A Symmetry Detector for Map Generalization and Urban-Space Analysis

Jan-Henrik Haunert

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

---

Abstract

This article presents an algorithmic approach to the problem of finding symmetries in building footprints, which is motivated by map generalization tasks such as 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 vector data that use algorithms for string matching. It detects both mirror symmetries and repetitions of geometric structures. In addition to the existing vector-based methods, the new method finds partial symmetries in polygons while allowing for small geometric errors and, based on a least-squares approach, computes optimally adjusted mirror axes and assesses their quality. Finally, the problem of grouping symmetry relations is addressed with an algorithm that finds mirror axes that are almost collinear. The presented approach was tested on a large building dataset of the metropolitan Boston area and its results were compared with results that were manually generated in an empirical test. The symmetry relations that the participants of the test considered most important were found by the algorithm. Future work will deal

with the integration of information on symmetry relations into algorithms for map generalization.

Keywords: GIS, Geometry, Algorithms, Generalization, Urban, Building, Pattern, Detection

---

1. Introduction

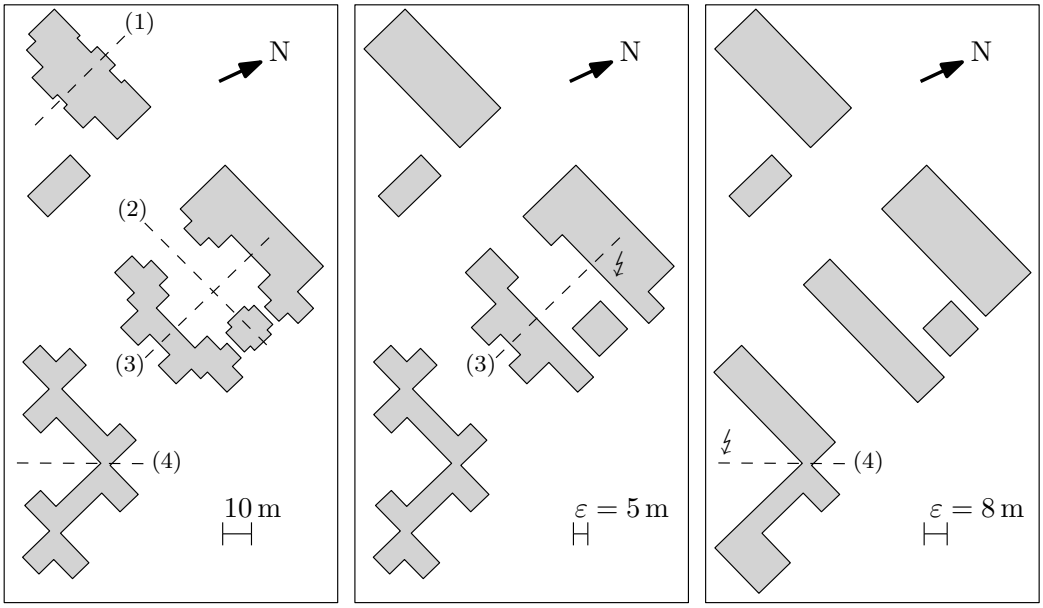
Symmetry is a fundamental element of design. In architecture, it is widely applied from the micro-level (facade decor) to the macro-level (urban design, landscape architecture) and for multiple reasons (functionality, aesthetics, low construction costs). A large number of empirical studies by psychologists suggests that humans are extremely good in detecting symmetries in shapes and generally perceive symmetries as important shape characteristics—literature surveys are given by Wagemans (1994) and Treder (2010). Since many geographic analysis tasks require methods for shape characterization, an automatic symmetry detector is needed. This article presents a new algorithm for the detection of symmetries and repetitive structures in polygons, which is tailored to deal with building footprints typically found in cadastral or topographic vector databases. It was tested for a building dataset of the metropolitan Boston area.

This work contributes to the general aim of enriching spatial data with information on geometric structures and patterns. Such information is valuable for multiple applications. The main motivation of this work is *map generalization*, which aims to decrease a map's level of detail while preserving its characteristic structures. With respect to buildings, both symmetry-preserving simplification and symmetry-aware aggregation are interesting

map generalization problems that have not been approached yet.

Several authors have proposed algorithms for *building simplification* (Mayer, 1998; Sester, 2005; Kada and Luo, 2006; Damen et al., 2008). We (Haunert and Wolff, 2010) have developed a new, optimization-based algorithm that allows multiple quality criteria to be integrated, for example, the preservation of a building's area and its dominating edge directions. Symmetry preservation, however, is currently not considered as a quality criterion in this method or in other methods, thus symmetries may become lost by automatic simplification, see Fig. 1. In an empirical test, which we will discuss in Sect. 4 in detail, 30 participants were asked to scale Fig. 1(a) by a factor of 0.5. To keep the building footprints as legible as in the original figure, the participants were allowed to simplify shapes. Among all non-trivial solutions (solutions with more than four but less than the original number of vertices) the overwhelming majority (81%) preserved the dominant symmetries of the input polygons. This confirms the assumption that symmetries are important for map generalization. A good strategy to preserve symmetries seems to be to first detect symmetries in the input buildings. Then, simplifications that destroy symmetries can be penalized by defining an appropriate cost function. This approach will hopefully yield simplification results that better reflect the characteristic properties of the input buildings.

The problem that a simplification algorithm destroys symmetries in a building footprint especially arises when generalizing a small-scale map into a medium-scale map that still allows the buildings to be displayed with a relatively large amount of detail. Another map generalization problem, which is more relevant when a small-scale map needs to be derived, is *building aggre-*

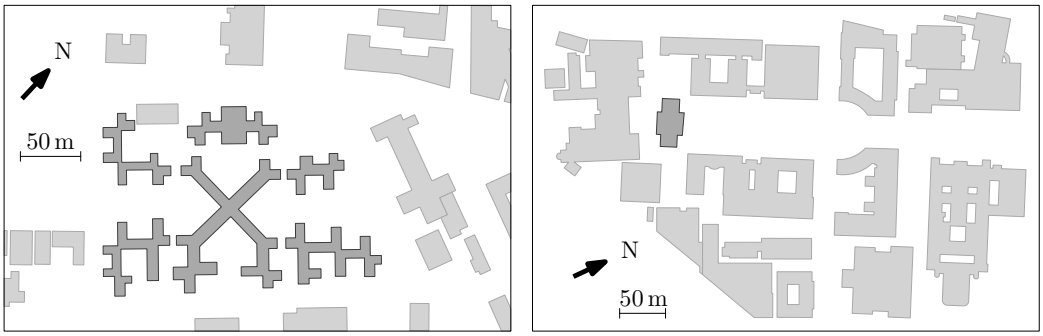


(a) six buildings with characteristic symmetries (dashed lines labeled with 1–4)

(b) simplification result with $\varepsilon = 5\,\mathrm{m}$ that violates symmetry 3

(c) simplification result with $\varepsilon = 8\,\mathrm{m}$ that violates symmetry 4

Figure 1: When simplifying building footprints with the method by Haunert and Wolff (2010), characteristic symmetries may become lost. The parameter ε is the error tolerance that defines the degree of simplification.



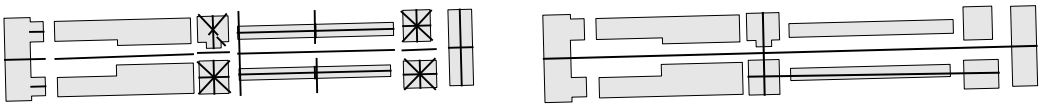
(a) six buildings (dark shaded) that, due to reflectional symmetry and repetitive structures, can be perceived as a group

(b) a building (dark shaded) whose importance is emphasized by symmetry as an element of architectural design

Figure 2: Two patterns of buildings in Boston, MA.

gation, which 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, too. For example, humans better perform on the task of deciding on the presence of a shape outline in an array of Gabor elements (white dots blurred in different directions) if the shape is symmetric (Machilsen et al., 2009). Similarly, we clearly perceive that six of the buildings in Fig. 2(a) form an ensemble, which presumably is because of their symmetric arrangement and repetitive structures. Therefore, replacing the ensemble by a single (larger) shape can be a favorable generalization action. Alternatively, we may select a subset of the input buildings for the small-scale map. Also in this case it makes sense to first find groups of buildings, since the characteristics of a group need to be preserved by the selection (Regnauld, 2001).

