Diplomarbeit

Generalization of building footprints derived from high resolution remote sensing data

Ausgeführt am

Institut für Photogrammetrie und Fernerkundung,
Technische Universität Wien

unter der Anleitung von

Univ.Prof. Dipl.-Ing. Dr.techn. Norbert Pfeifer
und
Dipl.-Ing. Dr.techn. Markus Hollaus

durch

Marieke Dutter
Obere Augartenstraße 72/14
A-1020 Wien

Wien, im März 2007 .....

Contents

Abstract

This work describes the generalization of two-dimensional building outlines. The building outlines are derived through image segmentation and classification processes from aerial images and consist of small line segments and many redundant points. The generalized building outlines are dedicated to serve as a base for the three dimensional modeling of buildings. Therefore simple geometric shapes are required. For this purpose a model driven method for generalization of two-dimensional building outlines has been developed. The basic idea is to fit a limited count of building primitives (a rectangle, the "L"-, "T"- and "Z"-form, as well as a combination of those structures) to the raw building outline, beginning with the simplest model, and after a quality check, proceeding with more complex models if necessary. This method has been implemented as a tool in ArcGIS. Due to the restrictions in the building primitives, not all buildings from the input data can be generalized automatically. Therefore an editing tool has been integrated in the generalization tool. It provides a simple assistance to the user in the manual post-processing.

Zusammenfassung

Diese Arbeit beschäftigt sich mit der Generalisierung zweidimensionaler Gebäudegrundrisse. Die Gebäudegrundrisse stammen aus einer Bildsegmentierung und -klassifizierung von Luftbildern und bestehen folglich aus kleinen Liniensegmenten und vielen redundanten Punkten. Die generalisierten Gebäude können als Basis für die dreidimensionale Gebäudemodellierung dienen und weisen aus diesem Grund einfache geometrischen Formen auf. Für diesen Zweck wurde eine modellgetriebene Methode zur Generalisierung zweidimensionaler Gebäudegrundrisse entwickelt. Ihr liegt die Idee zugrunde, den abgeleiteten Gebäudegrundrissen eine beschränkte Anzahl Gebäudeprimitive (Rechteck, "L"-, "T"- und "Z"-Form, sowie eine Kombination aus diesen) einzupassen, wobei mit dem einfachsten Modell begonnen wird und nach

einer Qualitätskontrolle bei Bedarf mit komplexeren Modellen fortgefahren wird. Diese Methode wurde in einem Tool in ArcGIS implementiert. Aufgrund der Beschränkungen in der Gebäudeform durch die Modelle können nicht alle Gebäude automatisch generalisiert werden. Deshalb wurde ein Editierwerkzeug, das den Benutzer durch den Editiervorgang führt, in das Generalisierungsprogramm integriert.

Acknowledgements

I would like to thank Dr. Markus Hollaus for his continuous support in professional and programming aspects and for the confiding positive working atmosphere, which enabled this thesis.
Further I would like to thank Prof. Dr. Norbert Pfeifer for his helping support and inspiring discussions, which provided me with a wider perspective on this work.

Chapter 1

Introduction

Two-dimensional building outlines are needed in many disciplines dealing with spatial data, as for example cartography and the field of GIS. Building footprints also serve as a base for computation of three-dimensional building models. They can be derived from classified height data (LIDAR data) or from image segmentation. In both cases the result of the classification process is very often a set of pixel-areas (so-called *blobs*). After vectorization of the blobs, the buildings are represented as polygons, which have a noisy outline. In most cases they are not adequate to serve as a base for the final application. Therefore, generalization of the raw building outline is essential. Manual generalization is time-intensive and is more and more replaced by automatic or semiautomatic methods. This leads to a more transparent and efficient work flow, from the data capture to the final visualization. Of course the use of automatic methods reduces time consumption and increases cost efficiency. Furthermore, the objectivity of the generalization process is enhanced since the generalization result can be reproduced.

1.1 Objectives

The generation of building outlines results in most cases in noisy polygons containing small structures, that do not represent the real boundary. That's

why a generalization of the raw building outline is essential. The aim of the work is the development of a generalization method and the implementation of the method in a tool that should be ready for use. The generalized building outlines are needed for visualization purposes and three-dimensional building modeling. There is no need of fine details of the building outline. In contrary, the focus is set on the acquisition of the inherent building structure. This should be realized in two ways: On the one hand the generalized buildings should consist of only rectangular angles. On the other hand the complexity of the generalized buildings should be limited by providing only a small amount of different building primitives that are fitted to the raw building polygons. Furthermore, particular emphasis should be laid on a high degree of automation. But since obviously not all buildings can be represented by generalized polygons that fulfill the conditions mentioned above (because the buildings consist of a more complex structure), the user has to edit manually the badly generalized buildings. In order to support the editing process, an appropriate tool should be developed and integrated into the generalization tool. The tool should be embedded into a well-known GIS environment (ArcGIS[1]), which provides all GIS-functionality for the user and at the same time the integrated VBA (Visual Basics for Applications) software development environment. Another benefit in the use of ArcGIS is the integration of both the automatic generalization and the manual editing process in one software. This fact frees the user from another data import and export procedure and thus reduces time consumption and possible errors.

1.2 Structure of the work

Section 2 starts with general descriptors for building boundaries and areas. It gives an overview on line generalization algorithms. In the last part of the chapter, a selection of generalization algorithms, developed in past and present days specifically for buildings, are presented.
In section 3, details on the algorithm, that has been developed and imple-

[1]ESRI ArcGIS Desktop 9.1

mented for this thesis, are described.
The practical use of the implemented generalization tool is presented in section 4. Furthermore, the work flow is described and examples of the generalization results are shown. Finally, the application of the manual editing tool is described in this section.
The generalization result is validated in section 5. It starts with the presentation of three quality measures, which describe the fitting quality of the generalized buildings. The chapter goes on with the application of those measures to the realized generalization result.
Based on the findings of the previous chapter, the last chapter (section 6) shows and discusses limitations of the presented algorithm and gives proposals for possible enhancement. Furthermore, a short outlook on other techniques, that could possibly be used for building generalization, is given.
In the appendix the generalization result, achieved with the test data, is fully documented in 12 tiles.

Chapter 2

State of the art

The term *generalization*, as used in the title of this work, has some aspects in common with the term *generalization*, as it is used in cartography. In order to distinguish between the different usages of the term, this chapter starts with a brief definition of cartographic generalization. Afterwards it will be presented, which aspect of cartographic generalization this work concentrates on. The way algorithms deal with data depends to a great extent on the way the data is stored. For this reason a selection of possible representations of boundaries and regions is presented. Moreover describing boundaries and regions provides information about geometric characteristics of the objects. Some descriptive parameters are presented in this chapter. Line generalization algorithms do not consider building characteristics and constraints on the shape that should represent a building. For this reason they are not a solution for the building generalization problem. Nevertheless, they often contribute to the building generalization process, and therefore, a selection of line generalization algorithms is presented in this chapter. The diversity of building generalization methods present in the literature can not be presented in this context in a satisfying way. A selection of them is described in the last subsection of this chapter.

2.1 Cartographic generalization

In the work of Weibel and Jones (Weibel and Jones 1998) the following definition of cartographic generalization can be found:

> [...] map generalization (or simply generalization) is the process of deriving from a detailed source spatial database a map or database, the contents and complexity of which are reduced, while retaining the major semantic and structural characteristics of the source data appropriate to a required purpose. (Weibel and Jones 1998)

A classic example would be the derivation of a topographic map at 1 : 100 000 from a source map of 1 : 50 000. However, map generalization should not be reduced to scale reduction:

> [...] it represents a process of informed extraction and emphasis of the essential while suppressing the unimportant, maintaining logical and unambiguous relations between map objects, maintaining legibility of the map image, and preserving accuracy as far as possible. (Weibel and Jones 1998)

Changing the scale of a map causes several visualization problems. Some objects may be to small and would not be readable any more by the user. The distance between objects may be to short and therefore neighbored objects could not be visually distinguished any more. Dealing with those problems leads to the need of cartographic generalization.
The generalization process is based on general rules. But it has to allow local, individual modifications of the rules, in order to preserve the characteristics of a situation in the generalized data set (Kelnhofer 2001).

Generalization strategies can be divided into two groups:

- Geometric generalization and
- geometric-semantic generalization.

Geometric generalization of an object means *simplification*, *enlargement* and *displacement* of the object. For example a building outline has to be simplified, because details would no longer be readable. An object has to be enlarged, if its dimensions are less than required for readability. It has to be displaced, if the distance to the neighboring object is to small.

Geometric-semantic generalization means *selection*, *aggregation*, *classification* and *exaggeration*. If, for example, the interspace between detached houses in rural area is too small, we have to select some of them to remain in the generalized data set. In that way the characteristics of individual houses are preserved. However a similar situation in urban area could be classified as closed building area and therefore the buildings would be merged into a block of houses.

