HAL
open science

Continuously Generalizing Buildings to Built-up Areas by Aggregating and Growing

Dongliang Peng, Guillaume Touya

To cite this version:

Dongliang Peng, Guillaume Touya. Continuously Generalizing Buildings to Built-up Areas by Aggregating and Growing. 3rd ACM SIGSPATIAL Workshop on Smart Cities and Urban Analytics (UrbanGIS'17), Nov 2017, Redondo Beach, CA, United States. pp.10, 10.1145/3152178.3152188 . hal-02095104

HAL Id: hal-02095104

https://hal.science/hal-02095104v1

Submitted on 10 Apr 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/320470854

Continuously Generalizing Buildings to Built-up Areas by Aggregating and Growing

Conference Paper · October 2017

DOI: 10.1145/3152178.3152188

CITATIONS
0

READS
52

2 authors:

Dongliang Peng
University of Wuerzburg
13 PUBLICATIONS **8** CITATIONS

SEE PROFILE

Guillaume Touya
Institut national de l'information géographique…
70 PUBLICATIONS **610** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project

Finding Optimal Area Aggregation Sequences with Comparing the A* Algorithm and Integer Linear Programming View project

Project

MapMuxing View project

All content following this page was uploaded by Dongliang Peng on 21 October 2017.

The user has requested enhancement of the downloaded file.

Continuously Generalizing Buildings to Built-up Areas by Aggregating and Growing

Dongliang Peng
Chair of Computer Science I, University of Würzburg
Würzburg, Germany
dongliang.peng@uni-wuerzburg.de

Guillaume Touya
COGIT, IGN
Saint-Mandé Cedex, France
guillaume.touya@ign.fr

ABSTRACT

To enable smooth zooming, we propose a method to continuously generalize buildings from a given start map to a smaller-scale goal map, where there are only built-up area polygons instead of individual building polygons. We name the buildings on the start map *original buildings*. For an intermediate scale, we aggregate the original buildings that will become too close by adding bridges. We grow (bridged) original buildings based on buffering, and simplify the grown buildings. We take into account the shapes of the buildings both at the previous map and goal map to make sure that the buildings are always growing. The running time of our method is in $O(n^3)$, where n is the number of edges of all the original buildings.

The advantages of our method are as follows. First, the buildings grow continuously and, at the same time, are simplified. Second, right angles of buildings are preserved during growing: the merged buildings still look like buildings. Third, the distances between buildings are always larger than a specified threshold. We do a case study to show the performances of our method.

CCS CONCEPTS

• **Applied computing** → **Cartography**;

KEYWORDS

multiple representation, zooming, simplification, buffer

ACM Reference Format:
Dongliang Peng and Guillaume Touya. 2017. Continuously Generalizing Buildings to Built-up Areas by Aggregating and Growing. In *UrbanGIS'17: UrbanGIS'17:3rd ACM SIGSPATIAL Workshop on Smart Cities and Urban Analytics , November 7–10, 2017, Redondo Beach, CA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3152178.3152188

1 INTRODUCTION

Digital multi-scale maps such as Google Maps and OpenStreetMap support zooming by displaying maps at different levels. This discrete strategy may results in sudden changes, which disturb user navigation. To provide better zooming experience, we try to produce a sequence of maps with small incremental changes from a level to another level. This process is known as *continuous generalization*.

A way to achieve continuous generalization is to use morphing. Often, a start map and a goal map, respectively at a larger scale and a smaller scale, are used as input, then maps at intermediate scales are produced while the start map is morphed to the goal map. In order to morph, correspondences between two maps need to be defined. For example, corresponding points between a pair of polylines have been investigated based on dynamic programming [17], Delaunay triangulations and binary line generalized tree (BLG-tree) [8], and simulated annealing [14]. From a point to its corresponding point, a straight-line trajectory is often used to interpolate. Peng et al. [19] defined trajectories based on least-square adjustment in order to obtain more reasonable intermediate-scale polylines. Using morphing, Peng et al. [20] continuously generalized administrative boundaries based on compatible triangulations. When the numbers of line features are different on the start and goal maps, a continuous selection is required and Chimani et al. [5] proposed to generate a selection sequence applicable for road network. They removed one road at each step while keeping the remaining roads connected.

These methods are interesting but only work on lines and our problem of building polygon interpolation cannot be achieved by similar morphings. Regarding the continuous generalization of polygon features, Danciger et al. [7] proposed to grow polygons when a map was zoomed out. Their method preserves polygons' topology, area-ratios, and relative positions. In the case where the goal map is an aggregated version of start land-cover map, Peng et al. [21] computed optimal aggregation sequences for land-cover areas.

Buildings are important elements on maps, many methods have been proposed to generalize them but not necessarily in a continuous way. For example, Haunert and Wolff [11] simplified a set of buildings based an integer program. Their simplification minimizes the number of edges of all the buildings and guarantees that the errors are smaller than a user-defined tolerance. At the same time, no topological conflict is introduced by the simplification. Buchin et al. [2] simplified buildings based on edge-move operations. Their method preserves orientations of edges.

When users zoom out on digital maps, buildings become smaller and the distances between buildings decrease. So simplification is not the only necessary operation, and buildings can also be aggregated to legible enough [29]. Several methods were proposed to aggregate buildings while preserving their shape (right angles remain) [6, 23, 24]. These algorithms can be used as inspirations to define a continuous transformation of buildings.

Algorithms were also proposed to create built-up area polygons (that appear in our goal map) from individual building polygons

ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of a national government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.
UrbanGIS'17, November 7–10, 2017, Redondo Beach, CA, USA
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5495-0/17/11…$15.00
https://doi.org/10.1145/3152178.3152188

(that appear in our start map). For instance, Chaudhry and Mackaness [4] identified the boundaries of urban settlement by calculating 'citiness' based on buildings. But the method cannot be adapted to provide a continuous transformation from the buildings to the built-up area.

