A new method for subdivision simplification with applications to urban-area generalization

A New Method for Subdivision Simplification with Applications to Urban-Area Generalization*

Kevin Buchin
k.a.buchin@tue.nl

Wouter Meulemans
w.meulemans@tue.nl

Bettina Speckmann
speckman@win.tue.nl

Dep. of Mathematics and Computer Science, TU Eindhoven, The Netherlands

ABSTRACT

We introduce a local operation for polygons and subdivisions called an edge-move. Edge-moves do not change the edge orientations present in the input and are thus suitable for iterative simplification or even schematization. Based on edge-moves we present a new efficient method for area- and topology-preserving subdivision simplification. We show how to tailor this generic method towards the specific needs of building wall squaring and urban-area generalization. Our algorithm is guaranteed to make further progress on any subdivision that has two or more faces and/or reflex vertices. Furthermore, our method produces output of high visual quality and is able to generalize maps with ≈ 1.8 million edges in a few hours.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Spatial Databases and GIS*; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling

General Terms

Algorithms, Theory

Keywords

Outline schematization, Building simplification, Map generalization

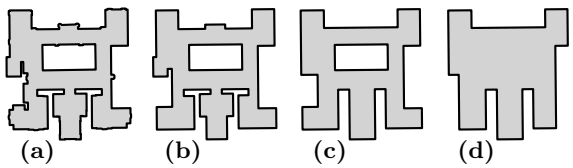


Figure 1: (a) Building (167 edges). (b-c) Simplified to 48 and 28 edges. (d) Generalized to 20 edges.

*W. Meulemans and B. Speckmann are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.022.707.

ACM SIGSPATIAL GIS '11, November 1-4, 2011. Chicago, IL, USA

1. INTRODUCTION

Maps are widely used to convey information. They employ a variety of visual representations and are targeted towards different purposes. Regardless of representation and purpose, visual clutter arises when placing too much detail in a small area of a map, greatly reducing legibility. To ameliorate this, the displayed elements are often simplified or even schematized. Simplification eliminates details from map elements. Schematization goes one step further, increasing legibility by distorting elements, often to adhere to some predefined orientations. The style and amount of detail on a map needs to be suitable for its target scale and purpose. Many maps are topographic maps, for example depicting an urban area by indicating buildings, roads, and rivers. Such maps are often not designed for a single purpose and try to display as much information as possible without causing clutter. When visual clutter is to be reduced for topographic maps, we speak of cartographic generalization. Encompassing more than just simplification, cartographic generalization of urban data distinguishes seven generalization operators [11]: elimination, displacement, aggregation, exaggeration, detail elimination, squaring, and typification. An example of outlines of the same building generalized for different scales is shown in Fig. 1.

Topographic detail is less relevant for other types of maps. For example, a metro map mainly needs to convey connectivity information. These maps are typically schematized, often using the four main orientations (horizontal, vertical, and both diagonals). Schematized maps of transport networks frequently show not only the network itself but also schematized region boundaries or subdivisions. In the remainder of this paper, we consider only maps that are simple polygonal subdivisions.

Results. We present a simple and fast method for subdivision simplification that can be tailored to support schematization as well as the generalization operators detail elimination, squaring, aggregation, and elimination. Our method preserves the area of each face in an input subdivision, uses (a subset of) the orientations of the input, and is topologically correct. It is based on a single operation called an *edge-move* (illustrated in Fig. 2). In Section 2, we introduce edge-moves for simple polygons. Edge-moves are *complete*, that is, they can reduce the complexity of any non-convex simple polygon in an area-, orientation-, and topology-preserving way. As their name suggests, edge-moves move edges and vertices: the output vertices are necessarily not restricted to be a subset of the input vertices. In Section 3, we describe our algorithm for polygons and briefly discuss a method to

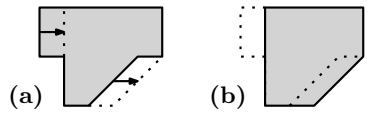


Figure 2: (a) Two edge-moves. (b) The result.

restrict orientations of polygons or subdivisions before simplification to allow for schematization. The same section also explains how to tailor our algorithm to wall squaring. Section 4 shows how to extend our method to polygonal subdivisions. In Section 5, we discuss the application of our method to urban-area generalization. We introduce a simple approach to aggregation that integrates easily into our simplification algorithm. Together with aggregation our method is complete for subdivisions as well. Results are illustrated and discussed per topic throughout the paper.

Related work. There is a significant body of work in the GIS literature that deals with line simplification. Of particular relevance to our paper is the work of Bose *et al.* [1] on area-preserving line simplification. Delling *et al.* [5] and Neyer [13] discuss $\mathcal{C}$-oriented schematization of routes or lines. However, it is generally not advisable to simplify or schematize each chain in a subdivision separately, especially when aiming for a result with low complexity. There are some approaches, developed in computational geometry, that preserve the topology of the input subdivision. For example, De Berg *et al.* [4] describe a method that simplifies a polygonal subdivision without introducing intersections or passing over special input points. However, such methods are not restricted to use the orientations present in the input and thus cannot be used for schematization purposes. Moreover, many simplification problems that minimize the number of edges in a subdivision are NP-complete [7].

Cartographic generalization is a very broad topic and we cannot hope to survey the literature here. Hence we highlight only a few relevant references. Our work is most closely related to the following papers on building generalization [3, 9, 16, 20]. Furthermore, algorithms for building wall squaring [10, 14, 15] and outline simplification [8, 14, 17] can also be applied to subdivision simplification and schematization. However, none of these methods is area-preserving and many do not provide topological guarantees. Swan *et al.* [18] and Wolff [19] give excellent overviews of methods for map schematization and metro map construction.

2. EDGE-MOVES

Here we introduce edge-moves and sketch an argument that shows that edge-moves can always be performed on a non-convex polygon in an area-, orientation-, and topology-preserving way.

