Enhancing building footprints with squaring operations

Guillaume Touya, Imran Lokhat

JOURNAL OF SPATIAL INFORMATION SCIENCE
Number 13 (2016), pp. 33–60
doi:10.5311/JOSIS.2016.13.276

RESEARCH ARTICLE

Enhancing building footprints with squaring operations

Imran Lokhat and Guillaume Touya

COGIT team, Institut géographique national (IGN), France

Received: January 13, 2016; returned: March 29, 2016; revised: May 11, 2016; accepted: August 15, 2016.

Abstract: Whatever the data source, or the capture process, the creation of a building footprint in a geographical dataset is error prone. Building footprints are designed with square angles, but once in a geographical dataset, the angles may not be exactly square. The almost-square angles blur the legibility of the footprints when displayed on maps, but might also be propagated in further applications based on the footprints, e.g., 3D city model construction. This paper proposes two new methods to square such buildings: a simple one, and a more complex one based on nonlinear least squares. The latter squares right and flat angles by iteratively moving vertices, while preserving the initial shape and position of the buildings. The methods are tested on real datasets and assessed against existing methods, proving the usefulness of the contribution. Direct applications of the squaring transformation, such as OpenStreetMap enhancement, or map generalization are presented.

Keywords: squaring, OpenStreetMap, shape, map generalization, least squares, orientation

1 Introduction

Geographical information abstracted and captured as vector data is error prone, and source errors [17] are an important part of these errors. The source errors can originate from the source itself, e.g., GPS imprecision, or satellite imagery resolution, or they can originate from the capture process, e.g., manual capture with GIS tools, or automatic image segmentation. When geographic information is voluntarily contributed, the errors can be more frequent as some sources might be less professional and the contributor skills more heterogeneous [9]. Buildings are mostly built to be perfectly square but the source errors may alter this squareness: for instance, some 90° angles may be captured as 87° angles on the

footprint. This is what Savino and Rumor call a shape regularity anomaly [31]. This alteration of squareness does not make the building footprints unusable because the defects are often nearly imperceptible. However, this adds noise to the cartographic visualization of such data, and may convey errors in further applications that use the footprint geometries. Is it possible to automatically correct the squareness alteration while preserving the positional accuracy and the shape of the building footprints? This paper tries to answer this question proposing two new algorithms to square building footprint polygons.

The next section presents related work on squareness and polygon squaring algorithms. Section 3 describes our first proposition to square building footprints, using a naive approach that squares polygon segments one after another. Section 4 presents our main contribution with a squaring algorithm based on a least squares adjustment. Section 5 details the experiments carried out to validate and evaluate our propositions, and compare them to existing methods. Finally, Section 6 proposes direct applications of the algorithm to enhance OpenStreetMap data or to improve cartographic generalization.

2 Related work

The need for squaring building footprints is not new, and some algorithms have already been proposed. The first proposition from [1] seeks to enhance footprints that are not totally squared in a topographic dataset, because of capture specifications that do not require strict square angles. The algorithm squares almost-right angles while preserving wall regularities such as symmetries and alignments. The algorithm is based on the optimization of an energy function, by moving all the points of the building footprint. The energy is computed according to several constraints:

- a *squareness* constraint to square almost-right angles,
- a *movement* constraint that prevents points from moving too far from their initial position,
- a *parallelism* constraint that forces parallel segments to remain parallel,
- an *alignment* constraint that preserves wall alignments between too close buildings, and
- a *topology* constraint that prevents from topology problems inside a footprint and with close buildings.

The results are quite compelling, particularly regarding the preservation of initial characteristics of the buildings, although the authors state that the algorithm fails on buildings that do not need squaring, because the walls are altered when they should be left unchanged.

Cartographic generalization is a process that seeks to simplify geographic information and display it in a legible way at a smaller scale. The aim of cartographic generalization is more to convey the geography behind the map, i.e., preserve and enhance spatial relations and structures, more than convey the exact positioning of map objects [20]. As a consequence, most generalization processes require exaggeration operations such as typification, parallelism enhancement and squaring [21]. Some squaring operations have been embedded into simplification algorithms dedicated to buildings, e.g., [15, 29]. Such algorithms are effective but are not applicable to buildings that should only be squared and not simplified. Additionally, specific squaring algorithms have been developed to complete

the simplification algorithms. The generalization process developed during the European project AGENT [3, 30] includes a squaring operation partially inspired from [1]. Gaffuri proposed another algorithm based on a multi-agent generalization process, namely GAEL, that displaces the vertices of the building so that all nearly right angles become perfectly right angled [8]. The algorithm uses the main wall orientations [7] as a principle governing the squaring process.

The squaring of polygons has also been tackled by more global cartographic generalization processes that try to solve generalization with an optimization framework: constraints, including the squareness of buildings are put in equations and an optimization method, such as the finite elements [16], or a least squares adjustment [13, 32], computes the position of the vertices of map features that balances the constraint for shape preservation with the constraints for map legibility. These methods are interesting because they show how squareness can be modeled in equations in an optimization framework, even though they cannot directly be used when simplification is not required.

Map schematization is a process to simplify maps to a lower graphical complexity compared to topographic maps, by caricature and minimization of non-functional details [19]. When creating schematized maps, squaring the angles is often a means to obtain lower graphical complexity, and this is the case for instance when generating schematized metro maps [4, 24, 37]. The methods to force angle squaring in such schematization are interesting sources of inspiration for our building footprint squaring issue. In the subdivision schematization method from [23], there is a rectilinearization to make all angles flat or right. Although the method is not directly reproducible in our context, where angles may be different from 90° or 180°, the techniques proposed to simplify the rectilinearized polygons could be used to make a building squaring algorithm.

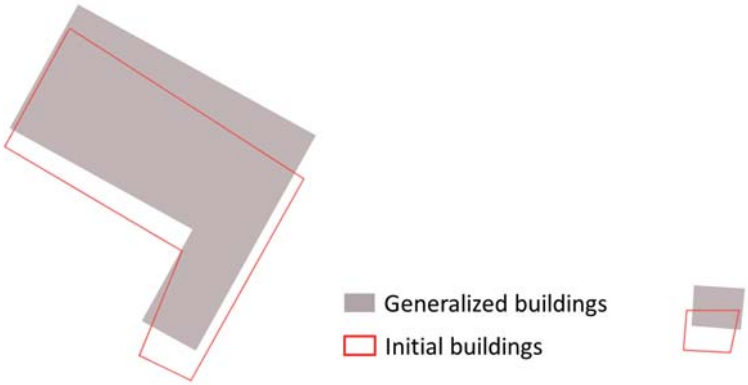

Figure 1: Two buildings squared with the GAEL algorithm where the squaring also caused a displacement.

Given these existing methods, we first attempted to use the GAEL squaring algorithm since it was available on our research platform, GeOxygene [12, 28]. However, experiments showed some problems with the algorithm: the squared footprints were sometimes displaced (Figure 1), and more rarely, the process diverges, causing large distortions: this algorithm iteratively moves points to a balanced position to optimize geometrical constraints satisfaction, but sometimes, the iterations only make the constraints balance worse, and

the algorithm fails. So we decided to propose two new algorithms: a naive and quick one, inspired from [1], presented in Section 3; and a least squares based algorithms following some principles of [32], presented in Section 4.

3 A naive polygon squaring algorithm

The approach examines each angle one by one, and makes adjacent line segments straight or perpendicular by moving one of the edges through a narrow angle band (i.e., 180°, such angles are referred to as "flat" in the remainder of the paper), right angled, or 45°, within a tolerance range, of 0.5°. This tolerance range, as well as the angle margin, are parameters of the algorithm. To achieve these squarings, a two pass process is performed:

- During the first pass, the almost-180° angles are flattened by projecting the vertex on the flat line joining the two points around it (Figure 2).
- Then, in a second step, we rotate one edge of a candidate angle (one that is almost 90° or 45° angle) in order to square the angle. The chosen edge is the one *less aligned*, i.e., the more orthogonal, with a vector co-linear to the overall orientation of the polygon. We could have chosen the orientation of the longest edge or the length of the smallest surrounding rectangle; we chose to use the main orientation of the polygon as defined in [7], which is a $[0,\pi]$ orientation computed by measuring the orientation of the longest edge of the smallest minimum bounding rectangle.

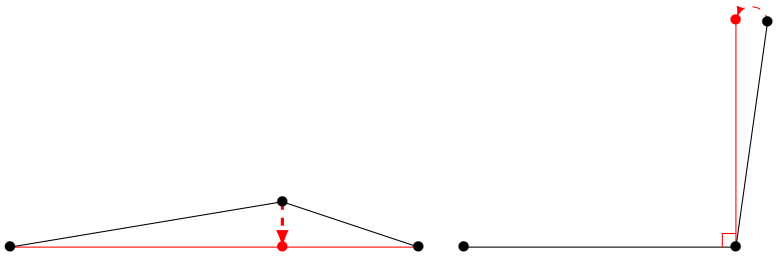

Figure 2: Flattening an almost-straight line.

Some minor conditions are evaluated before moving an edge (i.e., the vertex at its end): if the preceding angle is right angled and the edge to rotate is the one before the angle, or if the two angles surrounding the candidate angle are 90°, nothing is done, because squaring the angle would cause distortion of the polygon. The order of angle processing in the algorithm clearly matters: experiments showed that different orders of angle processing could lead to different squared shapes, although the number of angles that are squared remains stable. To avoid this instability, three squared shapes are computed with different orders (the order of vertices in the geometry, the reverse order, and a random order), and we keep only the squared solution that minimizes the total vertex displacement. We made this choice in order to keep the algorithm as quick as possible, and experiments showed that those three iterations were a good compromise between quality of the result and computation time.