Map generalization is not the only application this work aims for. Architects and city planners are also interested in evaluating the design of cities, for example, to reveal dependencies between an architectural design principle and the navigation performance of pedestrians (Wiener and Franz, 2004). In this context, the term *space syntax* is often used for concepts describing the structuring of towns by straight lines of sight and viewsheds (Hillier and Hanson, 1984). Jiang and Claramunt (2002) have dealt with the integration of space syntax analysis methods into geographic information systems. In order to characterize the space surrounding a pedestrian, Leduc et al. (2011)



(a) mirror axes found by string matching (b) lines supported by multiple axes in (a)

Figure 3: In a first step (a), mirror axes are detected that reflect symmetry relations between a connected piece of the boundary of a polygon and a connected piece of the boundary of another (or the same) polygon. In a second step (b), mirror axes are grouped to find symmetry relations involving multiple buildings or building parts.

have developed a method that classifies isovist polygons (that is, portions of a constant-height plane that are visible from given points) into classes like street canyon or public square. A symmetry detector could be used, for example, to quantify the degree of symmetry of an isovist polygon and, thereby, to measure the regularity of an architectural design, which certainly influences how users interact with space. As an example, consider the set of buildings in Fig. 2(b). Even though the dark shaded building is among the smallest buildings in the set, it has a high perceptual salience, which is due to the symmetric arrangement of the buildings. Therefore, it would probably be a good idea to refer to that building in a driving instruction. Moreover, we may assume that the building has an important representative function, like a town hall or a castle. The building is in fact the main building of Harvard Medical School. Symmetry can thus be used as a cue for both automatic *landmark selection* (that is, deciding which building serves best as a landmark in a driving instruction) and *building classification*, which are topical problems in geographic information science. A comparison of methods for landmark selection is given by 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.

This article is based on a conference paper (Haunert, 2011) that presented a basic string-matching method for symmetry detection. The original method finds basic symmetry relations between two polylines, each being a connected piece of the boundary of an input polygon. The symmetry relations we are interested in, however, may involve multiple buildings or building parts—consider, for example, a road where each building on the left side has a mirror-image counterpart on the right side. To handle such cases, the original method had to be extended. In particular, an aggregation step was needed, in which short line segments that represent mirror axes are grouped into longer line segments, see Fig. 3. With the revised method, multi-part symmetry relations that involve multiple buildings can be detected.

For every match found with the string-matching algorithm, a global geometric consistency test needs to be performed. For example, the mirror axis fitting best for all pairs of corresponding points of a match needs to be found, which can be done by least-squares adjustment. Based on the quality of the least-squares fit it is then possible to decide whether a match indeed corresponds to a reflection at a straight line. Computing the least-squares fit with the common iterative algorithm (Kraus, 2004, Appendix to Section 4.1), which was suggested in the paper preceding this article, turned out to be very time consuming. In contrast, the revised method uses a faster, non-iterative solution based on a singular value decomposition (SVD).

In summary, this article combines several existing techniques in an innovative way, namely, shape simplification, symmetry detection by string matching, grouping of lines (mirror axes), and least-squares adjustment. Together, these techniques yield a new symmetry detector for vector data that

is more widely applicable than the existing methods. More precisely, the new method allows multiple symmetry relations to be found of which each may involve multiple disconnected parts of a set of input polygons, even if the symmetric counterparts slightly differ from each other. Such a method has not been presented and tested on a large real-world instance before.

Furthermore, this article presents results of an empirical study in which humans were asked to manually solve three tasks (building simplification, building grouping, detection of mirror axes) to which the proposed symmetry detector can be applied. By comparing the automatically and the manually generated results, the symmetry detector is shown to indeed find the most salient symmetry relations in the data, but also open problems and ideas for future research are identified.

The article is structured as follows. We first discuss related work on data enrichment in map generalization and on algorithms for symmetry detection (Sect. 2). Then, the algorithm for symmetry detection is presented in detail (Sect. 3). In Sect. 4, we discuss the empirical study and experimental results obtained with the new algorithm. Section 5 concludes the article.

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; Steiniger, 2007; Neun et al., 2008). 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, for example, when selecting a subset of all roads for a map.

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 Yang (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, the characteristic 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 a polygon P as a string X, for example, as a sequence of angles and edge lengths, see Fig. 4.

We denote the reversal of a string A by A^{-1} and the concatenation of two strings A and B by AB. In order to find a mirror symmetry, the method of Wolter et al. (1985) tests whether the string X^{-1} is a substring of the string XX (i.e., the concatenation of X with itself). This test can be done in $\Theta(n)$ time where n is the number of 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 mirror symmetry. 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. Lladós et al. (1997) use an approach based on a string edit distance that, in order to cope with distorted shapes, allows sequences of symbols in the string representation of a polygon to become merged and the resulting new symbols to become matched. An alternative approach by Yang et al. (2008) is based on critical contour points, that is, vertices of a simplified version of the original contour. The method presented in this article is similar in the sense that the string matching is performed on a simplified version of the input polygon. Since symmetry-preserving algorithms for line and building simplification do not exist, however, a building simplification algorithm is applied with a conservative setting, primarily to remove redundant vertices.

In order to aggregate mirror symmetries found by string matching (recall Fig. 3) the corresponding mirror axes are grouped. The presented algorithm

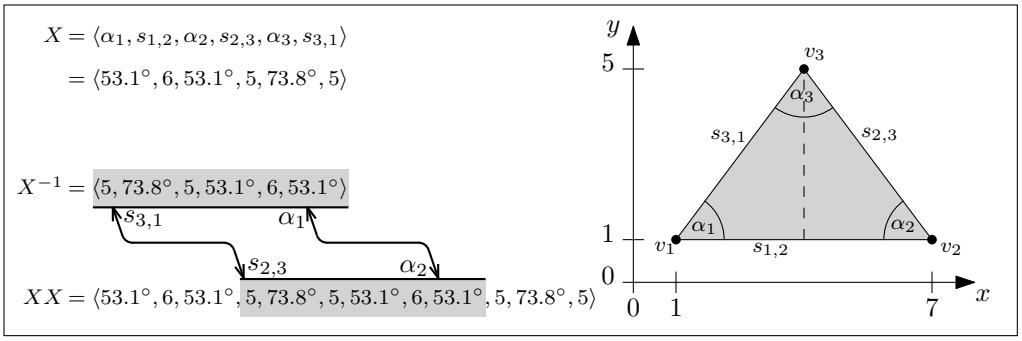


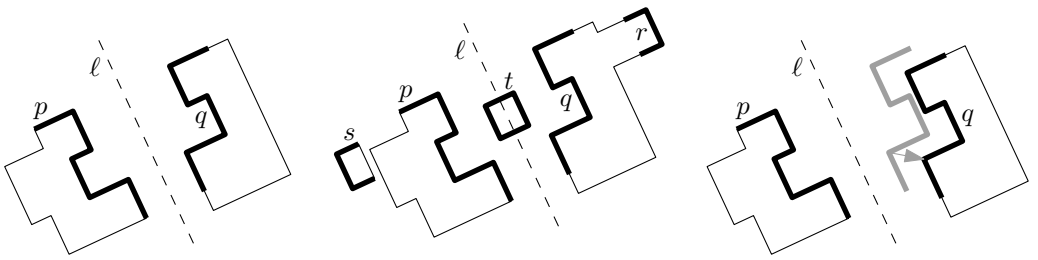
Figure 4: Principle of the algorithm for symmetry detection by Wolter et al. (1985). Since X^{-1} is a substring of XX, the polygon is mirror symmetric. For example, edge $s_{3,1}$ is a mirror image of edge $s_{2,3}$. The mirror axis is drawn as a dashed line.

builds up on two simple existing methods, namely a transformation of lines into Hough space (Hough, 1962) and a grouping algorithm for collinear line segments (Scher et al., 1982).

3. Methodology

3.1. Preliminaries

Generally, we define a *symmetry* s as a triplet (P, Q, T), where P and Q are sets of polylines and T is a geometric transformation such that applying T to the polylines in P yields (with some error tolerance) the polylines in Q. We require that each of the polylines in P and Q is a connected part of the boundary of an input polygon. If $|P| = |Q| = 1$, we say that s is *basic*. Note that in a basic symmetry the polyline in P and the polyline in Q may be the same. For instance, let p be equal to the entire boundary of the polygon in Fig. 4. Indeed, there is a non-trivial transformation T that maps p onto itself: the reflection at the vertical line through v_3. Therefore, $(\{p\}, \{p\}, T)$ is a basic symmetry.



(a) $(\{p\}, \{q\}, T)$ is basic and pure

(b) $(\{p, s, t\}, \{q, r, t\}, T)$ is pure but not basic

(c) $(\{p\}, \{q\}, T)$ is basic but not pure

Figure 5: Three types of reflections. The transformation T in (a) and (b) is the reflection at the straight line ℓ; in (c), T additionally has a translation component (gray arrow).

We allow the transformation T of a symmetry $s = (P, Q, T)$ to be of two different types, that is, it may either be

(a) a reflection at a straight line followed by a translation or

(b) a proper rigid transformation (i.e., a rotation followed by a translation).

If T is of type (a), we term s a *reflection*. If T is of type (b), we term s a *repetition*. We are particularly interested in *pure* reflections, that is, reflections that do not have a translation component. Figure 5 visualizes the definitions of pure and basic reflections on examples.

The symmetry detection comprises a matching stage and a grouping stage.

