2004; Shewchuk, 1996) with a lower angle bound of 30°. Right: More parsimonious mesh generated by our simplification method, respecting the same bound, but consisting of just 10 120 instead of 17 580 triangles. The histograms show the distribution of inner angles.

1. Introduction

The discretization of domains by means of meshes, in particular triangles meshes, is a cornerstone of numerous application areas. Simulation based on the finite element method (FEM) and related techniques is a prime example. The quality of the mesh is long-known to have a fundamental effect on performance and other important, theoretical as well as practical, aspects (Zlámal, 1968).

The quality of a mesh is a function of the quality of its individual elements (as well as further aspects like geometric domain approximation error). An important observation is that often not the average but rather the worst element is determinative of a mesh's overall quality and suitability for numerical methods.

*Corresponding author

Email addresses: `sthinderink@uos.de` (Steffen Hinderink), `manishmandad@gmail.com` (Manish Mandad), `campen@uos.de` (Marcel Campen)

In particular, the maximum eigenvalue of the stiffness matrix (and thereby its condition) is dominated by the single worst element (Shewchuk, 2002b; Kim et al., 2019; Fried, 1972), negatively affecting stability, efficiency, and convergence in practice (Sastry et al., 2014). Therefore major efforts have been spent to develop mesh generation methods that produce meshes with a focus on the quality of the worst contained element.

For the case of triangle meshes for 2D domains, where the triangle's inner angles are particularly discriminative, various techniques have been proposed to generate boundary conforming meshes with a lower bound (and by implication also upper bound) on the inner angles. Influential works include Chew (1989, 1993); Bern et al. (1994); Ruppert (1995); Shewchuk (2002a); Miller et al. (2003); Üngör (2004); Erten and Üngör (2009); Rand (2011).

We observe that meshes resulting from such methods as well as from other sources are not necessarily parsimonious. Typically simpler meshes (with a smaller number of elements) do exist, conforming to the same domain boundary, that have exactly the same quality in terms of minimum inner angle, cf. Figure 1. In a sense, the meshes are not tight on the lower bound; most inner angles are far from this bound. While parsimony may not be a relevant objective in some applications, in others it is; one recent example of a use case that, as an ingredient, asks for an as simple as possible constrained triangulation with a lower angle bound is the work in Mandad and Campen (2021).

1.1. Contribution

Based on this observation, we consider the problem of angle-bounded 2D triangle mesh simplification. We describe a method that takes as input a lower angle bound θ^*, a piecewise-linear domain boundary (possibly including internal feature or interface curves), and a boundary conforming triangle mesh M whose inner angles are larger than θ^* (with possible local exceptions). It outputs a mesh M' conforming to the same boundary, whose inner angles are still lower-bounded by θ^* (with the same or less local exceptions), such that $|M'| \leq |M|$, aiming for a low number of elements. A precise problem description follows in Section 3.

The method follows the general idea of incremental mesh decimation, as summarized by Kobbelt et al. (1998). We propose multiple strategies for the choice of mesh modification operators (halfedge, edge, and triangle collapses in combination with different prioritization modes) and in particular for the angle-driven settling of geometric degrees of freedom. These are systematically evaluated so as to arrive at a general recommendation for the most suitable approach.

2. Related Work

The topic of triangle mesh simplification or decimation has already received significant attention. Surveys such as those by Rossignac (1997); Ciampalini et al. (1997); Cignoni et al. (1998) provide a broad overview. Often, the focus is on the decimation of surface meshes in 3D. A pretty common objective therefore is the minimization of approximation error relative to the surface described by the input mesh, and a multitude of strategies to track and reduce this error in the decimation process have been proposed.

This objective, of course, is not relevant to the here addressed plane triangulation setting. Nevertheless, some of these methods may additionally take further aspects besides geometric approximation error into account. Kobbelt et al. (1998) describe the general framework of incremental mesh decimation, and mention the option to preclude decimation operations that violate certain fairness criteria. In this context, Botsch et al. (2010) concretely mention the roundness of triangles and the dihedral angles as possible criteria. These and further criteria like limits on edge lengths and aspect ratios of triangles are implemented, for instance, in the toolbox of Möbius and Kobbelt (2010).

The explicit consideration of inner angles in this context, however, is surprisingly rare. Li and Zhang (2006) use as criterion, in a surface setting, that the inner angles may not exceed an upper bound. This bound is fixed (90°). Edge collapses are used as decimation operator (cf. Section 4.1). We consider multiple operators, use a generic bound, and in particular propose strategies to optimize vertex positions in the process not driven by surface approximation error but in an angle-driven manner, towards the goal of increasing the efficacy of the simplification.

Besides the approach of taking an existing mesh (from arbitrary sources) as input and simplifying it, some techniques have been described, e.g., Erten and Üngör (2009); Tournois et al. (2008), to intervene during the mesh generation process, in particular inside the Delaunay refinement process, modifying it such that on average coarser meshes may be obtained (possibly at the expense of guarantees). Our approach is agnostic to the source of the mesh or its manner of generation; it can take any mesh as input and perform decimation, targeting any lower angle bound.

3. Problem Statement

Let us make precise the setting and the objective. First, assume the following input:

- L is a set of line segments in the plane, defining a piecewise-linear domain boundary, and possibly additional piecewise-linear feature curves in the domain's interior.
- $M = (V, E, T)$ is a triangle mesh, with vertices V, edges E and triangles T, that conforms to L, i.e. each line segment $l \in L$ coincides with a set of edges of M.
- θ^* is a user-set parameter that defines the desired lower bound on inner angles.

Furthermore, let $w(M, \theta^*)$ denote the number of angles in M that violate the bound. Based on this input, we generate a mesh M', by modification of M, such that

- M' conforms to L,
- $w(M', \theta^*) \leq w(M, \theta^*)$,
- $|M'| \leq |M|$.

Our objective is to minimize $|M'|$, the number of elements in M', which we approach in a greedy manner. Note that the second condition above implies that if M respects the angle bound θ^*, M' does so as well. If M partially violates it (e.g. near sharp corners prescribed by L, where this may be inevitable), M' does not contain any further (but possibly less) violations. Optionally, we may, even more strictly, want to prevent each individual violation from becoming more severe.

A typical source for the input mesh M are constrained triangulation algorithms with quality guarantees, like those mentioned in Section 1. For instance, using Delaunay refinement (Ruppert, 1995) lower angle bounds up to 28.6° (except in the vicinity of sharp angles prescribed by the constraints L) can be guaranteed.

4. Decimation

To be able to reliably and strictly prevent any kind of violation of conformity or angle bound, we generate M' out of M by means of a series of constrained mesh modification operations, within the general framework of incremental mesh decimation (Kobbelt et al., 1998). We design and constrain the operations such that conformity and quality are preserved, while the number of elements decreases strictly monotonically.

The angle requirement is maintained by only allowing those operations that lead to meshes not violating it. If some triangle corners form angles less than θ^* already in M, we only allow improvement, i.e. an increase, but not any further decrease of this angle. Therefore, the individual lower bound for a triangle corner that has an angle of β in M is $\theta = \min(\theta^*, \beta)$. A mesh modification operation is considered legal if $\beta' \geq \theta$, where β' is the angle of the same corner after deformation of the triangle due to the operation.

We briefly review the employed operators in Section 4.1 and address the question of how to settle the involved geometric degrees of freedom (the choice of vertex positions) so as to enable a large amount of decimation in Section 4.2.

4.1. Operators

We use and explore multiple local mesh modification operators for an incremental decimation of M, as outlined in the following.