Finally, some papers directly tackle the continuous transformation of buildings when scale is reduced. Li et al. [15] morphed between two buildings at different scales. They managed to preserve the orthogonal characteristics of buildings, but their algorithm cannot be used in our case as our goal map does not contain buildings anymore. Touya and Dumont [27] proposed a progressive transformation of buildings into a built-up area, where buildings are progressively replaced by the shape of the block they belong to. However, this last algorithm is not continuous enough, as each iteration directly transforms a set of buildings in a block into a polygon that covers the whole block. So there is no existing solution for the continuous generalization of building polygons into built-up area polygons.

Our contributions are as follows. In Section 2, we continuously generalize a start map, of buildings, to a smaller-scale goal map. The generalization consists of aggregating, growing, and simplifying. We aggregate original buildings which will be too close at an output scale by adding bridges. We grow (bridged) original buildings by buffering, where we use so-called *miter* joins to keep the right angles of buildings. Because of using this kind of joins instead of *round* ones, we have more problems. We show how to solve these problems. We also simplify the buildings at the output scale. Finally, an analysis of running time is presented at the end of this section. We carry out a case study and discuss the performances of our method in Section 3. We conclude our paper in Section 4.

2 METHODOLOGY

The input map is our start map. We denote the scale of the start map by $1 : M_s$. We generate a goal map at scale $1 : M_g$ ($M_g > M_s$) by generalizing the start map. We use time $t \in [0, 1]$ to define the continuous generalization process. We require that the generalization yields exactly the start map when $t = 0$ and the goal map when $t = 1$. The start map should be continuously changed to the goal map when t increases from 0 to 1. For the sake of convenience, we define parameter $M_t = M_s + t \cdot (M_g - M_s)$.

The continuous generalization is carried by dilating the original buildings. If dilated buildings become too close at time t, we aggregate the original buildings by adding bridges. We grow the (bridged) original buildings by buffering with miter joins. At any time t, the grown buildings need to be simplified to look like buildings. This simplification is carried out in two steps: the first one is to use dilation and erosion to remove "dents" and "bumps"; the second step is to remove vertices using Imai–Iri algorithm [12]. To make sure that buildings never shrink when t is increasing, we merge the shape of a building at time t and its shape at the immediately previous step (before t). Also, we clip the building using the shape on goal map to ensure some continuity.

The operators of the proposed method are presented in the following subsections:

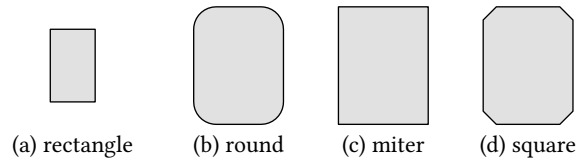
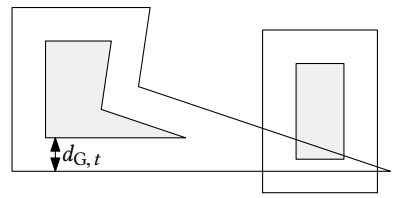
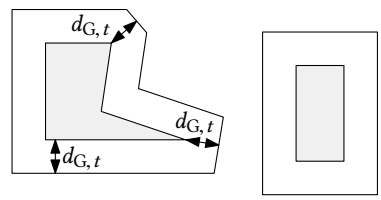


Figure 1: Buffering a polygon using round, miter, and square joins.



(a) buffering using miter joins, with distance $d_{G,t}$



(b) squaring if spikes are too long

Figure 2: Using square joins instead of miter joins to avoid long spikes.

2.1 Growing buildings by buffering

We denote by d_G the growth for goal map. At time t, the growth is

$$d_{G,t} = t \cdot d_G. \tag{1}$$

There are three typical joins when buffering a polygon, i.e., round, miter, and square joins (see Figure 1). We choose the miter joins to grow buildings in order to preserve the right angles. If an angle is acute, however, an excessively long *spike* will be produced. This spike may go across other buildings (see for example Figure 2a). To avoid this kind of interruptions, we require that if the tip of a spike is more than $\alpha d_{G,t} (\alpha \geq 1)$ away from the original vertex, a *square* join is applied (see Figure 2b). To keep right angles of buildings, we must have $\alpha \geq \sqrt{2}$. We set $\alpha = 1.5$. In this case, a square join will be applied when an angle is smaller (more acute) than 83.6°.

2.2 Simplifying grown buildings based on dilation and erosion

As mentioned earlier, building simplification methods have already been proposed. Damen et al. [6] generalized buildings using morphological operators. A drawback of this method is that the orientation of the buildings have to be identified. Meijers [16] simplified buildings using offset curves generated based on straight skeletons. Our method is similar to Meijers [16]. We dilate and erode the buildings to remove dents and bumps that can occur when buildings grow (see Figure 3).

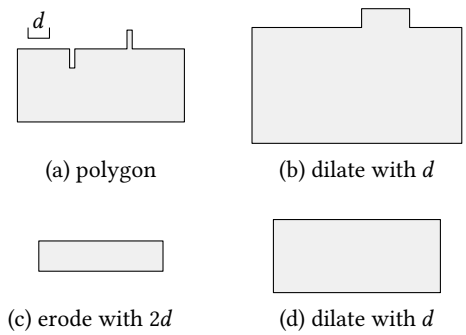


Figure 3: Removing a dent by dilating and then removing a bump by eroding. (a) A polygon with a dent and a bump. (b) Dilating the polygon in (a) with distance d to remove the dent. (c) Eroding the polygon in (b) with $2d$ to remove the bump. (d) Dilating the polygon in (c) with d so that the result has the same size as the polygon in (a).

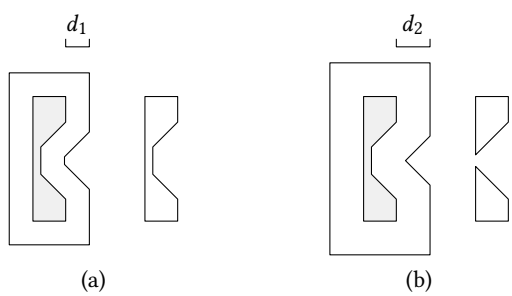


Figure 4: Dilating the gray polygon, then eroding the dilated polygon. (a) Dilating and eroding with distance d_1. (b) Dilating and eroding with d_2, where $d_2 > d_1$. In (a) the result of dilation and erosion is the same as the original polygon, while in (b) the result consists of two parts.