Obviously the task of generalization can not be performed by applying generalization rules sequentially. Rather it is a complex process where particular sub tasks impact upon others. Manual generalization requires knowledge about the generalization rules, a precise idea of the information that should be transported by the medium and a lot of experience. Automated digital generalization has been a field of research for many years, but totally automated generalization remains still a difficult task and is a work in progress (Meng 2001).

In digital cartography and GIS two branches of map generalization can be distinguished: cartographic generalization and database (or model) generalization (Weibel and Jones 1998).
Cartographic generalization is the process of deriving a visualization from a source database. It has to deal with the specific problems of graphical symbolization and comprises the generalization methods mentioned above. The aim is to produce a readable image and hence a message that can be clearly decoded.
Database (or model) generalization concentrates on the derivation of reduced databases from source databases. The aim is to save data storage, to increase

computational efficiency and to derive data sets of reduced accuracy and/or resolution. It can be a preprocessing step to cartographic generalization. Database generalization does not produce graphic output, but output that meets the accuracy specifications of the target database. Processes involved in database generalization can be formalized more easily than cartographic generalization.

Building generalization, in the way it is explored in this thesis, is among database (or model) generalization. It focuses on detecting the main characteristics of a building in the raw building outline data, that comes out from the segmentation and classification process. This leads at the same time to a reduction of points representing the building outline. In many cases the aim is not a change in scale, but a "cleaning" and regularization (which means e.g. the introduction of rectangular angles) of the outline. This is a prerequisite of storing the building outlines in a database for a further processing.

2.2 Description of boundaries and regions

2.2.1 Representation of boundaries and regions

Boundaries and regions can be represented in numerous ways. Some examples are presented in the following.

The coordinates of the boundary points can be stored in a *coordinate list*. The order of the points in the list is the same as they are encountered by traversing the boundary (for example counterclockwise).
A boundary can be represented by registering the *slope* of each line segment and the length of that segment. In other words the slope is a function of the distance. This line representation carries information about basic shape characteristics. For example straight line parts can be identified by horizontal lines in the slope functions. This descriptor has to be used with care, especially if the boundary comes from a raster data vectorization. It is not possible to make a statement about the overall shape of the boundary with-

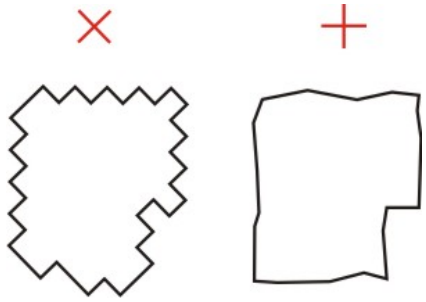


Figure 2.1: In boundaries coming from vectorized raster data, only four values for the slope can be found (indicated by the left cross for the left polygon). The cross in the right image indicates four mean values for the slope in the right polygon.

out further processing of the slope representation, because only four different values for the slope exist! In general the four directions are completely different from the approximated orientations of the boundary segments when neighboring points are considered (see example in figure 2.1).

Chain code is often used to represent the boundary of a pixel area and is a specialization of the representation described above. It is a connected sequence of straight-line segments of specified length and direction (Gonzalez and Woods 2003). The 4-directional chain code considers only four directions and a single length of a segment. The 8-directional chain code considers 8 directions an two different values for the length of a segment.

Coordinate pairs of boundary points can be treated as *complex numbers*:

$$s(k) = x(k) + iy(k) \tag{2.1}$$

where s is an arbitrary point of the boundary with the coordinates (x_k, y_k) for $k = 0, 1, 2, \ldots, K-1$. K is the count of points. The x-axis is treated as the real axis and the y-axis as the imaginary axis of a sequence of complex numbers.

A curve can be represented by its *fourier coefficients*, each coefficient being related to a spatial frequency.

A compact representation of a binary image (a region) is the *run-length code* (Jähne 2004). The image is "scanned" line by line and the count of pixels in a sequence of equal pixels is registered together with the value of

the pixels. Since the representation of an area requires only two pixel values (zero and one) and a sequence of zero-pixels is always followed by a sequence of one-pixels, there is no need to store the pixel value. Only the count of pixels in a sequence of equal pixels is stored.

2.2.2 Description of closed boundaries

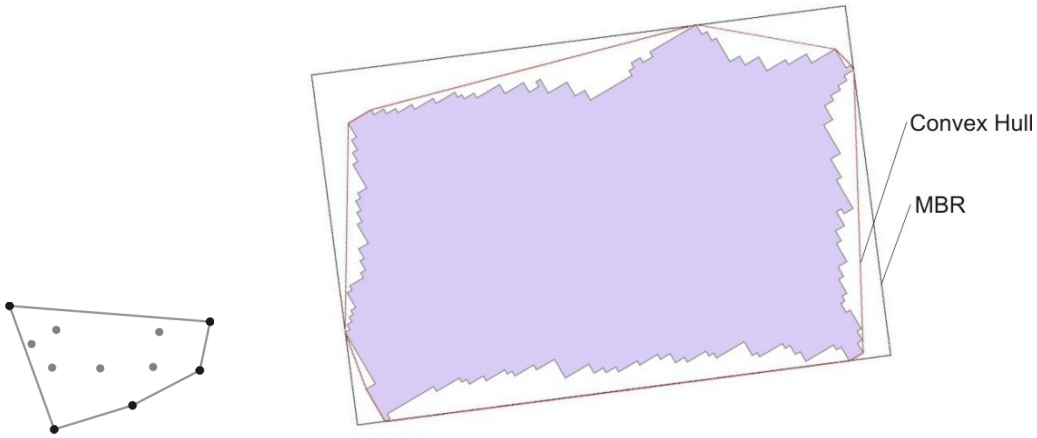
The main shape characteristics of a boundary are easily perceptible by human operators. In order to formalize them (thus to make them accessible to a potential generalization algorithm), it is necessary to find useful descriptions for boundaries. Some descriptors are presented in this section (Gonzalez and Woods 2003):

Scalar features

The simplest descriptor is the *length* of the boundary.
The *diameter* of a boundary is defined as the maximum distance between two boundary points:

$$\text{Diam}(B) = \max[D(p_i, p_j)] \tag{2.2}$$

where D is a distance measure and p_i and p_j are points on the boundary.
The line connecting the two points p_i and p_j, that comprises the diameter, is called the *major axis*.
The line being perpendicular to the major axis and having such a length, that the rectangle formed by both axes encloses the boundary and touches the boundary in four points, is the *minor axis*.
The box just described is called *basic rectangle* of the boundary. The ratio of the major to the minor axis is called *eccentricity* of the boundary.
Curvature is defined as the rate of change of slope in a point of the boundary. It can be used to detect straight parts or corner points. For example the change in a point of 10 % indicates, that the point is part of a nearly straight segment. It may be a corner point, if the change exceeds 90 %. As described for the slope-descriptor it has to be used with care with noisy data



(a) The convex hull (black points) of a set of points.

(b) Convex hull and MBR: The MBR has one side collinear to a side of the convex hull.

Figure 2.2: Convex hull and MBR of a set of points and of a polygon

or vectorized raster data. In order to provide meaningful results, it has to be interpretated in relation to the length of the boundary segments.

Convex hull

Given a set P of n points, the convex hull is defined as the smallest convex set containing P (Chan 1996). Several algorithms for computing the convex hull have been developed (e.g. *Jarvi's march* or *Graham's scan*). Let's consider the convex hull of the set of vertices of a polygon. It is the smallest convex polygon, that contains the original polygon. See figure 2.2(a) for a convex hull of a set of points, and figure 2.2(b) for the convex hull of a polygon.

Minimum bounding rectangle (MBR)

The *minimum bounding rectangle* of a closed polygon completely encloses the polygon and has the smallest area. One method of calculating the MBR is presented by (Freeman and Shapira 1975). The MBR of a polygon is the same rectangle as the MBR of the convex hull of the polygon. Furthermore, the MBR of a convex polygon has one side collinear with one of the edges of

the convex polygon. Therefore, the convex hull must be calculated first. By iterating through all edges of the convex hull and computing the enclosing rectangle that is oriented parallel to the actual edge, the smallest rectangle is found. This is the minimum bounding rectangle (see figure 2.2(b) for an example of an MBR).
As it will be detailed in section 3, the MBR can be used for the detection of the main orientation of a building.