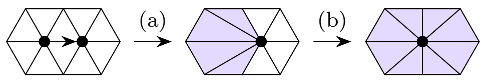


Figure 2: An edge collapse can be viewed as a halfedge collapse (a), followed by relocating the merger vertex (b) to a new position determined by some rule. The triangles whose shape changes in the process are highlighted.

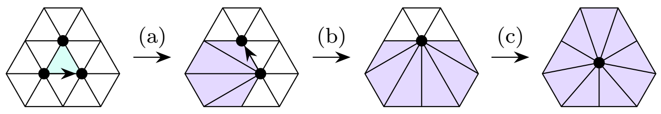


Figure 3: The collapse of a triangle (green) can be viewed as a sequence of two halfedge collapses (a, b), followed by relocating the merger vertex (c). The triangles whose shape changes in the process are highlighted.

Halfedge collapse. Each edge $e \in E$ can be considered to be consisting of two halfedges, pointing in opposite directions. A halfedge collapse for a halfedge h of e is a standard operation that simplifies a mesh by conceptually merging the vertex where h originates into the vertex that h points to, as illustrated in Figure 2a. This in particular deletes the two triangles incident to the edge e.

Edge collapse. In an edge collapse, the two vertices of an edge $e \in E$ are merged into one vertex at a new position. Both triangles incident to e are deleted. It can be implemented as a halfedge collapse followed by a vertex shift, as illustrated in Figure 2. In contrast to the halfedge collapse, this operation has geometric degrees of freedom, in form of the position of the merger vertex.

Triangle collapse. We additionally propose the use of a *triangle collapse* operation, merging all three vertices of a triangle $\Delta \in T$ into one vertex at a new position. The three triangles that are edge-adjacent to Δ vanish. While an operation of this kind has been used in the early days of mesh processing (Hamann, 1994), later work has often preferred (half)edge collapses for their finer granularity. For instance, Kobbelt et al. (1998) "recommend to make the topological operator itself as simple as possible", suggesting the exclusive use of the halfedge collapse. Indeed, the effect of a triangle collapse, in terms of connectivity, can be achieved by a sequence of two (half)edge collapses (cf. Figure 3). However, in our particular constrained setting, for the intermediate connectivity between the two edge collapses there may not be a legal geometric state within the angle-bounded configuration space. Especially as, in a sense, the triangle collapse operation is typically less anisotropic than an edge collapse, we may expect it to enable additional decimation operations. As our evaluation (cf. Section 5) shows, this is indeed the case to a certain extent.

For each of these operators, we need to define under which circumstances they are *legal* for which halfedge, edge, or triangle – in the sense that the desired angle bounds as well as boundary and feature conformance are maintained if the respective operator is applied. For the halfedge collapse this is a simple matter:

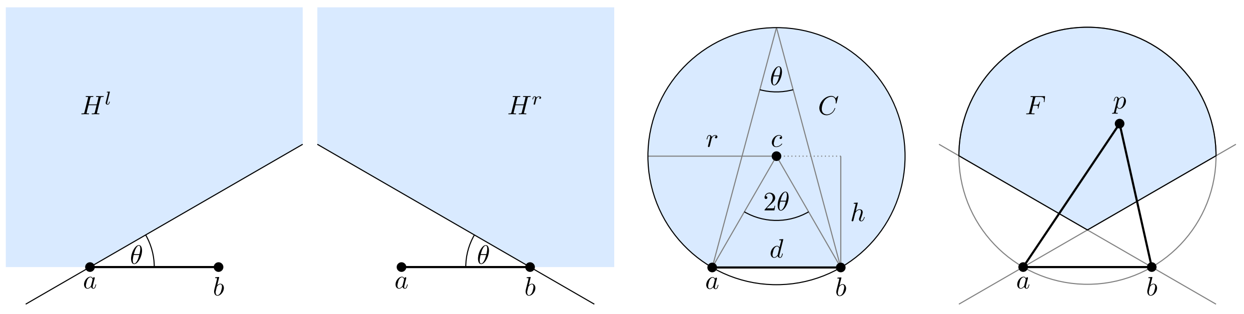
- Maintenance of angle bounds is easily checked by measuring tentative post-collapse angles of the affected triangles (those highlighted in Figure 2 center).
- In case the to-be-collapsed vertex lies on a constraint segment $l \in L$, conformance is preserved if exactly two of its incident edges, including the to-be-collapsed edge, lie on the same constraint segment $l \in L$.

Note that in this, angles need to be computed in a *signed* manner to prevent triangles from folding over.

The other operators, however, have geometric degrees of freedom; their legality depends on the choice of the new vertex position, as it affects the inner angles of the incident triangles. In the following we propose and discuss multiple options to settle these degrees of freedom.

4.2. Geometric Degrees of Freedom

The edge collapse as well as the triangle collapse operation have geometric degrees of freedom: the choice of the position p of the merger vertex. Recall that for every triangle all three angles must be greater than or equal to θ. The angles of the triangles that are incident to the merger vertex depend on its position p. Let us determine the *angle-bounded kernel*, the set $K \subseteq \mathbb{R}^2$ such that $p \in K$ if and only if all incident triangles have inner angles lower-bounded by their respective bound θ.



(a) Halfplane through a. (b) Halfplane through b. (c) Circle through a and b. (d) Common intersection.

Figure 4: Fan-shaped intersection of the halfplanes and the circle.

First, consider a single incident triangle, with vertices (in counter-clockwise order) at positions a, b, p. The edge between a and b is illustrated in Figure 4. In order to have positive inner angles, p must lie above this edge. Furthermore, each one of the triangle's three corners implies a constraint on the legal positions for p:

- For the angle at a to be large enough, p must lie in the closed halfplane H^l above the line passing through a that forms a counter-clockwise angle θ with $b - a$ (Figure 4a).
- For the angle at b to be large enough, p must lie in the closed halfplane H^r above the line passing through b that forms a counter-clockwise angle $-\theta$ with $b - a$ (Figure 4b).
- For the angle at p to be large enough, according to the inscribed angle theorem p must lie in the closed disk C formed by a circle with center c and radius r passing through a and b (Figure 4c). Center c lies on the bisector of the edge ab, at height h, computed as

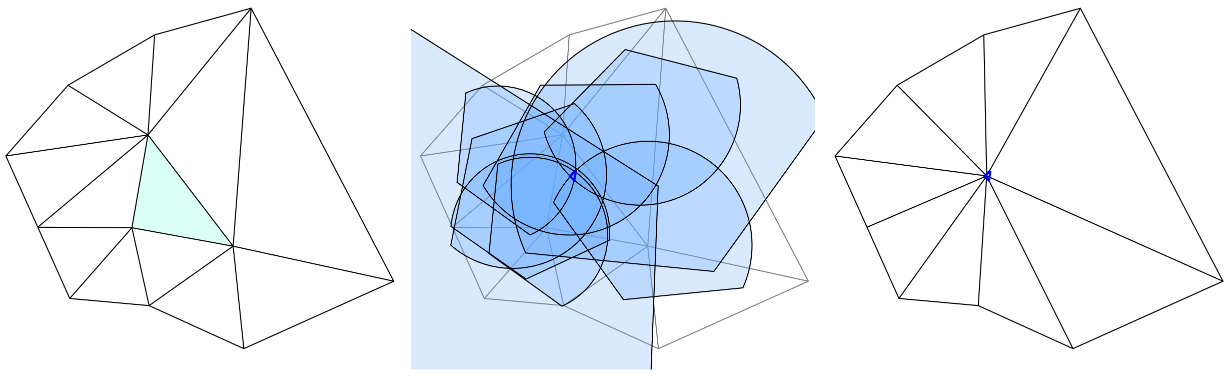
$$h = \frac{d}{2\tan\theta} \quad \text{and} \quad r = \frac{d}{2\sin\theta} = \sqrt{h^2 + \frac{d^2}{4}}\ , \quad \text{where} \quad d = \|b - a\|\ .$$

All conditions for this single triangle are satisfied if the point p lies in the fan-shaped intersection $F = H^l \cap H^r \cap C$ of these two halfplanes and the circular disk, illustrated exemplarily in Figure 4d for the case of $\theta = 30°$.