Intuitively, an edge-move operates on three consecutive edges, moving the vertices of the middle edge along the lines that coincide with the other edges while maintaining the orientation of the middle edge. Such an operation can add or remove area, depending on the direction of movement. A contraction is an edge-move where at least one edge vanishes due to reaching length zero. Two edge-moves—one adding area, the other removing area—are combined to obtain an area-preserving and orientation-preserving operation. Fig. 2 shows two of these complementary edge-moves. In subdivisions, edge-moves cannot always be applied. However, with a simple method for aggregation, we can actually prove completeness (see Theorem 2 in Section 5).

More formally, assume that we are given a simple polygon P. We call an edge of P convex or reflex if both its vertices are convex or reflex respectively. The exterior angle of a vertex is defined as the angle between one edge and the extension of the other. The exterior angle is sometimes also referred to as turning angle. The angle is negative if and only if the vertex is reflex.

A *configuration* G consists of three consecutive edges denoted by g_1, g_2, and g_3. Edges g_1 and g_3 are its *outer edges*, g_2 is its *inner edge*. The outer edges of a configuration define two tracks, infinite lines through the edges. An *edge-move* on G moves g_2 such that its orientation is preserved and its vertices are on the tracks, making the outer edges longer or shorter. An edge-move is valid if at least one of its vertices remains on its original outer edge and g_2 remains on the same side of or on the intersection point of the tracks (if any). An edge-move which causes one of the edges of G to reach length zero, is a *contraction*. Contractions are extremal edge-moves. An edge-move is *positive* if it adds area to P and *negative* if it removes area.

A configuration supports edge-moves, either positive, negative or both. Let g_2^+ denote the extremal position of g_2 after any valid positive edge-move, i.e. the position after a positive contraction. The *positive contraction region* of a positive configuration, $R^+(G)$, is the region enclosed by g_2, g_2^+, and the tracks. A *feasible positive configuration* is a configuration for which $R^+(G)$ is empty except for G. Similarly, we define the *negative contraction region* $R^-(G)$ and a feasible negative configuration. If a positive or negative configuration is feasible, then any valid positive or negative edge-move respectively is feasible. An example of an edge-move and its regions is shown in Fig. 3.

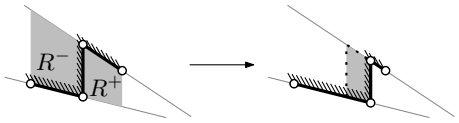


Figure 3: A valid positive edge-move.

Since we desire an approach that preserves the area of the polygon, we combine two *complementary* feasible configurations, one positive and one negative, executing an edge-move on both simultaneously. The one with the smaller contraction region is contracted, while the other is moved just far enough to compensate for the area change. Two configurations *conflict* when they share an edge, unless they share only outer edges and one of these has a convex and a reflex vertex. In this special case the two edge-moves both either shorten or lengthen the shared edge.

THEOREM 1. *Every simple non-convex polygon has a non-conflicting pair of complementary contractions.*

Here we give only some intuition for the proof of Theorem 1; the full proof is given by Buchin *et al.* [2].

By induction, it can be shown that at least one feasible negative configuration exists in a polygon. More specifically, such a configuration exists in a consecutive series of edges of a polygon that adhere to certain restrictions. By properly inverting the polygon (i.e. turning it "inside-out"), it follows that there must also be a feasible positive configuration. What remains to be shown is that there is a pair that is

non-conflicting. We can show the existence of either of the following: a series of edges that adhere to the restrictions such that any feasible negative configuration in the series cannot conflict the given feasible positive configuration; or a feasible positive configuration that does conflict the given negative configuration. In the latter case, we can prove that the negative configuration is situated such that there must be another nearby that does not conflict with the feasible positive configuration.

3. SIMPLE POLYGONS

Using the edge-moves defined in the previous section, we iteratively simplify simple polygons. However, a simple polygon might admit many edge-moves. As a heuristic, we select a complementary pair of edge-moves such that the size of smallest of the two contraction regions is minimized. By Theorem 1, there is always a complementary pair of two non-conflicting feasible configurations. Therefore, we can find a pair of configurations that are both feasible. Of all non-conflicting feasible configurations, the one that is closest (in the number of edges along the boundary) is chosen. For efficiency, each edge stores the number of edges that block the positive and negative contraction of the configuration of which that edge is the inner edge. These counters can be used to check in constant time whether a configuration is feasible. We obtain a quadratic-time algorithm, given in Algorithm 1. The correctness of the algorithm follows from Theorem 1 and the observation that a single configuration conflicts with at most 5 configurations. Since at most a constant number of edges change during the edge-moves, only a constant number of edges have to be checked to decrement and increment the counters of all edges in the polygon. Simplification results are shown in figures 4(b) and 5(b).

Algorithm 1 `SimplifyPolygon`(P, k)

Require: P is a simple polygon
Ensure: P is a convex polygon or has at most k edges

1: Initialize contraction counter for each edge
2: **while** $|P| > k$ and P is non-convex **do**
3: Find up to six of the smallest positive contractions and up to six of the smallest negative contractions
4: Determine minimal contraction pair (G_1, G_2)
5: Dec. counters for contractions blocked by G_1 or G_2
6: Contract (G_1, G_2)
7: Inc. counters for contractions blocked by G_1 or G_2
8: **return** P

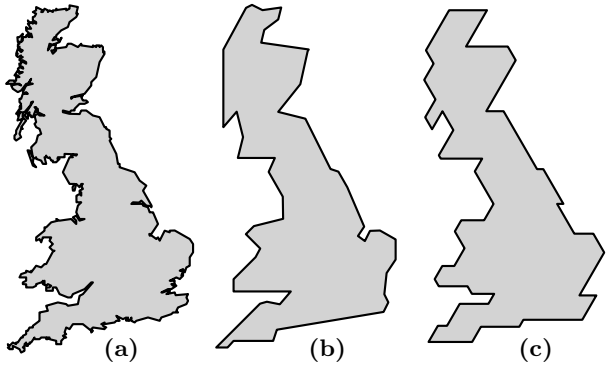


Figure 4: (a) Great Britain (549 edges). (b) Simplified to 50 edges. (c) Schematized hexagonally to 50 edges.

