Detecting Symmetries in Building Footprints by String Matching

Jan-Henrik Haunert

Chair of Computer Science I, University of Würzburg,
Am Hubland, 97074 Würzburg, Germany
jan.haunert@uni-wuerzburg.de

Abstract

This paper presents an algorithmic approach to the problem of finding symmetries in building footprints. The problem is motivated by map generalization tasks, for example, symmetry-preserving building simplification and symmetry-aware grouping and aggregation. Moreover, symmetries in building footprints may be used for landmark selection and building classification.

The presented method builds up on existing methods for symmetry detection in polygons that use algorithms for string matching. It detects both axial symmetries and repetitions of geometric structures. In addition to the existing string-matching approaches to symmetry detection, we consider the problem of finding partial symmetries in polygons while allowing for small geometric errors. Moreover, we discuss how to find optimally adjusted mirror axes and to assess the quality of a detected mirror axis using a least-squares approach.

The presented approach was tested on a large building data set of the metropolitan Boston area. The dominant symmetry relations were found. Future work is needed to aggregate the obtained symmetry relations, for example, by finding sets of mirror axes that are almost collinear. Another open problem is the integration of information on symmetry relations into algorithms for map generalization.

1 Introduction

Many buildings contain symmetric structures. No matter whether a symmetric building design was chosen for aesthetics, functionality, or simply for minimizing construction costs, humans perceive symmetry as an important building characteristic. Since many geographic analysis tasks require methods for shape characterization, an automatic symmetry detector is needed. This paper presents a new algorithm for the detection of symmetries in polygons. This algorithm is tailored to deal with building footprints that we typically find in cadastral or topographic databases. It was tested for a building data set of the metropolitan Boston area.

A building footprint consists of multiple polygonal rings (that is, one exterior ring and multiple interior rings). The presented method finds (partial) symmetries in one ring or between two rings, no matter whether the rings are interior or exterior. In the following we refer to each ring as a polygon.

The work presented in this paper contributes to the general aim of enriching spatial data with information on geometric structures and patterns. Such information is valuable for multiple applications. Our main motivation is *map generalization*, which aims to decrease a map's level of detail while preserving its characteristic structures. With respect to buildings, we particularly aim at symmetry-preserving simplification and symmetry-aware aggregation. Both problems have not been approached yet.

For *building simplification* we recently presented an algorithm based on discrete optimization (Haunert and Wolff, 2010). This algorithm allows us to integrate multiple quality criteria such as the preservation of a building's area and its dominating edge directions. Symmetry preservation, however, is currently not considered as a quality criterion in our method (and not in other methods), thus we may lose symmetric structures by simplification, see Fig. 1. In order to overcome this drawback, we need to detect symmetries in the input building. Then, we can define a cost function that penalizes those simplifications that destroy symmetries.

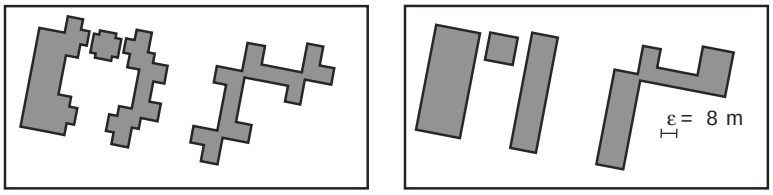

Fig. 1. Two buildings (left) and their simplifications (right) obtained with the building simplification method by Haunert and Wolff (2010) and the error tolerance $\varepsilon = 8$ m. The symmetry relations are lost.

Building aggregation means to find groups of buildings. Each group may be replaced by a single map object, for example, a building block. In map generalization, the grouping of objects is usually done according to Gestalt criteria, for example, alignment, similarity, and proximity, which model criteria of human perceptional grouping (Wertheimer, 1938). Obviously, symmetry is an important criterion for grouping. In Fig. 2 we clearly perceive that five buildings form an ensemble – this is because of their symmetric arrangement. Therefore, replacing the ensemble by a single shape can be a favorable generalization action.

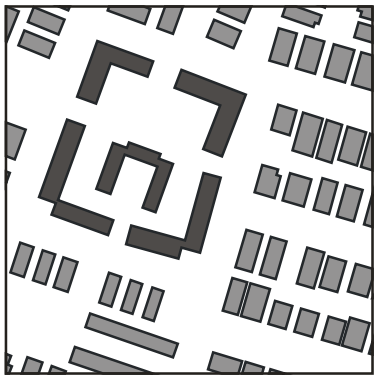

Fig. 2. An ensemble of five buildings (dark grey) that a human can identify based on symmetry.

Fig. 3. Because of symmetry relations, the dark grey building can be used as a landmark.

Map generalization is not the only application of symmetry detection. For example, buildings whose major symmetry axes are collinear with important lines of sights can serve as landmarks for navigation (see Fig. 3). Moreover, such buildings often have representative functions like town halls or castles. The dark grey building in Fig. 3, for example, is the main building of Harvard Medical School. Therefore, symmetry can be used as a cue for both *automatic landmark selection* (that is, deciding which building serves best as a landmark in a routing instruction) and *building classification*, which are topical problems in geographic information science. For a recent approach to compare different landmark selection methods we refer to Peters et al. (2010). Steiniger et al. (2008) and Werder et al. (2010) have proposed shape measures to classify building footprints and, more generally, polygons according to their functionality.