The algorithm is clearly dedicated to simple building polygons having fewer vertices and only one wall orientation [7], i.e., walls are more or less in the same direction or almost

perpendicular to this direction. On more complex buildings, the algorithm might fail to square the building, either because the main orientation is not significant (some buildings do not have a single orientation), or because moving an edge "un-squares" another angle. That is why we proposed a second algorithm presented in next section.

4 Least squares based polygon squaring

In order to complete the naive simple algorithm that we knew to be limited to simple polygons, we targeted a more generic framework for squaring, able to model more constraints such as the ones used in [1], and described in Section 2. Least squares adjustment is a mathematical technique used to approximate a solution for an overdetermined system, i.e., an equation system where there are more equations than unknowns. It is often used as an optimization technique in geographic information science, and is well adapted to spatial deformation. It has been used for instance for cartographic generalization [13, 32], for geometric conflation [22, 34], and for line morphing [26].

4.1 Algorithm description

The model is based on five constraints applied to the 2D coordinates of the polygon vertices:

- a constraint $\mathbf{C}_1$ to express the fact that we wish to minimize the movement of points from their initial position (similar constraints are used in [13, 32, 34]),
- a constraint $\mathbf{C}_2$ to force perfect squareness upon almost-right angles, i.e., the angles should be 90° with a small tolerance (e.g., 0.5°),
- a constraint $\mathbf{C}_3$ to force perfect squareness upon almost-45° or 135° angles,
- a constraint $\mathbf{C}_4$ to force the straightening of almost-straight lines, and
- a constraint $\mathbf{C}_5$ to preserve the parallelism of two parallel wall segments.

As in the simple squaring algorithm, the margin either side of 90°, 180°, or 45° angles is a parameter of the algorithm. This parameter should be rather small, but not necessarily the same for each type of angle (see Section 5.2 for the values used in the experiments). Thus each angle is only constrained by one of the three C_2 to C_4 constraints at most. Only the angles that are outside the tolerance (e.g., smaller than 89.5° or bigger than 90.5° for right angles) and inside the margin (e.g., bigger than 80° or smaller than 100° for right angles) are considered in the least squares adjustment, the other angles are fixed.

We note the three consecutive points of angle i: P_{i-1}, P_i and P_{i+1}. We define the two consecutive vectors, $\vec{u}$ and $\vec{v}$ from these points (Figure 3).

Then, simply using dot products and cross product (in fact the z component resulting from the cross product) on the points' coordinates, we define the functions F and G to express the constraints on potential angles candidates (angles not far from 45, 90, 135 or 180 degrees).
Let $P_i(x_i, y_i)$, $\vec{u} = \overrightarrow{P_{i-1}P_i}$ and $\vec{v} = \overrightarrow{P_iP_{i+1}}$, we have :

$$F = \vec{u} \cdot \vec{v} = (x_i - x_{i-1}) \cdot (x_{i+1} - x_i) + (y_i - y_{i-1}) \cdot (y_{i+1} - y_i) \quad (1)$$

$$G = \vec{u} \times \vec{v} = (x_i - x_{i-1}) \cdot (y_{i+1} - y_i) - (y_i - y_{i-1}) \cdot (x_{i+1} - x_i) \quad (2)$$

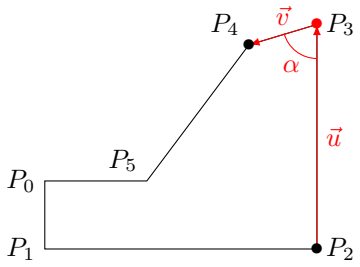

Figure 3: Angle constraints are tested with two consecutive vectors around its vertex.

Using equations 1 and 2, the constraints on an angle α can be translated into the following equations:

$$C_2 : (x_i - x_{i-1}) \cdot (x_{i+1} - x_i) + (y_i - y_{i-1}) \cdot (y_{i+1} - y_i) = 0 \tag{3}$$

$$C_3 : (x_i - x_{i-1}) \cdot (x_{i+1} - x_i) + (y_i - y_{i-1}) \cdot (y_{i+1} - y_i) = cos(\frac{\pi}{4}) \tag{4}$$

$$C_4 : (x_i - x_{i-1}) \cdot (y_{i+1} - y_i) - (y_i - y_{i-1}) \cdot (x_{i+1} - x_i) = 0 \tag{5}$$

The constraint C_5, which seeks to preserve the parallelism of two wall segments $[AB]$ and $[CD]$ with $A(x_a, y_a), B(x_b, y_b), C(x_c, y_c)$, and $D(x_d, y_d)$, means that $\overrightarrow{AB}$ and $\overrightarrow{CD}$ are co-linear. Using the cross product, it can be translated into Equation 6.

$$C_5 : (x_b - x_a) \cdot (y_d - y_c) - (x_d - x_c) \cdot (y_b - y_a) = 0 \tag{6}$$

A least squares adjustment requires linear equations but the functions F and G are nonlinear because the coordinates of the vertices are unknown. So we approximate the model with a linear one and the parameters are obtained by successive iterations, as in [22,26].

Let us consider a polygon formed by $(n+1)$ points of coordinates $(x_0, y_0), \ldots, (x_n, y_n)$. Let us suppose there are i nearly right angles, j nearly flat angles, k nearly 45° angles, and l nearly 135° angles. Cases of segment parallelism are ignored for simplicity but their inclusion in the equation is precisely the same.

Using the identity matrix $I_{2(n+1)}$, and the partial derivatives of (1) and (2) with respect to x and y for the three consecutive points of candidate angles we can form A the Jacobian Matrix of the model (Figure 4), which is a matricial representation of the equation system. In matrix A, the lines are related to the constraints instantiated on a single building, and the columns are related to the coordinates of each point of the building, which are unknown in the equation system. The first lines represent the constraint that points should not move (C_1). Then, the derivative equations for (C_2), (C_3), etc. are added in the matrix.

We note S the parametric model, X the vector of parameters to estimate, and Y the observations vector, where α_i^{90} is the ith almost-right angle, α_i^{180} is the ith almost-flat angle, etc.:

$$A = \underbrace{\begin{pmatrix}
1 & 0 & \dots & & & & & \dots & 0 \\
0 & 1 & 0 & \dots & & & & & \vdots \\
\vdots & & & & & & & & \vdots \\
\vdots & & & & & & & 1 & 0 \\
0 & \dots & & & & & \dots & 0 & 1 \\
0 & \dots & 0 & \frac{\partial F}{\partial x_{i-1}} & \frac{\partial F}{\partial y_{i-1}} & \dots & \frac{\partial F}{\partial y_{i+1}} & 0 \dots & 0 \\
\vdots & & & & & & & & \vdots \\
0 & \dots & \frac{\partial G}{\partial x_{j-1}} & \frac{\partial G}{\partial y_{j-1}} & \dots & \frac{\partial G}{\partial y_{j+1}} & 0 & \dots & 0 \\
\vdots & & & & & & & & \vdots \\
0 & \dots & & 0 & \frac{\partial F}{\partial x_{k-1}} & \frac{\partial F}{\partial y_{k-1}} & \dots & \frac{\partial F}{\partial y_{k+1}} & 0 \quad 0 \\
\vdots & & & & & & & & \vdots \\
\frac{\partial F}{\partial x_{l-1}} & \frac{\partial F}{\partial y_{l-1}} & \dots & \frac{\partial F}{\partial y_{l+1}} & 0 & \dots & & & 0 \\
\vdots & & & & & & & & \vdots \\
0 & \frac{\partial F}{\partial x_{l-1}} & \frac{\partial F}{\partial y_{l-1}} & \dots & \frac{\partial F}{\partial y_{l+1}} & 0 & \dots & & 0
\end{pmatrix}}_{2n+1}$$

$I_{2(n+1)}$; i lines; j lines; k lines; l lines

Figure 4: Jacobian matrix of the model.

$$Y = \begin{pmatrix}
x_0 \\ y_0 \\ \vdots \\ x_n \\ y_n \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ \cos(\frac{\pi}{4}) \\ \vdots \\ \cos(\frac{\pi}{4}) \\ \cos(\frac{3\pi}{4}) \\ \vdots \\ \cos(\frac{3\pi}{4})
\end{pmatrix} \begin{matrix} \} 2(n+1) \\ \\ \\ \\ \\ \} i \\ \\ \\ \} j \\ \\ \\ \} k \\ \\ \\ \} l \end{matrix} \qquad S(X^m) = \begin{pmatrix}
x_0^m \\ y_0^m \\ \vdots \\ x_n^m \\ y_n^m \\ F(\alpha_1^{90}) \\ \vdots \\ F(\alpha_i^{90}) \\ G(\alpha_1^{180})) \\ \vdots \\ G(\alpha_j^{180})) \\ F(\alpha_1^{45}) \\ \vdots \\ F(\alpha_k^{45}) \\ F(\alpha_1^{135}) \\ \vdots \\ F(\alpha_1^{135})
\end{pmatrix} \begin{matrix} \} 2(n+1) \\ \\ \\ \\ \\ \} i \\ \\ \\ \} j \\ \\ \\ \} k \\ \\ \\ \} l \end{matrix} \tag{7}$$

Then, with $B = Y - S(X)$ we can write the normal equations of the system as

$$(A^T PA)^{-1} dX = A^T PB \tag{8}$$

dX being the vector of increments, and P a weight matrix containing weights relative to the importance of each constraints (e.g., squareness is important with a weight of 10, while preventing vertex movement is less important with a weight of 1). Weight setting is further discussed in the next section.