At time t, we should grow buildings with distance $d_{\mathrm{G},t}$. We dilate with distance $d_{\mathrm{D},t}$ ($d_{\mathrm{D},t} > 0$), erode with $d_{\mathrm{D},t} + d_{\mathrm{E},t}$($d_{\mathrm{E},t} > 0$), and dilate back with $d_{\mathrm{E},t}$. A problem of this process is that a building may be split into several parts by erosion (see Figure 4 for example). The reason is that some parts of a building may be increased (by growing and dilating) with distance $d_{\mathrm{G},t}+d_{\mathrm{D},t}$, but can be decreased (by eroding) as much as $\alpha(d_{\mathrm{D},t}+d_{\mathrm{E},t})$. If $d_{\mathrm{G},t}+d_{\mathrm{D},t} < \alpha(d_{\mathrm{D},t}+d_{\mathrm{E},t})$ and the building is not thick enough, a thin part may disappear. In order to avoid this problem, we require that

$$d_{\mathrm{G},t} + d_{\mathrm{D},t} \geq \alpha(d_{\mathrm{D},t} + d_{\mathrm{E},t}),$$

which means

$$d_{\mathrm{D},t} \leq \frac{d_{\mathrm{G},t} - d_{\mathrm{E},t}}{\alpha - 1}. \tag{2}$$

We would like to use $d_{\mathrm{E},t} = \frac{l}{2} \cdot M_t$ so that any dents and bumps narrower than l will be removed. We set $l = 0.3\,\mathrm{mm}$ in the map, which was used as a length threshold by, for example, Regnauld [23]. Unfortunately, distance $d_{\mathrm{G},t}$ can be arbitrarily small according to Equation 1, but $d_{\mathrm{E},t}$ is at least $\frac{l}{2}M_{\mathrm{s}}$. In this case, $d_{\mathrm{G},t} - d_{\mathrm{E},t} \leq 0$ when t is small, which violates Equation 2, where $d_{\mathrm{D},t} > 0$. As a compromise, we set erosion distance

$$d_{\mathrm{E},t} = t \cdot \frac{l}{2} M_{\mathrm{g}}. \tag{3}$$

Still, we have to make sure that $d_{\mathrm{G},t} - d_{\mathrm{E},t} > 0$, which means $t \cdot d_{\mathrm{G}} - t \cdot \frac{l}{2}M_{\mathrm{g}} > 0$. As a result, we need to make sure that

$$M_{\mathrm{g}} < \frac{2d_{\mathrm{G}}}{l}. \tag{4}$$

when we grow a bridged building, A "bay" may appear (see Figure 5b). We remove such a bay by dilating (see Figure 5c) and then eroding (see Figure 5d) with distance $d_{\mathrm{D},t}$. We define the width of a bay as the diameter of the largest circle that can be placed in the bay. If the width of a bay is smaller than $2d_{\mathrm{D},t}$, then the bay can be removed by dilating with distance $d_{\mathrm{D},t}$. We wish to remove bays which have widths less than $2r_{\mathrm{h}}$. Variable $r_{\mathrm{h}} = 2\sqrt{a_{\mathrm{h}}/\pi}$ is the radius of a hole which is just large enough to be presented on map. Following [4], we set area $a_{\mathrm{h}} = 8\,\mathrm{mm}^2$ on map. Sometimes, our $d_{\mathrm{D},t}$ is not large enough to remove a bay with width r_{h} because of the limitation from Equation 2. We define

$$d_{\mathrm{D},t} = \min\left(\frac{d_{\mathrm{G},t} - d_{\mathrm{E},t}}{\alpha - 1}, r_{\mathrm{h}}M_t\right). \tag{5}$$

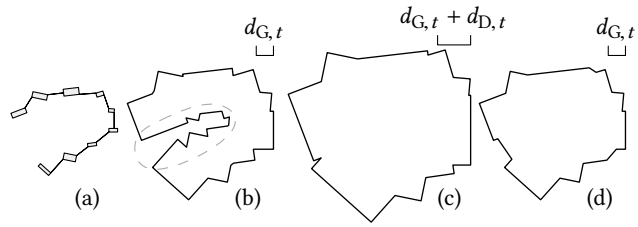


Figure 5: Removing a bay by dilating and eroding. (a) An aggregated building by adding bridges (see Section 2.3). (b) Growing the aggregated building with distance $d_{\mathrm{G},t}$, where the region marked by the dashed circle is a bay. (c) Dilating the grown building with $d_{\mathrm{D},t}$. (d) Eroding the dilated building with $d_{\mathrm{D},t}$.

2.3 Iteratively aggregating close buildings by adding bridges

We grow each original building and, as illustrated in Section 2.2, simplify the grown building. If some buildings become too close after these operations, we aggregate them by adding bridges (see for example Figure 6). Following Stoter et al. [25], we define that two buildings are too close if their distance is less than $\varepsilon = 0.2\,\mathrm{mm}$ on map. The real separation threshold at time t is

$$d_{\varepsilon,t} = \varepsilon \cdot M_t. \tag{6}$$

Our way of detecting close buildings is simple. We buffer buildings with distance $d_{\varepsilon,t}/2$ using round joins (see Figure 1b). We merge the buffers that overlap each other. The original buildings intersecting the same merged buffer are identified as a group of close buildings. For each pair of the original buildings in the same group, we connect them by adding a line segment linking the nearest points. The line segments may cross each other or even intersect buildings. To make the topology simple, we only select some of