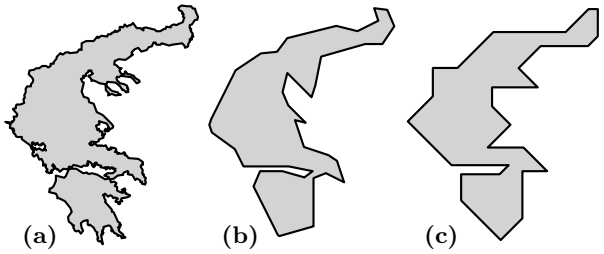


Figure 5: (a) Greece (616 edges). (b) Simplified to 40 edges. (c) Schematized octagonally to 27 edges.

It is also possible to modify the algorithm such that edge-moves with non-empty contraction areas can be used. To this end, the algorithm should not simply count the number of edges blocking a configuration, but keep track of the closest point in the contraction region. This allows to quickly decide whether a configuration can compensate for a change in area. A simple implementation of this would lead to a cubic-time algorithm. However, in reasonable outlines, edge-moves are obstructed typically by relatively few edges. Hence the actual execution time is expected to not differ much from the original algorithm. This modification is especially relevant for subdivisions as it gives more flexibility.

Since edge-moves preserve orientations, we can use it for schematization purposes as well. This is done by restricting angles beforehand. We describe in Section 3.1 how one can convert a polygon into an area-equivalent polygon with edges oriented according to some given set. Finally, we can also perform building wall squaring using edge-moves. However, this needs more preprocessing than just schematization. These additional steps are described in Section 3.2.

3.1 Imposing angular restrictions

Here, we describe how to convert a given subdivision to a new subdivision such that each face maintains its area and such that every edge in the result is parallel to an orientation in some given set $\mathcal{C}$. This method is based on the rectilinearization process described by Meulemans *et al.* [12].

We do not go into detail on the method in this paper. The basic idea is to create "staircases" for each edge that ensure area preservation and the adherence to the orientation set $\mathcal{C}$. Intersections can be avoided by using a sufficient (but finite) number of "steps". Examples of staircases are given in Fig. 6. Two schematization results of this method, followed by simplification, are shown in figures 4(c) and 5(c).

The orientation set $\mathcal{C}$ must consist of at least two distinct orientations (four directions). However, there are no other constraints on the orientation set, it may either be regular or irregular. The method can always deal properly with vertices of degree four. Vertices up to degree $2|\mathcal{C}|$ may be dealt with, if the distribution of the edges around these high-degree vertices is such that every edge can be assigned

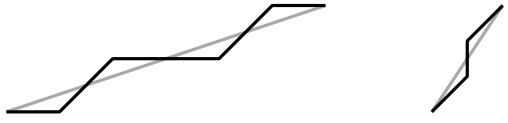


Figure 6: Staircases for octagonal schematization.

to the first or second orientation in either clockwise or counterclockwise direction. Since vertices of degree four or higher occur very infrequent in cartographic applications, this is considered to be sufficient.

3.2 Building wall squaring

Buildings are man-made objects and as such often have right-angle corners. However, given outlines may be inaccurate. Building wall squaring is the task of restoring these right-angle corners where appropriate. We perform squaring by imposing angular restrictions, followed by a sequence of edge-moves to return to the original complexity of the building. We first discuss how to detect arcs in a building outline, as squaring and deforming these should be avoided. We then discuss the detection of building orientations and finally show how to obtain a squared building outline.

Arc detection. Some buildings have characteristic arcs, such as a church having a circular chapel. While in a polygonal representation, the number of vertices used to represent such a feature may be high, its visual complexity is far lower, the arc being perceived as a single object. To avoid squaring and deforming such arcs, we detect these beforehand, marking the vertices as *arc vertices*. We characterize an arc as a sequence of vertices that have similar exterior angles and segment lengths. Given three parameters γ, δ and ϕ, we mark a sequence of vertices $v_1, \ldots, v_n$ as arc vertices if the following three conditions are met: the exterior angle between two adjacent vertices may not change more than γ; the exterior angle is also constrained to be contained in the interval $[-\delta; +\delta]$; the ratio in length between two adjacent edges may not exceed ϕ. In our experiments, we use $\gamma = \frac{\pi}{12}$ (15 degrees), $\delta = \frac{\pi}{6}$ (30 degrees) and $\phi = 3$. Note that the first and last vertices of the sequence are not constrained and thus any sequence of at most two vertices would satisfy the conditions. A sequence of three vertices, in our opinion, disregards the parameter γ and may as well be just a corner of the building. Therefore, we require that the sequence consists of at least four vertices. Finally, we must also make sure not to mark straight walls, represented by multiple vertices, as arcs. This is done be requiring that there exists a v_i $(1 < i < n)$ such that $\overline{v_i - v_1} \cdot \overline{v_n - v_1} < 0.99$.

Orientation detection. To detect the orientations in a building, we extend a method by Duchêne *et al.* [6]. Their method takes a set of candidate orientations, and for each such orientation computes the weighted sum of edge lengths of edges contributing to that orientation. An edge contributes to a candidate orientation if its orientation deviates only by a small angle from the candidate orientation. The weight of an edge depends linearly on this angle. We modify this method in two ways.

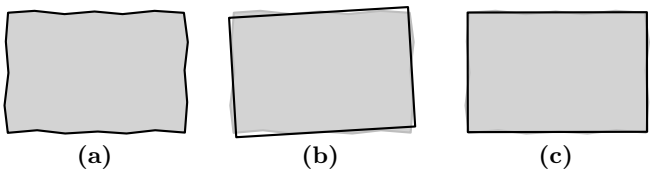


Figure 7: (a) A building with imprecise walls. (b) Result with linear contribution function. (c) Result with Gaussian contribution function.