X is then calculated iteratively, with X^0 being the vector of initial positions of the building vertices:

$$X^m = dX^m + X^{m-1} \tag{9}$$

with

$$dX^m = (A^T PA)^{-1} A^T PB \tag{10}$$

The approximate least squares solution for the linear system, obtained from equations 9 and 10, minimizes the sum of weighted squared residuals (the difference from the exact solution for each equation). In order to ensure the convergence of the iterating algorithm, we set a maximum number of iterations, and a minimum value for the residuals that means that the process stops when the last iteration causes negligible distortions.

4.2 Weight setting and stopping condition

The weight matrix P gives a relative weight to each of the equations encapsulated in the lines of matrix A. The five constraints that are translated into equations are not of equal importance, i.e., some constraints may be less satisfied than others, and the weight matrix allows for this. Squaring a building with angles not totally square requires the displacement of some vertices so we accept C_1 violation provided the other constraints are fully satisfied. We want C_1 matrix lines to have a smaller weight than the other constraints. Harrie [14] proposes four methods to set weights in a similar least squares based generalization system. We followed an empirical approach—exploring different weights. From this we determined the relative importance among the constraints (Table 1). Some experiments on the sensitivity of weights are presented in the following section.

C_1	C_2	C_3	C_4	C_5
vertex displacement	90° angles squaring	180° angles squaring	45° angles squaring	wall alignment preservation
5	100	15	10	10

Table 1: Weights used for the five constraints.

The proposed nonlinear least squares method is iterative so the condition that stops the iterations is crucial. After several tries based on the number of iterations or on the residual values, we used a twofold condition:

- a maximum number of iterations is reached, or
- the norm of dX vector is sufficiently small (the vertices displacement is negligible).

In at least 99% of cases, the iteration stops because of the dX condition, generally within 4 to 6 iterations but the maximum number of iterations is used to stop the rare cases where the algorithm diverges. We set this maximum number of iterations to 50 but we did not try to optimize it given the rarity of diverging cases. In order to set the dX vector condition, we tried several norms, including the L^1-norm, Frobenius norm (equal to L^2-norm for vectors), and the L^∞ norm. The L^∞ norm that takes the largest absolute values in the vector, appeared theoretically to be the best solution, and it proved right in our experiments. We empirically set the threshold to 0.001, which is one millimeter, and it gave a good balance between the number of iterations and the quality of the output. Some experiments with varying thresholds are presented in the next section.

We did not use the residuals for the stopping condition, but used it to detect cases where the algorithm under-performs. Cases where the maximum number of iterations is reached or where the L^2-norm of the residual vector is very large (threshold empirically set) can be marked as errors of the algorithm, i.e., the algorithm did not find an optimal solution.

The most complex operations of the algorithm are matrix products and matrix inversion, which gives a complexity of $\mathcal{O}(n^3)$ where n is the size of a row of the square matrix A, i.e., n is a little more than two times the number of vertices in the polygon (it depends on the number of almost-square angles, see Figure 4). The complexity of $\mathcal{O}(n^3)$ is valid for standard matrices, which is the case in our implementation, but if we use a sparse matrix model for A, as A really is a sparse matrix, the complexity decreases to $\mathcal{O}(n^2)$.

5 Experiments

In order to test both proposed algorithms, experiments were carried out on our open source Java platform GeOxygene [12] on standard desktop computers. The next subsection describes the experimental datasets. Results are presented, evaluated and compared with algorithms from the literature.

5.1 Experimental datasets

Three datasets were selected for the experiments. The first two are extracts from OpenStreetMap. One covers the small French town Orthez with a variety of building types (residential, apartment blocks, industrial, agricultural). In this area, most OSM buildings are semi-automatically imported from the image segmentation of scanned cadastral maps, which causes some imperfections in the building outline (Figure 5). The width of the building outline, the hatching, and the numbers that might cross the outline cause inaccurate generated outlines, with particularly unsquare angles. The dataset contains 7650 buildings.

The second OSM dataset is an extract of the city of Würzburg in Germany. Germany is interesting because most of buildings have been manually captured from satellite imagery, which also causes some imperfections that can be corrected by squaring operations. This dataset contains 10485 buildings.

The last experimental dataset is an extract of French reference dataset BD TOPO®, produced by the French national mapping agency, IGN. The extract is located on the Reunion Island and contains around 600,000 buildings with a 1 m precision. Buildings have been captured from high resolution aerial photographs and field survey. All these buildings are stored in a PostGIS database.

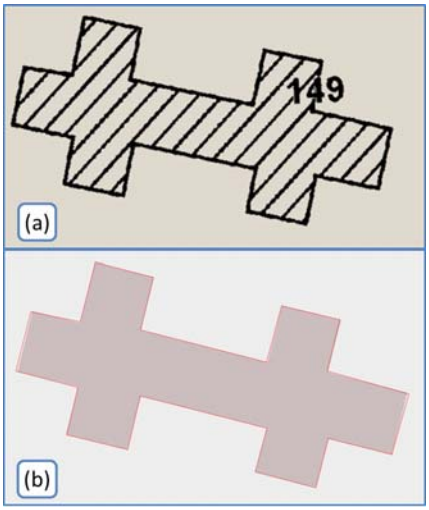

Figure 5: (a) French scanned cadaster imported by image segmentation in OSM. (b) the result building in OSM (red) squared by the proposed least squares algorithm.

5.2 Results

All three datasets are squared using both methods and the results are presented in Figures 6 to 12. In each experiment, we used a threshold of 15° for squaring right and flat angles, and 8° for 45° angles, which means, for instance, that angles from 76° to 104° are squared to 90°. The 15° threshold derives from cartographic knowledge and is often used in cartographic generalization [33].

In the Orthez area in France, where buildings are mostly captured by an automatic import from an image segmentation process, many buildings need to be squared. On simple shapes, both methods effectively square the buildings, but the least squares method tends to distort the footprints less (Figure 6). On complex building shapes, the least squares performs better according to our visual inspection (Figure 7).

In the Würzburg area, in Germany, where buildings have been mostly captured manually using aerial imagery, most of the buildings are already square. In this area, most of the remaining unsquared buildings are correctly squared by the algorithms (Figure 8) with a better visual result for least squares applied to complex shapes (Figure 9).

For the French NMA reference dataset, nearly 70% of the buildings are squared, with significant distortions (Figure 10), which was expected because the level of detail is supposed to be lower compared to the cadastral data used in OSM. The effectiveness of the least squares method on complex buildings is confirmed with this third dataset (Figure 11).

As predicted, the simple squaring algorithm fails for some complex buildings with a large number of vertices, but the least squares squaring does square these buildings properly (Figure 12).

Experiments with the C_5 constraint were also carried out with the least squares algorithm. The C_5 constraint aligns the almost-parallel walls of close buildings while squaring the building, and the results show a significant improvement compared to an individual squaring of the building (Figure 13). Two pre-processing steps are necessary to use this con-

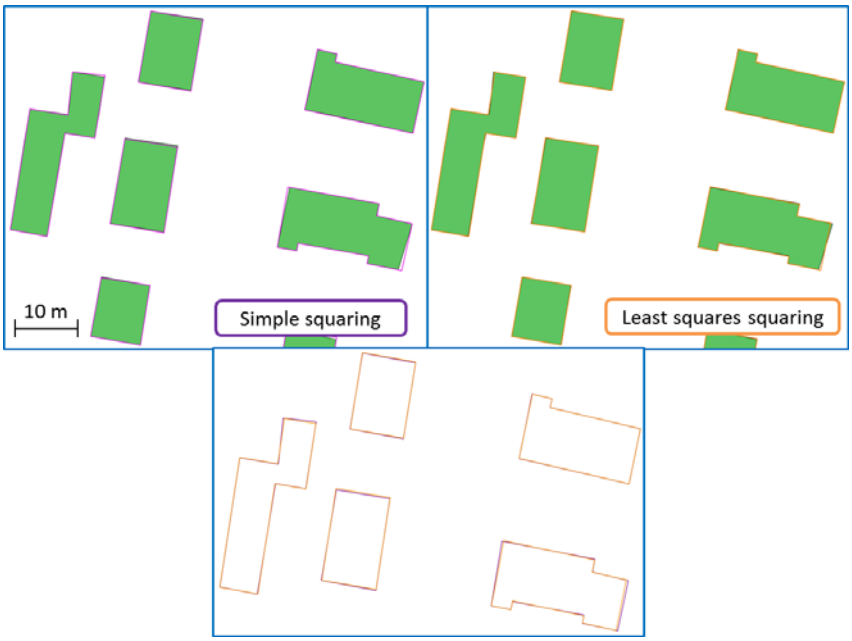

Figure 6: Extract of squared buildings with simple shapes from Orthez area, initial shapes in green, simple in purple, and least squares in orange.

straint in the least squares algorithm: first identify the buildings that should be processed together; then, identify the walls (i.e., the polygon segments) that should be aligned. In this experiment, the first step was done by manually picking aligned buildings, but many methods exist to automatically identify such alignments [6, 27, 38]. The second step is carried out automatically by considering the consecutive building pairs in the alignment: we search for the closest pair of almost-parallel segments, proximity being computed between the centers of the segments.