In the matching stage, basic symmetries are detected. To this end, a string-matching algorithm is applied that yields a set of pairs of polylines. Then, a least-squares adjustment is performed to find a transformation for each such pair. The square sum of residuals allows us to decide whether the pair (p, q) of polylines and the transformation T found for this pair indeed

constitute a basic symmetry $(\{p\}, \{q\}, T)$. If the square sum of residuals is too large, that triplet becomes rejected.

In the grouping stage, the set of basic symmetries is partitioned into groups. Basically, two or more basic symmetries become grouped if their transformations have similar parameters. Two pure reflections, for example, should only be grouped if their mirror axes are (almost) collinear. A simple iterative algorithm for grouping is applied that respects this criterion. At the end of the second stage, the least-squares method is applied to each group in order to find the transformation T that fits best for all polylines in that group. Again, we may reject a group of basic symmetries if this transformation is not good enough, that is, if the square sum of residuals is too large.

In the remainder of this section, we discuss the representation of polylines as strings (Sect. 3.2), the string-matching algorithm (Sect. 3.3), and the least-squares approach (Sect. 3.4) used in the matching stage. Then, the grouping algorithm is presented (Sect. 3.5); the least-squares method from Sect. 3.4 is also used in the grouping stage.

3.2. Symmetries in the string representation of polygons

We encode a polygon P as a string $X(P)$ of edge lengths and angles as it was shown in Fig. 4. Thereby, we obtain a shape representation that is invariant against rotations and translations. Figure 6 illustrates on an example how we can find *partial* symmetries in P based on string matches. Generally, each string match is a pair of strings, one of them being a substring of $X(P_1)X(P_1)$ and the other one being a substring either of $X(P_2)X(P_2)$ or of $X^{-1}(P_2)X^{-1}(P_2)$, where P_1 and P_2 are two potentially (but not necessarily) distinct polygons. String matches comprising a substring of $X(P_1)X(P_1)$ and

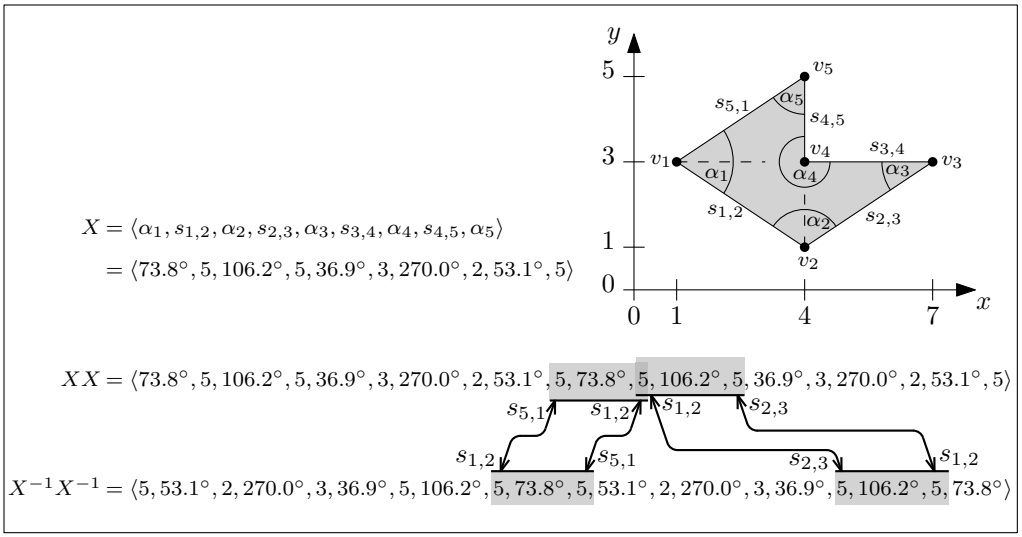


Figure 6: Principle of the algorithm for the detection of partial symmetries.

a substring of $X(P_2)X(P_2)$ define basic repetitions; string matches comprising a substring of $X(P_1)X(P_1)$ and a substring of $X^{-1}(P_2)X^{-1}(P_2)$ define basic reflections. Suppose that, in the example in Fig. 6, we search for common substrings of XX and $X^{-1}X^{-1}$ of at least three symbols, where X is the string encoding the given polygon. As the result, we find two matches, namely, $\langle s_{5,1}, \alpha_1, s_{1,2} \rangle = \langle s_{1,2}, \alpha_1, s_{5,1} \rangle = \langle 5, 73.8°, 5 \rangle$, which corresponds to a reflection at the horizontal line through v_1, and $\langle s_{1,2}, \alpha_2, s_{2,3} \rangle = \langle s_{2,3}, \alpha_2, s_{1,2} \rangle = \langle 5, 106.2°, 5 \rangle$, which corresponds to a reflection at the vertical line through v_2. Note that we would have found $\langle 5, 73.8°, 5 \rangle$ neither in X nor in X^{-1}, thus it was indeed necessary to concatenate X with itself and X^{-1} with itself before searching for common substrings. On the other hand, we find $\langle 5, 106.2°, 5 \rangle$ twice in XX and twice in $X^{-1}X^{-1}$, thus we have to keep in mind that we should avoid reporting the same match multiple times.

Since we want to tolerate small geometric differences between the shapes

that we match, we do not insist on the identity of the two strings x_1 and x_2 that form a string match but instead require the following conditions:

C1 The number k of symbols is the same in both strings.

C2 Both strings start with the same type of symbol, that is, either with a symbol representing an edge length or an angle.

C3 For $i = 1, 2, \ldots, k$, if the i-th symbol in x_1 and the i-th symbol in x_2 represent angles, both angles differ at most by $\Delta\alpha_{\max}$.

C4 For $i = 1, 2, \ldots, k$, 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 $1 + \Delta\ell_{\max}$.

The parameters $\Delta\alpha_{\max}$ and $\Delta\ell_{\max}$ allow us to specify our geometric error tolerance. In the example in Fig. 6, if we set $\Delta\alpha_{\max} = 35°$ and $\Delta\ell_{\max} = 0.5$, we can match the substring $s_1 = \langle 73.8°, 5, 106.2°, 5, 36.9°, 3, 270.0°, 2, 53.1°, 5\rangle$ of XX with the substring $s_2 = \langle 106.2°, 5, 73.8°, 5, 53.1°, 2, 270.0°, 3, 36.9°, 5\rangle$ of $X^{-1}X^{-1}$, thus the whole polygon P would be considered symmetric—a good (perhaps not the best) transformation for this string match would be the reflection at the bisecting line of α_4. Note that, in this example, we can find matches of strings in XX and $X^{-1}X^{-1}$ that are even longer than s_1 and s_2. Such string matches, however, are not interesting, since s_1 and s_2 already contain all symbols of X and X^{-1}. Likewise, we are not interested in matches of very short strings, since these would correspond to very unimportant symmetry relations. Therefore, we define a third parameter $k_{\min} \in \mathbb{N}$. We require that the cardinality of a string match, that is, its total number of symbols, must not be smaller than $k_{\min}$.

Next, we exclude string matches that are dominated by other string

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

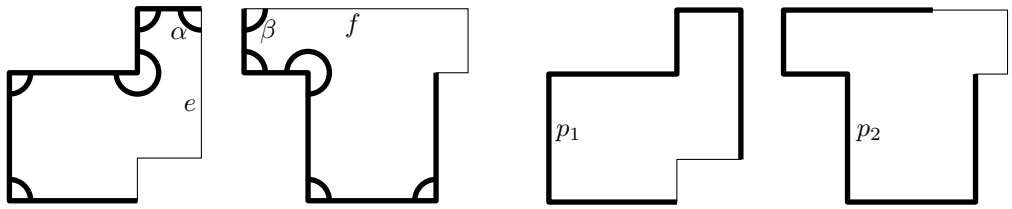
1. x_1 is a substring of y_1 and x_2 is a substring of y_2 and
2. x_1 has the same position in y_1 as x_2 in y_2, that is, the number of symbols in y_1 preceding x_1 equals the number of symbols in y_2 preceding x_2.

According to this definition, in the example in Fig. 6 with $\Delta\alpha_{\max} = \Delta\ell_{\max} = 0$, the match of the substring $\langle s_{1,2}, \alpha_2 \rangle$ of XX and the substring $\langle s_{2,3}, \alpha_2 \rangle$ of $X^{-1}X^{-1}$ would be dominated by the match of the strings $\langle s_{1,2}, \alpha_2, s_{2,3} \rangle$ and $\langle s_{2,3}, \alpha_2, s_{1,2} \rangle$.

If we have found a match of two strings that satisfies the above-mentioned conditions, we need to decode the two strings into two polylines, which are required as input for the least-squares method. The polylines p_1 and p_2 for the two strings x_1 and x_2 of a string match are computed as follows; the result is shown in Fig. 7. For each edge symbol in a string, we add the corresponding polygon edge to the polyline for that 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.