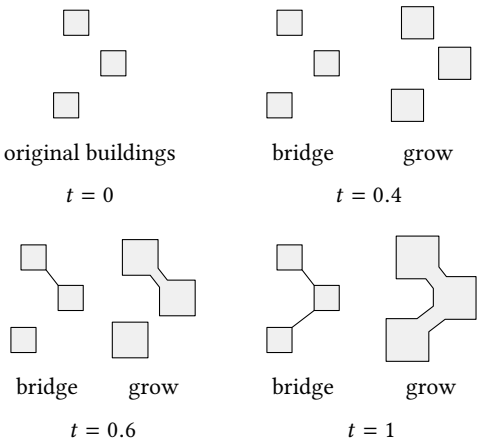


Figure 6: Aggregating original buildings in the same group by adding bridges. Then grow the bridged buildings.

the line segments as *bridges*. We consider each building as a node and each line segment as an edge, then we have a graph. Using Prim's algorithm [22], we find a minimum spanning tree (MST) in the graph, taking the lengths of the line segments as the weights. As a result, the line segments corresponding to the edges in the MST are identified as bridges. We aggregate the group of original buildings by adding these bridges.

Aggregated buildings may become too close because of the additional bridges. Therefore we have to iterate the aggregation process. Figure 7 shows such an example. For example, we grow and buffer buildings p, q, and r. As the buffers of q and r overlap (see Figure 7c), we aggregate buildings q and r by adding a bridge (see Figure 7d). There are two buildings left in Figure 7d. We then grow and buffer the buildings in Figure 7d, the buffer of building p overlap the buffer of the bridge for q and r (see Figure 7f). Finally, building p is aggregated with bridged q and r (see Figure 7g), and there is only one building left in Figure 7g. Then we grow and buffer again to get the final geometry of the group. As the number of buildings does not decrease comparing to last process (one building before, one building after), the iteration stops.

When buildings are grown and aggregated iteratively, bridges are added, and these bridges have a width of $2d_{\mathrm{G},t}$ at time t. This setting guarantees that no bridge is thin when $t = 1$. As we aggregated all buildings that will be too close, we are able to guarantee that there is no pair of buildings which has separation distance less than $d_{\varepsilon,t}$.

2.4 Simplifying aggregated buildings using Imai–Iri algorithm

As the scale is decreasing (M_t increases), we should remove more and more details. So, the aggregated and grown buildings are simplified using Imai–Iri algorithm [12]. First, this algorithm finds all the valid shortcuts of a polyline. A shortcut is valid for a segment if the distance between the segment and the shortcut is at most a specified value (see Figure 8). We set the value also as l. That is, at time t, the threshold distance is

$$d_{l,t} = l \cdot M_t. \tag{7}$$

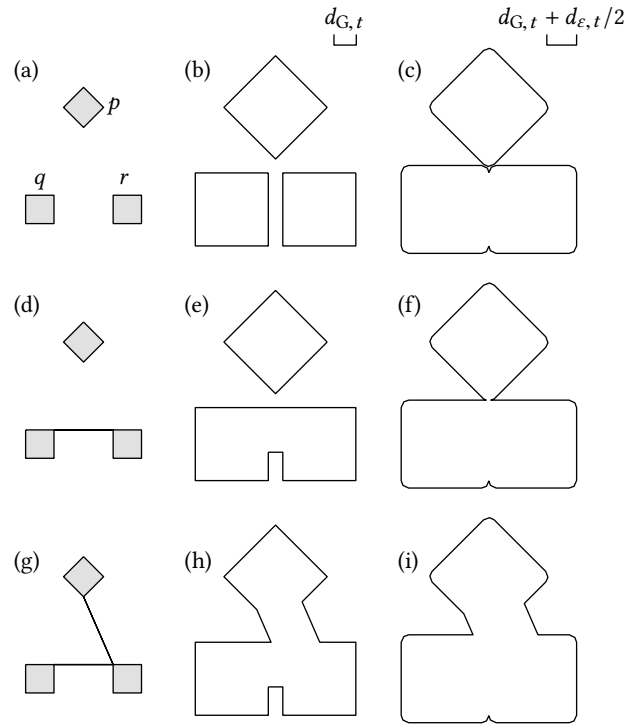


Figure 7: Iteratively aggregating close buildings by adding bridges. (a) Original buildings p, q, and r. (b) Growing the original buildings with distance $d_{\mathrm{G},t}$. (c) Buffering the grown buildings with distance $d_{\varepsilon,t}/2$ using round joins. (d) Aggregating buildings q and r by adding a bridge, as their buffer overlap each other in (c). (e) Growing the buildings in (d) with distance $d_{\mathrm{G},t}$. (f) Buffering the grown buildings in (e) with distance $d_{\varepsilon,t}/2$ using round joins. (g) Aggregating buildings p, q and r by adding bridges. (h) Growing the buildings in (g) with distance $d_{\mathrm{G},t}$. (i) Buffering the grown buildings in (h) with distance $d_{\varepsilon,t}/2$ using round joins.

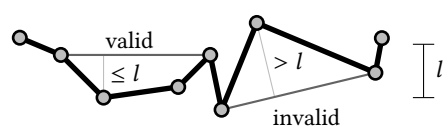


Figure 8: Valid and invalid shortcuts for Imai–Iri algorithm.

Second, the algorithm finds a sequence of valid shortcuts, from the beginning of a polyline to the end, using breadth-first search. The sequence of valid shortcuts is an approximation of the polyline and has the least number of line segments, with error smaller than $d_{l,t}$.

We add two more constraints for a shortcut to be valid. First, a shortcut must be completely inside the grown building. If a shortcut is outside, we may be not able to arrive at the shortcut by growing. Second, a shortcut is not allowed to intersect the aggregated building. If we allow this intersection, the building will have to be shrunk, which should be avoided.