We also carried out experiments to measure processing time for the least squares based algorithm, on a standard desktop computer with a i7 core processor. The median processing time for a building that requires squaring, i.e., a building with almost-square angles, is 5 ms, and the mean for all datasets is 400 buildings per second. Considering that this a Java implementation where sparse matrix operations are not optimized, we consider this as a good result, meaning it could be used for online rendering with some optimization of the implementation.

5.3 Parameterization experiments

We only focus on least squares in this section, and there are several types of parameters that can alter the results: weights, stopping thresholds, and thresholds defining what is "unsquare."

With regards to weights, past experiments with least squares show that their setting might be a problem because of sensitivity to the weights. In our case, the configuration of the constraints and their relative importance lead to a very low sensitivity to the weights.

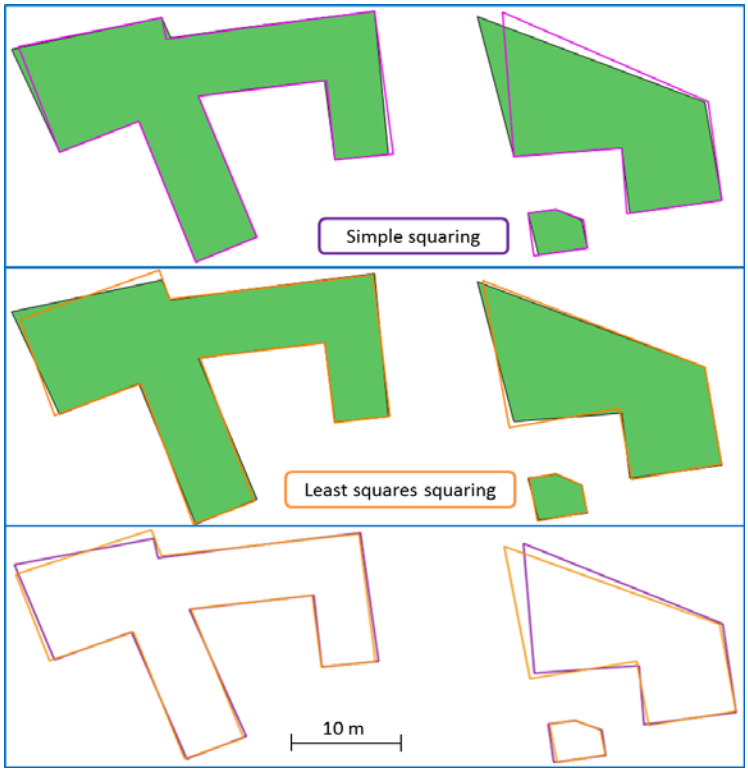

Figure 7: Extract of squared buildings with more complex shapes from Orthez area, initial shapes in green.

We tried different distributions of weights, but the results were very similar in 99% of the cases (only 6 buildings are squared differently in the French OSM dataset when weights vary) (Figure 14): test one (T1) was carried out with small variations over the optimum weights, test two (T2) has equal weights for all constraints, and test three (T3) has a greater weight on the constraint that prevents vertex displacement rather than on constraints forcing squareness.

There are two stopping criteria: the iteration number and the norm of the previous displacement. Tests were first carried out on the optimal number of iterations. On one of the datasets, iteration limits were raised from 5 to 500. The results were always the same as all buildings were squared in 5 or less iterations. This shows that our chosen value of 50 iterations was too conservative, so further testing on different types of buildings and different areas should be carried out to estimate an optimal value. We tested different values for the norm of displacements, ranging from 0.0001 m to 0.5 m, for a default value of 1 mm. Results show that from 0.0001 m to 0.1 m, the squaring is the same, and some squaring results start to deteriorate when the threshold is bigger than 0.1 m (Figure 15), because the algorithm is sometimes stopped before the optimal solution is reached. These tests indicate that keeping a 1 mm parameter value is a good solution to obtaining optimal results in the shortest time.

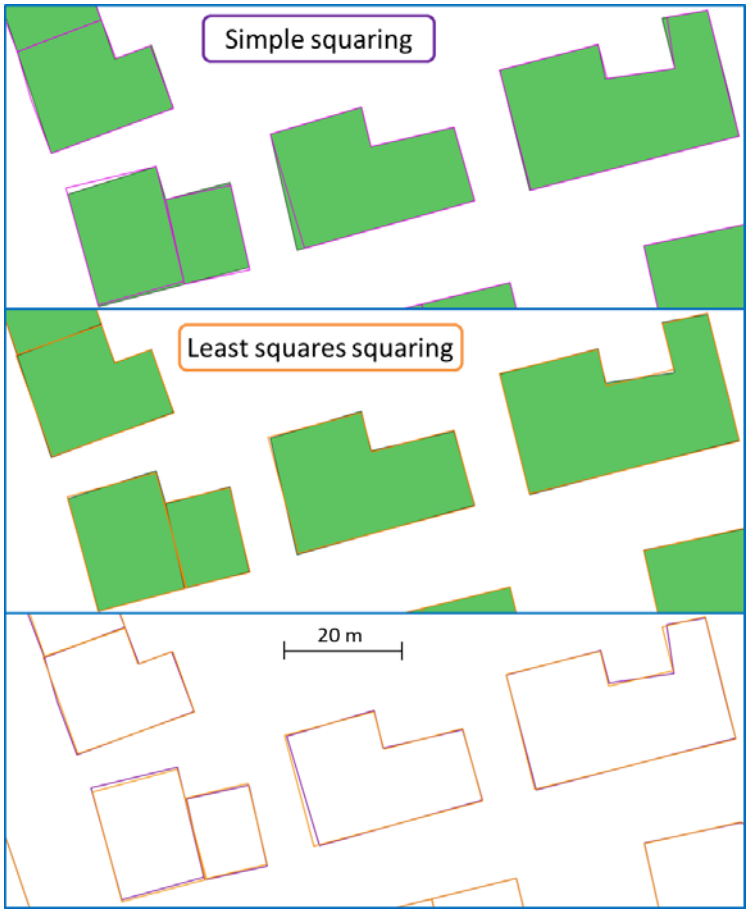

Figure 8: Extract of squared buildings with simple shapes from Würzburg area, initial shapes in green.

The main parameter of the algorithm is the angle tolerance threshold around 90° (respectively 180°, 45°, and 135°) to identify angles that need to be squared. The default empirical values are 15° for right and flat angles, and 8° for 45° angles, and derive from knowledge in cartographic generalization [33]. When these threshold vary, the squaring results clearly vary with more or less angles of the polygon being squared (Figure 16). So this parameter needs to be carefully set in relation to the aim of the squaring operation.

5.4 Evaluation

There are two ways to evaluate the proposed algorithms: first, assess how well polygons are squared and how much the initial shape is preserved (5.4.1); then compare the proposed algorithms to the existing algorithms, by applying them on the same buildings with the same parameters, in order to assess which algorithm performs better (5.4.2).

Figure 9: Extract of squared buildings with more complex shapes from Würzburg area, initial shapes in green.

5.4.1 Assessment of squared shapes

In order to assess how well polygons are squared by both proposed algorithms, the number of almost-right, almost-flat and almost-45° angles is counted before and after squaring, for every building footprints in the three experimental datasets. A good squaring method would set these three numbers to zero after squaring. Added to the count of almost-right angles, we sum the differences to the perfect angle: a building that has two almost-right angles with angle values of 87° and 86° will have a sum of almost-right angles equal to 7.

Only footprints that required squaring are kept in the assessment, the other footprints are exactly the same before and after. The ratio of buildings not exactly square varies from one dataset to another: 90% for the French OSM dataset, 80% for the French reference dataset, and 20% for the German OSM dataset. A summary of this assessment is presented in Table 2. Both methods clearly reduce the amount of angles almost-right (ARA) and almost-flat (AFA): the initial mean number of ARA (5.92 per building) and AFA (0.79 per building) falls to 0.51 and 0.16 respectively for least squares, and 1.59 and 0.22 respectively for the simple method. The least squares method outperforms the simple method as the median number shows that for most of buildings, there are no buildings in need of squaring. Even with the least squares method, some angles remain almost-square or almost-flat after squaring in some cases, but the angle differences sum show that the remaining ARA and AFA are close to $\pi/2$ and 0. The sum of ARA is only 0.51° in total which is very low, given that we previously used a tolerance of 0.5° to consider angles as perfectly square. Regarding the least squares method, the remaining ARA and AFA are due to too few iterations (see Section 4.2, or particular geometrical configurations). A closer inspection of the buildings with remaining ARA and AFA shows that they are mostly large and com-