3.3. String matching by dynamic programming

In this section we discuss a solution to the problem of finding all string matches for reflections that satisfy the conditions from Sect. 3.2. The string matches for repetitions can be found in a straightforward way. We first discuss the special case that $\Delta\alpha_{\max} = \Delta\ell_{\max} = 0$. In this case, a string match of



(a) two strings of angles (circular arcs) and lengths of edges (displayed fat) that match

(b) polylines (displayed fat) encoded by the strings of angles and edge lengths in (a)

Figure 7: Decoding of a string match (a) into a pair of corresponding polylines p_1 and p_2 (b). Note that the edges e and f have very different lengths. Still, e is added to p_1 and a part of f is added to p_2 since the angles α and β match.

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 string match defined by a maximal common substring is not dominated by any other string 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 $j = 0, 1, \ldots, n+1$ and $d_{i,0} = d_{i,n+1} = 0$ for

$i = 0, 1, \ldots, m+1$. For $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$ we define

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

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 indexes for rows and columns. Once we have computed the matrix D, we can easily find the maximal common substrings. For each pair (i, j) with $i \in \{1, 2, \ldots, m\}$ and $j \in \{1, 2, \ldots, n\}$ 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:

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

where both $m = |X(P_1)X(P_1)| = 2|X(P_1)|$ and $n = |X^{-1}(P_2)X^{-1}(P_2)| = 2|X^{-1}(P_2)|$ are even numbers. We define the relation $\approx$ in Eq. (2) according to the conditions C3 and C4 that we introduced in Sect. 3.2. The additional condition $d_{i-1,j-1} < \min\{m/2, n/2\}$ in the first line of Eq. (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 Eq. (2) by setting $d_{i,j}$ to one.

In order to avoid reporting the same string match twice, we only report a match of two strings x_1 and x_2 if x_1 ends in the second half of $X(P_1)X(P_1)$ and x_2 ends in the second half of $X^{-1}(P_1)X^{-1}(P_1)$.

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

3.4. Least-squares adjustment

As a result of the algorithm in Sect. 3.3 we obtain a set of string matches. We can use the decoding presented in Sect. 3.2 to find for each string match the two corresponding polylines p_1 and p_2, which both have the same number κ of vertices. It remains to find a transformation T such that $s = (\{p_1\}, \{p_2\}, T)$ is a basic symmetry.

Generally, we are interested in a transformation T (of a certain class of transformations) that minimizes

$$\sum_{i=1}^{\kappa} \|T(p_1(i)) - p_2(i)\|^2 , \tag{3}$$

where $p(i)$ is the coordinate vector of the i-th vertex of polyline p and $T(p(i))$ is the result of applying T to $p(i)$.

Obviously, T could be found with the least-squares approach that is common in photogrammetry and surveying (Kraus, 2004, Appendix to Section 4.1), that is, one could iteratively set up and solve the Gaussian normal equations. Fortunately, however, for the case that T is restricted to combinations of rotations and translations, Arun et al. (1987) have found a non-iterative method that is exact and much more efficient. In fact, the authors have dealt with the problem of aligning two sets of three-dimensional points; since we have to deal with two dimensions only, the method becomes even simpler.

The method first computes the center of gravity $c_1 = \frac{1}{\kappa}\sum_{i=1}^{\kappa} p_1(i)$ of the vertices of p_1 and the center of gravity $c_2 = \frac{1}{\kappa}\sum_{i=1}^{\kappa} p_2(i)$ of the vertices of p_2. Then, *reduced coordinates* $q_1(i) = p_1(i) - c_1$ and $q_2(i) = p_2(i) - c_2$ for $i = 1, 2, \ldots, \kappa$ are introduced. Finally, a rotation is searched that, when applied to the points $q_1(i)$, minimizes the square sum of residuals to the points $q_2(i)$. This is done by first computing the matrix $H = \sum_{i=1}^{\kappa} q_1(i)q_2(i)^T$ and then its singular value decomposition, that is, matrices U, Λ, and V such that $H = U\Lambda V^T$, both U and V are 2×2 unitary matrices, and Λ is a 2×2 rectangular diagonal matrix with nonnegative real numbers. The matrix $X = VU^T$ usually has determinant $+1$; in this case, it is a rotation matrix and the optimal solution is to apply the rotation defined by X followed by the translation defined by $c_2 - Xc_1$. If the determinant of X is -1, the algorithm fails. This, however, only occurs if there is no reasonably good rotation that would allow the two point sets to become matched. In this case, we can safely reject the match as a repetition.

Interestingly, if the determinant of X is -1, X is a reflection matrix, that

is, it defines a reflection at a straight line ℓ through the origin. Therefore, we can use the same method to find an optimal combination of a reflection and a translation for two polylines.

In any case, we compute the residuals of the transformation found and the a-posteriori variance of the vertex coordinates. If this exceeds a user-set threshold, we reject the match.

Since we are often interested in *pure* reflections at a straight line, we also test the result of setting the translation part of a reflection to zero. For this test we translate the mirror axis ℓ such that it contains the point $(c_1 + c_2)/2$, that is, the center of gravity of *all* points. We then reflect p_1 across ℓ and again compute the square sum of coordinate differences to p_2 in order to decide whether the transformation defines a new symmetry relation for the polylines p_1 and p_2.

3.5. Grouping of basic symmetries

In the grouping step, a partition $\mathcal{P}$ of the set of basic symmetries is computed. For every element $\{(\{p_1\}, \{q_1\}, T_1), \ldots, (\{p_k\}, \{q_k\}, T_k)\}$ of $\mathcal{P}$ (that is, for every *group*) there has to be a transformation T of the same type as $T_1, \ldots, T_k$ such that $(\{p_1, \ldots, p_k\}, \{q_1, \ldots, q_k\}, T)$ is a symmetry. That is, applying T to polyline p_i for $i = 1, \ldots, k$ yields, with only small residuals, polyline q_i. Preferably, the number of groups should be small. For particular types of transformations there might be efficient algorithms that yield optimal solutions to this problem. It is known, however, that similar problems such as the problem CLUSTERING (Garey and Johnson, 1990) are NP-hard, even under very simplifying assumptions. Therefore, an iterative heuristic method based on a Hough transform (Hough, 1962) is applied.

The algorithm inserts all basic symmetries of the same type into an accumulator matrix A. The cells of A correspond to disjoint equally-sized k-dimensional ranges, where k is the number of transformation parameters for the respective symmetry type. We insert every symmetry s as a *seed* into the cell c of A whose range contains the transformation parameters of s and a non-seed copy of s into each of the $3^k - 1$ adjacent cells of c. Thereby we ensure that two symmetries with similar transformations will indeed share a common accumulator cell and thus may become grouped.

The algorithm iteratively selects an ungrouped seed s from the accumulator matrix. Then, it attempts to assigns as many elements as possible to s by iterating over the ungrouped elements in the same cell. An element e is assigned to the group of seed s if the transformations of e and s have similar parameters. For all elements in a group, we finally find the best fitting transformation T using the least-squares approach from Sect. 3.4. Based on the square sum of residuals we decide whether or not we accept the group as a new (non-basic) symmetry.

For a pure reflection, the transformation is well defined with the mirror axis ℓ (a straight line), which we parametrize with the distance d between ℓ and the origin of the coordinate system and the angle $\alpha \in [0, \pi[$ between ℓ and the x-axis; the accumulator matrix A thus has two dimensions. For every mirror axis ℓ, however, we are additionally given a start and end point on ℓ defined by the spatial extent of the building parts reflected at ℓ, see Fig. 8. This information is useful, since the grouping of symmetries should also respect spatial proximity. Therefore, we use a method for merging line segments instead of grouping (infinitely long) straight lines. *Merging* a line

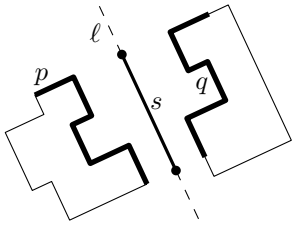
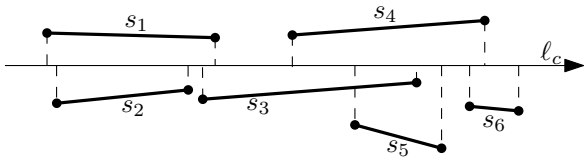


Figure 8: The line segment s for the basic symmetry $(\{p\}, \{q\}, T)$, where T is the reflection at line ℓ.



$S_c = \langle s_1, s_2, s_3, s_4, s_5, s_6 \rangle$

$E_c = \langle s_6, s_4, s_5, s_3, s_1, s_2 \rangle$

Figure 9: If s_3 is the seed segment, Algorithm GroupSegments attempts to merge line segments s_4, s_5, s_6, s_1, s_2 (in that order) with s_3.

segments t with a seed segment s means to select the two most distant vertices u, v of the four vertices of t and s and to set $s = (u, v)$.