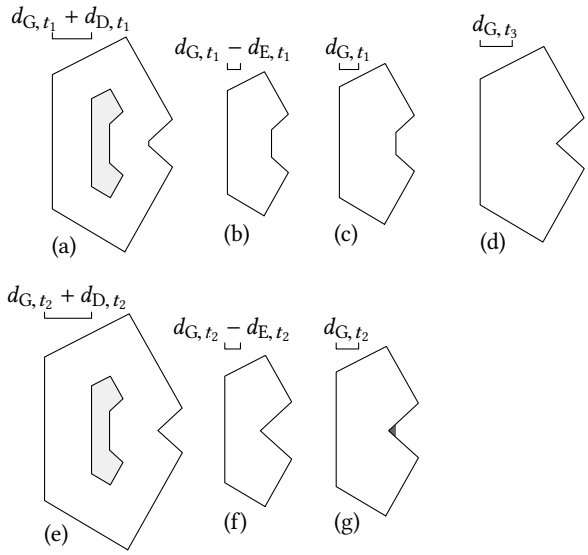


Figure 9: A building shrinks because of dilation and erosion, where $t_1 = 0.6$, $t_2 = 0.7$, and $t_3 = 1$. The gray polygons represent the original building. (a) Growing and Dilating the building with distances d_{G,t_1} and d_{D,t_1}, respectively; (b) Eroding the polygon in (a) with $d_{D,t_1} + d_{E,t_1}$; (c) Dilating the polygon in (b) with d_{E,t_1}. (d) The goal shape. The process of (e), (f), and (g) is the same as the process of (a), (b), and (c). The darker gray piece in (g) shows the part which is included in the polygon of (c), but not in the polygon of (g).

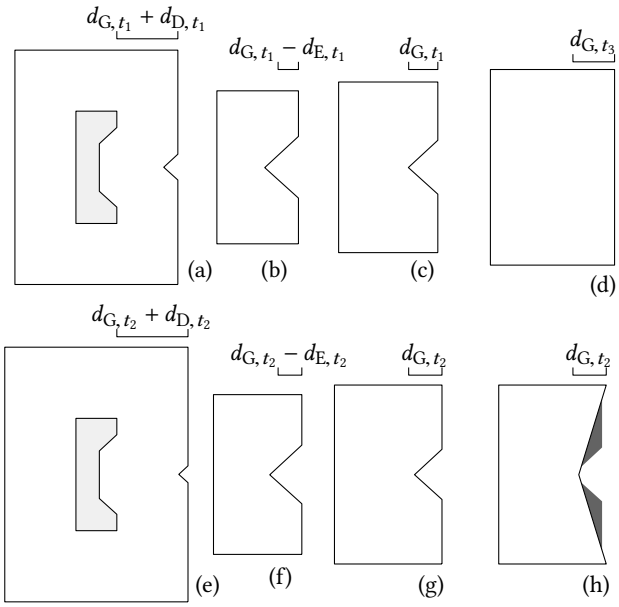


Figure 10: A building shrinks because of line simplification, where $t_1 = 0.7$, $t_2 = 0.8$, and $t_3 = 1$. The gray polygons represent the original building. The process of (a), (b), and (c), and the process of (e), (f), and (g) are the same as the processes in Figure 9. (d) The goal shape. (h) Simplifying the polygon in (g) using the Imai–Iri algorithm. Note that $d_{l,t_1} < d_{l,t_2}$ (see Equation 7), which is why the Imai–Iri algorithm does not remove any vertex of the polygon in (c), but removes two vertices of the polygon in (g). The darker gray pieces in (h) show the parts which are included in the polygon of (c), but not in the polygon of (h).

The classical way to filter points in a polyline or polygon is the Douglas–Peucker algorithm [9], but it did not filter enough the geometry, or damaged too much the shapes when the threshold was bigger, so the Imai–Iri algorithm was chosen as a better fit.

2.5 Generating buildings on intermediate-scale maps

Both the erosion and line simplification may result in shrinking a building. Figure 9 and Figure 10 show such examples respectively. To avoid these kinds of shrinking, for a building, we merge its shape at time t and its shape at the immediately previous step (before t). For example, we generate a sequence of 10 maps, which means $t \in \{0.1, 0.2, \ldots, 1\}$. Figure 9c shows the result at $t = 0.6$. In Figure 9g, the dark-gray part included in the result at $t = 0.6$, but not in the result at $t = 0.7$. In other words, the result at $t = 0.7$ shrinks at the dark-gray part. To prevent the shrinking at $t = 0.7$, we merge the result at $t = 0.7$ with the result of the immediate previous step, i.e., $t = 0.6$. The merged result is shown in Figure 11a. Similarly, Figure 11b shows the merged result of buildings in Figure 10c and Figure 10h. This merge also avoids bridges' shrinking. Figure 12 shows such an example.

Added to this shrinking problem, a building aggregate on an intermediate map should never exceed the goal shape of the aggregated building. Otherwise, the building will need to be shrunk to ensure continuity with the goal built-up polygon. To guarantee this consistency, we clip the building using the goal shape and remove the outside parts.

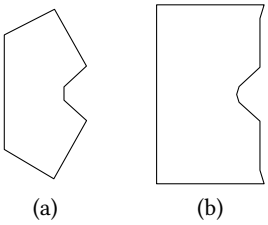


Figure 11: Merging a polygon with the polygon at the immediately previous step. (a) Merging the polygons of Figure 9c and Figure 9g. (b) Merging the polygons of Figure 10c and Figure 10h.

2.6 Eliminating small buildings

Following the previous steps of continuous generalization may result in the creation of small isolated building aggregates that do not belong to the goal map polygon because of their isolation. That is why we eliminate a building aggregate if its area is smaller than a threshold. Following Stoter et al. [25] and Chaudhry and