The paper is structured as follows. We first discuss related work on data enrichment in map generalization and on algorithms for symmetry detection (Section 2). Section 3 introduces a new algorithm for symmetry detec-

tion. In Section 4 we discuss experimental results with this algorithm. Section 5 concludes the paper.

2 Related Work

The gathering of knowledge on patterns and structures in geographic data, *data enrichment*, is often considered as a prerequisite for automatic map generalization (Mackaness and Edwards, 2002; Neun et al., 2008; Steiniger, 2007). Thomson and Brooks (2002) show how to find long sequences of (almost) collinear road segments in road datasets. Such sequences, so-called *strokes*, correspond to major road axes that need to be preserved during generalization. Heinzle and Anders (2007) present algorithms to find star-like structures, rings and regular grids in road networks in order to improve the generalization of networks. Christophe and Ruas (2002) as well as Ruas and Holzapfel (2003) present methods to find alignments of buildings. Gaffuri and Trévisan (2004) show how to deal with such patterns in a multi-agent system for map generalization. Methods for the grouping of buildings are proposed by Regnauld (2003) and Yan et al. (2008). These methods, however, do not consider symmetry as a criterion for grouping.

In contrast, symmetry detection has found much attention in the literature on image analysis and pattern recognition. Symmetry detection in images is often done based on local image features that are highly distinctive and invariant against certain transformations, for example, rotation and scale. Loy and Eklundh (2006) as well as Cho and Lee (2009), for example, use so-called SIFT (scale-invariant feature transform) descriptors. A comparative study on symmetry detection in images is given by Park et al. (2008). Mitra et al. (2006) present a method for finding symmetries in three-dimensional models. Similar to the symmetry detectors for images, their method relies on characteristic points. In this case, however, these points are defined based on the curvature of the model's surface. Point pairs that correspond by shape symmetry are found using RANdom SAmple Consensus (RANSAC).

In contrast to symmetry detection in images, symmetry detection in two-dimensional polygons is often done by string matching. The basic string matching approach of Wolter et al. (1985) is to encode the polygon P as a string X, for example, as a sequence of angles and edge lengths, see Fig. 4. In order to find an axial symmetry relation, we need to test whether the string X^{-1} (meaning the reversal of X) is a substring of the string XX (meaning the concatenation of X with itself). This test can be done in $\Theta(n)$

time where n is the number elements in XX by using the algorithm of Knuth et al. (1977). In the example in Fig. 4, the string X^{-1} is indeed a substring of XX. Its location within XX yields the axial symmetry relation. Similarly, we can find a rotational symmetry relation by finding X itself within XX. We need to avoid trivial solutions, however, that match X to the first or second half of XX. This can be done by removing the first and the last element from XX before matching. Based on a similar approach by string matching, the algorithm of Atallah (1985) finds *all* axes of symmetry of a polygon with n vertices in $\Theta(n \log n)$ time. In order to cope with geometric distortions, Lladós et al. (1997) use an approach based on a string edit distance.

Yang et al. (2008) present an approach to symmetry detection based on critical contour points. The critical points are the vertices of a simplified version of the original contour. However, since symmetry-preserving algorithms for line and building simplification do not exist, we need to be careful with this approach. In the preprocessing, we use a building simplification algorithm with a conservative setting in order to remove marginal but potentially disturbing details.

The general string-matching approach seems to be applicable for symmetry detection in building footprints. Not considered in the string-matching approaches discussed, however, is the problem of finding *partial* symmetry relations (only parts of the shape are symmetric). The next section presents a solution to this problem. Furthermore, we address the problem of generating optimal mirror axes by least-squares adjustment.

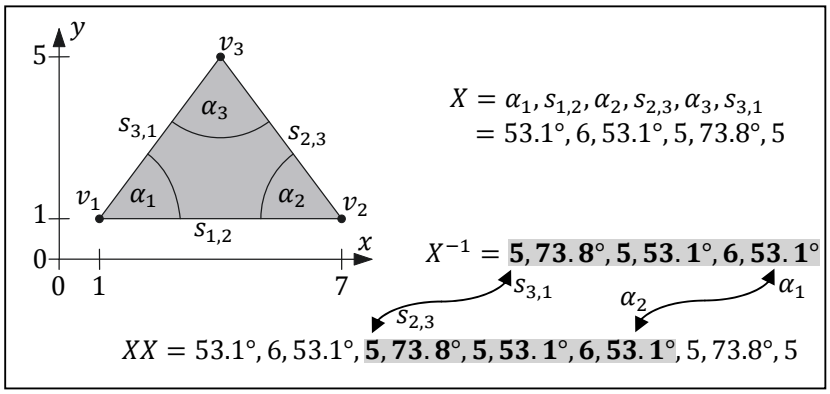

Fig. 4. Principle of the algorithm for symmetry detection by Wolter et al. (1985). By finding string X^{-1} in string XX, it becomes clear that the polygon has an axial symmetry relation. According to that relation, for example, edge $s_{3,1}$ is a mirror image of edge $s_{2,3}$.