Let n be the number of triangles incident at the vertex. The sought angle-bounded kernel is the intersection $K = \bigcap_{i\in[n]} F_i$ of the fan-shaped regions of these triangles. $[n]$ denotes the index set $\{1, \ldots, n\}$. Note that K, as the intersections of convex sets (halfplanes and disks), is convex. Furthermore, its boundary is an arc-polygon, formed by line segments and circular arc segments. For the case of a triangle collapse, the kernel K for the merger vertex is illustrated in Figure 5. Note that each edge in the triangle's *link* (the nine outer edges in Figure 5a) contributes one fan-shaped region.

Only if the position p of the merger vertex of an edge collapse or triangle collapse is chosen such that it lies inside K, may this collapse be legal. If K is empty, the edge e or the triangle Δ, respectively, is generally not collapsible. One may hypothesize that a choice far in the interior of K is beneficial compared to a choice on or close to its boundary, because it leaves a larger gap towards the lower angle bound, potentially leaving more flexibility for legal subsequent collapses. We consider multiple options for the choice of p in the following, and evaluate their relative performance in Section 5.

Note that if any of the two or three vertices involved in an edge or triangle collapse lies on a constraint from L, for simplicity we do not consider this collapse, so as to avoid multiple case distinctions and special case handling. Instead, such vertices are left to be handled by L-constrained halfedge collapses described in Section 4.1.



(a) Triangle and surround before the collapse. (b) All nine fans and their intersection K. (c) Possible configuration after the collapse.

Figure 5: Angle-bounded kernel (bold blue), defined as the intersection of fans (light blue), when collapsing a triangle (green).

4.2.1. Centroid

A simplistic (and computationally cheap) choice for p is the center of the edge or the triangle that is collapsed, i.e. the average of the two or three involved vertices. If this choice implies violating angles, i.e. if the centroid does not lie inside the angle-bounded kernel K, the collapse is considered illegal.

This simple choice is conservative: a collapse may be prohibited even if K is non-empty. The three further (more expensive) strategies proposed in the following, by contrast, always yield some point inside the angle-bounded kernel if it is non-empty.

4.2.2. Kernel Mean Construction

One way to choose a point inside the angle-bounded kernel K is to explicitly construct this set, and then pick a point from it. For instance, we can compute the corners $\{p_1, \ldots, p_k\}$ of the arc-polygonal kernel boundary and choose the mean $p = \frac{1}{k} \sum p_i$, which – due to convexity of K – is guaranteed to lie inside.

The corners p_i are exactly those intersection points of any two of the involved kernel-defining lines and circles that lie within all of the respective closed halfplanes H and disks C, i.e. on or above the lines in Figure 4ab and on or inside the circle in Figure 4c, respectively. A simple way for their computation therefore is the calculation of pairwise intersections (line-line, line-circle, and circle-circle intersections), followed by filtering out those intersection points that lie outside of any halfplane or disk.

While this simple approach has cubic run time complexity (and asymptotically more efficient incremental convex arc-polygon clipping algorithms can be imagined), note that it is cubic in the valence n of the vertex only, and that this valence is upper-bounded by a small constant due to the inner angle bound θ. In particular, if the post-collapse valence exceeds $360°/\theta$ (e.g. 12 in case of a constant bound $\theta = 30°$), we can immediately conclude that K must be empty.

4.2.3. QCQP Formulation

The angle-bounded kernel K of a vertex can be characterized by a set of linear and quadratic constraints. Let $o_i + \lambda v_i$ for $\lambda \in \mathbb{R}$, $\|v_i\| = 1$, $i \in [2n]$ be the lines that bound the involved halfplanes H_i to their left, and let c_j and r_j be the centers and radii of the involved disks C_j for $j \in [n]$, where n is the number of edges of the link. Furthermore, define the lines' normals (s_i, t_i) via $s_i = -v_{iy}$, $t_i = v_{ix}$, and their signed distance to the origin as $u_i = v_{iy} o_{ix} - v_{ix} o_{iy}$. Then K is defined by the following set of convex linear and quadratic constraints:

$$\begin{aligned} s_i p_x + t_i p_y + u_i &\geq 0 \quad \forall i \in [2n] \quad \text{and} \\ {r_j}^2 - {c_j}_x^{\,2} - {c_j}_y^{\,2} + 2{c_j}_x p_x + 2{c_j}_y p_y - {p_x}^2 - {p_y}^2 &\geq 0 \quad \forall j \in [n] \ . \end{aligned}$$

A point $p = (p_x, p_y)$ inside K can therefore be found using a solver for quadratically constrained programs. In particular, we can model the following problem within the QCQP class of optimization problems (Boyd et al., 2004), that additionally aims to yield not just any point within K, but a point that is well-centered:

$$\begin{aligned} &\text{Maximize} && \delta && \text{(1a)}\\ &\text{subject to} && s_i p_x + t_i p_y + u_i - \delta \geq 0 \quad \forall i \in [2n] && \text{(1b)}\\ & && {r_j}^2 - {c_j}_x^{\,2} - {c_j}_y^{\,2} + 2{c_j}_x p_x + 2{c_j}_y p_y - {p_x}^2 - {p_y}^2 - \delta \geq 0 \quad \forall j \in [n]\ . && \text{(1c)} \end{aligned}$$

Here a dummy variable δ is introduced that effectively measures the minimum over the signed distances of p to the to the lines (1b) and the signed squared distances to the circles (1c) – and this minimum is maximized. If $\delta \geq 0$ in the program's solution, p is valid; otherwise K is empty.

However, distances to lines and to circles are treated differently in this formulation. For an even better-centered solution point p, we can define a modified QCQP:

$$\begin{aligned} &\text{Maximize} && \delta && \text{(2a)}\\ &\text{subject to} && (s_i p_x + t_i p_y + u_i)^2 - \delta \geq 0 \quad \forall i \in [2n] && \text{(2b)}\\ & && {r_j}^2 - {c_j}_x^{\,2} - {c_j}_y^{\,2} + 2{c_j}_x p_x + 2{c_j}_y p_y - {p_x}^2 - {p_y}^2 - \delta \geq 0 \quad \forall j \in [n] && \text{(2c)}\\ & && s_k p_x + t_k p_y + u_k \geq 0 \quad \forall k \in [2n]\ . && \text{(2d)} \end{aligned}$$

In this, distances to circles as well as distances to lines are taken into account consistently in squared form; notice the difference between (2b) and (1b). The additional constraints (2d) ensure that p lies on the proper side of each line constraint; they are necessary because the squared distance in (2b) is unsigned.

4.2.4. Smooth Quasiconcave Maximization

The strategy in the previous section maximizes the minimum (squared) distance of p to the lines and circles, and therefore yields the point p inside K most distant from the boundary of K (where it would imply an inner angle tight on the lower bound). This does not imply, however, that this point actually maximizes the minimum inner angle it implies.