Algorithm GroupSegments defines the iterative merging process in detail.

```
GroupSegments()
 1 Fill matrix A with one seed plus eight non-seed copies per segment.
 2 Get longest seed segment s in A and mark all its copies in A as grouped.
 3 S = an empty set that will contain sets (i.e., groups) of line segments
 4 while s ≠ nil do
 5     G = {s}, i.e., a group of segments (initially) containing only the seed
 6     c = cell of A from which s was taken
 7     foreach ungrouped segment r in S_c after s do
 8         Try to merge r with s. If successful, insert r into G and mark all
           its copies as grouped.
 9     foreach ungrouped segment r in E_c after s do
10         Try to merge r with s. If successful, insert r into G and mark all
           its copies as grouped.
11     Insert G into S.
12     Get longest ungrouped seed segment s in A and mark all its copies in
       A as grouped.
13 return S
```

To control the order in which the line segments are merged with a seed segment s in such a way that s will grow by bridging small gaps, every accumulator cell c keeps its line segments in two arrays S_c and E_c. For every segment s in c we compute a line interval by projecting s orthogonally onto the line ℓ_c that is defined with the parameters lying in the center of the parameter range represented by c. The array S_c keeps the segments in the order in which their intervals start on ℓ_c. Similarly, E_c is *inversely* ordered by interval ends, see Fig. 9. A segment t is merged with a seed segment s if the gap between s and t is small compared to the lengths of s and t. More precicely, we use the evaluation function $f(s,t) = (L(s)+L(t))/L(s) \cdot (L(s)+L(t))/L(M(s,t))$ defined by Scher et al. (1982), where $L(s)$ is the length of a segment s and $M(s,t)$ is the segment that results from merging two segments s and t. A merge of s and t is accepted if both $f(s,t)$ and $f(t,s)$ exceed a

user-set threshold $f_{\min}$.

4. Experimental Results

In this section we discuss experiments that were performed with the proposed symmetry detector. Section 4.1 describes the input data and the parameter setting and Section 4.2 gives an overview of the results. In Sect. 4.3 we discuss results on examples, which show that the symmetry detector indeed allows interesting groups of buildings to be found. The symmetries can be classified according to their cardinality, which allows hierarchies of symmetries to be found; this is discussed in Sect. 4.4. We conclude the section with the discussion of an empirical study in Sect. 4.5.

4.1. Experimental setup

The presented algorithms were implemented in C++ and tested for a dataset of 4553 building footprints of the metropolitan Boston area, which 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 redundant vertices that would hinder the matching process, the building footprints were simplified with the algorithm by Haunert and Wolff (2010) and an error tolerance ε of 0.1 m.

The symmetry detector requires two files as input, one file containing the building footprints and a second file containing edges, each of them connecting two buildings. For every single building and for every pair of buildings connected by an edge, the string-matching algorithm is applied.

[1] http://www.mass.gov/mgis/lidarbuildingfp2d.htm (accessed 2-3-2012)

parameter name	**symbol**	**value**
tolerance for building simplification	ε	0.1 m
distance threshold for neighborhoods of buildings	$D_{\max}$	50 m
tolerance for differences of angles	$\Delta\alpha_{\max}$	0.15 rad
tolerance for differences of edge lengths	$\Delta\ell_{\max}$	30%
minimum cardinality for basic symmetries	$k_{\min}$	5
maximum variance for basic symmetries	$s^2_{\max}$	2 m^2
size of intervals for angles (for Hough transform)	$\delta\alpha$	2°
size of intervals for distances (for Hough transform)	$\delta\ell$	2 m
parameter for segment grouping	$f_{\min}$	0.5

Table 1: Parameters used for the presented experiments.

Two methods for generating the file of edges were tested. These were implemented in a separate program that uses the CGAL[2] library for computational geometry. The first method simply generates an edge between two buildings if their distance is smaller than a user-set threshold $D_{\max}$. The second method computes a constrained Delaunay triangulation for the set of building footprints and defines an edge for each two buildings that are connected by a triangle edge. The advantage of this method is that it does not require a parameter to be set and that it guarantees the number of edges to be linear in the number of building footprints. On the downside, the triangulation-based method will fail to find a symmetry between two buildings if a third building lies between them. In the following, if not stated otherwise, we will discuss results obtained with the distance-based method, setting $D_{\max} = 50$ meters. With this value, both methods yielded very similar results.

Table 1 summarizes the parameters applied, which were found by experi-

[2]http://www.cgal.org/ (accessed 2-3-2012)

ments. Since the sequence of 90° and 270° turns of building outlines is often very characteristic, the tolerance for edge lengths is set to a relatively large value (30%) and to the relatively small value of 0.15 rad for angles. Setting $k_{\min} = 5$ for the string-matching algorithm implies that we find many small symmetries, most of them probably being uninteresting on their own. These small symmetries, however, may become grouped and thus contribute to more interesting non-basic symmetries. Basic symmetries of small cardinality that do not become grouped can still be rejected in a post-processing step. With the least-squares approach from Sect. 3.4, the residuals for non-basic symmetries were relatively large compared to the residuals for basic symmetries. Therefore, the variance for a non-basic symmetry was allowed to be $2s^2_{\max}$ with $s^2_{\max}$ being the maximally allowed variance for basic symmetries.

4.2. Overview of results

For the dataset of 4553 buildings, the symmetry detector needed 14 seconds on a Windows PC with 3 GB RAM and a 3.00 GHz Intel dual-core CPU. This time includes the string matching, the least-squares adjustment, and the grouping of symmetries. The program found 13095 basic reflections and 27584 basic repetitions, which means that, for every building, 2.9 basic reflections and 6.1 basic repetitions were found on average. Figure 10 shows the number of basic pure reflections for different cardinalities. About 50 percent of basic pure reflections have cardinality eight, which is due to the fact that the dataset contains many rectangular buildings (of which each has at least two reflections with four edges and four angles). Except for this special case, it can be observed that reflections of odd cardinality are generally more frequent than reflections of even cardinality. The reason for this somehow

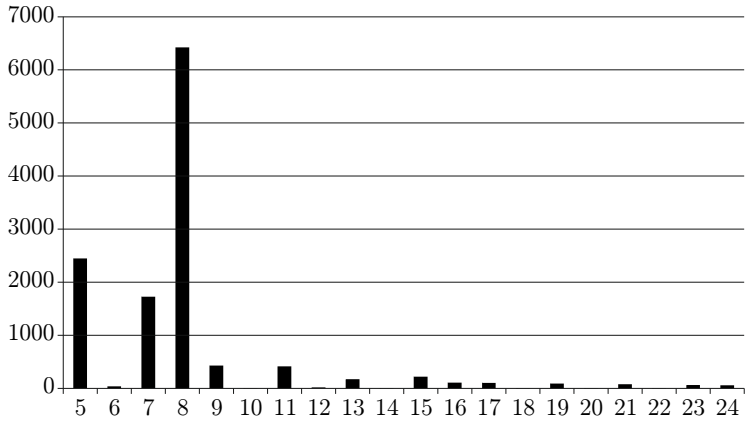


Figure 10: Histogram of cardinalities for basic pure reflections. Cardinalities greater 24 exist but are not shown here.

surprising fact is that for a basic pure reflection (p, q, T) it often holds that $p = q$ and that p is not a closed polygon. This implies that the strings corresponding to p and q either both start and end with an edge or that both start and end with an angle, thus each string contains an odd number of symbols. The histogram of cardinalities for non-basic pure reflections (see Fig. 11) shows that the cardinalities of basic reflections in the same group often sum up to multiples of eight. This is because the dataset contains many detached, rectangular houses that line up and have a common mirror axis.

Analyzing the set of basic repetitions found, it turned out that for a basic repetition (p, q, T), the angle of the rotation component of the transformation T is very likely to be 0°, 90°, or 180°. This is because buildings that share a common design are often aligned to the same axes.

4.3. Detecting building groups

We can use the symmetry detector for the grouping of buildings by analyzing the graph $G = (V, E)$ that contains a vertex for every building and an

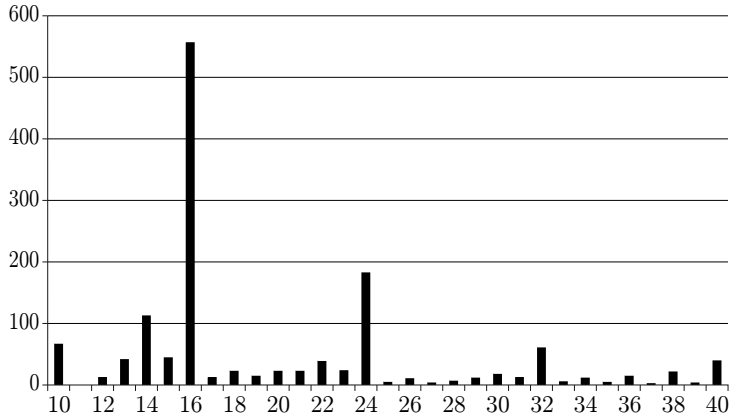


Figure 11: Histogram of cardinalities for non-basic pure reflections. Cardinalities greater 40 exist but are not shown here.

edge for every repetition found. More precisely, G is a multigraph, since there can be multiple repetitions for a pair of buildings, and G may contain loops, that is, edges connecting vertices with themselves. Every edge can be labeled with multiple attributes, for example, the cardinality of the corresponding repetition, its variance resulting from the least-squares adjustment, and the lengths of the polylines involved. These attributes may be condensed into a weight for each edge. Then, groups can be found by partitioning the weighted graph into components, for example, by removing edges of low weight.