3 Methodology for Symmetry Detection

Generally, the symmetry relations we aim to detect are geometric transformations of which each maps a continuous part p_1 of a building outline onto (or sufficiently close to) a continuous part p_2 of a second building outline. Both parts may either belong to different polygons or to the same polygon. Moreover, both p_1 and p_2 may be the same. For instance, let p_1 and p_2 be equal to the entire polygon in Fig. 4. Indeed, there is a non-trivial transformation that maps p_1 onto itself: the reflection at the vertical line through v_3. Reflections, however, are but one type of transformation we can detect with the presented method. More generally, we allow the following two types of transformations:

[1] p_2 is obtained by (successively) translating and rotating p_1

[2] p_2 is obtained by (successively) reflecting, translating, and rotating p_1.

Accordingly, we term the pair (p_1, p_2) a *type-1 match* or a *type-2 match*. In particular, we are interested in axial symmetries, that is, type-2 matches that correspond by a reflection on a straight line.

We first formalize the problems of finding type-1 and type-2 matches as a string matching problem (Section 3.1) and then discuss a solution by dynamic programming (Section 3.2). Finally, we discuss an approach based on least-squares adjustment that allows us to find axial symmetries in the detected set of type-2 matches (Section 3.3).

3.1 Symmetry Relations in the String Representation

By encoding a polygon P as a string $X(P)$ of edge lengths and angles, we obtain a shape representation that is invariant against rotations and translations. This allows us to define each type-1 match based on a pair of *similar* strings, one of them being a substring of $X(P_1)X(P_1)$ and the other one a substring of $X(P_2)X(P_2)$, where P_1 and P_2 are two potentially distinct polygons. Similarly, we define each type-2 match based on a pair of similar strings, one of them being a substring of $X(P_1)X(P_1)$ and the other one a substring of $X^{-1}(P_2)X^{-1}(P_2)$. Two strings x_1 and x_2 are called similar if the following four criteria hold:

[1] The number k of symbols is the same in both strings.

[2] Both strings start with the same type of symbol, that is, either with a symbol representing an edge length or an angle.

[3] For $i = 1,2,\dots,k$, if the i-th symbol in x_1 and the i-th symbol in x_2 represent angles, both angles differ at most by $\Delta\alpha_{\max}$.

[4] For $i = 1,2,\dots,k$, if the i-th symbol $x_1(i)$ in x_1 and the i-th symbol $x_2(i)$ in x_2 represent edge lengths, the ratio $\max\{x_1(i), x_2(i)\}/\min\{x_1(i), x_2(i)\}$ does not exceed $1 + \Delta l_{\max}$.

The parameters $\Delta\alpha_{\max} \in \mathbb{R}_0^+$ and $\Delta l_{\max} \in \mathbb{R}_0^+$ allow users to specify the geometric error tolerance. Furthermore, we define the number k of symbols as the *cardinality* of a match. We are not interested in matches of single line segments, which have cardinality one. In order to exclude such insignificant matches, a user needs to define a third parameter $k_{\min} \in \mathbb{N}$. The cardinality of a match must not be smaller than $k_{\min}$.

Next, we exclude matches that are *dominated* by other matches: A match of two strings x_1 and x_2 is dominated by a match of two strings y_1 and y_2 if

- x_1 is a substring of y_1 and x_2 is a substring of y_2 and
- x_1 has the same position in y_1 as x_2 in y_2, that is, the number of symbols in y_1 preceding x_1 equals the number symbols in y_2 preceding x_2.

Additionally, we need to take care that we do not select a substring of a string XX that is longer than the original string X representing the polygon and we should avoid reporting a match of two polygon parts twice.

Finally, if we have found a match of two strings that satisfies the above-mentioned criteria, we need to decode the two strings into two shapes, for example, to visualize the matching result. The shapes p_1 and p_2 for the two strings x_1 and x_2 of a match are computed as follows.

For each edge symbol in a string, we add the corresponding polygon edge to the shape for the string. If the string begins (or ends) with a symbol for an angle, we add both polygon edges that form this angle. With this approach, however, the first (or last) edge of p_1 and the first (or last) edge of p_2 get very different lengths. Therefore, we shorten the longer edge of both unmatched edges such that they get the same lengths.

3.2 String Matching by Dynamic Programming

In this section we discuss a solution to the problem of finding all type-2 matches satisfying the criteria from Sect. 3.1. The type-1 matches can be found in a straightforward way. We first discuss the special case that $\Delta\alpha_{\max} = \Delta l_{\max} = 0$. In this case, a type-2 match of maximum cardinality

can be found by solving the *longest (or maximum) common substring problem* for the strings $X(P_1)X(P_1)$ and $X^{-1}(P_2)X^{-1}(P_2)$.

The longest common substring problem can be solved in linear time using a generalized suffix tree (Gusfield, 1997). We are interested, however, in finding multiple symmetry relations. Therefore, we search for *all maximal common substrings* of $X(P_1)X(P_1)$ and $X^{-1}(P_2)X^{-1}(P_2)$. Note that there is a difference between a maximum and a maximal common substring of two strings x_1 and x_2: a common substring x of x_1 and x_2 is *maximum* if no other common substring of x_1 and x_2 is longer than x; for x being a *maximal* common substring, however, it suffices that there is no other common substring of x_1 and x_2 that contains x, that is, a match defined by a maximal common substring is not dominated by any other match.