A further strategy we consider therefore is the direct maximization of the minimum inner angle implied. To this end, let

$$\varphi : \mathbb{R}^2 \to \mathbb{R}^{3n}$$

denote the $3n$ inner angles of all n triangles incident on a vertex, dependent on its position p. Each incident triangle, with, besides p, constant vertex positions a and b, contributes three elements φ_i to φ, and each φ_i is of one of three forms:

$$\varphi^{<}_{ab}(p) = \arccos \left\langle \frac{b-a}{\|b-a\|}, \frac{p-a}{\|p-a\|} \right\rangle,$$
$$\varphi^{\wedge}_{ab}(p) = \arccos \left\langle \frac{a-p}{\|a-p\|}, \frac{b-p}{\|b-p\|} \right\rangle,$$

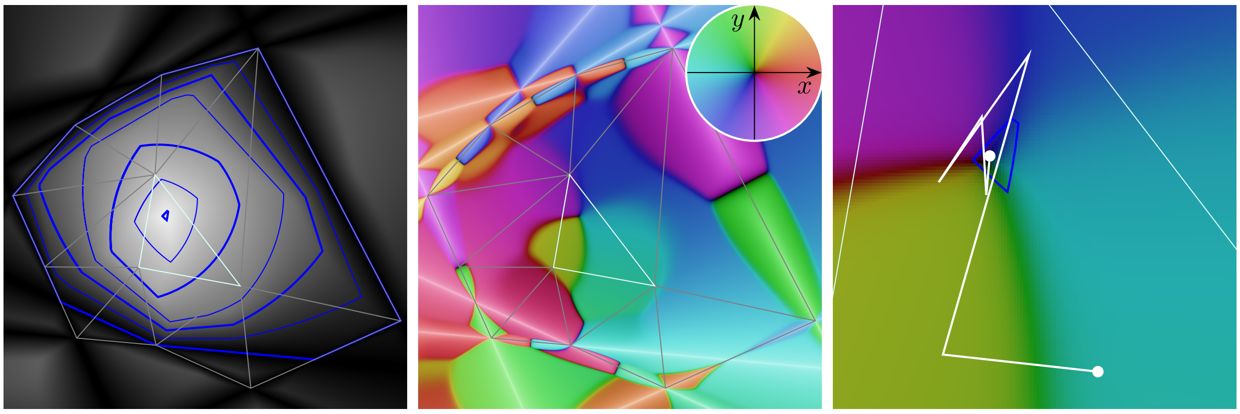
and $\varphi^{>}_{ab} = \varphi^{<}_{ba}$, measuring the angles at a, p, and b, respectively. We would like to choose p such that

$$f^*(p) = \min_{i \in [3n]} \varphi_i(p) \to \max\,. \tag{3}$$

This raises the question whether finding this maximum, given the nonlinear nature of f^*, is practically feasible.

Proposition 1. *f^* is quasiconcave inside of the visibility kernel of the central vertex's link polygon.*

Proof. The point p lying inside the visibility kernel is equivalent to all incident triangles having positive orientation. In this regime $\varphi^{<}_{ab}(p)$, $\varphi^{\wedge}_{ab}(p)$, and $\varphi^{>}_{ab}(p)$ clearly are quasiconcave, and therefore each φ_i for $i \in [3n]$. As the minimum of quasiconcave functions is quasiconcave, f^* is quasiconcave. □



(a) Greyscale visualization of the smooth minimum f over all angles affected by the collapse of a triangle (green). Level sets (0°, 5°, 10°, 15°, 20°, 25°, 30°) shown in blue; the outermost (0°) forms the visibility kernel of the link polygon.

(b) Color-coded gradient ∇f on the same configuration. The gradient's direction is hue-coded, its magnitude brightness-coded as shown in the inset.

(c) The path of a gradient ascent, following the gradient towards the global maximum, here lying inside the angle-bounded kernel for $\theta = 30°$ (blue).

Figure 6: Illustration of minimum inner angle maximization for the case of a triangle collapse.

As an alternative argument, recall that the angle-bounded kernel K defined in Section 4.2 is convex for any angle $\theta > 0$. Notice that these angle-bounded kernels are exactly the upper level sets of f^*, i.e. $K = \{p \mid f^*(p) \geq \theta\}$. Convexity of all upper level sets implies quasiconcavity as well (Arrow and Enthoven, 1961).

Quasiconcavity implies a unique maximum (Mangasarian, 1994) and enables us to reliably find it using a gradient ascent – which, however, additionally requires differentiability. f^*, unfortunately, is not differentiable because of the involved minimum. We can, however, instead employ a differentiable *smooth minimum* and define $f(p) = \mathrm{smin}_\alpha(\varphi(p))$ with

$$\begin{aligned} \mathrm{smin}_\alpha &: \mathbb{R}^{3n} \to \mathbb{R} \\ t &\mapsto \frac{1}{\alpha} \log \sum_{i \in [3n]} \exp(\alpha t_i) \ . \end{aligned}$$

This smin_α is known as the (scaled) LogSumExp function. For $\alpha < 0$ it is a smooth minimum-like function, for $\alpha \to -\infty$ it converges to the minimum function. Note that smin_α, for $\alpha < 0$, is a concave function and therefore f like f^* is quasiconcave, such that it can be used for a gradient ascent. Furthermore, $f(p) \leq f^*(p)\ \forall p \in \mathbb{R}^2$.

Figure 6a illustrates this function f (with $\alpha = -10^2$, our default used throughout) for the same local mesh configuration as in Figure 5. Some isocurves of f^* are marked in blue, the outermost one ($f^* = 0°$) bounds the visibility kernel, and the innermost one ($f^* = 30°$) is exactly the intersection K from Figure 5b.

Gradient. According to the chain rule the gradient is given by $\nabla f(p) = J_{\mathrm{smin}}(\varphi(p)) \cdot J_\varphi(p)$ with J being the respective Jacobian matrices. The partial derivatives of smin_α are simply

$$\frac{\partial}{\partial t_i} \mathrm{smin}_\alpha(t) = \frac{\exp(\alpha t_i)}{\sum_{j \in [3n]} \exp(\alpha t_j)} \ .$$

The partial derivatives of the elements of φ are a bit lengthier, but still easy to evaluate:

$$\frac{\partial}{\partial x_i}\varphi^{<}_{ab}(p) = -\left(\frac{b_{x_i} - a_{x_i}}{\|b-a\|\|p-a\|} - \frac{(p_{x_i} - a_{x_i})\langle b-a, p-a\rangle}{\|b-a\|\langle p-a, p-a\rangle^{\frac{3}{2}}}\right)\left(1 - \frac{\langle b-a, p-a\rangle^2}{\langle b-a, b-a\rangle\langle p-a, p-a\rangle}\right)^{-\frac{1}{2}} \quad \text{and}$$

$$\frac{\partial}{\partial x_i}\varphi^{\wedge}_{ab}(p) = -\left(\frac{-a_{x_i} - b_{x_i} + 2p_{x_i}}{\|a-p\|\|b-p\|} + \frac{(a_{x_i} - p_{x_i})\langle a-p, b-p\rangle}{\|b-p\|\langle a-p, a-p\rangle^{\frac{3}{2}}} + \frac{(b_{x_i} - p_{x_i})\langle a-p, b-p\rangle}{\|a-p\|\langle b-p, b-p\rangle^{\frac{3}{2}}}\right) \cdot \left(1 - \frac{\langle a-p, b-p\rangle^2}{\langle a-p, a-p\rangle\langle b-p, b-p\rangle}\right)^{-\frac{1}{2}}.$$

Figure 6b illustrates the gradient ∇f for the configuration from Figure 6a.

Gradient Ascent. Using the above gradient, we can perform an iterative gradient ascent. As long as we start from a point within the visibility kernel (i.e. a point that implies no flipped triangle), it will converge towards the maximum of f. In the ascent step, we employ a backtracking line search with a simple adaptive step size selection: the step size is halved until the current step leads to an improved target function value, i.e. an increase in f, and the next step's line search starts with twice the successful step size from the previous iteration. We terminate the line search if the step size drops below $10^{-3}\times$ the shortest edge length in the vertex's link, and impose a limit of 100 halvings. Figure 6c shows a zoomed in version of Figure 6b with the steps of the gradient ascent, starting from the triangle's center point, drawn in white.