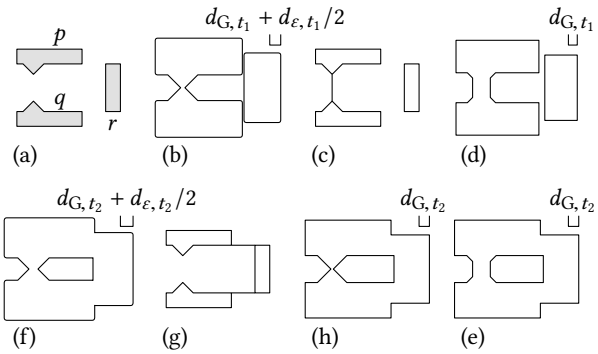


Figure 12: Avoiding shrinking resulted by moved bridges, where $t_1 = 0.5$ and $t_2 = 0.6$. (a) Original buildings. (b) Growing buildings with distance d_{G,t_1} and dilating with $d_{\varepsilon,t_1}/2$; buildings p and q are identified as in the same group. (c) Aggregating p and q by adding a bridge. (d) Growing the (bridged) buildings in (c) with distance d_{G,t_1}. (f) Growing original buildings with distance d_{G,t_2} and dilating with $d_{\varepsilon,t_2}/2$; all the three buildings are in the same group (g) Aggregating by adding bridges according to the MST. (h) Growing the bridged building in (g) with distance d_{G,t_2}. The bridge in (d) shrinks comparing to (h). (e) Avoiding this shrinking at time t_2 by merging buildings of (d) and (h).

Mackaness [4], we set this threshold as $a = 0.16\,\text{mm}^2$ on map. The real threshold at time t is

$$a_t = a \cdot M_t^2.$$

For a group of buildings that will be aggregated into one built-up area at time $t = 1$, we consider the area sum of all the buildings at any time t, instead of considering each building individually.

As mentioned in Section 2.2, we remove holes that have area less than $a_h = 8\,\text{mm}^2$ on map. The real area threshold for a hole at time t is

$$a_{h,t} = a_h \cdot M_t^2.$$

2.7 Running time

Suppose that our input has n edges in total. Operations like growing, dilation, erosion, merge, and clip cost time $O(n^2)$ [10, 18]. We iteratively aggregate in Section 2.3. In the worst case, we need to repeat $O(n)$ times, which increases our running time to $O(n^3)$. It is unlikely that we need to repeat the aggregation more than twice, though. Simplifying polygons using Imai–Iri algorithm needs time $O(n^3)$. The improved version of Imai–Iri algorithm by Chan and Chin [3] does not help in our case because we have more constraints when simplifying (see Section 2.4). As a result, the running time of our method is in $O(n^3)$.

3 CASE STUDY

We have implemented our method based on C# (Microsoft Visual Studio 2015) and ArcObjects SDK 10.4.1. The code is available in open source on Github[1]. The offsetting function and clipping function are available from library Clipper of Johnson [13], which is

[1] https://github.com/IGNF/Continuous-Generalisation

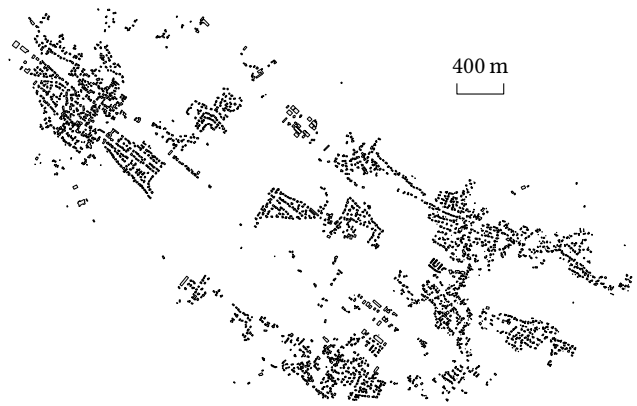


Figure 13: Data. There are 2,590 buildings, which in total have 19,255 edges and have area 448,996.8 m^2.

based on the clipping algorithm of Vatti [28]. The offsetting function is used for the buffering, dilation, erosion, and merge operations. We ran our case study under 64-bit Windows 7 on a 3.3 GHz dual core CPU with 8 GB RAM. We measured processing time by the built-in C# class *Stopwatch*. Our testing data is extracted from a dataset produced by the French Mapping Agency (IGN); see Figure 13. The data is at scal 1 : 15,000, which means $M_s = 15{,}000$. It represents the buildings of four towns, i.e., Aussevielle, Denguin, Poey-de-Lescar, and Siros, in the Pyrénées-Atlantiques county, south-western France. IGN also stores a dataset at scale 1 : 50,000. This dataset was basically obtained from the data at scale 1 : 15,000 by buffering with distance 25 m, where sometimes distance 50 m was also used in order to identify towns. A restriction for the town from Boffet [1] is that the longest edge in an MST of the buildings should be shorter than 100 m.

We set our goal scale at 1 : 50,000, which means $M_g = 50{,}000$, and $d_G = 25$ m so that we can compare our result with the existing data. Also, this setting makes Equation 4 hold, where $l = 0.3$ mm. In Equation 5, the first part is always smaller than the second part according to our settings. As a result $d_{D,t} = t \cdot 35$ m, where $d_{E,t} = t \cdot 7.5$ m according to Equation 3 and $\alpha = 1.5$.

Our program took 93.6 s to compute the goal shapes of the built-up areas. The 56 built-up areas have 2,095 edges before line simplification. Using the Imai–Iri algorithm, we have 1,102 edges left. In comparison, there are 1,597 edges left when we simplified using the Douglas–Peucker algorithm.

Figure 14 shows the bridged original buildings as well as our result of built-up areas at time $t = 1$ (almost the same as the goal shapes). The transparent polygons in Figure 14 are the data of built-up areas at scale 1 : 50,000 from IGN. No small building was removed in our result or the IGN data. The boundaries of our built-up areas are more straight than that of IGN data, where we have 1,135 edges while the data has 4,968 edges. From this perspective, we would say that our result is more reasonable than the existing data. A questionnaire, however, is needed to make a more convincing comparison.