The problem of finding all maximal common substrings of two strings x_1 with m symbols and x_2 with n symbols can be solved in $\Theta(mn)$ time by dynamic programming. To specify this approach, we define the $m \times n$ matrix D of integers. We denote the number in row i and column j of D by $d_{i,j}$. Additionally, we define $d_{0,j} = d_{m+1,j} = 0$ for $j = 0,1,\dots,n+1$ and $d_{i,0} = d_{i,n+1} = 0$ for $i = 0,1,\dots,m+1$. For $i = 1,2,\dots,m$ and $j = 1,2,\dots,n$ we define

$$d_{i,j} = \begin{cases} 1 + d_{i-1,j-1} & \text{if } x_1(i) = x_2(j) \\ 0 & \text{else} \end{cases}, \qquad (1)$$

where $x_1(i)$ denotes the i-th symbol in x_1 and $x_2(j)$ the j-th symbol in x_2. The values of D can be computed in increasing order of the indices for rows and columns.

Once we have computed the matrix D, we can easily find the maximal common substrings. For each pair $i \in \{1,2,\dots,m\}$, $j \in \{1,2,\dots,n\}$ with $d_{i,j} > 0$ and $d_{i+1,j+1} = 0$, the substring of x_1 starting at index position $(i - d_{i,j} + 1)$ and ending at index position i corresponds to one maximal common substring of x_1 and x_2. In x_2, this substring starts at index position $(j - d_{i,j} + 1)$ and ends at index position j.

In order to deal with geometric differences between the two building parts of a match and to avoid the selection of substrings that are longer than the original encoding of the building polygon, we define the values $d_{i,j}$ in a slightly different way:

$$d_{i,j} = \begin{cases} 1 + d_{i-1,j-1} & \text{if } x_1(i) \approx x_2(j) \text{ and } d_{i-1,j-1} < \min\{m/2, n/2\}, \\ 1 & \text{if } x_1(i) \approx x_2(j) \text{ and } d_{i-1,j-1} = \min\{m/2, n/2\}, \\ 0 & \text{else} \end{cases} \quad (2)$$

We define the relation $\approx$ according to the similarity criteria 3 and 4 that we introduced in Sect. 3.1. The additional condition $d_{i-1,j-1} < \min\{m/2, n/2\}$ in the first line of equation (2) avoids that we generate strings that are too long, that is, if $d_{i-1,j-1}$is equal to the length of the string for one of the involved polygons, we do not further extend the corresponding match but start with the construction of a new match. This is done in the second line of equation (2) by setting $d_{i,j}$ to one.

In order to avoid reporting the same match twice, we need to introduce a small modification to the procedure for finding the maximal common substrings in D: instead of considering each pair of indices $i \in \{1,2,\dots,m\}$, $j \in \{1,2,\dots,n\}$ for defining the two ends of the corresponding substrings, we only consider each pair of indices $i \in \{m/2 - 1, m/2, \dots, m-1\}$, $j \in \{n/2 - 1, n/2, \dots, n-1\}$.

Note that, when implementing the presented method, we should avoid comparing edges with angles. Therefore, we can use two matrices D_α and D_e, each of dimension $m/2 \times n/2$, instead of one matrix D of dimension $m \times n$. We use D_α for the comparisons of angles and D_e for the comparisons of edge lengths.

3.3 Least-Squares Adjustment

As a result of the algorithm in Sect. 3.2 we obtain a set of matches, each represented as a pair of strings. We can use the decoding presented in Sect. 3.1 to find the corresponding pair of shapes. The two shapes p_1 and p_2 of a match are polylines, both having the same number κ of vertices. For $i = 1,2,\dots,\kappa$, the i-th vertex of p_1 corresponds with the i-th vertex of p_2.

If p_1 and p_2 correspond by axial symmetry, we can compute the mirror axis by choosing any pair of corresponding vertices v_1 and v_2 and computing a straight line that is perpendicular to the vector $\overrightarrow{v_1 v_2}$ and passes through the midpoint between v_1 and v_2. If we do this for each type-2 match, we obtain candidates for mirror axes. These axes, however, are not very accurate, because we used a single pair of vertices for their construction. In order to obtain more accurate mirror axes, we apply a least-squares

adjustment that uses the information given with *all* pairs of corresponding vertices. The main benefit of this approach is that, in addition to the adjusted mirror axis, it offers a standard deviation that allows us to conclude whether the match indeed corresponds to an axial symmetry, or whether another type of transformation is involved, for example, a transformation or rotation.

For the adjustment we use a Gauss-Helmert model, which has the general form

$$\Psi\left(\underline{\tilde{X}}, \underline{\tilde{L}}\right) = \underline{0}, \tag{3}$$

where $\underline{\tilde{X}}$ is the vector of unknowns without errors and $\underline{\tilde{L}}$ the vector of observations without errors (Niemeier, 2002). The aim of the adjustment process is to add a vector $\underline{v}$ of corrections to the vector of erroneous observations $\underline{L}$ and to estimate the vector of unknowns such that the system of equations (3) holds and the square sum $\underline{v} \cdot \underline{v}$ of the corrections is minimized.