4.3. Prioritization

Having a set of local decimation operators as well as methods for the choice of vertex positions at hand, a remaining question is where and in which order these operators shall be applied. As a collapse affects certain inner angles that are relevant also for other collapses and their legality, the order in which potential collapses are attempted affects how many collapses will actually be legally possible.

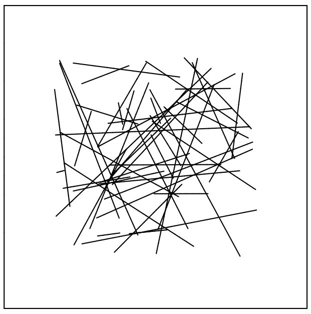
A baseline option is the random choice of order, i.e. halfedges, edges, or triangles are selected in random order, and their collapse (using one of the above vertex positioning choices) executed if legal.

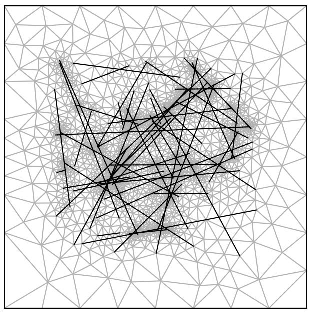
Alternatively, we can proceed greedily following some priority. A potentially beneficial option is to always perform that collapse that, among all legal collapses, implies the maximum minimum angle among all affected inner angles – in the hope that this leaves maximum freedom for subsequent collapses to succeed. To efficiently implement this, initially the badness of each possible collapse is evaluated: the minimum angle that would be formed if the collapse was executed. The collapses are then sorted in a priority queue accordingly. One after the other they are then taken from the queue (starting from the least bad) and executed if possible. Whenever a collapse is executed, the mesh's connectivity and its inner angles change in a very local neighborhood region. The badness scores of queue elements associated with this region are updated and the queue resorted accordingly. A heap-based implementation is appropriate for that matter. Compared to the random order, this sorted order causes additional computational cost, in the form of maintaining the sorted priority queue and of computing badness scores multiple times per potential operations.

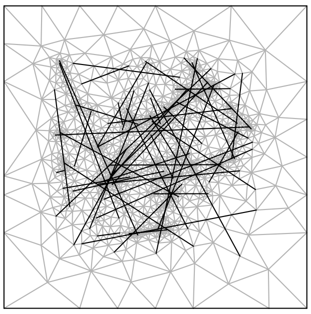
Additionally, prioritizing triangle collapses, which affect a larger region, over edge collapses could have a positive effect, so we also evaluate this option.

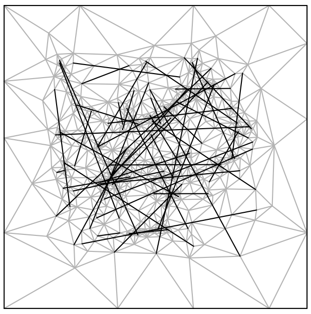
5. Evaluation

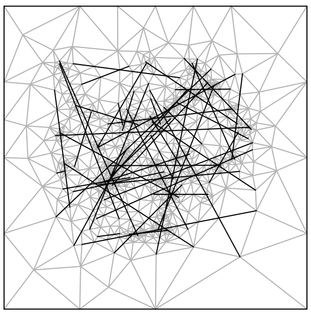
With a number of decimation operators, vertex positioning choices, and prioritization options at hand, we aim to evaluate the relative performance of the various combinations, in particular in terms of the degree of simplification that is achieved and the computational time required.


(a) L.


(b) M_{30°, $|T| = 10\,887$.


(c) $M'^{30^\circ}_{30^\circ}$, $|T| = 7386$.


(d) $M'^{20^\circ}_{30^\circ}$, $|T| = 3498$.


(e) M_{20°, $|T| = 3606$.

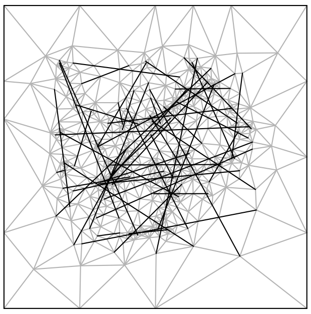

(f) $M'^{20^\circ}_{20^\circ}$, $|T| = 2982$.

Figure 7: Example random constraint set L, meshes M_ψ generated using a constrained meshing algorithm with angle parameter ψ, and meshes $M'^{\theta^*}_\psi$ obtained by simplifying M_ψ using our method (using the ■ configuration) maintaining a lower angle bound θ^*.

5.1. Input Data

We generate several thousand test input meshes for our evaluation. To this end, for each instance, we generate a constraint set L (cf. Section 3) by placing into a bounding box a fixed number of line segments, with end points randomly chosen inside the box (with some margin), and splitting these segments at points of intersection. Figure 7a shows one such resulting constraint set L. We then apply the Delaunay refinement based algorithm of Üngör (2004) as implemented in the Triangle library (v1.6) (Shewchuk, 1996) to generate a triangulation M conforming to L, setting the target minimum angle parameter, referred to as ψ in the following, to 10°, 20°, 25°, or 30°. We generate 1000 such input meshes per angle parameter setting. Figure 7b shows an example for the 30° case, Figure 7e for the 20° case.

We tune the number of random line segments such that the meshes M created that way have around 10 000 triangles for the 30° case. Tests with meshes of sizes at different orders of magnitude did not reveal any significant differences, so we focus our attention on this exemplary dataset.

5.2. Experiments

We perform experiments with various combinations of the options presented above. In terms of collapse types we use either just halfedge collapses, or halfedge and edge collapses, or the full set: halfedge, edge, and triangle collapses. In terms of order we may use prioritization by the minimum implied angle, and we may prioritize triangle collapses whenever possible. All in all, we explore the following combinations, and use the listed symbols to indicate them in upcoming plots:

- ○ halfedge collapses
- ● halfedge collapses + angle priority
- ◇ halfedge, edge collapses
- ◆ halfedge, edge collapses + angle priority
- △ halfedge, edge, triangle collapses

- ▲ halfedge, edge, triangle collapses + angle priority
- ■ halfedge, edge, triangle collapses + angle priority + triangle priority

For each combination that involves not only halfedge collapses, we can furthermore employ one of five different vertex positioning strategies (Section 4.2) which we indicate by colors:

- ● Centroid
- ● Kernel Mean
- ● QCQP (1)
- ● QCQP (2)
- ● Min Angle Maximization (3)

For the latter strategy, there is a parameter α in the employed smooth minimum function. If $|\alpha|$ is chosen too small, the minimum is smoothed heavily, resulting in an ascent to an inaccurate maximum point. A large value, by contrast, may slow down the ascent due to the consequent near-discontinuity of the gradient. Tests lead to $\alpha = -10^2$ as a good trade-off; larger values only slow down the process without noticeable benefit. Also experiments with gradual adjustment of the value in the course of the ascent showed no consistent benefit.

Remark. We additionally experimented with patch collapses (contracting not a single but multiple adjacent triangles to a merger vertex at once) as well as with interleaved vertex shifts (optimizing all vertex positions by means of the angle maximizing gradient ascent from Section 4.2.4). Effects were minuscule, so we exclude them from further consideration.

5.3. Results

We start by considering the case of $\theta^* = 30^\circ$, a practically important case because meshes with angle bounds up to around that value can be generated with high reliability by various methods, while sources for and occurrences of meshes with significantly higher bounds are scarce to begin with.