Instead of using a linear contribution function, we use a Gaussian function with a peak equal to the edge length and a width of 7.5 degrees. This change results in a higher contribution to orientations that are close to the orientation of the edge. Whereas a linear contribution function ensures that a maximal value always occurs at the orientation of one of the edges, the Gaussian function does not have this property, allowing the method to detect "intermediate" orientations in noisy data (see Fig. 7). Moreover, instead of using a sampled approach to find an orientation with maximal contribution, we compute the desired orientation by numerical methods.

A building may have multiple orientations, varying between parts of the outline, as shown in Fig. 8. Instead of locating only the main orientation, we remove the edges that are close to the detected orientation (or its perpendicular) and repeat the detection process until no edges remain.

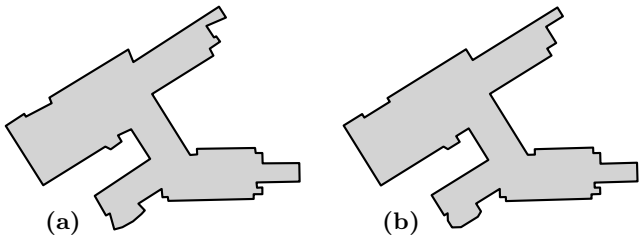


Figure 8: A building with two distinct orientations. (a) Original building. (b) Squared outline.

Squaring. Now that arcs and orientations have been detected, we perform two steps to obtain the squared outline.

We start by imposing angular restrictions as described in Section 3.1. This is based on the detected orientations. However, we do not change the edges that end in two arc vertices such that arcs are not deformed in the process. If both edges of a degree-2 non-arc vertex have the same associated orientation, then it is marked as being *superfluous*. These superfluous vertices are vertices along a (nearly) straight wall and should be removed after squaring.

To obtain the squared outline, we now apply edge-moves on the building outline. Instead of applying those with smallest area, we apply those that cause the inner edge of the configuration to move the least, making sure not to move arc vertices. We stop when the number of vertices is at most the number of non-superfluous vertices of the original outline.

For further simplification, we can "release" the arc vertices and impose the angular restriction on these edges as well. This allows for simplification (and at some point elimination) of arcs from the building. Regardless of releasing the arc vertices, simplification continues by selecting edge-moves with the smallest contraction regions. If arc vertices are to be released at all, this should be done before any further simplification. This avoids large deformations that may prove to be unnecessary after releasing the arc vertices.

As an optional postprocessing step, we align walls using edge-moves. If two (non-arc) walls with the same orientation are nearly colinear, we can try to apply edge-moves such that the walls become exactly aligned. For two walls that are similarly directed (e.g. the interior is above the wall for both), one can always do this while preserving area. However, if the walls are oppositely directed, this need not be possible. Even in cases where this is possible (due to

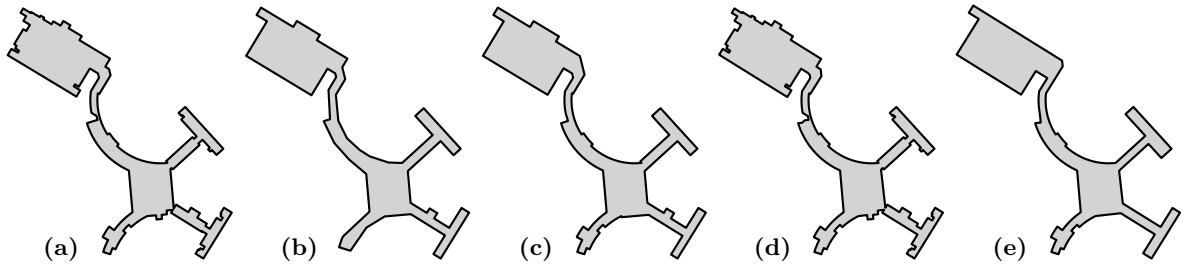


Figure 9: (a) Airport of Boston (361 edges). (b) Simplified to 54 edges. (c) Simplified to 295 edges with arc detection. (d) Squared outline with 360 edges. (e) Simplified to 283 edges with arc detection after squaring.

different lengths of the walls), experiments indicated that this yields unsatisfactory results. Other edge-moves could also be used to compensate for the area gain or loss, but it may cause a deformation that is worse than the misaligned walls and it risks a misalignment of walls that were aligned. By relaxing the area-preservation constraint, the misaligned walls with opposite directions can move towards each other.

Fig. 9 shows various results for the main building of the Boston airport. The result without arc detection obviously obtains the lowest (measured) complexity. However, the perceived complexity is not reduced for some arcs, it actually increased. The building has quite a few small arcs that appear to be a simple offset of a larger arc. Since these vertices are fixed by the arc detection, these details are not eliminated by simplification. For simplification with detected arcs, it would be desirable to move such arcs to merge with the larger one. However, the details of such an operation are left as future work. Fig. 10 shows three results for another complex building. Our method neatly squared the building after detecting arcs (see Fig. 10(b)). However, walls that are not part of an arc but that do end in two arc vertices are not modified, for example the vertical wall in the bottom right. Releasing the arc vertices and imposing the angular restrictions on the edges allows us to further simplify the building with satisfying results (Fig. 10(c-d)).