In our case, there are two unknowns, m and b, which define the mirror axis in the form $y = mx + b$. The vector of observations contains the coordinates of the vertices, which means that it has 4κ elements.

For each pair of corresponding vertices v_1 (with coordinates x_1 and y_1) and v_2 (with coordinates x_2 and y_2), we introduce two constraints.

The first constraint means that the midpoint between v_1 and v_2 lies on the mirror axis:

$$(y_1 + y_2)/2 = m\,(x_1 + x_2)/2 + b \tag{4}$$

The second constraint means that the vector $\overrightarrow{v_1 v_2}$ is perpendicular to the mirror axis:

$$(x_2 - x_1) + (y_2 - y_1)m = 0 \tag{5}$$

In order to estimate the corrections and unknowns we linearize equations (4) and (5) and apply the common iterative adjustment procedure (Niemeier, 2002). In each iteration, we update the unknowns m and b. The initial mirror axis is defined based on the two corresponding vertices with the maximum distance. In addition to the estimates for m and b we obtain a standard deviation s based on the corrections $\underline{v}$. A mirror axis is selected if s does not exceed a user-specified value $s_{\max} \in \mathbb{R}_0^+$.

4 Experimental Results

The presented algorithms were implemented in C++ and tested for a data set of 5134 building footprints of the metropolitan Boston area. The data set is freely available as part of the Massachusetts Geographic Information System, MassGIS[1]. According to the data specifications, the building footprints were manually extracted from LiDAR data.

In order to remove marginal details that would hinder the matching process, the building footprints were automatically generalized with an error tolerance ε of 1 m. A neighborhood for the polygons was defined based on a triangulation of the free space not covered by polygons. This triangulation was obtained by using the CGAL[2] library for computational geometry. Each two polygons that were connected with a triangle edge are defined as neighbors.

For each single building and for each pair of neighboring buildings, the type-1 and type-2 matches were searched. Furthermore, for each type-2 match, five iterations of the least-squares adjustment were applied. Together, these computations took 8 seconds on a Windows PC with 3 GB RAM and a 3.00 GHz Intel dual-core CPU.

Table 1. Parameters used for the presented experiments

parameter name	symbol	value
tolerance for building simplification	ε	1 m
tolerance for differences of angles	$\Delta\alpha_{\max}$	0.15 rad ($\approx 8.6°$)
tolerance for differences of edge lengths	$\Delta l_{\max}$	30%
minimum cardinality for matches	$k_{\min}$	8
maximum standard dev. for mirror axes	$s_{\max}$	1 m

Table 1 summarizes the parameters applied, which were found by experiments. Note, however, that setting $k_{\min} = 8$ implies that two symmetry axes are found for a rectangle. Setting $k_{\min}$ to a higher value implies that no symmetry axes are found for a rectangle. Therefore, $k_{\min} = 8$ is, in a way, a natural choice. The sequence of 90° and 270° turns of building outlines is often very characteristic, thus the tolerance for edge lengths is set to a relatively large value (30%) and to the relatively small value of 8.6° for angles (which can be interpreted as roughly 10% of a right angle).

According to the defined criteria, 11528 type-1 matches and 14100 type-2 matches were found. This means that, for each building, 2.2 type-1

[1] http://www.mass.gov/mgis/lidarbuildingfp2d.htm (accessed 21-10-10)
[2] http://www.cgal.org/ (accessed 21-10-10)

matches and 2.7 type-2 matches were found on average. It is interesting that the type-2 matches are more frequent than the type-1 matches, as it shows that a reflection is indeed a preferred concept in building design (compared to pure repetition). Using the approach based on least-squares adjustment, 10477 mirror axes were found, that is, on average, 2.0 for each building. This also implies that 74% of the type-2 matches indeed represent (pure) axial reflections. We now discuss some selected samples from the data set.

Figure 5 (left) illustrates all type-2 matches found for a set of five apartment buildings. For each match, the corresponding building parts are shown as bold lines. Note that the same part may be involved in multiple matches. Additionally, the figure shows the hypotheses for mirror axes (thin lines). Obviously, many hypotheses are wrong, that is, a translation and/or a rotation need to be performed in addition to the axial reflection in order to match the two shapes. Figure 5 (right), however, shows that correct mirror axes are found by filtering the matches based on the standard deviation that we obtained by least-squares adjustment. Additionally, the least-squares adjustment yields accurate axes. The adjustment process is visualized in Fig. 6 for two buildings. In these examples, the initial axes are very inaccurate, but after five iterations we obtain results that are good enough, for example, for visualization.