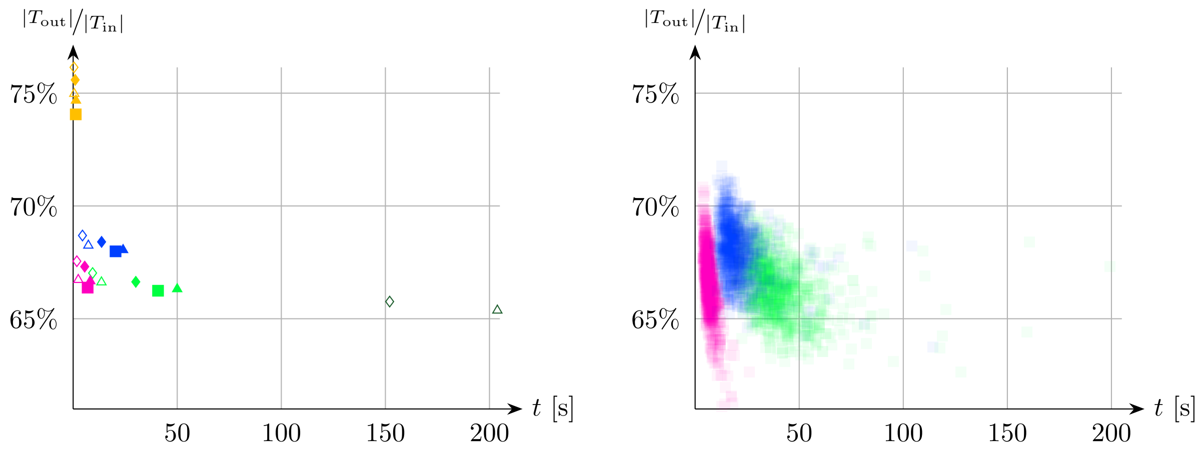
Figure 8 shows the experimental results when applying the various configurations of the simplification method with $\theta^* = 30^\circ$ to the dataset of 30° input meshes, i.e. the existing lower bound is maintained. More precisely, Figure 8a shows the reduction $|T_{\text{out}}|/|T_{\text{in}}|$ and the run time[1] for the various algorithm configurations, averaged over the 1000 test input meshes (cf. Section 5.1).

Positioning strategy. It is observable that the choice of vertex positioning strategy has a particularly clear effect on the results – in terms of run time as well as in terms of achieved reduction. The simple centroid strategy is by far worse than the other strategies in terms of reduction, while the run time advantage over some other configurations is not particularly large. In other words, putting some effort into computing the position of merger vertices does pay off.

Collapse types. Within each positioning strategy, it can be observed that the types of employed collapses matter, too. The more collapse types are used, the greater is the reduction but also the run time. The result of using only halfedge collapses is not shown in the plot; this simplistic approach can be considered impractical, only reducing to $|T_{\text{out}}|/|T_{\text{in}}| \approx 94\%$ on average (albeit in $t \approx 0.1$s).

Prioritization. Regardless of the choice of positioning strategy and the choice of collapse types, the prioritization based on the implied minimum angle further reduces the number of triangles, albeit to a small degree only. Notice how the cost of maintaining the priority queue is reflected in these experiments; in particular, this prioritization in combination with the positioning strategy using QCQP (2) takes impractically long, $t \approx 4$min (achieving a reduction of $|T_{\text{out}}|/|T_{\text{in}}| \approx 65\%$), such that we omit them from the plot. We experimented with various initializations for the QCQP (2) strategy, starting from the position resulting from other positioning strategies, but did not find a setting with a consistently positive effect on run time.

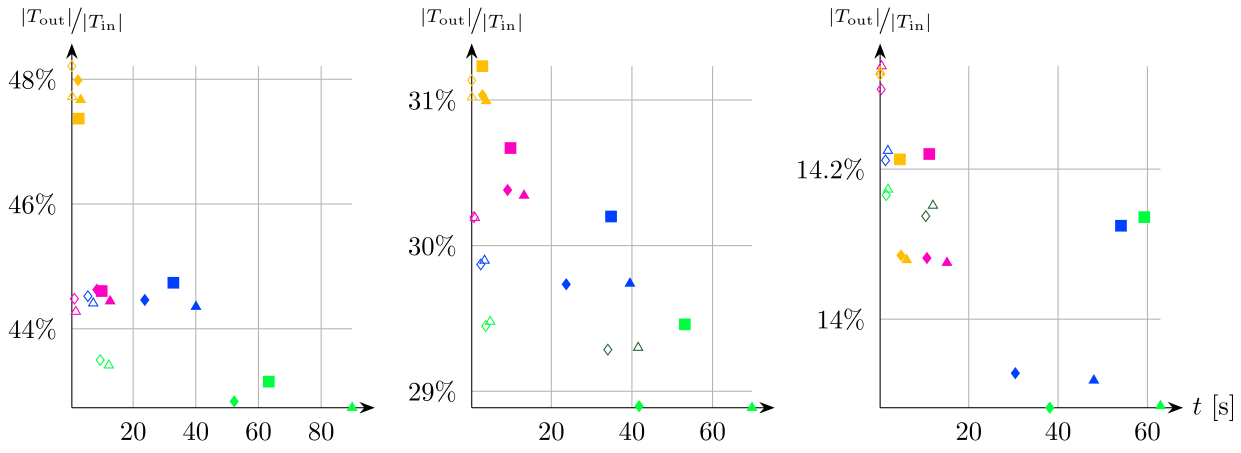
[1] All timings stem from single-threaded execution on an AMD 3970X processor; the QCQP problems were solved using the Gurobi Optimizer 9.5 (`www.gurobi.com`).



(a) $\psi = 30° \;\rightarrow\; \theta^* = 30°$.

(b) $\psi = 30° \;\rightarrow\; \theta^* = 30°$.

Figure 8: Reduction of the number of triangles against run time for various algorithm configurations. Left: averages. Right: per-mesh results for a subset of configurations.



(a) $\psi = 30° \;\rightarrow\; \theta^* = 25°$.

(b) $\psi = 30° \;\rightarrow\; \theta^* = 20°$.

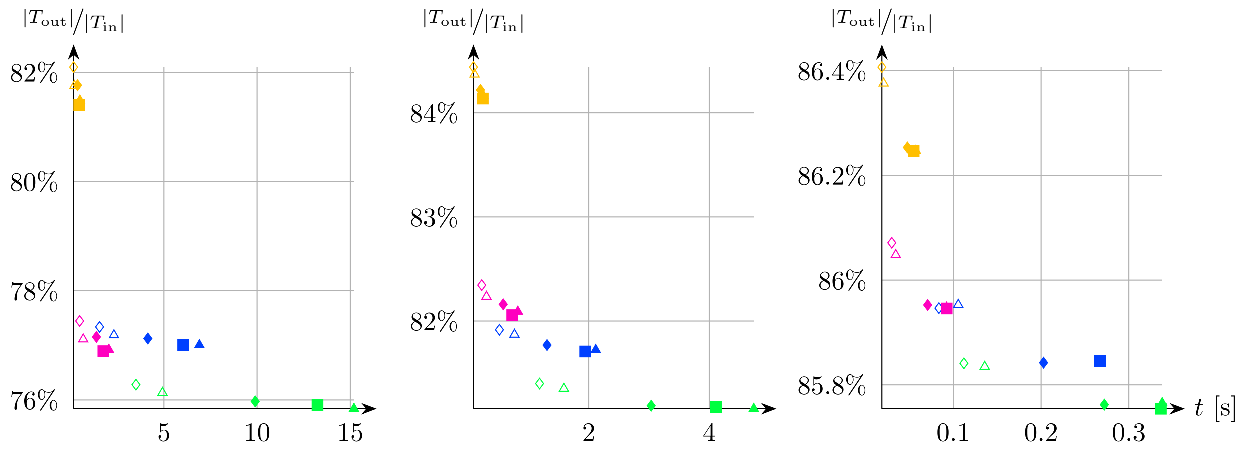
(c) $\psi = 30° \;\rightarrow\; \theta^* = 10°$.