Fourier Descriptors

Having reduced the boundary from a 2-D representation to a 1-D representation by treating the coordinates of the boundary's points as complex numbers $s(k)$ (see eq. 2.1), the complex fourier coefficients (or fourier descriptors) of the boundary can be calculated as follows:

$$a(u) = \frac{1}{K} \sum_{k=0}^{K-1} s(k) e^{-i2\pi uk/K} \tag{2.3}$$

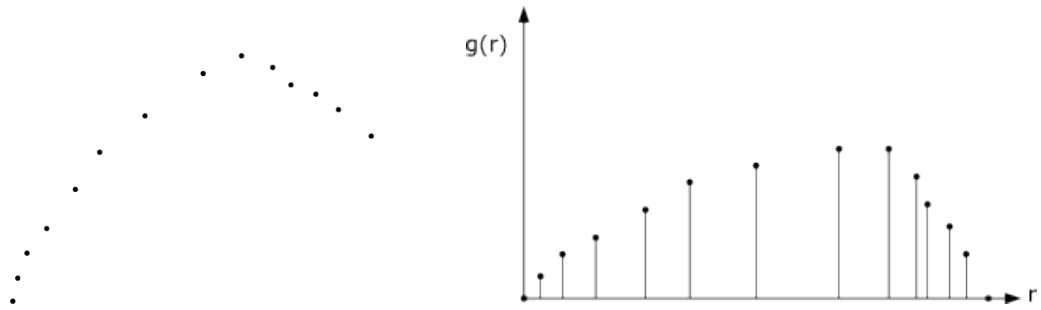
for $u = 0, 1, 2, \ldots, K-1$. It is the Discrete Fourier Transform (DFT) in the complex form.
By means of the coefficients, the boundary can be fully reconstructed (inverse Fourier transform):

$$s(k) = \sum_{u=0}^{K-1} a(u)^{i2\pi uk/K} \tag{2.4}$$

The high-frequency components (thus the last coefficients) account for fine detail and low-frequency components determine the global shape. This means that using only some of the first coefficients for reconstruction of the boundary results in smoothing the boundary.

Statistical Moments

The shape of boundary segments can be described quantitatively by using simple statistical moments. See figure 2.3 (a) and (b) for a simple example. The boundary is represented as a 1-D function $g(r)$ of an arbitrary variable



(a) Points of a boundary segment

(b) Boundary segment represented as a 1-D function $g(r)$

Figure 2.3: Statistical moments of a boundary segment

r. The distances from the boundary points to a reference line (e.g. the line connecting the first and last point) are treated as realizations of a discrete random variable v. From these values statistical moments such as the mean, the variance and higher-order moments, can be calculated. To achieve valid results, the length of the boundary segments has to be taken into account. That means that boundary points have to be equally distributed along the boundary. As it will be detailed in section 3, the standard deviation of the random variable v is used by the developed generalization method for the quality check of the fitting of a straight line segment.
Another approach is to normalize $g(r)$ to unit area and to consider it to represent a histogram. This means, that $g(r_i)$ is now treated as the probability of the occurring of value r_i. In this case, r is treated as the random variable and the statistical moments provide information about the shape. For example, the second moment $\mu_2(r)$ measures the spread of the curve about the mean value of r, and the third moment $\mu_3(r)$ measures the symmetry with reference to the mean.

2.2.3 Description of regions

Scalar features

The *area* of a region is defined as the number of pixels in the region multiplied by the area of one pixel. If the region is represented by a boundary vector

polygon, then for example the Gaussian trapezoidal formula (Gruber 2001) can be used to calculate the area:

$$A = \frac{1}{2}\sum_{i=1}^{n}(y_i + y_{i+1})(x_i - x_{i+1}) \tag{2.5}$$

The *perimeter* P of a region is the length of its boundary.
A very useful descriptor for the *compactness* of a region is:

$$c = \frac{P^2}{A} \tag{2.6}$$

where P is the perimeter and A the area of the region. It is a minimum for a disc-shaped region ($c = 4\pi$ for a perfect circle). Another useful formula describing compactness of an arbitrary formed region is the *Forman-Godron-Formindex* FI (Meng 2001):

$$FI = \frac{P}{2\sqrt{\pi A}} \tag{2.7}$$

The minimal value is 1 ($FI = 1$ for perfect circles). $(FI)^2$ is proportional to c.

Moment invariants

A statistical method used in object recognition and automated feature extraction is the calculation of moment invariants. Moment invariants provide characteristics of an object that uniquely represent its shape. They are calculated from statistical moments of a function. If $f(x, y)$ is a digital image (e.g. a value from 0 to 255 is mapped to every pair (x_i, y_i)), the *moment* of order $(p + q)$ is defined as

$$m_{pq} = \sum_{x}\sum_{y} x^p y^q f(x, y). \tag{2.8}$$

The *central moments* are defined as

$$\mu_{pq} = \sum_x \sum_y (x - \overline{x})^p (y - \overline{y})^q f(x, y) \tag{2.9}$$

where

$$\overline{x} = \frac{m_{10}}{m_{00}} \qquad \text{and} \qquad \overline{y} = \frac{m_{01}}{m_{00}}$$

are the mean of the x- and y-coordinates. The central moments are invariant to translation since the mean is always subtracted.
The central moments can be normalized using the following formula:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \tag{2.10}$$

where

$$\gamma = \frac{p+q}{2} + 1$$

for $p + q = 2, 3, \ldots$ These are the *normalized central moments*. They are dimensionless, thus invariant to scale change.
Finally a set of seven *invariant moments* can be derived:

$$\begin{aligned}
\phi_1 &= \eta_{20} + \eta_{02} && (2.11)\\
\phi_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 && (2.12)\\
\phi_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 && (2.13)\\
\phi_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 && (2.14)\\
\phi_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\
&\quad (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] && (2.15)\\
\phi_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + \\
&\quad 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) && (2.16)\\
\phi_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\
&\quad (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] && (2.17)
\end{aligned}$$

These moments are invariant to translation, rotation and scale change.
Moment invariants are a very good feature to use when dealing with par-

ticular types of shapes such as aircrafts or alphanumeric characters. In the work of Keyes and Winstanley (Keyes and Winstanley 2001) calculation of moment invariants was applied to topographic data like buildings, parcels and roads represented by closed polygons. The aim of their work was to investigate the usefulness of moment invariants for the identification of general shapes on maps. For each object that had to be classified, a set of moment invariants was calculated. The set of moment invariants was treated as a vector in multidimensional space. To be able to measure the similarity of the shapes of two objects, the length of the difference-vector of the corresponding moment invariants vectors was calculated. The conclusion was, that moment invariants alone are not sufficient for satisfying classification and that additional descriptors are necessary.
Moment invariants can be used for the derivation of three-dimensional building parameters from laser scanning data (Maas 1999). Using only first and second order invariant moments, a number of basic parameters of a building (position, orientation, length, width, height, root type and roof steepness) can be determined. Using higher order moments, more complex roof shapes can be modeled as well.

Principal Components Analysis

Principal Components Analysis (PCA) is a common technique for image compression and for finding patterns in higher dimensional data. It is also known from statistics as the (discrete) *Karhunen-Loève transform* or from image processing as the *Hotelling transform*. The task is to transform correlated data into uncorrelated data and to preserve the information contained in the data set. For example the values of corresponding pixels (i.e. which have the same coordinates) in the three component images of a color RGB image are correlated. This data can be represented as a random vector with three components:

$$\mathbf{x} = (x_1, x_2, x_3)^T$$

For each of the K pixel elements one population vector exists.
Considering an object, that is represented by a binary image, the coordinates

of the pixels can be treated as a random vector with two components populated by the x-values and the y-values of the pixels. This is the case for the points of a vector boundary:

$$\mathbf{x} = (x_1, x_2)^T$$

This data is also correlated.
The mean vector of a random vector is defined as:

$$\mathbf{m_x} = E\{\mathbf{x}\} \tag{2.18}$$

Information about the correlation of the random variables gives the covariance matrix $\mathbf{C_x}$:

$$\mathbf{C_x} = E\{(\mathbf{x} - \mathbf{m_x})(\mathbf{x} - \mathbf{m_x})^T\} \tag{2.19}$$

If two random variables x_i and x_j of a random vector are uncorrelated, their covariance will be zero. Thus the covariance matrix of uncorrelated data is a diagonal matrix.
The mean vector can be approximated from the data by using:

$$\mathbf{m_x} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{x}_k \tag{2.20}$$

and the covariance matrix:

$$\mathbf{C_x} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{x}_k\mathbf{x}_k^T - \mathbf{m_x}\mathbf{m_x}^T \tag{2.21}$$

Let $\mathbf{e}_i$ be the eigenvectors and λ_i be the eigenvalues of $\mathbf{C_x}$ (with $i = 1, 2, \ldots, n$ - n is count of elements of the random vector). The eigenvalues are arranged in descending order so that $\lambda_i \geq \lambda_{i+1}$(i=1,2,...,n-1). Let $\mathbf{A}$ be a matrix which rows are the eigenvectors of $\mathbf{C_x}$, ordered in such a manner, that the first row of $\mathbf{A}$ is the eigenvector corresponding to the largest eigenvalue, and the last row is the eigenvector corresponding to the smallest eigenvalue.

Finally the hotelling transform is defined as:

$$\mathbf{y} = \mathbf{A}(\mathbf{x} - \mathbf{m_x}) \tag{2.22}$$

Some of the properties of the Hotelling transform are:

- The mean of $\mathbf{y}$ is zero.
- The covariance matrix $\mathbf{C_y}$ of $\mathbf{y}$ is a diagonal matrix. It follows that the population of the vector $\mathbf{y}$ is uncorrelated.
- $\mathbf{C_x}$ and $\mathbf{C_y}$ have the same eigenvalues and eigenvectors.