In Fig. 12 we see that this approach already works well if we only consider the cardinality of a repetition. In the example, the repetitions with cardinality of at least eight define the edges of G, which has multiple connected components. Each connected component indeed contains buildings sharing a common design, for example, a stair-shaped or square-wave-shaped boundary. Figure 13 (left) shows that we find the group of six buildings from Fig. 2(a) with this method. For the example in Fig. 2(b) it is more interest-

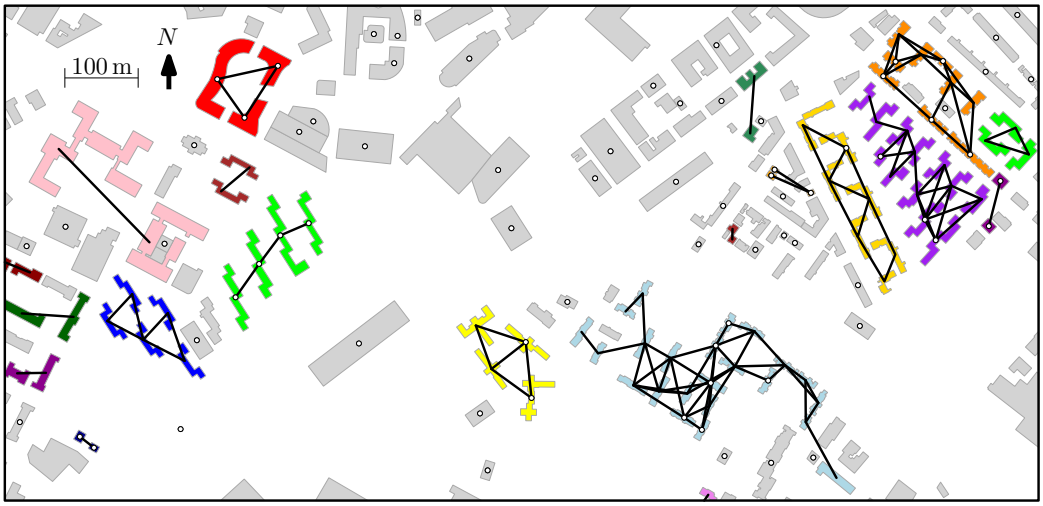


Figure 12: Repetitions with cardinality of at least eight, found for a part of the Boston dataset. The bold black links show for which pairs of buildings at least one repetition was found. A white dot stands for a repetition that involves two parts of the same building. Different colors show different connected components of the graph that contains an edge for each repetition.

ing to look at the detected reflections, which are shown in Fig. 13(right) for the triangulation-based neighborhood. We discuss the mirror axes yielded by the proposed method in more detail in the next section.

4.4. Discussion of the cardinality-based importance rating

Figure 14 shows a part of the Boston dataset with the mirror axes found. This figure shows that symmetry relations often form hierarchies and that the symmetry detector can reveal them. The maximum-cardinality mirror axis of the building labeled with (a), for example, is oriented towards Copley Square and thus structures the front and back view of the building, while a second mirror axes of lower cardinality structures the side views. Some mirror axes (blue) of very small cardinalities were found for the corners of the building. It may be interesting for some applications to distinguish such

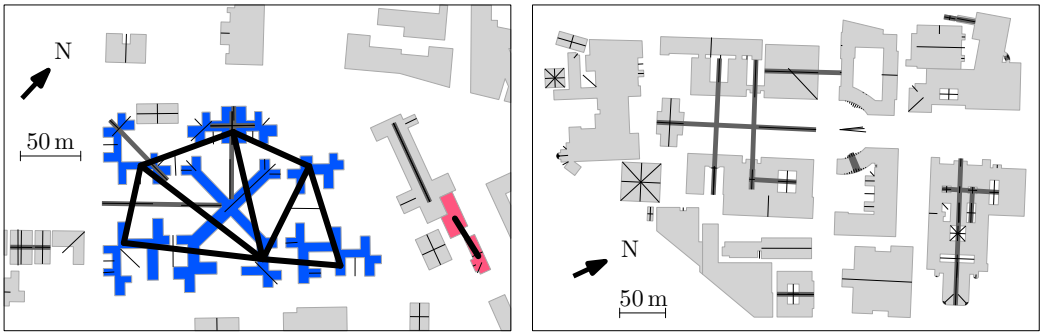


Figure 13: Results for the examples from Fig. 2. In the left example, repetitions between two distinct buildings are shown using the symbology from Fig. 12. Additionally, mirror axes for basic pure reflections (thin black lines) and mirror axes resulting from the grouping algorithm (bold gray lines) are shown. In the right example, one large connected component including almost all buildings was found. The width of the horizontal corridor between the buildings in the right example exceeds $D_{\max} = 50\,\mathrm{m}$, thus some of the symmetries shown were found only with the triangulation-based neighborhood.

"diagonal" mirror axes from mirror axes that are orthogonal to the edges of the building outline, since the latter axes are those that typically structure the front views of buildings and thus often have a high importance. This differentiation, however, is currently not done by the presented method, thus it (over)rates, for example, the mirror axis labeled with (b) as relatively important. Another problem of the cardinality-based weighting of mirror axes appears for polygons that contain small line segments representing circular ars, for example, for the multi-story car park (c), which has two round towers with spiral ramps. For such circles, many mirror axes of high cardinality are found. This result is correct, of course, but for most applications circles are perhaps not the most interesting structures we should search for. We could avoid too large weights for circular arcs by representing every circular arc in a polygon P as a single symbol in the corresponding string $X(P)$. This would require us, however, to detect arcs in a preprocessing step.

Finally, it is clear that the proposed method fails to find a symmetry if the geometric error tolerance is exceeded. For example, for the north facade of building (d), two mirror axes were found that are oriented in north-south direction (depicted blue). In contrast, no mirror axis was found for the south facade. In fact the mirror axes for the south facade were found by the string-matching algorithm, but in both cases the variance resulting from the least-squares adjustment was larger than the maximally allowed value of $2\,\mathrm{m}^2$; the variance was $3.5\,\mathrm{m}^2$ for the mirror axis of the south-west wing (d1) and $2.5\,\mathrm{m}^2$ for the mirror axis of the offset in the south facade (d2). In both cases, the geometric differences seem to be negligible when taking the very large size of the building into account. Therefore, it would be promising to define the maximally allowed variance depending on the size of the building. This, however, is currently not implemented in the method.

4.5. Comparison of automatically and manually generated results

In order to find out whether humans take symmetries into account when manually simplifying a shape and to test whether the symmetries found by the algorithm are indeed symmetries that humans would rate as important, a paper-based test was conducted, in which 30 students of age between 17 and 37 years were asked to manually solve three tasks. We now discuss this study in detail.

Task 1. The participants were asked to draw the set of six building footprints in Fig. 1(a) in a rectangle whose sides were by a factor of 0.5 shorter than the sides of the given drawing. They were asked to keep the drawing as legible as the original drawing and, to achieve this, were suggested to simplify shapes. The symmetry axes in Fig. 1(a) were not shown.

The results of Task 1 were assessed by counting for each of the four mirror

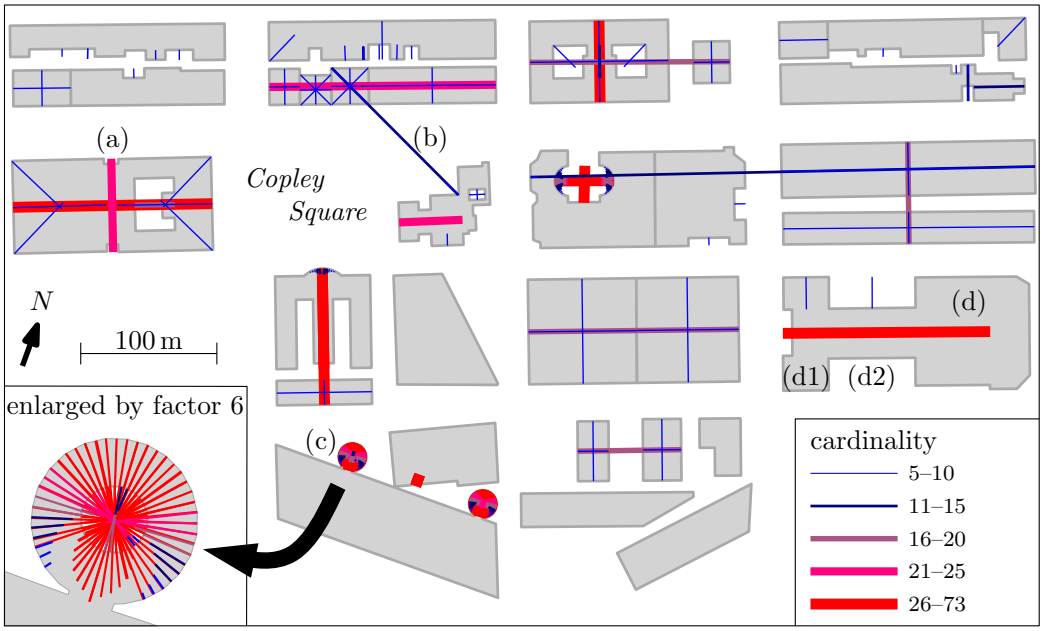


Figure 14: Mirror axes found for a part of the Boston dataset. The width and color of the mirror axes encode the cardinality of the symmetry.

axes in Fig. 1(a) how many of the 30 participants had simplified at least one of the buildings involved in the symmetry relation in a non-trivial way, that is, the simplified polygon contained more than four but less than the original number of vertices. For each non-trivial simplification it was assessed next whether or not the simplification was still symmetric with respect to the mirror axis. Table 2 summarizes the results: the overwhelming majority (81%) of all non-trivial simplifications preserved the symmetry relation. This allows us to conclude that symmetries are indeed important shape characteristics that should be considered in map generalization.