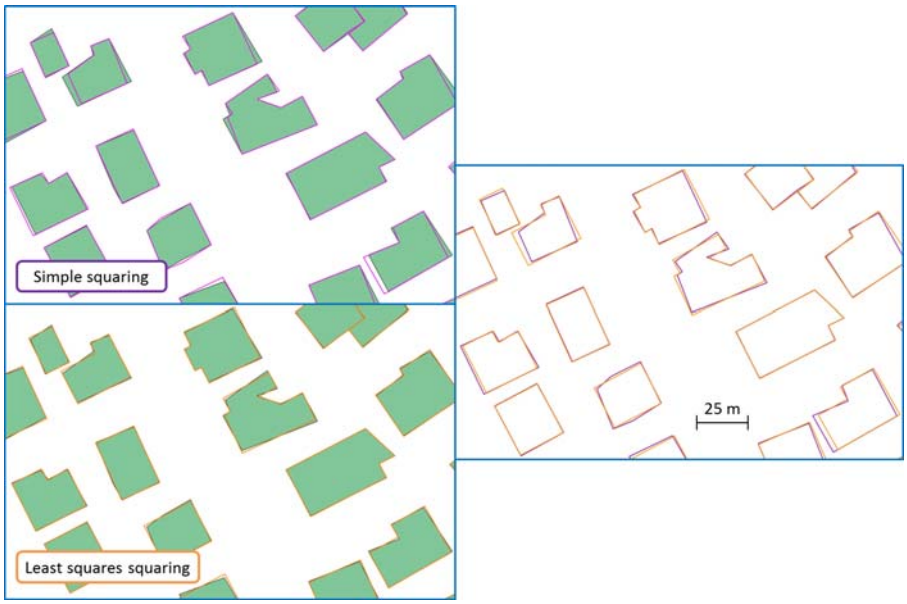

Figure 10: Extract of squared buildings with simple shapes from IGN reference dataset, initial shapes in green.

plex buildings, with unusual (e.g., round) shapes, such as the ones in Figure 11. On the other hand, the particular geometrical configurations can contain very long or very short segments in a building, which might be badly handled in the adjustment [34]: very small segments might lead to unacceptably large angle distortions, while very long segments might prevent angle distortions because they cause too long vertex displacement. Further testing is required to better handle very short and very long segments.

Regarding the preservation of the initial shape, we first measure how much the outline of the footprint has been distorted by the squaring operation. We are not interested here in measuring vertex movement, which changes the shape of the initial polygon, but instead are interested in the preservation of the general shape of the polygon and its main characteristics. Several polygon similarity measures were used to assess this shape preservation. Changing the representation space of the outline can help describe the exterior line work of the polygons, as proposed by the turning function [2] and the polygon signature function [36].

The turning function gives the orientation of the line with respect to the x and y axes, at a given curvilinear abscissa, i.e., it gives the orientation of each segment of the polygon. The turning function distance computes the area between the turning functions of two lines (Figure 17). To illustrate, if the orientation of a segment changes from 87° to 90°, which happens frequently with the squaring methods, the second turning function will be a little above the first one at the curvilinear abscissa of the segment. The distance value is normalized by the minimum value of both perimeters, in order to avoid penalizing the largest buildings. This measure seems particularly appropriate as squaring mainly changes angles and segment orientations.

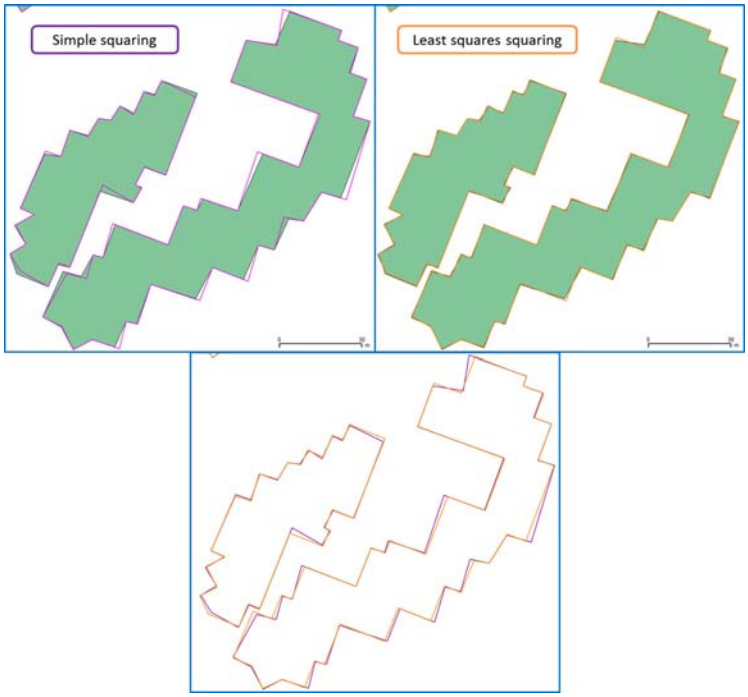

Figure 11: Extract of squared buildings with more complex shapes from IGN reference dataset, initial shapes in green.

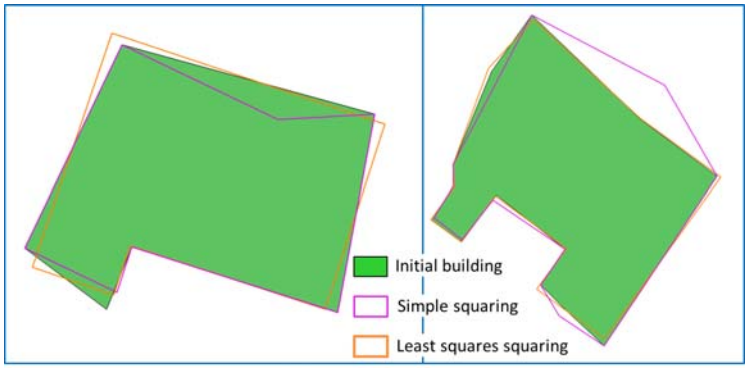

Figure 12: two buildings from Würzburg area where simple squaring fails.

The polygon signature is a function from [0,1] to $\mathbb{R}$ which represents the radial distance from the centroid of the polygon to each of its vertices [36]. The distance is computed the same way as the turning function distance (Figure 17). It has been successfully used for assessing polygon similarity in feature matching problems [36]. To illustrate, this distance is zero if there is only a translation between the compared polygons.

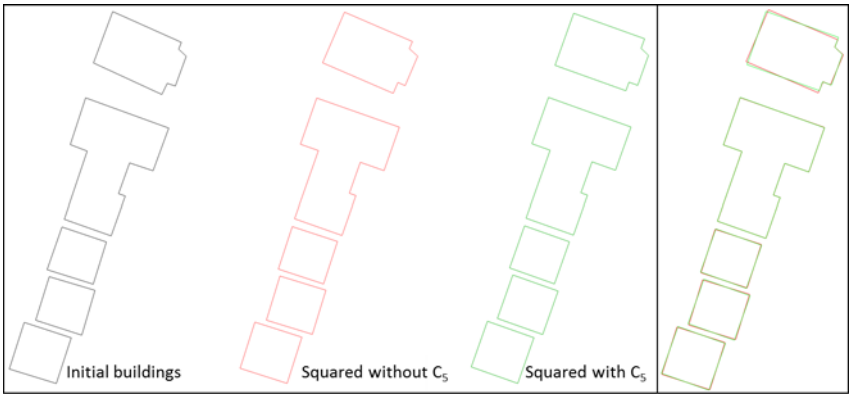

Figure 13: Aligned buildings squared together using the C_5 constraint.

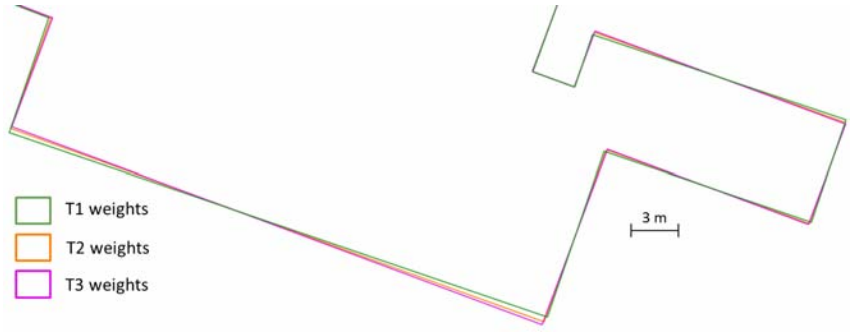

Figure 14: One of the buildings where weight-setting alters the squaring results.

In order to measure how polygons are the same at the same location, the surfacic distance [36] is complementary to the previous measurements: it is simply computed by comparing areas of the intersection and union of the two polygons (Equation 11).

$$\text{surfacicDistance}(A, B) = 1 - \frac{Area(A \cap B)}{Area(A \cup B)} \tag{11}$$

The results of the assessment are summarized in Table 3. All three measures indicate that the least squares squaring method is effective in preserving the initial shape of the building; the simple squaring method often does not. Regarding the surfacic distance, a mean of 0.013 is very low, as most of the matched lake footprints from OpenStreetMap and IGN reference data had a surfacic distance of between 0.05 and 0.4, according to [9].

5.4.2 Comparison with existing methods

It is useful to compare these two methods with other implementations. Comparisons were carried out with two other methods: first, the GAEL-based squaring method from [8], and then the squaring algorithm implemented in 1Spatial software Radius Clarity ™ during the AGENT European project [3], inspired from [1]. The comparison is carried out visually

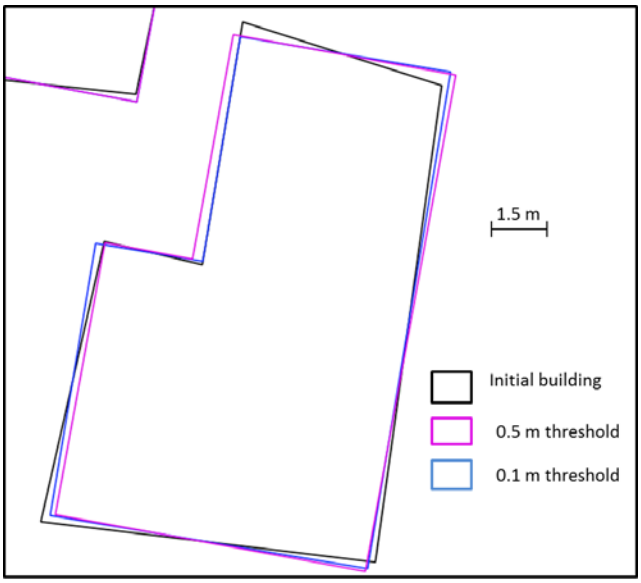

Figure 15: The stopping value starts altering the results when it is bigger than 0.1 m.