The result of the Hotelling transform is a vector $\mathbf{y}$ containing the coordinates of the translated and rotated binary object. The origin of the new coordinate systme is now located in the center of mass of the object. The axes of the new coordinate system are oriented in direction of the eigenvectors of $\mathbf{C_x}$. Thinking of the given (two dimensional) object rotating around an arbitrary axis (defined by two arbitrary points of the object), the moment of inertia is minimal for the principal axis of inertia. This axis is found by the hotelling transform. It is the x-axis of the new coordinate system.

2.3 Line simplification algorithms

Automated line generalization includes simplification, smoothing, displacement and enhancement (McMaster 1987). These elements must be considered as interrelated operations in order to guarantee adequate generalization results.
The purpose of line simplification is to reduce the number of points required to represent the line, without changing the main characteristics of the line. In the following some algorithms for line simplification will be presented. All presented algorithms have in common that a certain amount of salient or critical points are selected. No displacement of points is accomplished.
The following classification, that proposes five categories of algorithms, is

made by McMaster (McMaster 1987). The algorithms are classified by examining the "neighborliness" of the underlying mathematical processing.

2.3.1 Independent point routines

An example in this category is the very simple thinning algorithm, where only every n^{th} point is retained in the data. The result is of random nature and can only be used for thinning out unnecessarily dens data.

2.3.2 Local processing routines

When a point is tested for its geometrical relevance, algorithms in this class take information about the neighboring points into account. In the following the ordered set of points of the line will be referred to as p_i with $i = 1, \ldots, n$. The precedent point of p_i is p_{i-1}, the successive point of p_i is p_{i+1}. The euclidean distance between two points will be formalized as $\|\overrightarrow{p_i, p_j}\|$ and the perpendicular distance from a point p_i to a line segment $\overrightarrow{p_j, p_k}$ as $\|\overrightarrow{p_i, \overrightarrow{p_j, p_k}}\|$. The angle $\measuredangle(p_i, p_j, p_k)$ is defined as the smaller angle between line segment $\overrightarrow{p_j, p_i}$ and $\overrightarrow{p_j, p_k}$.

- A point can be tested for its euclidean distance toward the successive point. If they are closer together than a defined value, the point is removed from the data:

$$\|\overrightarrow{p_i, p_{i+1}}\| < const \Rightarrow p_i \text{ removed} \tag{2.23}$$

  The generalization result depends on the orientation of the traversing of the points!

- Another algorithm takes into account two neighbors of a point: The previous and successive point are connected by a straight line and the perpendicular distance of the examined point to the line is calculated. If the distance is smaller than a defined tolerance value, the point is assumed to be not essential for the shape of the line and it is removed

from the data:

$$\|\overrightarrow{p_i, \overrightarrow{p_{i-1}, p_{i+1}}}\| < const \Rightarrow p_i \text{ removed} \tag{2.24}$$

- An essential point can be detected by the angular change between two vectors. If point p_i is tested, the angle $\measuredangle(p_{i+1}, p_{i-1}, p_i)$ must be calculated. If it is smaller than a predefined value, the point is removed from the line:

$$\measuredangle(p_{i+1}, p_{i-1}, p_i) < const \Rightarrow p_i \text{ removed} \tag{2.25}$$

  This algorithm provides different generalization results for the same line depending on the orientation of the traversing of the points!

- Another algorithm in this group is testing the two neighbors p_{i-1} and p_{i+1} of a point p_i for three parameters D_1, D_2, A. These are the distance $\|\overrightarrow{p_{i-1}, p_i}\|$, the distance $\|\overrightarrow{p_{i-1}, p_{i+1}}\|$ and the angle $\measuredangle(p_{i-1}, p_i, p_{i+1})$. If one of the distances is smaller than its specific threshold, p_i is removed from the list. If both distances are larger than its threshold and the angle is larger than A, the point is also removed from the list:

$$\begin{aligned} &[\ (\|\overrightarrow{p_{i-1}, p_i}\| \ < \ D_1) \ \vee \ (\|\overrightarrow{p_{i-1}, p_{i+1}}\| \ < \ D_2)\ ] \ \vee \ \ldots \\ &\ldots \ (\measuredangle(p_{i-1}, p_i, p_{i+1}) \ > \ M) \Rightarrow p_i \text{ removed} \end{aligned} \tag{2.26}$$

2.3.3 Unconstrained extended local processing routines

An example of this type of algorithm is the Reumann-Witkam routine. The first point of the line is the first point of the generalized line. A corridor of predefined width is calculated parallel to the first line segment. Looking at the points sequentially, point p_i is considered to be the first point lying outside the corridor. Point p_{i-1} is the second point retained in the generalized line. Now the next corridor is calculated parallel to the line segment connecting point p_i and p_{i+1}. This procedure is continued until the end of the line is reached.

2.3.4 Constrained extended local processing routines

Like the Reumann-Witkam algorithm in the section before, the constrained extended local processing routines evaluate not only points and neighbors, but extended parts of the line. However, either the count of points being removed or the length of the generalized line segment is constrained. Some examples are algorithms by Opheim, Lang and Johannsen (McMaster 1987).

2.3.5 Constrained global routines

While testing the individual points for their importance, algorithms in this group take the whole line into consideration.

The well-known *Douglas-Peucker algorithm* (Douglas and Peucker 1973) takes the first point of the line as anchor point and the last point as a floating point. The distances from all other points to the line connecting the anchor point and the floating point must be calculated. If the greatest distance is below a certain tolerance parameter (thus it is located outside of a corridor which is defined by the connection line between the anchor and the floating point and has a width of the tolerance value), the straight line is a good approximation for the original line. If it is greater, the furthest point becomes the new floating point and the procedure is repeated. The floating point advances toward the anchor point until no point, that is located outside of the corridor, is left. The last floating point is stored as point of the simplified line and becomes the new anchor point. Again, the last point of the line is defined as floating point and the whole procedure is repeated.

All points from the original line, that are located between two points of the simplified line, are located within the tolerance distance.

In figure 2.4 an example of a line generalization by means of the Douglas Peucker algorithm can be examined: The first and last point of the line are connected by a straight line (figure a). Point 1 is the anchor point, point 17 is the floating point. Now it is searched for the farthest point from the