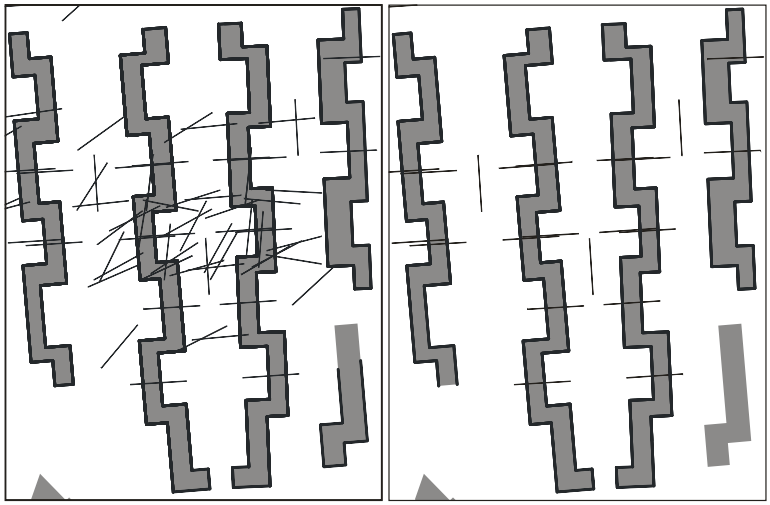

Fig. 5. Hypotheses for mirror axes (left) and selected mirror axes after adjustment (right). The selection of the axes is based on the variance of coordinates that is estimated based on residuals at the polygon vertices. Bold parts of the polygon outlines correspond by symmetry according to the mirror axes shown.

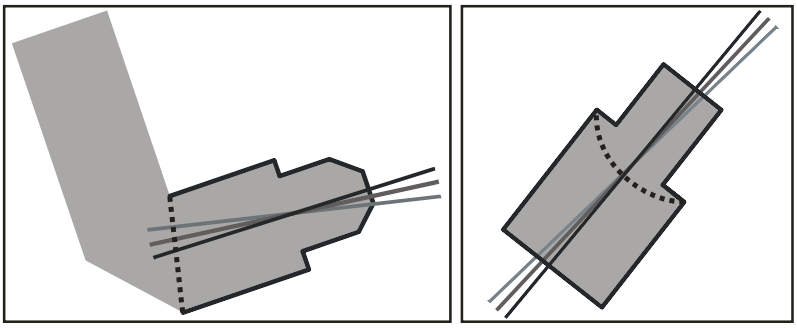

Fig. 6. Illustration of the adjustment process for two buildings with symmetry relations. The figures show the building parts that correspond by symmetry (bold parts of the polygon outlines), the initial mirror axes (light grey lines) the mirror axes after one iteration of the adjustment process (dark grey lines) and after five iterations (black lines). The dashed lines show which pairs of polygon vertices were used to compute the initial axes.

Figures 7, 8, and 9 show the mirror axes that were detected for the samples in Figures 1, 2, and 3, respectively. Generally, the results are satisfactory, that is, the most obvious symmetry relations were found. There are, however, a few open problems that we discuss for the result in Fig. 9.

In some cases we would probably like to report a symmetry relation though each continuous part of the building outline in the relation is small. For example, the mirror axis in Fig. 9 labeled with (1) is noticeable but not detected by the algorithm. In this example, there are two continuous building parts that contribute to the symmetry relation, the front façade and the back façade of a building. Since each of the two parts is small (that is, the corresponding string contains less than 8 symbols), the mirror axis is not detected. Together, however, both parts would have the required size. The aggregation of small matches is a problem that still needs to be solved.

Furthermore, the approach based on string matching relies on pair-wise correspondences of polygon vertices or edges. This is problematic, since two shapes can be similar without having such correspondences. We tried to ease this restriction by applying an algorithm for building simplification that removes potentially disturbing details. The problem, however, still occurs in some cases, especially, if the buildings have curved outlines. The mirror axis in Fig. 9 labeled with (2) corresponds to a symmetry relation of two buildings with circular arcs. The arcs of both buildings were digitized in two very different ways, thus no vertex or edge correspondences were found. This problem could be solved by detecting arcs in the building outline. For buildings that have a rectilinear shape, however, the algorithm yields good results.

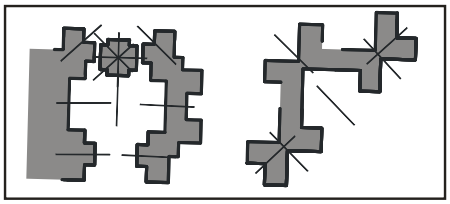

Fig. 7. Detected mirror axes (thin lines) and corresponding building parts (bold lines) for the sample in Fig. 1.

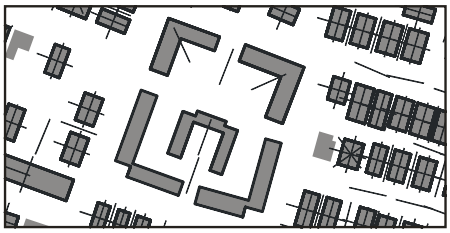

Fig. 8. Detected mirror axes (thin lines) and corresponding building parts (bold lines) for the sample in Fig. 2.