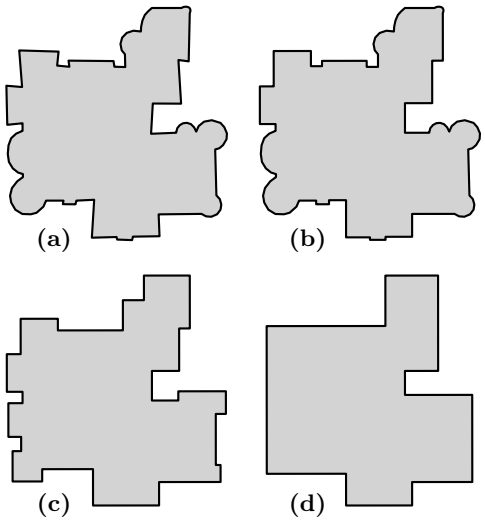


Figure 10: (a) Building with 95 edges. (b) Squared outline with 95 edges. (c) Simplified to 38 edges. (d) Simplified to 14 edges.

4. POLYGONAL SUBDIVISIONS

In this section we show how to simplify and schematize entire subdivisions. We consider planar polygonal subdivisions, which are induced by planar straight-line embeddings of graphs. These graphs need not be connected. We define a *component* of a subdivision to be a maximal subgraph that is not contained in any finite face of the embedding. Any subgraph that is embedded in a finite face is considered to be part of the subgraph that encloses that face. Note that this differs slightly from the typical definition of a connected component in a subdivision (see Fig. 11).

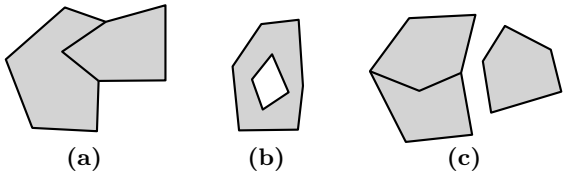


Figure 11: (a-b) Subdivisions with a single component. (c) Subdivision with two components.

The method for imposing angular restrictions, briefly discussed in Section 3.1, is already defined for subdivisions rather than polygons. However, the definition of an edge-move is based on polygons. In this section, we describe what has to be taken into account when applying edge-moves in subdivisions. For area preservation, it is important to combine only edge-moves of which the inner edges are incident on the same faces. In addition to this requirement, we must define edge-moves in the presence of vertices of degree three or higher. A configuration with interior vertices of degree two supports edge-moves as defined for polygons. However, if one of the interior vertices is degree four or higher, we do not allow moving that vertex and as a consequence the interior edge cannot be moved. We do not allow to move vertices of degree four or higher as this would either violate the topology-preserving or area-preserving property of the method. This would be possible when allowing more than two simultaneous edge-moves, at the cost of a higher computational cost. For vertices of degree three, there is some flexibility possible. There are three cases that can occur for a degree-three vertex v for an edge-move with inner edge e, incident to v (refer to Fig. 12):

- **(a)** The other edges of v have the same orientation. These edges must lie on different sides of the line through e. When moving e in one direction or the other, vertex v can slide along the other two edges.
- **(b)** The other edges of v do not have the same orientation, but lie on different sides of the line through e. When

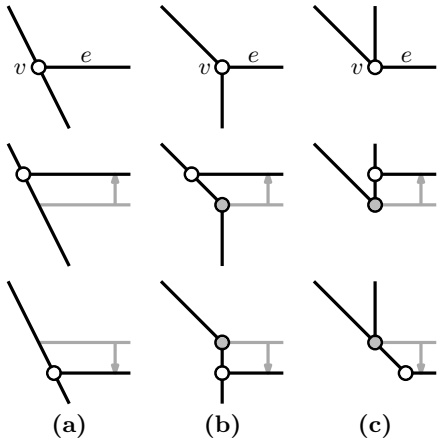


Figure 12: The three cases for a vertex of degree three (white) when moving an edge e. New vertices are given in gray. Top row: original situation; middle row: situation after moving e upward; bottom row: situation after moving e downward.

moving e in one direction or the other, vertex v can slide along the edge on that side, but an extra vertex must be introduced on its original position.

(c) The other edges of v do not have the same orientation, but lie on the same side of the line through e. When moving e in the direction of the other edges, vertex v can slide along the edge that makes the smallest angle to e, but an extra vertex must be introduced on its original position. When moving e in the other direction, vertex v can slide along the extension of either edge. We use the edge that has the largest angle to e, unless this edge has the same orientation as e (an angle of π). A new vertex must be introduced on the original position of v and this vertex has degree 3.

When combining edge-moves that introduce a new vertex, it is important to make sure that the combined effect of the edge-moves still reduces the overall complexity of the subdivision. Predicting the total complexity reduction of a single contraction is straightforward. Finally, to ensure correct topology, it is important that a vertex of degree three is never removed or moved on top of another vertex having a degree higher than two. Figures 13 and 14 both show a simplification and a schematization of a subdivision obtained by applying our method on subdivisions.

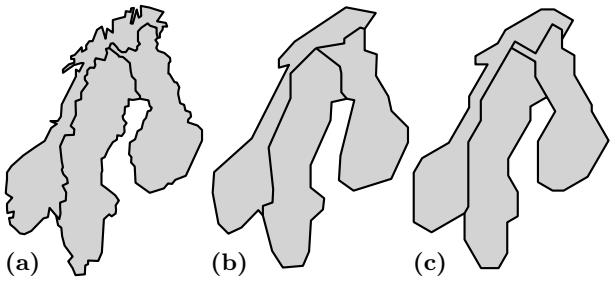


Figure 13: (a) Norway, Sweden, and Finland (238 edges). (b) Simplified to 54 edges. (c) Schematized dodecagonally (six orientations) to 54 edges.

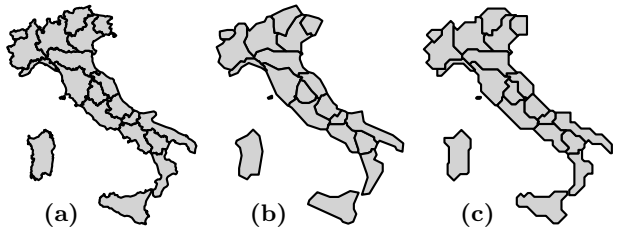


Figure 14: (a) The provinces of Italy (1297 edges). (b) Simplified to 244 edges. (c) Schematized octagonally to 244 edges.