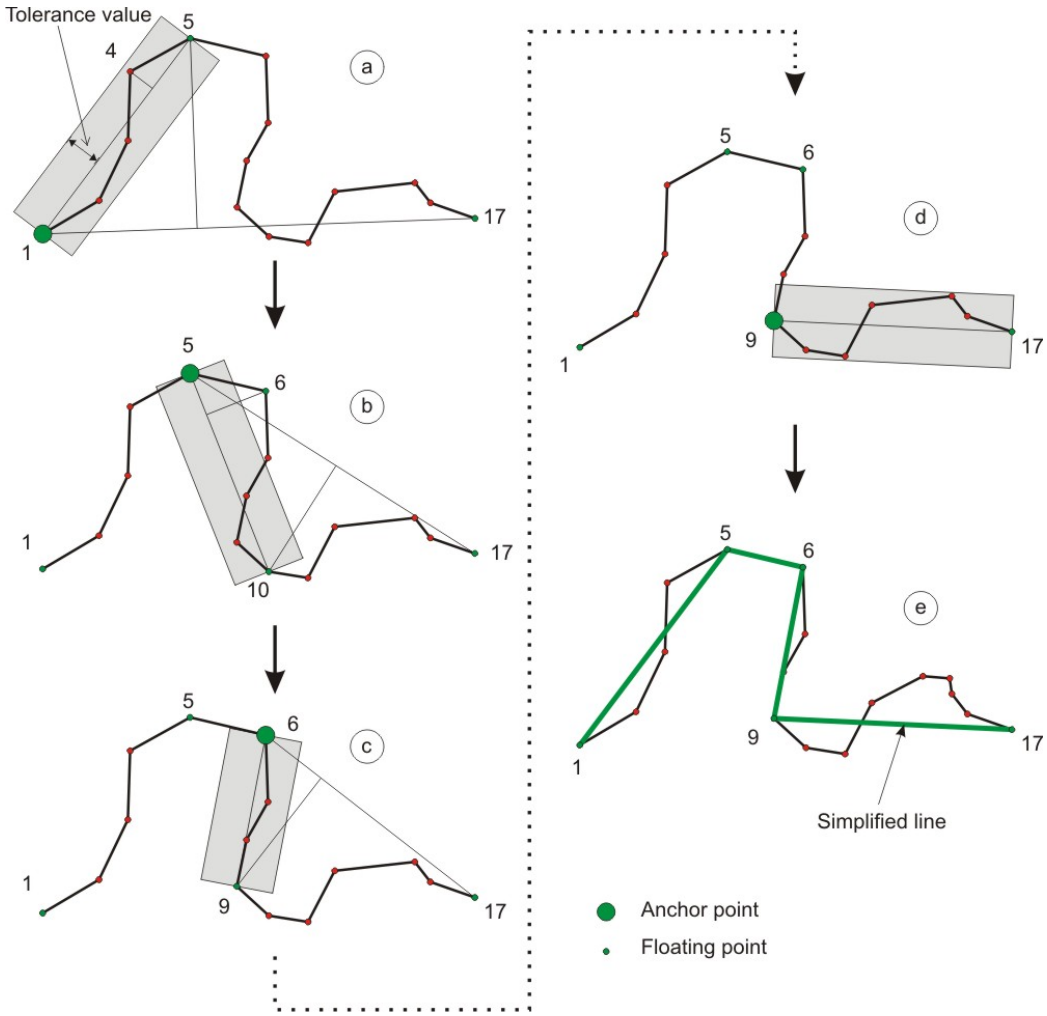


Figure 2.4: Douglas-Peucker algorithm

connection line. This is point 5. It is located outside the corridor, so it becomes the new floating point. A new base line from point 1 to point 5 and the distances of the points 2 to 4 are calculated. All distances are smaller than the tolerance value. Therefore, point 5 is retained in the simplified line. It becomes the new anchor point. The procedure is repeated with point 5 as anchor point and point 17 as floating point (figure b). The distances of point 5 to 16 are tested. Point 10 is the farthest and becomes the new floating point. From point 6 to 9, point 6 is the farthest from the base line connecting point 5 and 10. No more points are left, therefore point 6 is retained in the simplified line and becomes the new anchor point. The procedure is repeated (figure c and d). The result of the simplification can be seen in figure e. With the selected tolerance value, the points 1, 5, 6, 9 and 17 remain in the simplified line.

The later developed Visvalingam-algorithm (Visvalingam and Whyatt 1993) doesn't focuse on point distances but on area. The effective area of a point is defined as the area enclosed by the predecessor of the point, the point itself and the successor. The effective area of all points between the first and last point is calculated. The point with the least effective area is eliminated and the effective area of the neighboring points is recalculated. This procedure is repeated until the desired count of points is left. In this way small features of the line are eliminated first. It is ensured that small spikes can not be selected as critical points, as it might be the case using the Douglas-Peucker algorithm (such a point might have the greatest distance from the anchor-floater line).

2.4 Building generalization algorithms

Line generalization algorithms can not simply be applied to building polygons, because additional constraints like rectangularity and parallelism have to be taken into account (Sester 2000). Building generalization is essential in the case, that building outlines are derived from raster data. Not only data

reduction by line simplification is demanded but more often recognition of the structure of the building. Applying known line generalization algorithms to buildings is difficult, because parameters have to be set individually in order to achieve satisfying results and this is at odds with the need of automation. Building generalization aims at

- simplification of the building outline,
- computing "plane" building edges,
- preserving essential characteristics of the shape and
- applying constraints like rectangularity and parallelism.

Application fields for building generalization algorithms are all areas, where large-scale building footprints are needed. This is the case for all kinds of visualizations in 2D and as a base for 3D-building models. Moreover a large application area is the civil protection (flood maps, mud flows, avalanches, volcanic eruptions, etc.). In the insurance industry a growing demand of large-scale building models can be noticed. A special application is the computation of noise dispersion maps. In this case it is especially important to dispose of clean building edges. Particular attention must be turned on the correct detection of the orientation of the building.
Enhancement of the accuracy of a building outline through geometric generalization is possible under certain conditions. It is possible if it can be assumed, that only random errors and no gross errors exist in the data and if the task is to reconstruct all details (other than the random errors) appearing in the data. However, if the geometric generalization does not only aim in eliminating random errors, but is additionally applied for the simplifying and elimination of details of the building outline, the accuracy can obviously not be enhanced. In contrary, a loss of information must be accepted. If there exist classification errors in the data, the generalization process may eliminate those gross errors, in case of the degree of generalization is high enough. But this is a "side effect" and can not be controlled, because algorithms in general can not a priori differentiate between details of the building outline

Generalization action	Order
Elimination of a corner edge	1
Aggregation of two short edges	2
Elimination of two corner edges	3
Elimination of insignificant porches or indentations	4
Elimination of an offset	5
Accentuation of significant porches or indentations	6

Table 2.1: Generalization action hierarchy

and gross errors (coming e.g. from image classification).
Building generalization algorithms are mostly designed in the face of available data and in the face of the application, which the generalized result is dedicated to. Typically the input data must fulfill several properties: a specific data format, specific building outline characteristics (e.g. approximate rectangularity), adjacency of buildings (e.g. disjoint buildings or restriction of connection of buildings to straight lines), etc. Furthermore, the generalization result is completely determined by the algorithm. It may be controlled in a certain manner by e.g. tolerance values or selection of building primitives, that are provided by the algorithm, but in general not every generalization result can be achieved by any algorithm. In this section some known building generalization algorithms are presented.

2.4.1 Work by W. Staufenbiel

An early approach to automatic building generalization was made by Staufenbiel (Staufenbiel 1973). An operation work flow for cartographic generalization is presented. A major part of the work deals with building generalization. The base idea is preserving the perception thresholds for length and area when changing the scale of a map. Therefore the building generalization algorithm searches for edges of the building, that are smaller than a fixed threshold. They are either eliminated, adapted or accentuated. A specific order of the processing steps has to be preserved, otherwise no meaningful results are achieved. Table 2.4.1 shows the generalization work flow, that is

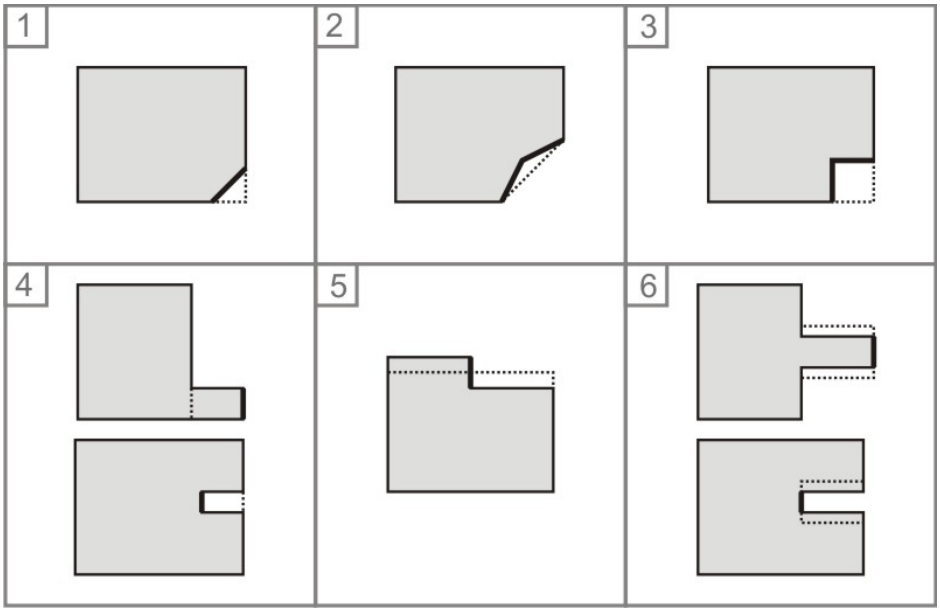


Figure 2.5: Successive edge processing steps (Staufenbiel-algorithm). The bold line segments are shorter than the specified threshold and have to be edited.

applied to each building. The succession is indicated in the right column. An example for every simplification step can be found in figure 2.5. The processing scheme developed by Staufenbiel was later implemented in the software CHANGE (Sester 2000).

2.4.2 Adaptation of the Douglas-Peucker algorithm to buildings

How the Douglas-Peucker-algorithm can be adapted to closed building polygons is described in (Kanani 2000). Initially the point farthest from the center of mass of the polygon is defined as the anchor point. The point farthest from the anchor point is defined as the floating point. These two points define a straight segment and separate the building polygon into two polylines. These two points are at the same time two detected corners of the simplified building. Now, the Douglas-Peucker-algorithm is computed with both polylines separately as described in section 2.3.5. The "tricky" part of the implementation is the determination of a suitable tolerance parameter.

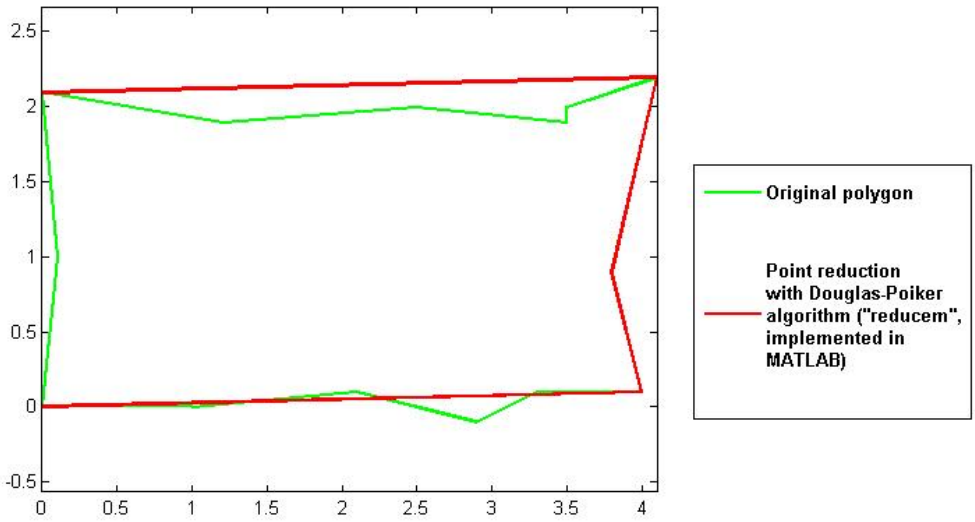


Figure 2.6: Application of the MATLAB function *reducem* on a building polygon. The green polygon is the original building polygon. The tolerance value of 0.3 was used.