Task 2. Given the set of buildings in Fig. 12, the participants were asked to find the ten most salient groups. As the criterion for grouping, the participants were asked to use characteristic repetitions in the shapes. To avoid misunderstandings, one group was already marked in the given drawing.

mirror axis in Fig. 1(a)	**(1)**	**(2)**	**(3)**	**(4)**	**sum**
trivial solutions among all solutions	47%	23%	20%	47%	34%
non-trivial solutions among all solutions	53%	77%	80%	53%	66%
symmetry-preserving solutions among all non-trivial solutions	88%	91%	71%	75%	81%

Table 2: Results of Task 1 for each of the mirror axes in Fig. 1(a); the aggregated result (sum) means that 81% of all non-trivial simplification preserved the symmetry relation.

To assess the results of Task 2, for each pair (a, b) of buildings the number $N(a, b)$ of participants who had put a and b into a common group was determined. Figure 15 visualizes $N(a, b)$ for each two buildings a and b that, according to the distance-based neighborhood defined in Sect. 4.1, are adjacent. We observe that most of the groups in Fig. 12, which had been found by the algorithm, are also present in Fig. 15, that is, the buildings within the automatically found groups had also been grouped by many participants. The large group labeled with $*$ in Fig. 15, however, had not been found by the algorithm. It is likely that the participants had found this group based on other criteria than repetitive structures. One participant actually commented that he had identified that group based on proximity. To assess the results of Task 2 quantitatively, it was tested whether the number $N(a, b)$ and the maximum cardinality $C(a, b)$ of a match between two buildings a and b are correlated. As the result, a correlation coefficient of 0.61 was found, indicating that the automatically found groups are indeed similar to the groups the participants had drawn.

Task 3. Given the set of buildings in Fig. 14, the participants were asked to find the five most salient symmetry axes. They were explicitly asked to ignore small geometric differences and they were allowed to cover multiple buildings with one axis. One symmetry axis was already marked in the given drawing.

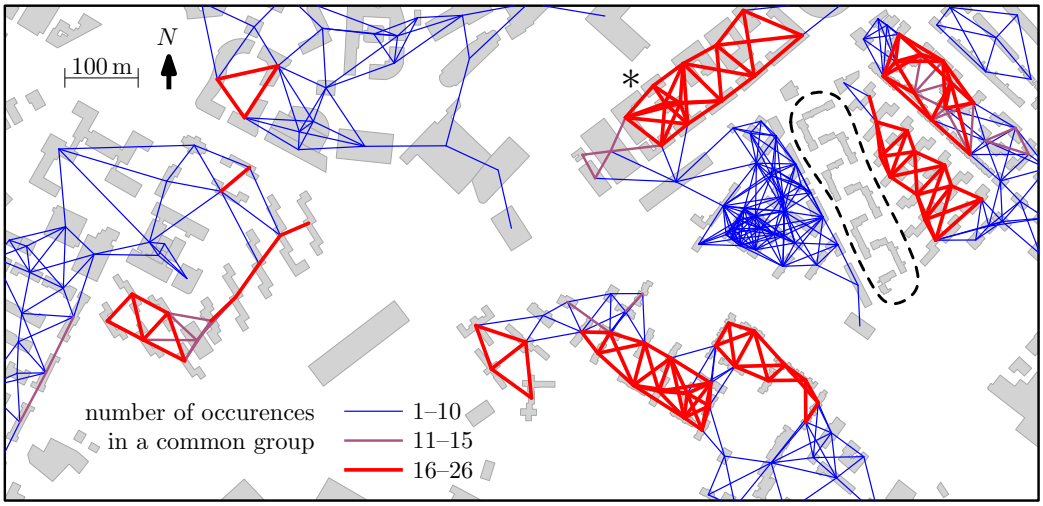


Figure 15: Results of Task 2. Edges connecting two buildings are classified according to the number of participants who put both buildings into a common group. The group that was given to the participants is displayed with a dashed outline.

Figure 16 visualizes the results of Task 3. Each mirror axis a that had been found by at least one participant is shown as a line whose color and width represents the number $N(a)$ of participants who had drawn this mirror axis. We can compute the probability that a axis mirror selected by a human is also found by the algorithm as $\sum_{a \in A_{\text{human}} \cap A_{\text{alg}}} N(a) / \sum_{a \in A_{\text{human}}} N(a)$, where A_{human} is the set of mirror axes found by the participants and A_{alg} the set of mirror axes found by the algorithm (recall Fig. 14); this probability turns out to be $159/184 = 0.86$. Note that some of the 30 participants had drawn more than five mirror axes, thus $\sum_{a \in A_{\text{human}}} N(a) = 184 > 30 \cdot 5$. The result of this assessment allows us to conclude that with high probability the important symmetries are found by the algorithm. On the other hand, there are many unimportant mirror axes in the result of the algorithm. In particular, among the 150 symmetries of highest cardinality in Fig. 14, 138 symmetries result

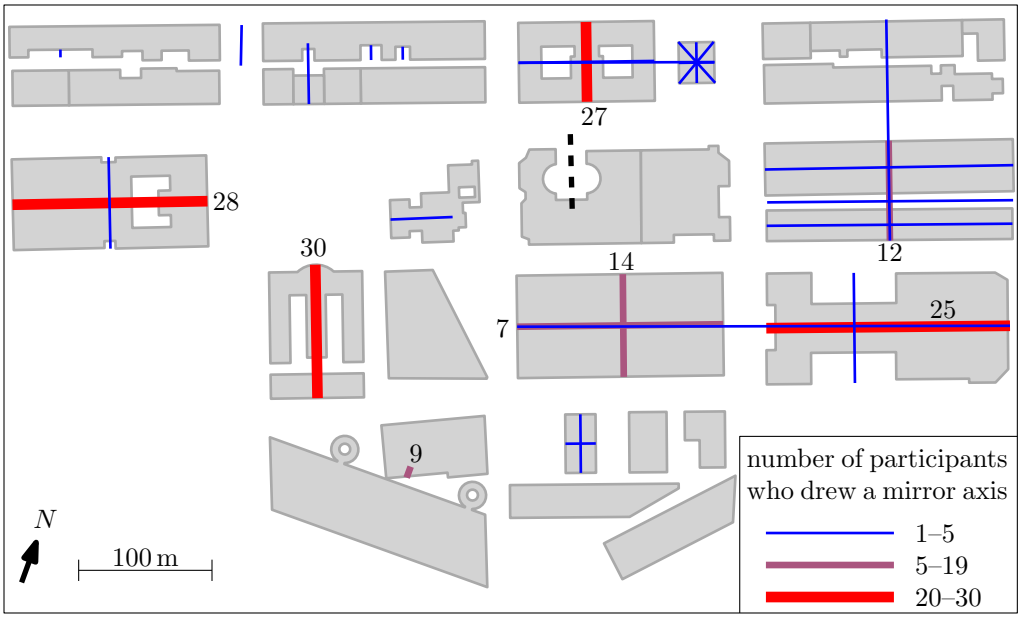


Figure 16: Mirror axes that participants drew in Task 3. The mirror axis displayed with a dashed line was given to the participants. The number of participants who drew a mirror axis is shown if it exceeds four.

from circular arcs that are approximated by short straight-line segments. If we neglect these problematic cases, however, then each of the five symmetries of highest cardinality in Fig. 14 had also been selected by a large number of participants.

5. Conclusion

We have discussed the problem of finding symmetry relations in geospatial datasets of buildings. This problem is important for the solution of map generalization problems, the grouping of buildings, landmark detection, and building classification. The presented algorithm for symmetry detection combines a very efficient string-matching approach based on dynamic programming, a least-squares approach based on a singular value decomposition,

and a grouping method based on a Hough transform. The algorithm copes with both 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.