5. URBAN-AREA GENERALIZATION

Urban-area generalization can be achieved in part by simplification. A collection of building outlines can be interpreted as a subdivision. Therefore, the method that we outlined in previous sections can be applied to obtain generalized versions of urban areas as well. However, depending on the size of the area and the target scale, simplification alone does not suffice; other generalization operators are required as well. In this section, we deal with the integration of aggregation and elimination into our simplification method. We assume that we have detailed and accurate building outlines available. Hence, we do not use squaring. Squaring could even cause misaligned walls for buildings on opposite sides of streets, greatly reducing the impression of a street caused by the empty region in between. This is illustrated in Fig. 15.

Our support for aggregation is two-fold: an *interior merge* aggregates two faces (buildings) within a component; an *exterior merge* aggregates two separate components. We also present a method to resolve high-degree vertices that cannot be handled by these merge operations.

An interior merge involves two faces and requires that these share at least one edge. The interior merge removes all edges that are incident to both faces. The cost of an interior merge equals the area of the smaller of the two faces multiplied by some parameter α_{in}. This cost can be compared to the cost (area of contraction region) of an edge-move in order to decide which operation should be done. Increasing or decreasing α_{in}, respectively, makes the process less or more likely to merge faces rather than performing edge-moves.

An exterior merge is initiated by making a zero-width connection of two edges between the two nearest points on the outer faces of the components. At least one of these new edges allow a (small) positive edge-move. This edge-move is performed and the area gain is compensated for by executing a negative edge-move in one of the faces. To prevent introducing a new orientation, we can pick two distinct orientations from the components and create a staircase for each of the newly introduced edges (see Section 3.1). The cost of an exterior merge equals the area of the smaller of

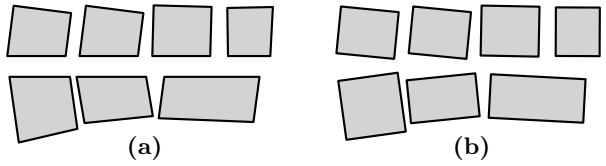


Figure 15: (a) Original outlines give the impression of a street. (b) Misaligned walls would cause the impression of the street to diminish.

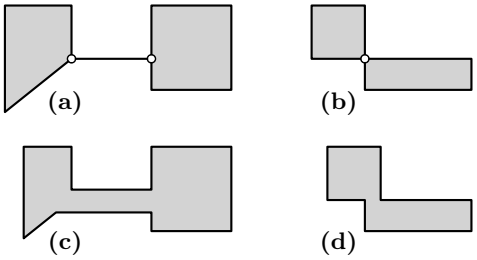


Figure 16: (a) An edge incident only to the infinite face. (b) A degree-four vertex alternatingly incident to the infinite face. (c-d) Results after eliminating a high-degree vertex. The area deformation is exaggerated to clarify the effect.

the two components multiplied by some parameter α_{ex}. We require that $\alpha_{ex} \geq \alpha_{in}$ holds. This guarantees that when an exterior merge occurs, the smaller of the two components consists of only one face, i.e. it is a simple polygon. In our experiments, we use $\alpha_{\text{in}} = 0.2$ and $\alpha_{\text{out}} = 0.5$.

It is possible for a component to consist of multiple faces, while no two faces share an edge (see Fig. 16(a-b) for examples). There are two cases possible: either a single edge has the infinite face on both sides or a vertex of degree four or higher is alternatingly incident to a finite face and the infinite face. A single edge that is incident only to the infinite face can be eliminated as follows: if it has a vertex of degree one, then it can be removed without affecting the area. Otherwise, we can duplicate the edge, creating a zero-area region in between the edge and its duplicate. At least one of these allows a positive edge-move. One of the faces at the ends of the edge can perform a negative edge-move to compensate for the change in area. This effectively merges the two faces. For a vertex of degree-four (or more), we perform a positive edge-moves to "cut" the vertex, merging two "adjacent" incident faces. Again, the increase in area can be compensated for by an edge-move in one of the merged faces. The cost of this operation is the area of the smallest merged face multiplied by α_{in}.

By combining these four methods of aggregating faces and components, we obtain a method that is complete for subdivisions. This is formulated in the following theorem.

THEOREM 2. *Let S be a simple polygonal subdivision. Unless S is a simple convex polygon, subdivision S can be generalized by either a complementary pair of feasible edge-moves, an inner or outer merge, or the elimination of a high-degree vertex, while preserving area, orientations, and planarity.*

PROOF SKETCH. If the subdivision is a simple polygon, there is always a complementary pair of feasible contractions (Theorem 1). If the subdivision is not a simple polygon, but has a component that is a simple polygon, then this component can always be used for an exterior merge with the nearest other component.

If the subdivision is not a simple polygon, and no component is a simple polygon, then either an interior merge can be performed or a high degree vertex can be eliminated, reducing the number of faces in the component. Since the dual of a single component is a tree, it must have a leaf: that is, at least one face consists of three or more vertices, of which only one has a degree higher than 2. □

By the theorem above, it is always possible to generalize a simple polygonal subdivision while preserving the total area and orientations inherent in the input. However, for our results, we do not apply the additional steps to ensure these properties. Orientations are not preserved as no angular restrictions are used beforehand, making the restriction unnecessary. For aggregation, we do not desire area preservation since it may cause apparent area *loss*: when nearby components merge into a single component, the space between them should be "absorbed", to have the neighborhood (which includes some of the empty space in between) maintain its apparent area. Hence, we do not compensate for the area increase caused by the exterior merge and even perform edge-moves on both newly introduced edges such that a maximal area is added. Also, it is undesirable to merge small components with a component that is far away. Such "outlier" components can simply be eliminated from the input. A component is considered an outlier when the nearest other building is further away than three times its diameter.