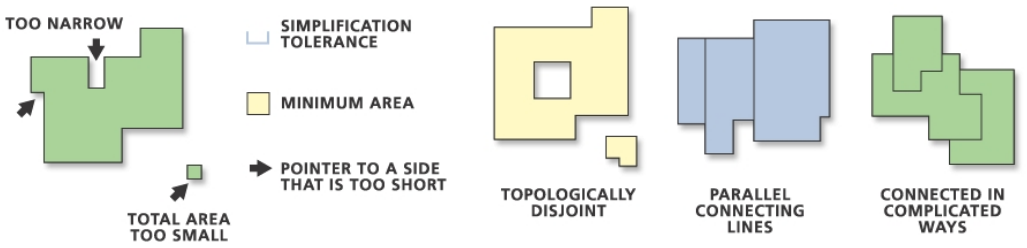
It is dependent on the characteristics of the input data and on the desired level of detail. The building model achieved by the algorithm can be refined by applying a least squares adjustment. In this way rectangularity or parallelism of building edges can be achieved.

The Douglas-Peucker algorithm is implemented in the function *reducem* in the software MATLAB[1] (The MathWorks). An example of a (fictive) building polygon, simplified by means of this function is shown in figure 2.6. A tolerance value of 0.3 was used.

2.4.3 *Simplify buildings*

ArcGIS provides the tool *Simplify buildings* (ESRI, Environmental Systems Research Institute). It takes a coverage file containing building polygons as input data and computes simplified buildings (A coverage is a data model

[1]The MathWorks MATLAB (`http://www.mathworks.com`)



(a) A distance and an area tolerance value are specified by the user. Too short segments are further processed (eliminated or widened).

(b) Disjoint buildings are simplified by itself (left). Connected buildings are simplified as a group, if the connection line is a straight line (middle). If the connection line is more complex, the buildings are not simplified (right).

Figure 2.7: Principles of the *Simplify buildings*-algorithm available from ArcGIS (image taken from the Help file).

for the storage of vector data. It contains both the spatial (location) and attribute (descriptive) data for geographic features). The algorithm preserves and enhances orthogonality of the buildings. This means, that near-90° angles become exactly 90°. The algorithm works on large scale data, where each building is represented individually. Topologically disjoint buildings are simplified independently. Connected buildings will only be simplified, if the common boundary is a straight line. They are handled as a group. If the boundary in common is more complicated, the buildings are not simplified. Two parameters are specified by the user:

- Simplification tolerance: Boundary segments shorter than the simplification tolerance are further processed. That means, that e.g. isolated, small intrusions are either filled or widened. Isolated small extrusions are cut off.

- Minimum area: Buildings that have a smaller area than the given minimum area are eliminated.

Compendiously, the number of vertices is reduced, but the measured area remains roughly the same as the original. The maximum degree of simplification is reached when a building is reduced to a rectangle.

The tool provides a friendly user-interface and is well integrated in the GIS-software. This makes it easy for GIS users and cartographers to perform generalization sessions (ESRI 1996). However the algorithm is not advantageous to data coming from raster data vectorization. Since the algorithm only eliminates or widens small boundary segments, the "pixel-shape" of the boundary is preserved and accentuated instead of a straightening of the edge. The result is e.g. not suitable for a visualization of 3D building models.

2.4.4 Generalization based on least squares adjustment

Sester presented a building generalization method, that uses the least squares adjustment (Sester 2000). The algorithm works in two steps. First an approximated model for each building is generated. This model is afterwards introduced in the least squares adjustment. It is adapted by means of the original building edges, that serve as observations. Characteristics of buildings like rectangularity and parallelism and characteristic building parts are not only preserved but can be exaggerated by means of the stochastic model. In a first step the buildings are processed individually one after the other. In a similar approach as described in section 2.4.1, the algorithm tries to replace building edges, which are shorter than a predefined minimal length. These edges are substituted according to some given rules, depending on the geometry of the adjacent building sides. The simplified building serves as an approximate building model and is then transformed into a parametric representation. These form parameters are the unknowns introduced in the least squares adjustment. The observations are the original building edges. The stochastic model, which describes the accuracy of the observations, is modeled in terms of the overlap with the approximate building model. An example of the results produced by the algorithm is shown in figure 2.8.

2.4.5 Boundary regularization

A method, that uses the boundary points to determine the parameters of a regularized polygon with perpendicular directions is presented by Sampath and Shan (Sampath and Shan 2004). The method was developed in

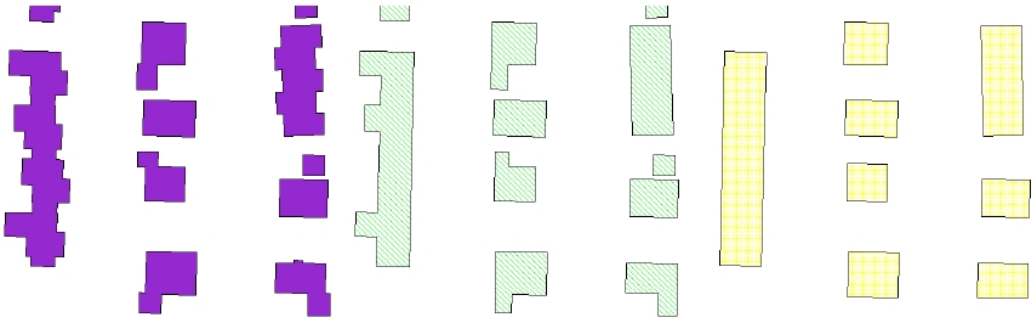


Figure 2.8: Example buildings, simplified with the algorithm presented by (Sester 2000). The original buildings are on the left. Buildings simplified with a small minimal length (3 m) can be seen in the middle and with a large minimal length (7 m) on the right. From: (Sester 2000).

order to transform lidar data points, which were classified as building points, into a building representation, that can be stored in a geospatial database. Therefore the building boundary points are first extracted by means of a "tracing"-algorithm. The resulting points, representing an irregular shape and possible artifacts, are introduced into the regularization process, that is described in the following.
The main idea of the algorithm is the extraction of long line segments as well as the two main directions of the segments from the original points and afterwards to introduce all points, segments and directions into a hierarchical least squares adjustment. Four steps are involved in the process:

1. The first step is to find all group of points, that represent straight line segments (so called "long line segments"). This is achieved by examining the slope of line segments, formed by consecutive points, and to pool points on consecutive edges with similar slopes into a group. For further processing (step 2 to 4) the largest groups are selected, because the longer the line segments, the more likely they represent the dominant directions of the building.

2. In the next step a line $A_i x + B_i y + 1 = 0$ is fitted into each group of points. The slope of each line is calculated: $M_i = -A_i/B_i$. The lines are again sorted into two groups, based on their slope. Approximately parallel lines have similar slopes and the product of the slopes

of approximately perpendicular lines is around -1.

3. The third step is again the determination of the long line parameters, but this time the constraint of parallelism or perpendicularity (step 2) is introduced. The result is a set of parameters of each long line and the dominant slopes of the building.

4. The final regularization step includes all (long and short) line segments into the least squares solution. The slopes obtained from the previous step are used as approximated values. In this step no explicit constraint is enforced, but the slope parameters for long line segments are given high weights. The line segments of the resulting polygon are fitted to the dominant directions of the building and at the same time, they are fitted to the lidar boundary points.

Because of the hierarchical approach (shorter line segments are processed after longer lines), the method is robust to possible errors in building segmentation and boundary tracing (Shan and Sampath 2006). Another advantage of the presented method is that the errors of the final extracted buildings can be evaluated through the least squares adjustment process. The method provides a global optimization solution, since no points or line segments are taken as fixed reference.