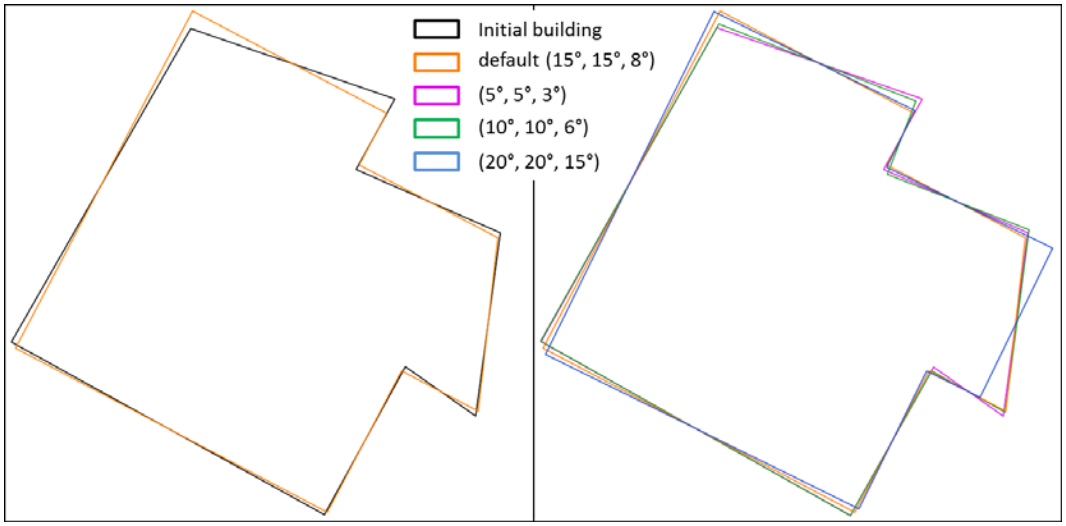

Figure 16: Different squaring results when the angle tolerance thresholds change.

but also with the same measures as the shape preservation analysis: the number of almost-right and -flat angles, the turning function distance, the polygon signature distance, and the surfacic distance.

We first compare the least squares to GAEL squaring. Table 4 shows how differently they reduce the ARA and the AFA: least squares clearly reduces the number of ARA and AFA, while similarly preserving the initial shape (turning and surfacic distance indicate

Measure	Mean	Median	Maximum
initial nb of ARA	5.92	4	146
initial nb of AFA	0.79	0	43
initial sum of ARA (°)	13.13	7.0	497.0
initial sum of AFA (°)	2.66	0.0	197.0
nb of ARA for least squares	0.51	0	63
nb of AFA for least squares	0.16	0	29
sum of ARA for least squares	0.51	0.0	87.0
sum of AFA for least squares	0.65	0.0	69.0
nb of ARA for simple squaring	1.59	1	51
nb of AFA for simple squaring	0.22	0	8
sum of ARA for simple squaring	10.55	7.0	363.0
sum of AFA for simple squaring	1.54	0.0	82.0

Table 2: Reduction of almost-right angles (ARA) and almost-flat angles (AFA) per building for the proposed squaring methods.

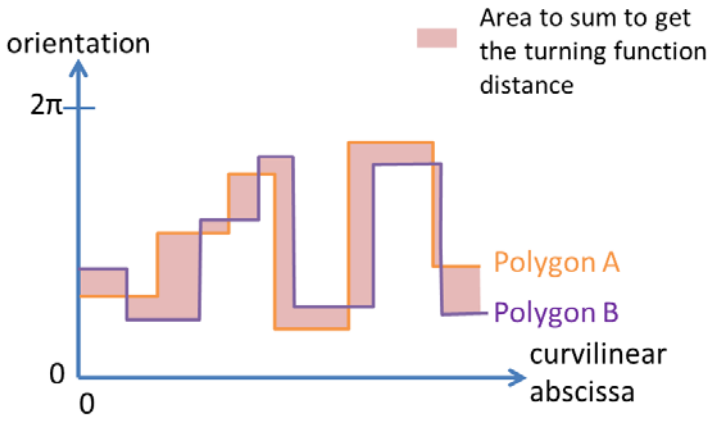

Figure 17: Illustration of the principles of the turning function distance between two polygons.

that shapes are better preserved by least squares but the signature distance indicates that they are better preserved by GAEL squaring). The three measures confirm that polygons are squared slightly differently (e.g., a mean turning distance of 0.1).

Regarding the comparison with the squaring method from the AGENT project, Table 4 shows that they reduce squareness in a similar manner, with a slight advantage for the least squares. But the shape comparisons show that the least squares better preserves the initial

Measure	Mean	Median	Standard deviation	Maximum	Minimum
Turning distance initial to simple	1.51	0.99	1.10	9.64	0.11
Turning distance initial to least squares	0.13	0.11	0.15	7.03	0.0
Signature distance initial to simple	5.88	2.31	24.42	476.03	0.06
Signature distance initial to least squares	0.10	0.02	0.85	35.2	0.0
Surfacic distance initial to simple	0.82	0.84	0.13	1.0	0.17
Surfacic distance initial to least squares	0.013	0.008	0.022	0.735	0.0

Table 3: Shape preservation measures between initial and squared buildings.

	ARA mean	AFA mean	ARA sum mean	AFA sum mean	Turning distance	Signature distance	Surfacic distance
Least squares	0.51	0.16	0.507	0.651	0.125	0.097	0.013
GAEL	1.16	0.35	1.997	0.789	0.128	0.071	0.044
AGENT	0.51	0.17	3.25	4.53	0.264	4.706	0.247

Table 4: Comparison of least squares, GAEL and AGENT squaring.

shape: with a 0.26 mean value for turning distance, a 4.7 mean value for signature distance and a 0.25 value for surfacic distance, the AGENT squared buildings are quite different from the initial buildings.

A visual analysis confirms that GAEL-based and least squares based methods are the best ones to square even complex buildings, with slightly better results for the least squares based method, even when the GAEL-based method does not cause any translation error (Figure 18).

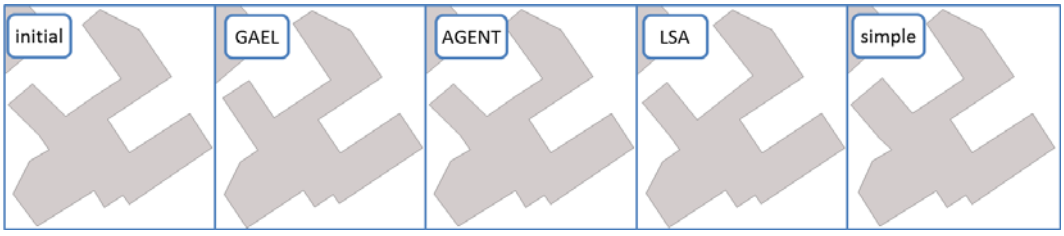

Figure 18: Comparison of all squaring method for a given complex building footprint, LSA stands for Least Squares Adjustment.

To conclude this comparison, none of the existing methods assessed outperforms our proposed least squares squaring method regarding squareness problems reduction and shape preservation.

6 Possible applications

This section reviews three types of possible applications for the squaring algorithms: enhancing OSM features to ease their use in numerous applications, improving image-based change detection, and agent-based cartographic generalization.

6.1 Enhancing OpenStreetMap building layer

The first obvious application of enhancing OpenStreetMap building layers is to improve the quality of the multiple maps derived from OSM. But OSM is not only used for visualization purposes, and many applications are now based on OSM, taking advantage of the freely available data covering the globe. For instance, some projects try to create 3D city models from OSM building footprint by extrusion [10,25]. Such projects would benefit from square buildings thus producing more realistic 3D buildings.

Other OSM-based applications include indoor mapping projects [11], or cadastral mapping [18].

The proposed squaring methods are dedicated to buildings but the diversity of OSM objects also includes other square geographical features, such as sports fields, tennis courts, and pools. Sports fields were extracted in a large region of France to test the squaring methods on such features. Results show that most sports fields in this area were already squared and the few (nearly 5%) features that are transformed are only slightly distorted. The simple squaring proves to be sufficient in this case, and should be preferred because of its speed (50 to 100 times quicker than the least squares and GAEL methods in the experiments).

6.2 Enhancing remotely sensed building footprints

Change detection from satellite imagery may require the generation of remotely sensed footprints [5]. In order to compare the generated footprints to an existing vector dataset that needs to be updated, the footprints first need to enhanced to be closer to the existing footprints in terms of shape, particularly regarding squareness. For instance, the change detection process from [35] generates sketchy building footprints by calculating digital elevation difference (Figure 19a).