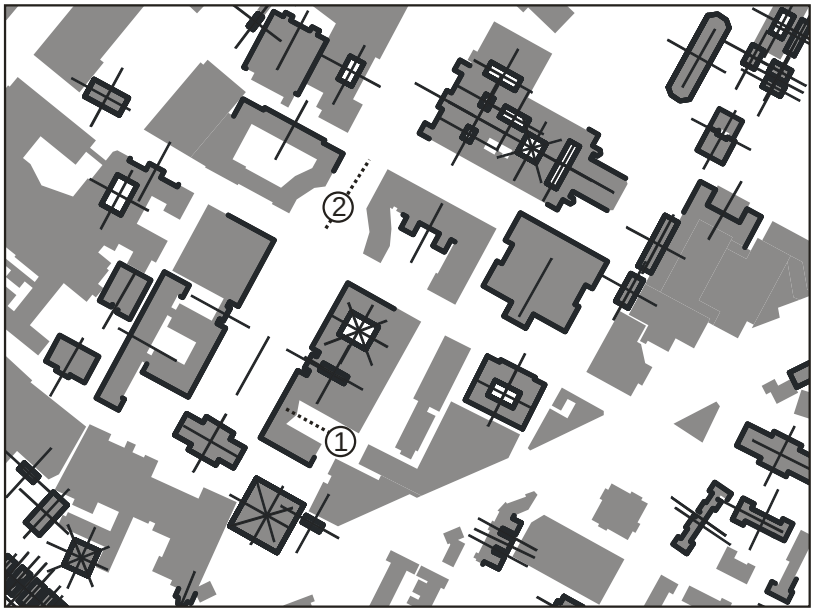

Fig. 9. Detected mirror axes (thin continuous lines) and corresponding building parts (bold lines) for the sample in Fig. 3. The dashed lines labeled with (1) and (2) display axes that were not detected.

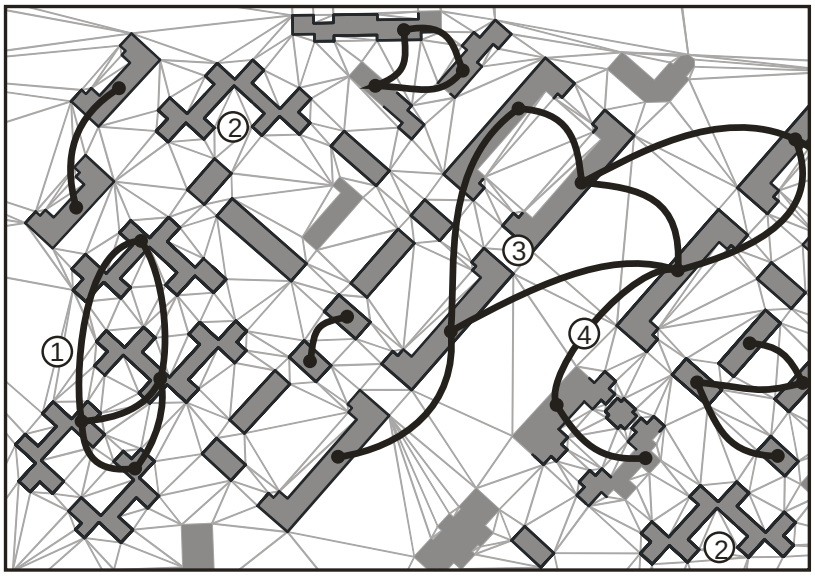

Fig. 10. Detected repetitions (type-1 matches) in building polygons. The bold polygon parts were matched with some other part. The bold arcs link polygons whose parts were matched. The grey lines show edges of the triangulation that was computed to define the neighborhood relation for the buildings. The numbers are referred to in the text.

Finally, we discuss the type-1 matches (that is, repetitions of building parts) yielded by the string-matching method. If we aim to group the buildings according to their similarity, we may be interested in the graph $G(V, E)$ where V is the set of buildings and E contains an edge for each pair of buildings for which at least one type-1 match was found. This graph is illustrated in Fig. 10 (bold arcs). We observe that the connected components of G define a grouping where each group indeed contains buildings of a similar design. For example, the group of four buildings labeled with (1) contains buildings of two different designs that are similar. We find buildings of the same design in different parts of the data set, for example, the buildings labeled with (2). These buildings are not matched because they do not have a similar neighbor. This reflects the proximity criterion in perceptional grouping. Occasionally, we fail to find repetitions (3) or we find matches between buildings that are relatively dissimilar (4). Therefore, additional research on similarity-based grouping is needed. For example, we need to decide how to consider both axial symmetries and repetitions for grouping.

5 Conclusion and Outlook

We have discussed the problem of finding symmetry relations and mirror axes in geospatial datasets of buildings. This problem is important for the solution of map generalization problems, landmark detection, and building classification. The presented algorithm for symmetry detection uses a very efficient string-matching approach based on dynamic programming. Mirror axes are found using an approach based on least-squares adjustment. The algorithm copes both with geometric errors and partial symmetries.

The results that we discussed in this paper show that the proposed method allows us to process large datasets fast (that is, several thousands of buildings in a few seconds) and to find most of the dominant symmetry relations. On average, for each building two mirror axes were found. In addition to the symmetry axes, the algorithm yields matches of similar building parts.

Future work is needed to aggregate symmetry relations. This is important, since symmetry relations involving multiple disconnected building parts are currently not considered in the algorithm proposed.

Furthermore, it is planned to integrate the derived information into methods for map generalization. We can expect that using the information derived with the presented algorithm will clearly improve the results of map generalization, in particular, building simplification and aggregation.

References

Atallah, M. J. (1985). On Symmetry Detection. IEEE Transactions on Computers, c-34(7), 663–666.