2.4.6 Recursive rectangle approximation

Very useful in case of building polygons, which are noisy due to the segmentation process, is the approximation by polygons, consisting exclusively of rectangular angles. The basic idea is to subtract all non-building parts from the smallest surrounding rectangle, that has the same orientation as the orientations of the boundary edges (Gross, Thoennessen, and Hansen 2005). Since a rectangular approximation is wanted, the rectangular approximation of the non-building parts has to be computed first. This leads to a recursive procedure, which is schematically visualized in figure 2.4.6.
Let A' be the area of the building. The procedure starts with the calculation

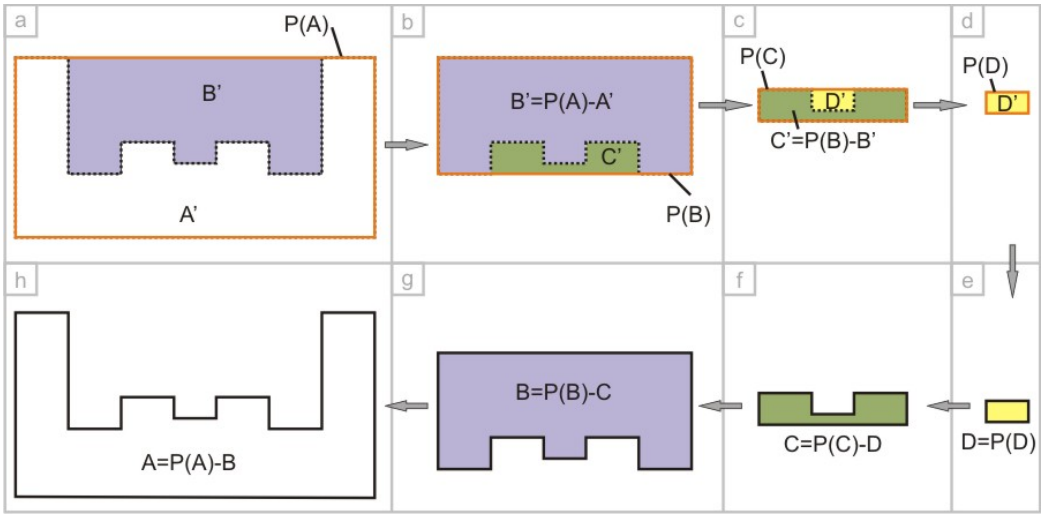
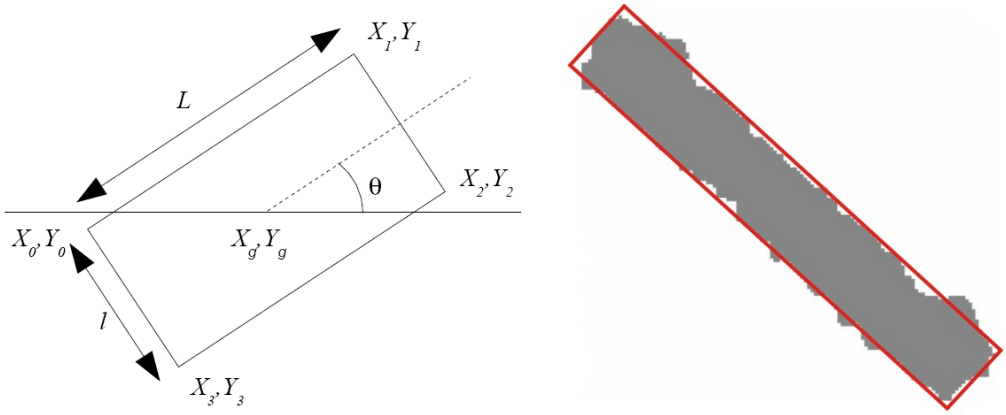


Figure 2.9: Recursive rectangle approximation: A dotted polygon line indicates noisy segments (from segmentation process). The smallest surrounding rectangle is marked in orange.

of the smallest surrounding rectangle with the same orientation as the orientations of the boundary edges P(A) (see step **a**). The non-building part B' is the difference of the surrounding rectangle P(A) and the original polygon A'. Step **b** is identical with step **a**, but now B' is taken as input polygon instead of A'. The smallest surrounding rectangle of B is computed (gives P(B)). The difference P(B) minus B' gives C' and so forth. In step **d** the appropriate depth of recursion is reached. The surrounding rectangle of D is computed: P(D). This is at the same time the rectangular approximation of D (step **e**). Now the exact rectangle D can be subtracted from the surrounding rectangle P(C) and the result is the rectangular polygon C (step **f**). In the same way step **g** and **h** are computed. Result is the rectangular approximation of the building.

When computing the difference of the surrounding rectangle and the building polygon, the result is in general composed of more than one polygon. An area threshold has to be defined. The described recursive algorithm is processed with all resulting polygons which have an area not smaller than the threshold. In case of very noisy polygon edges, the resulting rectangle



(a) A rectangle is determined by five parameters: center coordinates, orientation, length and width. From: (Vinson, Cohen, and Perlant 2001).

(b) Example of a building with estimated rectangle. From: (Vinson, Cohen, and Perlant 2001)(digitized).

Figure 2.10: Approximation of a (raster blob) building by a rectangle

approximation can not serve as the final approximation. The resulting approximation can serve as a suitable model, but the final location of the edges must be adapted to fit the noisy edges.

2.4.7 Extraction of rectangular buildings from aerial images

The method described in (Vinson, Cohen, and Perlant 2001), implements ideas from the principal components analysis (see section 2.2.3) for the estimation of rectangles from given raster blobs. A relation between the eigenvalues of the covariance matrix and the rectangle dimensions is established. The algorithm estimates five parameters over any given blob, which determine an approximating rectangle. The parameters are the center-coordinates X_g and Y_g, the orientation Θ, the length L and the width l (see figure 2.10(a)). The coordinates of the center of mass of the blob are taken as center-coordinates of the rectangle (see eq. 2.20). The orientation of the principal axis of the blob is taken as an estimation of the orientation of the rectangle.

It is calculated as follows:

$$\tan 2\Theta = \frac{2M_{xy}}{M_{xx} - M_{yy}} \tag{2.27}$$

The length and width of the rectangle is calculated by means of the eigenvalues:

$$\lambda_{max/min} = \frac{M_{xx} - M_{yy} \pm \sqrt{(M_{xx} - M_{yy})^2 + 4M_{xy}^2}}{2} \tag{2.28}$$

The analytic resolution of eq. 2.28 gives the eigenvalues:

$$\lambda_{max} = \frac{L^2 - 1}{12} \quad \text{and} \quad \lambda_{min} = \frac{l^2 - 1}{12} \tag{2.29}$$

The width and length of the estimated rectangle can be calculated as follows:

$$L = \sqrt{12\lambda_{max} + 1} \quad \text{and} \quad l = \sqrt{12\lambda_{min} + 1} \tag{2.30}$$

The Hausdorff-distance is used as a measure of similarity between the original blob and the approximated rectangle. An example of a rectangle approximation of a building can be seen in figure 2.10(b).
Furthermore, a method for refining the rectangular building model for complex buildings is presented. The automatic algorithm splits the blob into several parts. Each part is approximated by a rectangle. So the number of rectangles and the overlap between rectangles is minimized. The size of the rectangles is maximized. This optimization process is controlled by the Hausdorff-distance, which serves as a quality measure. An example of the rectangle decomposition is shown in figure 2.11.

2.4.8 Hough Transformation

The so called Hough Transformation is a method to detect lines, circles, ellipses etc. from a set of points. It is a technique for isolating features that share common characteristics (Vozikis 2005). From a set of points, those points are detected by the Hough Transformation, that are elements of a line. The level of detail can be controlled by defining either the number of

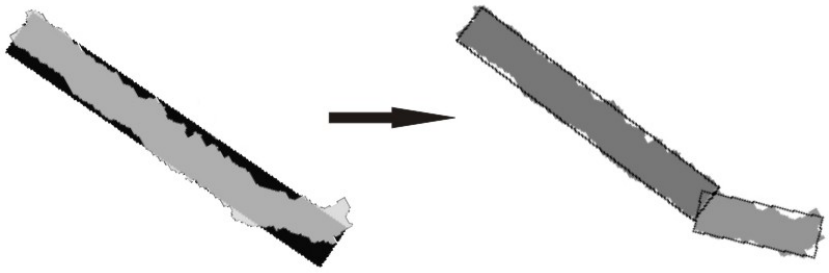


Figure 2.11: Refinement of the rectangle approximation by splitting up into several rectangles. From: (Vinson, Cohen, and Perlant 2001).