We produced a sequence of 10 maps, i.e., $t \in \{0.1, 0.2, \ldots, 1\}$. This production costs 668.2 s in total. We show such a sequence of maps in Figure 15 for the marked region in Figure 14. We counted

Figure 14: Bridged original buildings, goal shapes at scale 1 : 50,000 (darker polygons), and the built-up areas at scale 1 : 50,000 from IGN (transparent polygons). Some built-up areas from IGN are split because of streets' crossing. There are 56 goal shapes, which have 1,135 edges in total. We show a sequence of maps in Figure 15 for the region marked by the dashed circle.

the numbers of buildings in our results, and compared them to the numbers calculated by Töpfer's radical law (see Figure 16) and our *areal law*. We have unexaggerated-area symbols, so for Equation 2 of Töpfer and Pillewizer [26] we used C_{b2} and C_{z3}. As a result, we computed the numbers according to

$$n_t = n_s \frac{M_s}{M_t},$$

where $n_s = 2{,}590$ is the number of buildings on start map and n_t is the number of buildings on a map at scale $1 : M_t$. For the areal law, we argue that the number of buildings in a unit area on map should be fixed, which leads to

$$n'_t = n_s \left(\frac{M_s}{M_t}\right)^2.$$

According to Figure 16, our numbers decrease faster than the numbers of both the radical law and the areal law.

4 CONCLUSION

We proposed a method to continuously generalize buildings to built-up areas by aggregating and growing. We managed to produce a sequence of maps in which the buildings are always growing and, at the same time, are simplified. For the goal map at scale 1 : 50,000, the shapes of our built-up areas are more reasonable than the data from IGN.

It is always interesting to know the quantity we should keep on a map. We compared the numbers of buildings, and it is more consistent with the areal law than Töpfer's radical law. Another interesting problem is to compare the area of the buildings. Eventually, our result is a set of settlement boundaries. An interesting problem is to compare our method with Chaudhry and Mackaness [4]. Our method is supposed to provide a smooth transition between representations with individual buildings and representations with built-up areas, but the only way to verify that smoothness is achieved is to carry out a user survey where our approach is compared to non continuous generalization approaches.

REFERENCES

[1] Annabelle Boffet. 2000. Creating urban information for cartographic generalisation. In *9th International Symposium on Spatial Data Handling (SDH'00)*. Beijing, China.

[2] Kevin Buchin, Wouter Meulemans, and Bettina Speckmann. 2011. A new method for subdivision simplification with applications to urban-area generalization.. In *GIS*, Isabel F. Cruz, Divyakant Agrawal, Christian S. Jensen, Eyal Ofek, and Egemen Tanin (Eds.). ACM, 261–270. http://dblp.uni-trier.de/db/conf/gis/gis2011.html#BuchinMS11

[3] W.S. Chan and F. Chin. 1996. Approximation of Polygonal Curves with Minimum Number of Line Segments or Minimum Error. *International Journal of Computational Geometry & Applications* 06, 01 (1996), 59–77. https://doi.org/10.1142/S0218195996000058 arXiv:http://www.worldscientific.com/doi/pdf/10.1142/S0218195996000058

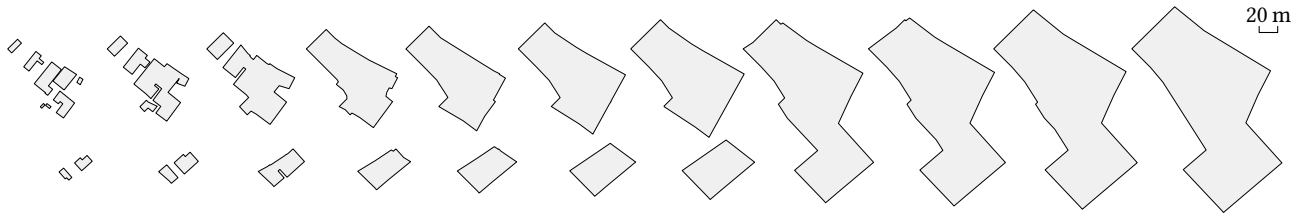


Figure 15: A sequence of maps at time $t \in \{0, 0.1, 0.2, \ldots, 1\}$ for the region marked in Figure 14.

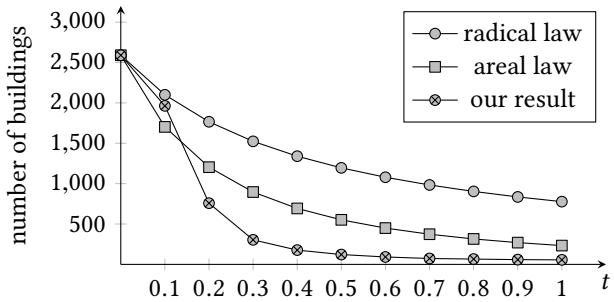


Figure 16: A comparison of the numbers of buildings between our result, areal law, and Töpfer's radical law.

[4] Omair Chaudhry and William A. Mackaness. 2008. Automatic identification of urban settlement boundaries for multiple representation databases. *Computers, Environment and Urban Systems* 32, 2 (2008), 95–109. https://doi.org/10.1016/j.compenvurbsys.2007.09.001

[5] Markus Chimani, Thomas C. van Dijk, and Jan-Henrik Haunert. 2014. How to eat a graph: computing selection sequences for the continuous generalization of road networks. In *22nd ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems (ACMGIS'14)*. 243–252. http://doi.acm.org/10.1145/2666310.2666414

[6] Jonathan Damen, Marc van Kreveld, and Bert Spaan. 2008. High quality building generalization by extending the morphological operators, In 12th ICA Workshop on Generalisation and Multiple Representation. *11th ICA Workshop on Generalization and Multiple Representation, Montpellier, France*. June 20–21.

[7] Jeff Danciger, Satyan L. Devadoss, John Mugno, Don Sheehy, and Rachel Ward. 2009. Shape deformation in continuous map generalization. *Geoinformatica* 13 (2009), 203–221.

[8] Min Deng and Dongliang Peng. 2015. Morphing linear features based on their entire structures. *Transactions in GIS* 19, 5 (2015), 653–677.

[9] David H. Douglas and Thomas K. Peucker. 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica* 10, 2 (1973), 112–122.