In this case, the building is already squared because the polygon is generated from a set of pixels, so it only contains square angles. So, the first step is to simplify the building shape with the building simplification algorithm from [3, 30] (Figure 19b). After simplification, the sketchy outline is removed but the building is still unsquare and we applied the least squares squaring algorithm to obtain a fully enhanced footprint (Figure 19c) that can be compared to the buildings of geographical dataset to complete the change detection process.

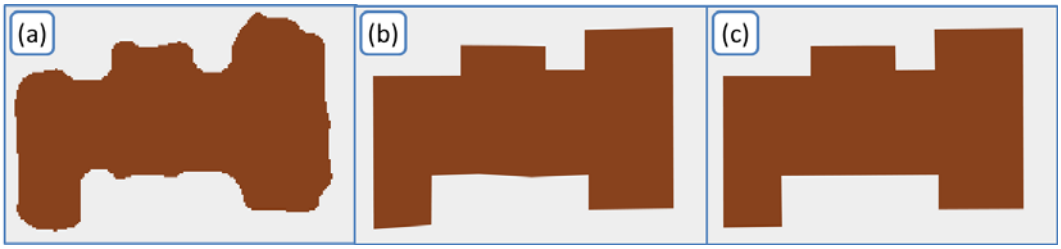

Figure 19: (a) a building detected by the change detection process of [35] (b) the building after simplification (c) the squared building.

6.3 Agent-based cartographic generalization

Squaring is a useful operation for the cartographic generalization of buildings, because almost-right angles blur the legibility of the buildings in the map [30]. Multi-agents based generalization processes such as AGENT [3, 30] already use squaring operations to caricature the almost-square buildings. For instance, the AGENT implementation in 1Spatial software Radius Clarity ™ makes use of an algorithm derived from [1], and the AGENT implementation in the open source software CartAGen [28] makes use of the squaring algorithm from [8]. But such agent-based systems can benefit from several complementary algorithms for the same operation thanks to a trial-error mechanism (Figure 20).

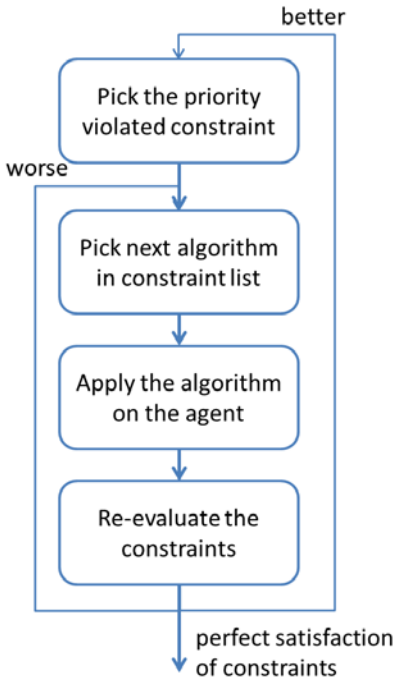

Figure 20: The simplified workflow of a building agent.

In an agent-based generalization process, squaring algorithms are governed by a squareness constraint that analyzes the building shape to identify angles that should be squared. We used both presented algorithms to improve the AGENT implementation in CartAGen [28], where the squareness constraint previously proposed only the GAEL squaring algorithm. We used the results of the evaluation to derive some heuristics of which algorithms should be applied when:

- when buildings are simple (e.g., less than ten vertices), try the **naive polygon squaring** first;
- when buildings are complex and contain almost-flat and/or almost-45° angles, try the **least squares squaring** first; else
- try the **GAEL squaring** first.

Of course, when one algorithm under-performs or crashes, another one can be used, which ensures all buildings are squared by the end of the process. This improvement was tested on the French OSM dataset that was generalized to the 1:25000 scale with the initial heuristic, and with the improved heuristic. The final results are very similar (Figure 21) and often identical. It is due to the simplification and enlargement operations used in the building generalization prior to the squaring operation. But the computation time is reduced by 20% with the new heuristic. However, it appears that the GAEL squaring algorithm diverges for 1–2% of the buildings, and in this case, having backup algorithms is very useful: these buildings are squared with one of the two proposed algorithms instead of remaining unsquared, as they did before.

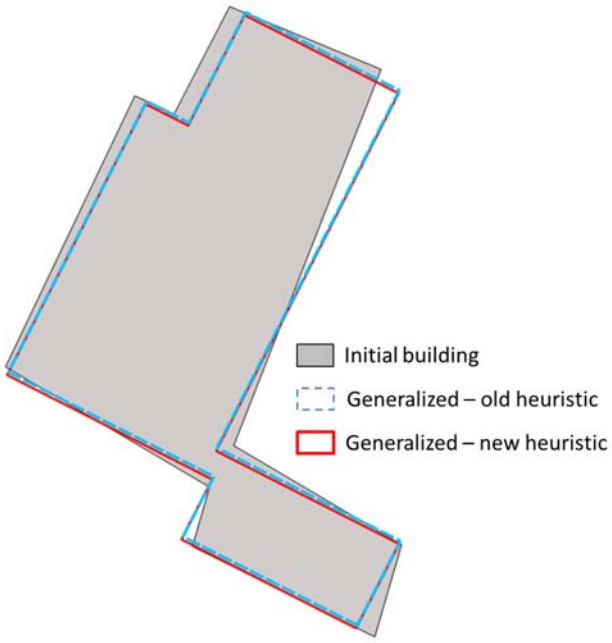

Figure 21: a building generalized to the 1:25000 scale with old and new heuristics.

7 Conclusion and further work

To conclude, this paper proposed two new methods to square building footprints: a quick and naive method that proved effective for quite simple polygons, and a more robust and flexible method based on nonlinear least squares. The methods have been successfully tested on large building datasets coming from reference data and OSM. The methods have also been compared to two existing methods proving their usefulness. The contribution of this research is also highlighted in three applications of the squaring methods: the enhancement of OSM features, the enhancement of automatic footprint extractions for change detection, and agent-based cartographic generalization. Both proposed methods are available in the open source platform GeOxygene [12].

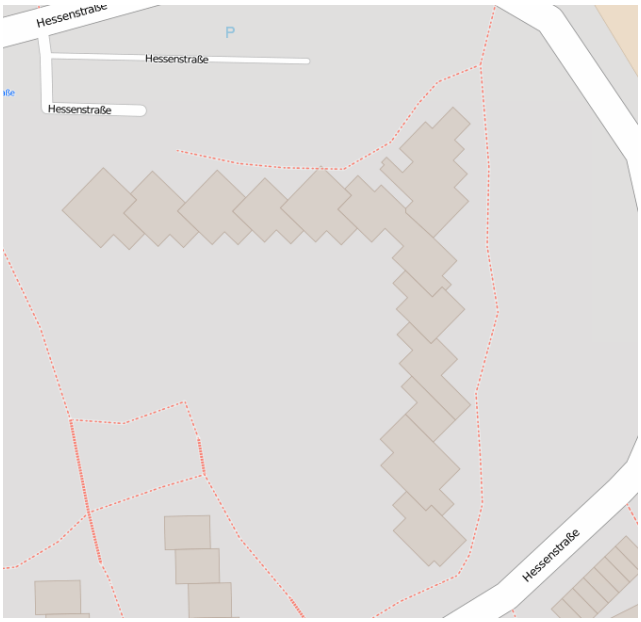

Figure 22: A complex building composed of connected buildings.

To go further, there are several ways to improve the least squares squaring methods. First, many OSM buildings are captured in a "cadaster" way, i.e., with several adjacent features for buildings with different owners or functions, instead of a single footprint (Figure 22). The method currently processes buildings one by one, potentially damaging the adjacency relation. This issue is quite easy to overcome by identifying the vertices shared by several buildings, and processing them only once in a single adjustment. That is how connected roads are processed for instance in [13]. The automatic identification of cases where the constraint C_5 should be used, also has to be investigated, as in the proposed experiments, we identified these cases manually. For OSM data, the enhancement has not been used to modify the world dataset, but we believe that creating an automated tool to correct OSM building would improve the quality of OSM. The proposed applications of the algorithm are offline, but would it be possible to use it online, in an application that renders the vectors from OSM, rather than the usual tiled raster maps? We tried the

least squares algorithm on twenty random buildings of the reference dataset, and squaring takes less than half a second, on a standard desktop computer, with a non-optimized implementation. This shows that there is a potential for online rendering; further testing is clearly required, particularly regarding the management of complex buildings. But the squaring enhancement has to be used carefully: this is mostly a rendering enhancement, and not necessarily a quality enhancement. On remotely sensed buildings, or on French OSM buildings, we believe that the squared shapes are closer to ground truth than the initial shapes, but that is not guaranteed by our algorithm; our ambition is more guided by visualization caricature, than by preserving ground truth. So the algorithm should not be used to massively "correct" the building footprints in OpenStreetMap. Finally, it would be interesting to measure the impact of squaring on the quality of all these applications making use of OSM building, with a fitness for use perspective. It could help adapting the required level of squareness for a given application.

Acknowledgments

The authors would like to acknowledge Anthony Wiart for his proposition of the change detection application, and Cécile Duchêne for her valuable comments on the use of orientation measures to guide squaring. The authors also acknowledge the three anonymous reviewers for their valuable and exhaustive comments.

References

[1] AIRAULT, S. De la base de données à la carte: une approche globale pour l'équarrissage de bâtiments. *Revue Internationale de Géomatique 6*, 2–3 (1996), 203–217.