Figure 9: Results when simplifying towards bounds θ^* that are lower than the input meshes' minimum angle ψ.

Finally, additionally giving priority to triangle collapses over other collapses yields the best results in terms of reduction – while interestingly improving the run time, in essence because it reduces the number of position (re)calculations.

Figure 8b shows the individual results per input mesh from the test data set for three algorithm configurations with different vertex positioning strategies. It can be observed that the results of the different configurations are well-clustered and the variance over the different inputs is relatively small, i.e. the relative performance of the different algorithm configurations is quite consistent, at least across the used class of input meshes. Reasonable conclusions are therefore drawable by considering averages, as in Figure 8a. As an individual example, the result of the ■ configuration (all collapse types, minimum angle maximization, prioritization) on the input from Figure 7b is shown in Figure 7c.

Other target bounds θ^*. The results for $\theta^* = 25°$, $\theta^* = 20°$, and $\theta^* = 10°$ are presented in Figure 9. As expected, the lower the bound θ^*, the more reduction can be achieved. Quite interestingly, though, the lower the bound θ^*, the smaller is the difference between the various configurations in terms of reduction: While



(a) $\psi = 25^\circ \;\rightarrow\; \theta^* = 25^\circ$. (b) $\psi = 20^\circ \;\rightarrow\; \theta^* = 20^\circ$. (c) $\psi = 10^\circ \;\rightarrow\; \theta^* = 10^\circ$.

Figure 10: Results for lower initial angles ψ and target bounds θ^*.

in Figure 8a they span more than 10 percentage points, in Figure 9c they span just half a percentage point. It is notable that the gradient ascent approach, which provided the probably practically most reasonable option for the case considered in Figure 8a, is no longer the best choice in the low-angle cases – though the differences are small. Interestingly, the simple centroid strategy provides a good trade-off between run-time and reduction performance in the case of Figure 9c. The prioritization based on angles is still beneficial in most cases, while the prioritization of triangles does not pay off here. One example result for the case $\theta^* = 20^\circ$ is illustrated in Figure 7d.

Other starting bounds ψ. In Figure 10 we consider the case of taking meshes with lower angles ($\psi = 25^\circ$, $\psi = 20^\circ$, and $\psi = 10^\circ$, again 1000 meshes each) as input. As an example, Figure 7e shows one input for the $\psi = 20^\circ$ case, and Figure 7f the corresponding simplified result. The overall picture is similar to the one in Figure 8, in terms of relative performance of the various algorithm configurations, with the kernel mean strategy showing a small relative improvement compared to the $\psi = \theta^* = 30^\circ$ case. In absolute numbers, less reduction is possible – at least when the input meshes are already tailored to the desired bound, as is the case for the test input meshes generated using the Triangle library.

Summary. We conclude that the use of simple halfedge collapses is not a reasonable choice for angle-bounded decimation. The use of edge collapses in combination with a simple centroid (i.e. edge mid point) position choice ◇♦ is only reasonable if run time efficiency is important. If, by contrast, reduction performance is of utmost importance and run time does not matter at all, the full battery together with QCQP (2) ■ is the configuration of choice.

For less extreme scenarios, where a reasonable balance between run time and reduction performance is sought, the positioning strategies min angle maximization ● and kernel mean ● are to be recommended for most cases, where the former has some advantage in terms of implementational simplicity, not requiring a QCQP solver. Especially when the mesh to be simplified is already tailored to some extent to the desired lower bound, the ■ configuration proves to be a good general choice. Figures 1 and 11 show the simplification of several example meshes following that recommendation.

According to our experience from operating with different types of input meshes (larger, smaller, more or less constraints) results can of course differ in terms of absolute numbers, regarding run time as well as regarding the degree of reduction. The above relative conclusions, however, appear to hold quite generally.

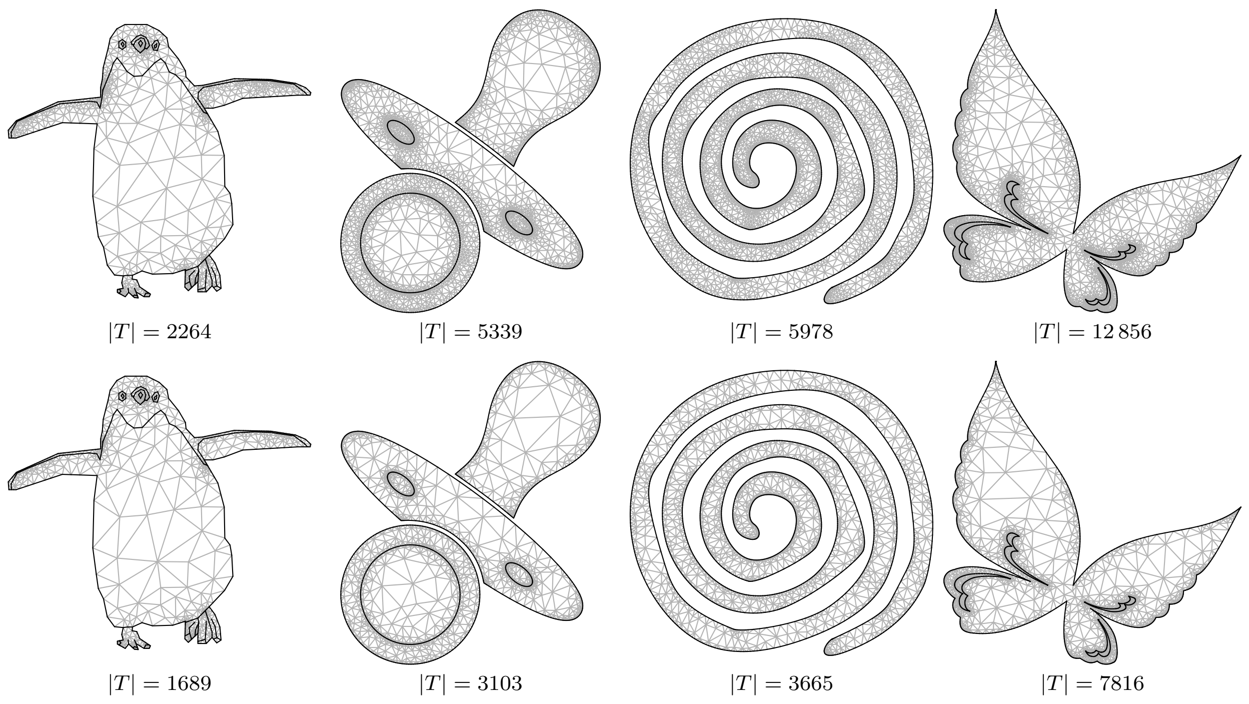
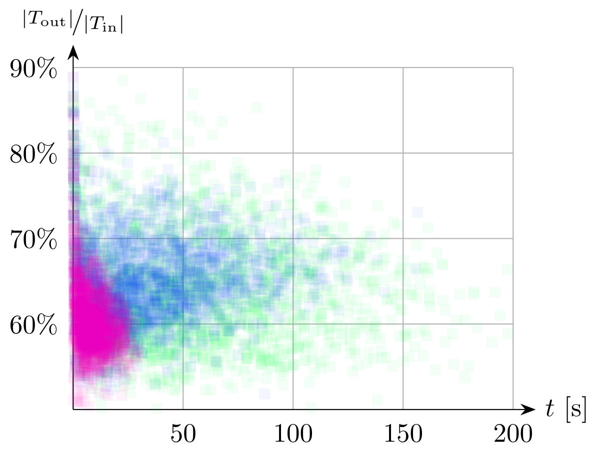


Figure 11: Some meshes before (top) and after (bottom) simplification using the 🟪 algorithm configuration for $\theta^* = 30°$.