[10] Günther Greiner and Kai Hormann. 1998. Efficient Clipping of Arbitrary Polygons. *ACM Trans. Graph.* 17, 2 (1998), 71–83. https://doi.org/10.1145/274363.274364

[11] Jan-Henrik Haunert and Alexander Wolff. 2010. Optimal and Topologically Safe Simplification of Building Footprints. In *18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACMGIS'10)*, A. E. Abbadi, D. Agrawal, M. Mokbel, and P. Zhang (Eds.). San Jose, CA, USA, 192–201. http://dl.acm.org/citation.cfm?doid=1869790.1869819

[12] Hiroshi Imai and Masao Iri. 1988. Polygonal Approximations of a Curve — Formulations and Algorithms. In *Machine Intelligence and Pattern Recognition*, Godfried T. Toussaint (Ed.). Computational Morphology: A Computational Geometric Approach to the Analysis Of Form, Vol. 6. Elsevier, 71–86.

[13] Angus Johnson. 2014. Clipper – an open source freeware library for clipping and offsetting lines and polygons. http://www.angusj.com/delphi/clipper.php. (2014). Accessed: 2017-08-22.

[14] Jingzhong Li, Tinghua Ai, Pengcheng Liu, and Min Yang. 2017. Continuous Scale Transformations of Linear Features Using Simulated Annealing-Based Morphing. *ISPRS International Journal of Geo-Information* 6, 8 (2017). https://doi.org/10.3390/ijgi6080242

[15] Jingzhong Li, Xingong Li, and Tian Xie. 2017. Morphing of Building Footprints Using a Turning Angle Function. *ISPRS International Journal of Geo-Information* 6, 6 (2017). https://doi.org/10.3390/ijgi6060173

[16] Martijn Meijers. 2016. Building simplification using offset curves obtained from the straight skeleton. In *19th ICA Workshop on Generalisation and Multiple Representation (ICAGW'16)*. Helsinki, China.

[17] Martin Nöllenburg, Damian Merrick, Alexander Wolff, and Marc Benkert. 2008. Morphing polylines: a step towards continuous Generalization. *Computers, Environment and Urban Systems* 32, 4 (2008), 248–260.

[18] Peter Palfrader and Martin Held. 2015. Computing Mitered Offset Curves Based on Straight Skeletons. *Computer-Aided Design and Applications* 12, 4 (2015), 414–424. https://doi.org/10.1080/16864360.2014.997637 arXiv:http://dx.doi.org/10.1080/16864360.2014.997637

[19] Dongliang Peng, Jan-Henrik Haunert, Alexander Wolff, and Christophe Hurter. 2013. Morphing polylines based on least squares adjustment. In *16th ICA Workshop on Generalisation and Multiple Representation (ICAGW'13)*. Dresden, Germany.

[20] Dongliang Peng, Alexander Wolff, and Jan-Henrik Haunert. 2016. Continuous Generalization of Administrative Boundaries Based on Compatible Triangulations. In *Geospatial Data in a Changing World: Selected papers of the 19th AGILE Conference on Geographic Information Science (AGILE'16) (Lecture Notes in Geoinformation and Cartography)*, Tapani Sarjakoski, Yasmina Maribel Santos, and Tiina L. Sarjakoski (Eds.). Springer, Helsinki, Finland, 399–415. https://doi.org/10.1007/978-3-319-33783-8_23

[21] Dongliang Peng, Alexander Wolff, and Jan-Henrik Haunert. 2017. Using the A* Algorithm to Find Optimal Sequences for Area Aggregation. In *Advances in Cartography and GIScience: Selections from the 28th International Cartographic Conference (ICC'17) (Lecture Notes in Geoinformation and Cartography: Publications of the International Cartographic Association (ICA))*, Michael P. Peterson (Ed.). Springer, Washington DC, USA, 389–404. https://doi.org/10.1007/978-3-319-57336-6_27

[22] Robert C. Prim. 1957. Shortest connection networks and some generalizations. *The Bell System Technical Journal* 36, 6 (1957), 1389–1401. https://doi.org/10.1002/j.1538-7305.1957.tb01515.x

[23] Nicolas Regnauld. 2001. Contextual Building Typification in Automated Map Generalization. *Algorithmica* 30, 2 (2001), 312–333. https://doi.org/10.1007/s00453-001-0008-8

[24] Nicolas Regnauld and Patrick Revell. 2007. Automatic Amalgamation of Buildings for Producing Ordnance Survey 1:50 000 Scale Maps. *The Cartographic Journal* 44, 3 (2007), 239–250. https://doi.org/10.1179/000870407x241782

[25] Jantien Stoter, Dirk Burghardt, Cécile Duchêne, Blanca Baella, Nico Bakker, Connie Blok, Maria Pla, Nicolas Regnauld, Guillaume Touya, and Stefan Schmid. 2009. Methodology for evaluating automated map generalization in commercial software. *Computers, Environment and Urban Systems* 33, 5 (2009), 311–324. https://doi.org/10.1016/j.compenvurbsys.2009.06.002 Geo-information Generalisation and Multiple Representation.

[26] F. Töpfer and W. Pillewizer. 1966. The Principles of Selection. *The Cartographic Journal* 3, 1 (1966), 10–16. https://doi.org/10.1179/caj.1966.3.1.10 arXiv:http://dx.doi.org/10.1179/caj.1966.3.1.10

[27] Guillaume Touya and Marion Dumont. 2017. Progressive Block Graying and Landmarks Enhancing as Intermediate Representations between Buildings and Urban Areas. In *20th ICA Workshop on Generalisation and Multiple Representation*. Washington DC, USA.

[28] Bala R. Vatti. 1992. A Generic Solution to Polygon Clipping. *Commun. ACM* 35, 7 (1992), 56–63. https://doi.org/10.1145/129902.129906

[29] Robert Weibel. 1997. Generalization of spatial data: Principles and selected algorithms. In *Algorithmic Foundations of Geographic Information Systems*, Marc van Kreveld, Jürg Nievergelt, Thomas Roos, and Peter Widmayer (Eds.). Lecture Notes in Computer Science, Vol. 1340. Springer, Chapter 5, 99–152. https://doi.org/10.1007/3-540-63818-0_5

View publication stats