Discussion. We present results for a data set of building outlines of Boston[1]. The complete data set, with 295,781 components and 1,750,427 edges, is depicted in Fig. 17. We ran our algorithm on the complete data set and on subsets. In the following, we discuss only two subsets: North End (131 components; 2035 edges) and the northern part of Roxbury (2192 components; 15,192 edges). We refer to the latter simply as "Roxbury". Some results for the complete data set are available online[2].

Fig. 18 shows the input and results for Roxbury. While the generalized version uses about 20% less edges than the simplified one, we think its visual quality is at least comparable to that of the simplified result. Some differences are highlighted in Fig. 19. Small buildings are merged when generalizing the urban area. This has little impact on the visual quality but greatly reduces the complexity. As a result, other buildings can even use some more edges, retaining more of their original shape. In Fig. 21, we show the input for North End, as well as our simplification and generalization result. Again, we conclude that the generalized result is at least comparable to the simplified result, while using almost 17% less edges.

The Boston data set has been used to analyze other simplification methods, such as the method presented by Haunert and Wolff [8]. One of their results for North End is shown in Fig. 21(d). Our simplification result has a similar complexity as their result. The visual quality of the results are comparable. We highlight typical differences by the example shown in Fig. 20. The buildings shown are simplified more accurately by the method of Haunert and Wolff (and use more edges), but the area as a whole is simplified better by our method as it preserves the impression of a street.

The main advantage of our method is that it is simple, fast, and more scalable. While their solution involves solving an integer linear program (an NP-hard problem) with in the worst-case $O(n^6)$ constraints, our method has a worst-case complexity of $O(n^3)$ when allowing infeasible configurations to compensate for area change. Since a component can be affected only by "nearby" components, we can obtain

[1]This data set is freely available as part of the Massachusetts Geographic Information System, MassGIS: `http://www.mass.gov/mgis/lidarbuildingfp2d.htm`, accessed on February 8, 2011.

[2]`http://www.win.tue.nl/~wmeulema/results.html#boston`

Figure 17: Data set of Boston with North End indicated by the small rectangle and a part of Roxbury indicated by a larger rectangle. The data set has 295,781 components and 1,750,427 edges.

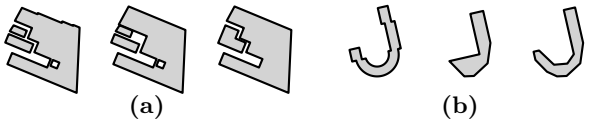


Figure 19: Highlights of Roxbury, found in the far southwestern corner and the southern area. Each case depicts input, simplification, and generalization. (a) By aggregation, these buildings are represented with less edges without a significant loss of visual quality at small scale. (b) Aggregation elsewhere allows buildings to use more edges, resulting in a higher visual quality.

Figure 18: (a) Roxbury, Boston (2192 components and 15,192 edges). (b) Simplified to 9965 edges. (c) Generalized to 1593 components and 8071 edges.

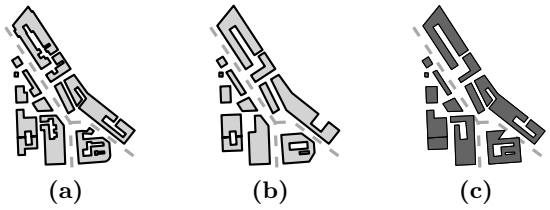


Figure 20: Highlight of North End, found in the northwestern corner of North End. (a) Buildings give the impression of streets (dashed lines). (b) In our result, the impression is preserved. (c) In the result of Haunert and Wolff [8], the impression of the street is reduced. Also observe the apparent topology violation in the bottom left building.

Figure 21: (a) North End, Boston (131 components and 2035 edges). (b) Simplified to 939 edges. (c) Generalized to 242 components and 783 edges. (d) Result of Haunert and Wolff (Fig. 10 in [8]). (e) Simplified to 939 edges allowing "invalid" edge-moves. (f) Generalized to 242 components and 783 edges allowing "invalid" edge-moves.

an efficient implementation. Our method is approximately as fast for small data sets, but is far more scalable to large data sets. Table 1 shows the execution time on North End, Roxbury and the complete data set.

Another advantage of our method is that it preserves the area of each face exactly. While Haunert and Wolff do take this into account, no guarantees can be given. Furthermore, our method integrates with a simple aggregation method, while this has to be done separately for the method of Haunert and Wolff. Finally, we observe a difference in what is considered topologically safe. Their method guarantees that there are no intersections in the result, but treats each building as a separate entity[4], disregarding shared walls and holes. Hence, some apparent holes may vanish (see Fig. 20) or adjacent buildings may disconnect. In contrast, our method works on subdivisions and—without aggregation—preserves the dual graph of the subdivision.

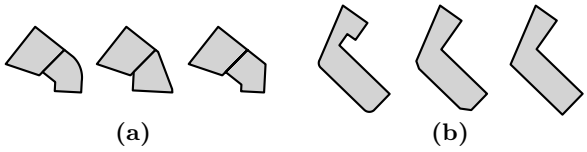
Table 1: Execution times, measured until no further operations can be performed. Last column lists execution times of the Haunert and Wolff method [8].

Data set	Simplified	Generalized	ILP
North End	2.288s	2.954s	2.06s
Roxbury	3.439s	9.037s	44s[3]
Boston	2h17m44s	4h57m07s	N/A

Our method, as described, cannot eliminate short convex and reflex edges easily, as these are not allowed to move outward or inward respectively. The effect is that round corners are moved inward completely or that small cutoffs are not removed (as illustrated in Fig. 22). This is caused by the definition of valid edge-moves, i.e., only edge-moves of which at least one of the vertices remains on the original outer edge are allowed. However, allowing "invalid" edge-moves is possible and causes no algorithmic problems. Since this type of edge-moves is not strictly required for completeness, we can put restrictions on the use of such operations without interfering with this property of the method. We allow an invalid edge-move with inner edge e only if the intersection of the tracks is in the direction of the edge-move and the orthogonal distance between this intersection and

[3]Personal communication with J.-H. Haunert, May 2011.