Cho, M. and Lee, K. M (2009). Bilateral Symmetry Detection via Symmetry-Growing. In: Proc. British Machine Vision Conference (BMVC '09).

Christophe, S. and Ruas, A. (2002). Detecting Building Alignments for Generalisation Purposes. In: Proc. ISPRS Commission IV Symposium on Geospatial Theory, Processing and Applications. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. XXXIV, part 4.

Gaffuri, J. and Trévisan, J. (2004). Role of Urban Patterns for Building Generalisation: An Application of AGENT. In: Proc. 7th ICA Workshop on Generalisation and Multiple Representation.

Gusfield, D. (1997). Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press.

Haunert, J.-H. and Wolff, A. (2010). Optimal and Topologically Safe Simplification of Building Footprints. Pages 192–201 of: Proc. 18th ACM

SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM-GIS'10).

Heinzle, F. and Anders, K.-H. (2007). Characterising Space via Pattern Recognition Techniques: Identifying Patterns in Road Networks. Chap. 12, pages 233–254 of: Mackaness, W., Ruas, A., and Sarjakoski, T. L. (eds), Generalisation of geographic information: Cartographic modelling and applications. Elsevier.

Knuth, D. E., J. H. Morris, Jr., and Pratt, V. R. (1977). Fast Pattern Matching in Strings. Siam Journal on Computing, 6(2), 323–350.

Lladós, J., Bunke, H., and Martí, E. (1997). Using Cyclic String Matching to Find Rotational and Reflectional Symmetries in Shapes. Pages 164–179 of: Intelligent Robots: Sensing, Modeling and Planning. Series in Machine Perception and Artificial Intelligence, vol. 27. World Scientific.

Loy, G. and Eklundh, J.-O. (2006). Detecting Symmetry and Symmetric Constellations of Features. Pages 508–521 of: Proc. 9th European Conference on Computer Vision (ECCV '06), Part II. Lecture Notes in Computer Science, vol. 3952. Springer.

Mackaness, W. and Edwards, G. (2002). The Importance of Modeling Pattern and Structure in Automated Map Generalisation. In: Proc. Joint ISPRS/ICA Workshop on Multi-Scale Representations of Spatial Data.

Mitra, N. J., Guibas, L. J., and Pauly, M. (2006). Partial and Approximate Symmetry Detection for 3D Geometry. ACM Transactions on Graphics, 25(3), 560–568.

Neun, M., Burghardt, D., and Weibel, R. (2008). Web Service Approaches for Providing Enriched Data Structures to Generalisation Operators. International Journal of Geographic Information Science, 22(2), 133–165.

Niemeier, W. (2002). Ausgleichungsrechnung. Walter de Gruyter.

Park, M., Lee, S., Chen, P.-C., Kashyap, S., Butt, A. A., and Liu, Y. (2008). Performance Evaluation of State-of-the-Art Discrete Symmetry Detection Algorithms. In: Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '08).

Peters, D., Wu, Y. H., and Winter, S. (2010). Testing Landmark Identification Theories in Virtual Environments. Pages 54–69 of: Spatial Cognition. Lecture Notes in Computer Science, vol. 6222. Springer.

Regnauld, N. (2003). Algorithms for the Amalgamation of Topographic Data. In: Proc. 21st International Cartographic Conference (ICC '03).

Ruas, A. and Holzapfel, F. (2003). Automatic Characterization of Building Alignments by Means of Expert Knowledge. In: Proc. 21st International Cartographic Conference (ICC '03).

Steiniger, S. (2007). Enabling Pattern-Aware Automated Map Generalization. PhD thesis, University of Zürich.

Steiniger, S., Burghardt, D., Lange, T., and Weibel, R. (2008). An Approach for the Classification of Urban Building Structures Based on Discriminant Analysis Techniques. Transactions in GIS, 12(1), 31–59.

Thomson, R. C. and Brooks, R. (2002). Exploiting Perceptual Grouping for Map Analysis, Understanding and Generalization: The Case of Road and River

Networks. Pages 148–157 of: Proc. 4th International Workshop on Graphics Recognition Algorithms and Applications. Lecture Notes in Computer Science, vol. 2390. Springer.

Werder, S., Kieler, B., and Sester, M. (2010). Semi-Automatic Interpretation of Buildings and Settlement Areas in User-Generated Spatial Data. In: Proc. 18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS '10). To Appear.

Wertheimer, M. (1938). Laws of Organization in Percetional Forms. Pages 71–88 of: A Source Book of Gestalt Psychology. Routledge & Kegan Paul.

Wolter, J. D., Woo, T. C., and Volz, R. A. (1985). Optimal Algorithms for Symmetry Detection in two and three Dimensions. The Visual Computer, 1(1), 37–48.

Yan, H., Weibel, R., and Yang, B. (2008). A Multi-Parameter Approach to Automated Building Grouping and Generalization. GeoInformatica, 12(1), 73–89.

Yang, X., Adluru, N., Latecki, L. J., Bai, X., and Pizlo, Z. (2008). Symmetry of Shapes Via Self-Similarity. Pages 561–570 of: Proc. 4th International Symposium on Advances in Visual Computing, Part II. Lecture Notes In Computer Science, vol. 5359. Springer.