We have seen that the string-matching algorithm detects many small symmetry relations. Many of such basic symmetries, for example, reflectional symmetries for rectangular or circular buildings, are rather uninteresting. Therefore, additional effort was spent to group symmetries and it was proposed to rate their importance based on their cardinality, that is, the number of angles and edges of the matches found. This approach yielded meaningful groups of buildings and important mirror axes in a building dataset of Boston, which are similar to the groups and mirror axes that humans had found.

A drawback of the presented method is that multiple parameters have to be set. In particular, the building footprints have to be generalized with a carefully chosen error tolerance (which may depend on the application) before the proposed symmetry detector can be applied. On the other hand, symmetries need to be known to ensure a good solution to the map generalization problem. It is interesting to develop algorithms for map generalization that exploit known symmetry relations, but, since symmetry detection and map generalization depend each other, we should also strive for integrated approaches to both problems. A feasible idea could be to find symmetries in a building's shortcut graph (Haunert and Wolff, 2010) that defines all possible simplifications of a building and to select shortcuts (and thereby the generalized building) under consideration of this information.

References

Arun, K. S., Huang, T. S., Blostein, S. D., 1987. Least-squares fitting of two 3-d point sets. IEEE Transactions on Pattern Analysis and Machine Intelligence 9 (5), 698–700.

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

Cho, M., Lee, K. M., 2009. Bilateral symmetry detection via symmetry-growing. In: Proc. British Machine Vision Conference (BMVC '09). BMVA, pp. 4.1–4.11.

Christophe, S., Ruas, A., 2002. Detecting building alignments for generalisation purposes. In: Advances in Spatial Data Handling – Proc. 10th International Symposium on Spatial Data Handling. Springer-Verlag, Berlin, Germany, pp. 419–432.

Damen, J., van Kreveld, M., Spaan, B., 2008. High quality building generalization by extending the morphological operators. In: Proc. 11th ICA Workshop on Generalisation and Multiple Representation.
URL `http://aci.ign.fr/montpellier2008/papers/04_Damen_et_al.pdf`

Gaffuri, J., Trévisan, J., 2004. Role of urban patterns for building generalisation: An application of agent. In: Proc. 7th ICA Workshop on Generalisation and Multiple Representation.
URL `http://aci.ign.fr/Leicester/paper/Gaffuri-v2-ICAWorkshop.pdf`

Garey, M. R., Johnson, D. S., 1990. Computers and Intractability; A Guide

to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA.

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

Haunert, J.-H., 2011. Detecting symmetries in building footprints by string matching. In: Advancing Geoinformation Science for a Changing World – Proc. 14th AGILE International Conference on Geographic Information Science. Lecture Notes in Geoinformation and Cartography. Springer-Verlag, Berlin, Germany, pp. 319–336.

Haunert, J.-H., Wolff, A., 2010. Optimal and topologically safe simplification of building footprints. In: Proc. 18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM-GIS'10). ACM, pp. 192–201.

Heinzle, F., Anders, K.-H., 2007. Characterising space via pattern recognition techniques: Identifying patterns in road networks. In: Mackaness, W., Ruas, A., Sarjakoski, T. L. (Eds.), Generalisation of geographic information: Cartographic modelling and applications. Elsevier, Oxford, UK, Ch. 12, pp. 233–254.

Hillier, B., Hanson, J., 1984. The Social Logic of Space. Cambridge University Press, Cambridge, UK.

Hough, P. V. C., 1962. Method and means for recognizing complex patterns. U.S. Patent 3069654.

Jiang, B., Claramunt, C., 2002. Integration of space syntax into GIS. Transactions in GIS 6 (3), 295–309.

Kada, M., Luo, F., 2006. Generalisation of building ground plans using half-spaces. In: Proc. ISPRS Commission IV Symposium on Geospatial Databases for Sustainable Development. Vol. 36 (part 4) of International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. ISPRS, on CD-ROM.

Knuth, D. E., J. H. Morris, J., Pratt, V. R., 1977. Fast pattern matching in strings. Siam Journal on Computing 6 (2), 323–350.

Kraus, K., 2004. Photogrammetry – Geometry from Images and Laser Scans, 2nd Edition. de Gruyter, Berlin, Germany.

Leduc, T., Chaillou, F., Ouard, T., 2011. Towards a "typification" of the pedestrian urban space: analysis of the isovist using digital processing method. In: Advancing Geoinformation Science for a Changing World – Proc. 14th AGILE International Conference on Geographic Information Science. Lecture Notes in Geoinformation and Cartography. Springer-Verlag, Berlin, Germany, pp. 275–292.

Lladós, J., Bunke, H., Martí, E., 1997. Using cyclic string matching to find rotational and reflectional symmetries in shapes. In: Intelligent Robots: Sensing, Modeling and Planning. Vol. 27 of Series in Machine Perception and Artificial Intelligence. World Scientific, pp. 164 – 179.

Loy, G., Eklundh, J.-O., 2006. Detecting symmetry and symmetric constellations of features. In: Proc. 9th European Conference on Computer Vision

(ECCV '06), Part II. Vol. 3952 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, pp. 508–521.

Machilsen, B., Pauwels, M., Johan, W., 2009. The role of vertical mirror symmetry in visual shape detection. Journal of Vision 9 (12), 1–11.

Mackaness, W., Edwards, G., 2002. The importance of modelling pattern and structure in automated map generalisation. In: Proc. Joint ISPRS/ICA Workshop on Multi-Scale Representations of Spatial Data.
URL http://www.ikg.uni-hannover.de/isprs/workshop/macedwards.pdf

Mayer, H., 1998. Model-generalization of building outlines based on scale-spaces and scale-space events. In: Proc. ISPRS Commission III Symposium on Object Recognition and Scene Classification from Multispatial and Multisensor Pixels. Vol. 37 (part 3) of International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. ISPRS, pp. 530–536.

Mitra, N. J., Guibas, L. J., Pauly, M., 2006. Partial and approximate symmetry detection for 3d geometry. ACM Transactions on Graphics 25 (3), 560–568.

Neun, M., Burghardt, D., Weibel, R., 2008. Web service approaches for providing enriched data structures to generalisation operators. International Journal of Geographical Information Science 22 (2), 133–165.

Park, M., Lee, S., Chen, P.-C., Kashyap, S., Butt, A. A., 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).
URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4587824

Peters, D., Wu, Y. H., Winter, S., 2010. Testing landmark identification theories in virtual environments. In: Spatial Cognition. Vol. 6222 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, pp. 54–69.

Regnauld, N., 2001. Contextual building typification in automated map generalization. Algorithmica 30 (2), 312–333.

Regnauld, N., 2003. Algorithms for the amalgamation of topographic data. In: Proc. 21st International Cartographic Conference (ICC '03). ICA, pp. 222–234.

Ruas, A., Holzapfel, F., 2003. Automatic characterization of building alignments by means of expert knowledge. In: Proc. 21st International Cartographic Conference (ICC '03). ICA, pp. 1604–1616.

Scher, A., Shneier, M., Rosenfeld, A., 1982. Clustering of collinear line segments. Pattern Recognition 15 (2), 85–91.

Sester, M., 2005. Optimization approaches for generalization and data abstraction. International Journal of Geographical Information Science 19 (8–9), 871–897.

Steiniger, S., 2007. Enabling pattern-aware automated map generalization. Phd thesis, University of Zürich.

Steiniger, S., Burghardt, D., Lange, T., 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., Brooks, R., 2002. Exploiting perceptual grouping for map analysis, understanding and generalization: The case of road and river networks. In: Proc. 4th International Workshop on Graphics Recognition Algorithms and Application (GREC '01). Vol. 2390 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, pp. 148–157.

Treder, M. S., 2010. Behind the looking-glass: A review on human symmetry perception. Symmetry 2 (3), 1510–1543.

Wagemans, J., 1994. Detection of visual symmetries. Spatial Vision 9 (1), 9–32.

Werder, S., Kieler, B., 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). ACM, pp. 330–339.

Wertheimer, M., 1938. Laws of organization in percetional forms. In: A source book of Gestalt psychology. Routledge & Kegan Paul, London, UK, pp. 71–88.

Wiener, J. M., Franz, G., 2004. Isovists as a means to predict spatial experience and behavior. In: Spatial Cognition. Vol. 3343 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, pp. 42–57.

Wolter, J. D., Woo, T. C., Volz, R. A., 1985. Optimal algorithms for symmetry detection in two and three dimensions. The Visual Computer 1 (1), 37–48.

Yang, W., 2008. Identify building pattens. In: Proc. XXIst ISPRS Congress. Vol. 37 (part B2) of International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. ISPRS, pp. 391–397.

Yang, X., Adluru, N., Latecki, L. J., Bai, X., Pizlo, Z., 2008. Symmetry of shapes via self-similarity. In: Proc. 4th International Symposium on Advances in Visual Computing, Part II. Vol. 5359 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, pp. 561–570.