[4]This causes the complexity measures given by Haunert and Wolff to differ slightly from those we give.



(a) **(b)**

Figure 22: Issues that can be solved by allowing invalid edge-moves. Both cases depict input, regular result, and result when allowing invalid edge-moves. Examples are found in central and southern North End, respectively. (a) Rounded corners can only shrink. (b) Small cutoffs cannot be removed.

the line through e is at most the length of e. Fig. 21(e-f) shows the result of our method on North End when allowing these additional edge-moves. The problems indicated have been alleviated. But this introduces another problem. The impression of the street at some locations vanishes similar to the results of Haunert and Wolff (see Fig. 20).

An advantage of the algorithm by Haunert and Wolff is that it is parameterized by error tolerance, which is quite easily obtained from a target scale. Our method is parameterized by desired complexity, which does not directly correspond to a target scale. However, it is straightforward to change the algorithm such that it performs only edge-moves that stay within some error tolerance of the original shape. Depending on the used measure, this leads to an increased worst-case execution time. Such a parametrization to scale would likely solve some issues of components "contending" over edges (as indicated in Fig. 19(b)).

6. CONCLUSION

We described a simple and fast method based on edge-moves for the simplification of polygonal subdivisions. Our algorithm is able to process large data sets and yields results of high visual quality. Our method preserves the area of each face of the input subdivision, uses a subset of the orientations present in the input, and results in a simple, topologically correct subdivision. Since our method preserves orientations, it can also be used for schematization, by imposing angular restrictions on the input beforehand. By adding two more preprocessing steps, our method can also be used for building wall squaring, one of the operators in cartographic generalization. For generalizing entire urban areas, we introduced a simple method for aggregation that integrates seamlessly with our simplification algorithm.

7. REFERENCES

[1] P. Bose, S. Cabello, O. Cheong, J. Gudmundsson, M. van Kreveld, and B. Speckmann. Area-preserving approximations of polygonal paths. *Journal of Discrete Algorithms*, 4(4):554–566, 2006.

[2] K. Buchin, W. Meulemans, and B. Speckmann. Area-Preserving C-Oriented Schematization. In *Abstracts 27th European Workshop on Computational Geometry*, pages 163–166, 2011.

[3] J. Damen, M. van Kreveld, and B. Spaan. High quality building generalization by extending morphological operators. In *Proc. 11th ICA Workshop on Generalization and Multiple Representation*, 2008.

[4] M. de Berg, M. van Kreveld, and S. Schirra. Topologically correct subdivision simplification using the bandwidth criterion. *Cartography and Geographic Information Science*, 25(4):243–257, 1998.

[5] D. Delling, A. Gemsa, M. Nöllenburg, and T. Pajor. Path schematization for route sketches. In *Proc. 12th Scandinavian Workshop on Algorithm Theory*, LNCS 6139, pages 285–296, 2010.

[6] C. Duchêne, S. Bard, X. Barillot, A. Ruas, J. Trévisan, and F. Holzapfel. Quantitative and qualitative description of building orientation. In *Proc. 5th ICA Workshop on Progress in Automated Map Generalisation*, pages 1–10, 2003.

[7] L. Guibas, J. Hershberger, J. Mitchell, and J. Snoeyink. Approximating polygons and subdivisions with minimum-link paths. *Int. Journal of Computational Geometry and Applications*, 3:383–415, 1993.

[8] J.-H. Haunert and A. Wolff. Optimal and topologically safe simplification of building footprints. In *Proc. 18th Int. Conference on Advances in GIS*, pages 192–201, 2010.

[9] S. Lamy, A. Ruas, Y. Demazeau, M. Jackson, W. Mackaness, and R. Weibel. The application of agents in automated map generalisation. In *Proc. 19th Int. Cartographic Conference*, 1999.

[10] H. Mayer. Scale-spaces for generalization of 3d buildings. *Int. Journal of Geographical Information Science*, 19:975–997, 2005.

[11] R. McMaster and K. Shea. *Generalization in Digital Cartography*. Association of American Cartographers, Washington D.C., 1992.

[12] W. Meulemans, A. van Renssen, and B. Speckmann. Area-preserving subdivision schematization. In *Proc. 6th Int. Conference on GIScience*, LNCS 6292, pages 160–174, 2010.

[13] G. Neyer. Line simplification with restricted orientations. In *Proc. 6th Int. Workshop on Algorithms and Data Structures*, LNCS 1663, pages 13–24, 1999.

[14] N. Regnauld, A. Edwardes, and M. Barrault. Strategies in building generalisation: modelling the sequence, constraining the choice. In *Proc. ICA '99 Workshop on Progress in Automated Map Generalization*, 1999.

[15] A. Ruas. *Modèle de généralisation de données géographiques à base de contraintes et d'autonomie*. PhD thesis, Université de Marne la Vallée, 1999.

[16] M. Sester. Generalization based on least squares adjustment. *Int. Archives of Photogrammetry and Remote Sensing*, 13:931–938, 2000.

[17] M. Sester. Optimization approaches for generalization and data abstraction. *Int. Journal of Geographical Information Science*, 19:871–897, 2005.

[18] J. Swan, S. Anand, M. Ware, and M. Jackson. Automated schematization for web service applications. In *Web and Wireless GISystems*, LNCS 4857, pages 216–226, 2007.

[19] A. Wolff. Drawing subway maps: A survey. *Informatik - Forschung und Entwicklung*, 22(1):23–44, 2007.

[20] H. Yan, R. Weibel, and B. Yang. A multi-parameter approach to automated building grouping and generalization. *GeoInformatica*, 12:73–89, 2008.