To check whether this is the case also for other (non-random) types of input, we applied our algorithm to examples from the dataset of discretized clipart domains published by Hu et al. (2019), originating from Openclipart[2]. Analogously to Figure 8b, the algorithm's performance (for $\theta^* = 30°$) over all examples whose initial mesh has at most 20 000 triangles is indicated in the inset scatter plot. The three rightmost cases in Figure 11 are examples from this test. As can be seen, the relative behavior of the different algorithm configurations is similar. On average, the amount of simplification is even better than on the randomly generated test cases, likely because there are larger regions free of constraints.



6. Conclusion

We explored the problem of triangulation simplification while preserving quality by means of maintaining a strict lower bound on inner angles of the triangular elements. In particular we proposed multiple novel strategies to choose vertex positions in the process of incrementally reducing mesh complexity through local collapse operators. The study of their relative performance in combination with various options for the choice of operators and the choice of order revealed interesting patterns and the presented results enable the choice of the most adequate configuration for a given use case.

A reference implementation with all discussed options is available[3].

[2]https://openclipart.org
[3]https://github.com/SteffenHinderink/Plugin-AngleBounded2DSimplification

6.1. Future Work

By design, while preserving the angle bound, our method increases the number of low-quality elements. As we are targeting a simpler triangulation, this is inevitable to a certain extent. Nevertheless, improvements may be possible (in case this is relevant for an application), for instance by combining additional remeshing or vertex teleportation operators with the decimation approach, either interleaved or in form of a postprocess.

As vertices are greedily removed one by one, the process is affected by local minima regardless of prioritization. The consideration of a longer sequence of collapses as atomic operator could potentially provide advantages in this regard, albeit at an increased computational cost.

Further directions may include the extension of the underlying ideas to other types of meshes, in particular volumetric tetrahedral meshes, in which, e.g, tetrahedron collapses can be used as operator for simplification (Chopra and Meyer, 2002), or quadrilateral meshes, where local diagonal as well as edge collapses (Kinney, 1997; Tarini et al., 2010), but also global poly-chord collapses (Daniels et al., 2008) may play a role for simplification.

Acknowledgements

The authors wish to thank Mario Dyczka for his help in the development of an early experimental implementation as well as Lina-Vanessa Lügtenaar for her help with input preparation.

References

Arrow, K.J., Enthoven, A.C., 1961. Quasi-concave programming. Econometrica 29, 779–800.

Bern, M., Eppstein, D., Gilbert, J., 1994. Provably good mesh generation. Journal of Computer and System Sciences 48, 384–409.

Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., Lévy, B., 2010. Polygon mesh processing. CRC Press.

Boyd, S., Boyd, S.P., Vandenberghe, L., 2004. Convex optimization. Cambridge University Press.

Chew, L.P., 1989. Guaranteed-quality triangular meshes. Technical Report. Department of Computer Science, Cornell University.

Chew, L.P., 1993. Guaranteed-quality mesh generation for curved surfaces, in: Proc. 9th annual symposium on Computational geometry, pp. 274–280.

Chopra, P., Meyer, J., 2002. TetFusion: An algorithm for rapid tetrahedral mesh simplification, in: Proc. IEEE Visualization 2002, pp. 133–140.

Ciampalini, A., Cignoni, P., Montani, C., Scopigno, R., 1997. Multiresolution decimation based on global error. Visual Computer 13, 228–246.

Cignoni, P., Montani, C., Scopigno, R., 1998. A comparison of mesh simplification algorithms. Computers & Graphics 22, 37–54.

Daniels, J., Silva, C.T., Shepherd, J., Cohen, E., 2008. Quadrilateral mesh simplification. ACM Transactions on Graphics (TOG) 27, 1–9.

Erten, H., Üngör, A., 2009. Quality triangulations with locally optimal Steiner points. SIAM Journal on Scientific Computing 31, 2103–2130.

Fried, I., 1972. Condition of finite element matrices generated from nonuniform meshes. AIAA Journal 10, 219–221.

Hamann, B., 1994. A data reduction scheme for triangulated surfaces. Computer Aided Geometric Design 11, 197–214.

Hu, Y., Schneider, T., Gao, X., Zhou, Q., Jacobson, A., Zorin, D., Panozzo, D., 2019. TriWild: Robust triangulation with curve constraints. ACM Transactions on Graphics (TOG) 38, 1–15.

Kim, T., De Goes, F., Iben, H., 2019. Anisotropic elasticity for inversion-safety and element rehabilitation. ACM Transactions on Graphics (TOG) 38, 1–15.

Kinney, P., 1997. CleanUp: Improving quadrilateral finite element meshes, in: Proc. 6th International Meshing Roundtable, pp. 437–447.

Kobbelt, L., Campagna, S., Seidel, H.P., 1998. A general framework for mesh decimation, in: Proc. 24th Graphics Interface, pp. 43–50.

Li, J.Y.S., Zhang, H., 2006. Nonobtuse remeshing and mesh decimation, in: Proc. 4th Eurographics symposium on Geometry processing, pp. 235–238.

Mandad, M., Campen, M., 2021. Guaranteed-quality higher-order triangular meshing of 2d domains. ACM Transactions on Graphics (TOG) 40, 1–14.

Mangasarian, O.L., 1994. Nonlinear programming. SIAM.

Miller, G.L., Pav, S.E., Walkington, N.J., 2003. When and why Ruppert's algorithm works, in: Proc. 12th International Meshing Roundtable, pp. 91–102.

Möbius, J., Kobbelt, L., 2010. OpenFlipper: An open source geometry processing and rendering framework, in: Proc. 7th International Conference on Curves and Surfaces, pp. 488–500.

Rand, A., 2011. Where and how Chew's second Delaunay refinement algorithm works, in: Proc. 23rd Canadian Conference on Computational Geometry, pp. 157–162.

Rossignac, J., 1997. Simplification and compression of 3d scenes, in: 18th Eurographics Tutorials.

Ruppert, J., 1995. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. Journal of Algorithms 18, 548–585.

Sastry, S.P., Shontz, S.M., Vavasis, S.A., 2014. A log-barrier method for mesh quality improvement and untangling. Engineering with Computers 30, 315–329.

Shewchuk, J.R., 1996. Triangle: Engineering a 2d quality mesh generator and Delaunay triangulator, in: Applied Computational Geometry: Towards Geometric Engineering, pp. 203–222.

Shewchuk, J.R., 2002a. Delaunay refinement algorithms for triangular mesh generation. Computational Geometry 22, 21–74.

Shewchuk, J.R., 2002b. What is a good linear finite element? Interpolation, conditioning, anisotropy, and quality measures. Unpublished preprint.

Tarini, M., Pietroni, N., Cignoni, P., Panozzo, D., Puppo, E., 2010. Practical quad mesh simplification. Computer Graphics Forum 29, 407–418.

Tournois, J., Alliez, P., Devillers, O., 2008. Interleaving Delaunay refinement and optimization for 2d triangle mesh generation, in: Proc. 16th International Meshing Roundtable, pp. 83–101.

Üngör, A., 2004. Off-centers: A new type of Steiner points for computing size-optimal quality-guaranteed Delaunay triangulations, in: Proc. 6th Latin American Symposium on Theoretical Informatics, pp. 152–161.

Zlámal, M., 1968. On the finite element method. Numerische Mathematik 12, 394–409.