[2] ARKIN, E. M., CHEW, L. P., HUTTENLOCHER, D. P., KEDEM, K., AND MITCHELL, J. S. B. An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 3 (1991), 209–216. doi:10.1109/34.75509.

[3] BARRAULT, M., REGNAULD, N., DUCHÊNE, C., HAIRE, K., BAEIJS, C., DEMAZEAU, Y., HARDY, P., MACKANESS, W. A., RUAS, A., AND WEIBEL, R. Integrating multi-agent, object-oriented, and algorithmic techniques for improved automated map generalisation. In *Proc. 20th International Cartographic Conference* (2001), vol. 3, ICA, pp. 2110–2116.

[4] CABELLO, S., DE BERG, M., AND VAN KREVELD, M. Schematization of networks. *Computational Geometry 30*, 3 (2005), 223–238. doi:10.1016/j.comgeo.2004.11.002.

[5] CHAMPION, N., BOLDO, D., PIERROT-DESEILLIGNY, M., AND STAMON, G. 2D change detection from satellite imagery: Performance analysis and impact of the spatial resolution of input images. In *Proc. IEEE International Geoscience and Remote Sensing Symposium (IGARSS)* (2011), IEEE, pp. 1421–1424. doi:10.1109/igarss.2011.6049332.

[6] CHRISTOPHE, S., AND RUAS, A. Detecting building structures for generalisation purposes. In *Proc. 10th International Symposium on Spatial Data Handling (SDH)* (Ottawa (Canada), 2002).

[7] DUCHÊNE, C., BARD, S., BARILLOT, X., RUAS, A., TREVISAN, J., AND HOLZAPFEL, F. Quantitative and qualitative description of building orientation. In *Proc. 5th Workshop on Progress in Automated Map Generalisation* (Paris, France, 2003).

[8] GAFFURI, J. Three reuse example of a generic deformation model in map generalisation. In *Proc. 24th International Cartographic Conference* (2009).

[9] GIRRES, J.-F., AND TOUYA, G. Quality assessment of the French OpenStreetMap dataset. *Transactions in GIS 14*, 4 (2010), 435–459. doi:10.1111/j.1467-9671.2010.01203.x.

[10] GOETZ, M. Towards generating highly detailed 3D CityGML models from OpenStreetMap. *International Journal of Geographical Information Science 27*, 5 (2013), 845–865. doi:10.1080/13658816.2012.721552.

[11] GOETZ, M., AND ZIPF, A. Extending OpenStreetMap to indoor environments: Bringing volunteered geographic information to the next level. In *Urban and Regional Data Management: Udms Annual 2011*, M. Rumor, S. Zlatanova, and H. Ledoux, Eds. Delft, The Netherlands, 2011, pp. 47–58.

[12] GROSSO, E., PERRET, J., AND BRASEBIN, M. GEOXYGENE: An interoperable platform for geographical application development. In *Innovative Software Development in GIS*, B. Bucher and F. Le Ber, Eds. John Wiley & Sons, 2012, ch. 3, pp. 67–90.

[13] HARRIE, L. E. The constraint method for solving spatial conflicts in cartographic generalization. *Cartography and Geographic Information Science 26*, 1 (1999), 55–69. doi:10.1559/152304099782424884.

[14] HARRIE, L. E. Weight-setting and quality assessment in simultaneous graphic generalization. *Cartographic Journal 40*, 3 (2003), 221–233. doi:10.1179/000870403225012925.

[15] HAUNERT, J.-H., AND WOLFF, A. Optimal simplification of building ground plans. In *Proc. ISPRS Commission II, WG II/3* (2008), ISPRS.

[16] HOJHOLT, P. Solving space conflicts in map generalization: Using a finite element method. *Cartography and Geographic Information Science* (2000), 65–74. doi:10.1559/152304000783548028.

[17] HUNTER, G. J., AND BEARD, K. Understanding error in spatial databases. *Australian Surveyor 37*, 2 (1992), 108–119. doi:10.1080/00050326.1992.10438784.

[18] KALANTARI, M., AND LA, V. Assessing OpenStreetMap as an open property map. In *OpenStreetMap in GIScience*, J. Jokar Arsanjani, A. Zipf, P. Mooney, and M. Helbich, Eds., Lecture Notes in Geoinformation and Cartography. Springer, 2015, pp. 255–272. doi:10.1007/978-3-319-14280-7_13.

[19] MACKANESS, W., AND REIMER, A. Generalisation in the context of schematised maps. In *Abstracting Geographic Information in a Data Rich World*, D. Burghardt, C. Duchêne, and W. Mackaness, Eds., Lecture Notes in Geoinformation and Cartography. Springer, 2014, ch. 10, pp. 299–328. doi:10.1007/978-3-319-00203-3_10.

[20] MACKANESS, W. A., BURGHARDT, D., AND DUCHÊNE, C. Map generalisation: Fundamental to the modelling and understanding of geographic space. In *Abstracting Geographic Information in a Data Rich World*, D. Burghardt, C. Duchêne, and W. Mackaness, Eds., Lecture Notes in Geoinformation and Cartography. Springer, 2014, pp. 1–15. doi:10.1007/978-3-319-00203-3_1.

[21] MCMASTER, R., AND SHEA, K. S. *Generalization in Digital Cartography*. Association of American Geographers Press, 1992.

[22] MENEROUX, Y., AND BRASEBIN, M. Towards a generic method for buildings-parcels vector data adjustement by least squares. In *Proc. 9th International Symposium on Spatial Data Quality (ISSDQ)* (sep 2015).

[23] MEULEMANS, W., RENSSEN, A., AND SPECKMANN, B. Area-preserving subdivision schematization. In *Geographic Information Science*, S. I. Fabrikant, T. Reichenbacher, M. Kreveld, and C. Schlieder, Eds., vol. 6292 of *Lecture Notes in Computer Science*. Springer, 2010, pp. 160–174. doi:10.1007/978-3-642-15300-6_12.

[24] NÖLLENBURG, M., AND WOLFF, A. Drawing and labeling high-quality metro maps by mixed-integer programming. *IEEE Transactions on Visualization and Computer Graphics 17*, 5 (2011), 626–641. doi:10.1109/tvcg.2010.81.

[25] OVER, M., SCHILLING, A., NEUBAUER, S., AND ZIPF, A. Generating web-based 3D city models from OpenStreetMap: The current situation in Germany. *Computers, Environment and Urban Systems 34*, 6 (2010), 496–507. doi:10.1016/j.compenvurbsys.2010.05.001.

[26] PENG, D., HAUNERT, J.-H., WOLFF, A., AND HURTER, C. Morphing polylines based on least-squares adjustment. In *Proc. 16th ICA Workshop on Generalisation and Multiple Representation* (Dresden, Germany, 2013).

[27] REGNAULD, N. Contextual building typification in automated map generalization. *Algorithmica 30*, 2 (2001), 312–333. doi:10.1007/s00453-001-0008-8.

[28] RENARD, J., GAFFURI, J., DUCHÊNE, C., AND TOUYA, G. Automated generalisation results using the agent-based platform CartAGen. In *Proc. 25th International Cartographic Conference (ICC'11)* (Paris, France, 2011), ICA.

[29] RUAS, A. Généralisation d'immeubles. Master's thesis, ENSG, 1988.

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

[31] SAVINO, S., AND RUMOR, M. Detecting errors in formally correct geodatabases. In *Extended Abstract Proceedings of GIScience 2014* (Vienna, Austria, 2014), K. Stewart, E. Pebesma, G. Navratil, P. Fogliaroni, and M. Duckham, Eds., vol. 40 of *GeoInfo*, Vienna University of Technology, pp. 218–223.

[32] SESTER, M. Optimization approaches for generalization and data abstraction. *International Journal of Geographical Information Science 19*, 8 (2005), 871–897. mydoi10.1080/13658810500161179.

[33] STOTER, J. E., BAELLA, B., BLOK, C., BURGHARDT, D., DUCHÊNE, C., PLA, M., REGNAULD, N., AND TOUYA, G. State-of-the-art of automated generalisation in commercial software. Tech. rep., EuroSDR series on research projects, Amsterdam, 2010.

[34] TOUYA, G., COUPÉ, A., JOLLEC, J., DORIE, O., AND FUCHS, F. Conflation optimized by least squares to maintain geographic shapes. *ISPRS International Journal of Geo-Information 2*, 3 (2013), 621–644. doi:10.3390/ijgi2030621.

[35] VALLET, B. Homological persistence for shape based change detection between digital elevation models. In *ISPRS Annals – Volume II-3/W2, WGIII/3 ISA13 – The ISPRS Workshop on Image Sequence Analysis* (Antalya, Turkey, 2013), ISPRS, pp. 49–54.

[36] VAUGLIN, F., AND BEL HADJ ALI, A. Geometric matching of polygonal surfaces in GISs. In *Proc. ASPRS Annual Meeting* (Tampa, FL, 1998).

[37] WARE, M. J., TAYLOR, G. E., ANAND, S., AND THOMAS, N. Automated production of schematic maps for mobile applications. *Transactions in GIS 10*, 1 (2006), 25–42. doi:10.1111/j.1467-9671.2006.00242.x.

[38] ZHANG, X., AI, T., STOTER, J., KRAAK, M.-J., AND MOLENAAR, M. Building pattern recognition in topographic data: Examples on collinear and curvilinear alignments. *GeoInformatica 17*, 1 (2013), 1–33. doi:10.1007/s10707-011-0146-3.