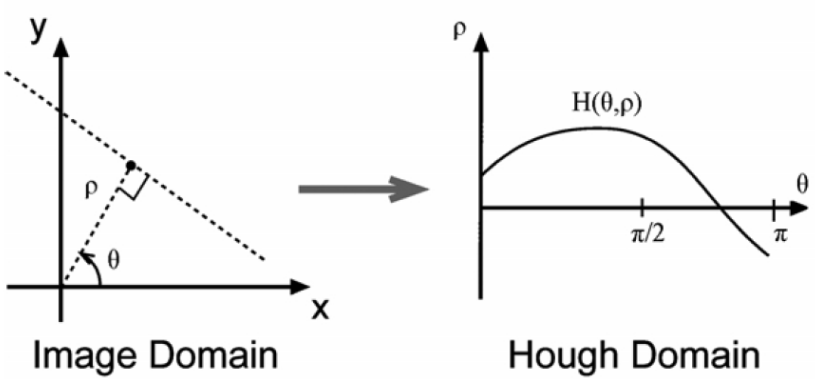


Figure 2.12: Transformation of a point from the image space to the parameter space. From: (Vozikis 2005).

edges to be found, or the number of points lying on one edge. Furthermore, constraints like orthogonal angles can be applied. This makes it very useful for the simplification of noisy building outlines.
The general idea of the Hough Transformation is to transform the information from the image space into a parameter space and apply there an analysis. A general representation of a line in 2D-space is the Hesse Normal Form:

$$x \cos\theta + y \sin\theta - \rho = 0 \tag{2.31}$$

where ρ is the perpendicular distance from the origin to the line and Θ is the angle of the perpendicular vector in the range 0 to π counted counterclockwise from the x-axis (see figure 2.12 on the left). Each point from the image space

is transformed into the parameter space by using the following formula:

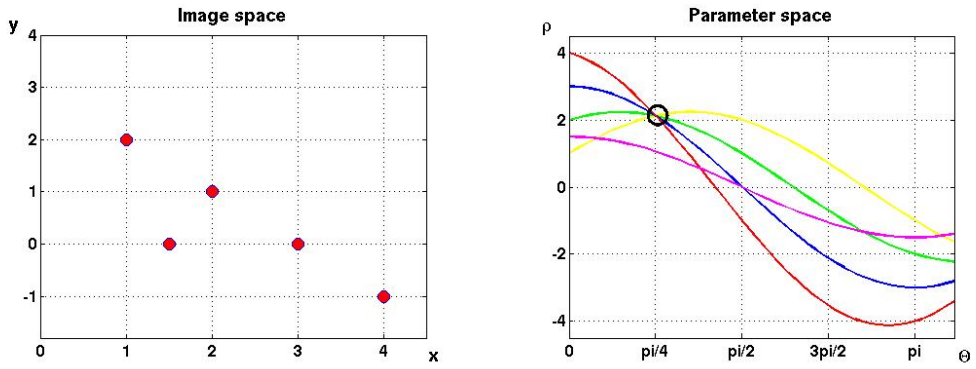
$$\rho = x \cos\theta + y \sin\theta \tag{2.32}$$

The points in the image space produce sinusoids in the parameter space. Each line in the image space corresponds to a point in the parameter space. Thus if some points lie on the same line in the image space, the sinusoids in parameter space will intersect in a single point. The transformed sinusoids are analyzed in respect of intersection points in the parameter space. An example of five points, being transformed into parameter space, is illustrated in figure 2.13. Four of the sinusoids intersect in a single point, hence the corresponding points are elements of the same line.

The computational attractiveness of the Hough Transformation arises from the subdividing of the parameter space into so-called accumulator cells. This corresponds to a discretization of the Hough Transformation. The value of all accumulator cells, which are covered by a sinusoid, is incremented by the value 1. After all points have been transformed into the parameter space, the intersection points of the sinusoids can be found easily by searching for maximum values of the accumulator cells.

Another advantage is, that the computational accuracy can be adjusted easily by varying the cell size. The smaller the cells are, the better becomes the accuracy, but concurrently the computation time increases. Unfortunately the points of a noisy building edge lie not exactly on a line, but close to it. That means, that the sinusoids in the Hough domain intersect not exactly in one point and it has to be searched for a distribution of intersecting points. The distribution is not normal, therefore the straightforward solution (taking the local maximum as an estimation for the line parameters) is not suitable. Instead the center of the distribution is defined as the weighted mean of all intersection points that are bigger than a predefined threshold.

After finding the local maxima and processing the back transformation, the resulting lines have to be intersected in order to find the final building polygon. To avoid the problem of intersecting the wrong lines, it has to be searched only for intersections that lie close to the vector data. In figure 2.14



(a) Five points in image space. Four of them lie exactly on a line.

(b) The same points in parameter space. Four of the sinusoids intersect in one point.

Figure 2.13: Example for a transformation of points from image into parameter space. Four points lie on a line and therefore intersect in parameter space in a single point.

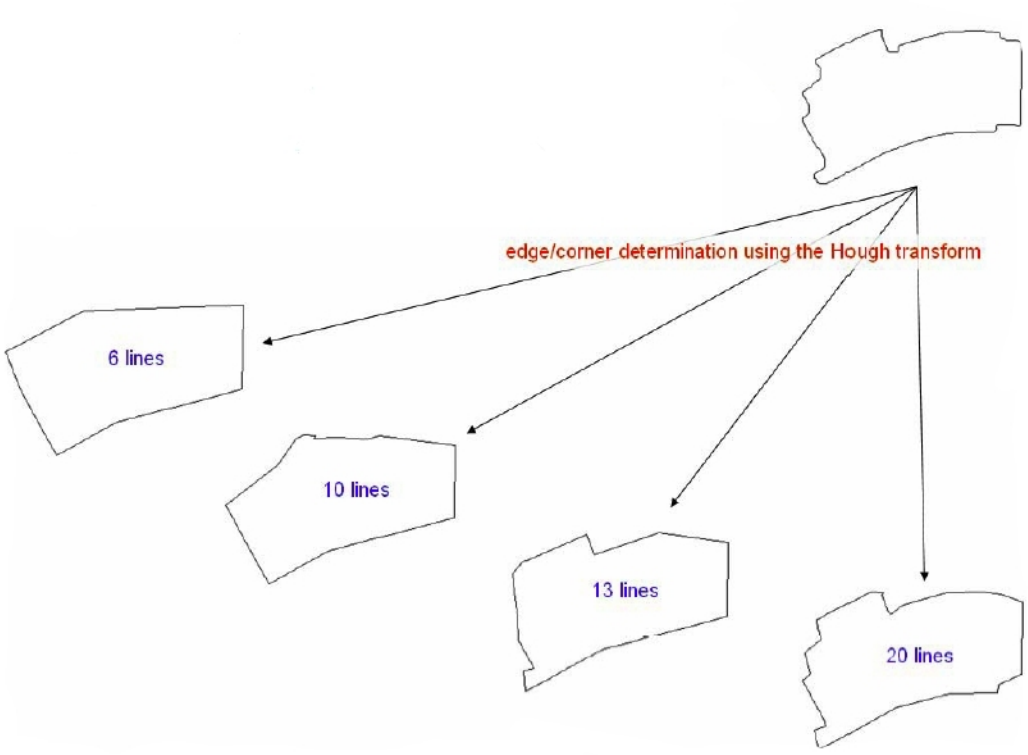


Figure 2.14: The level of detail can be controlled by the count of lines, thus the count of searched local maxima. From: (Vozikis 2005).

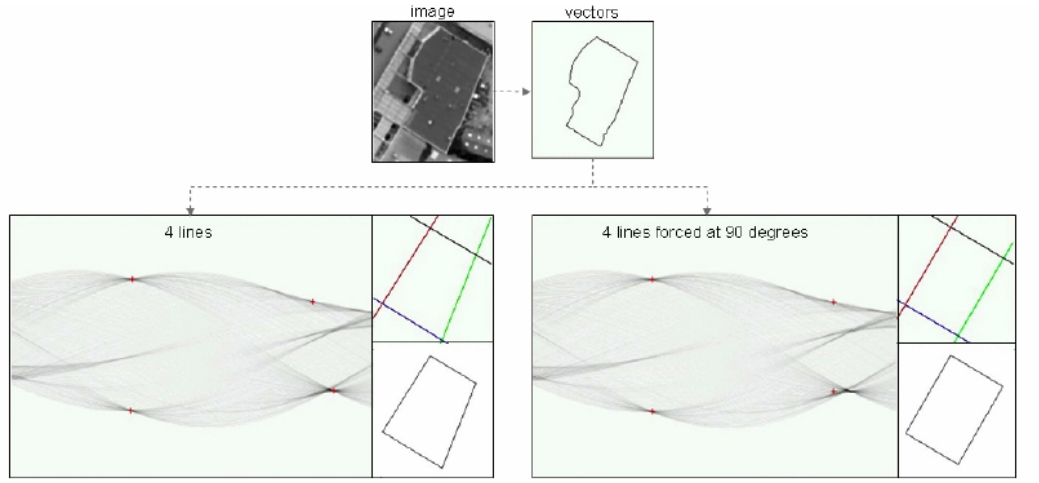


Figure 2.15: In the left image, four intersection points are searched and no angle constraint is applied. For the right example an adjustment is calculated, in order to force the lines at 90 degrees. From: (Vozikis 2005).

an example of different simplification results by means of the Hough Transformation is shown. The degree of generalization is determined by the count of detected lines.

Angle constraints between intersecting lines will be achieved, if the horizontal distance between the corresponding local maxima in parameter space is equal to the wanted angle. Thus the final locations of the used points in Hough domain are calculated through an adjustment. Figure 2.15 shows an example with four intersection points. In the left part of the figure, no angle constraint is applied. In the right part of the figure, an adjustment is calculated , in order to force the lines at 90 degrees. It can be seen, that the intersecting points corresponding to perpendicular lines, have a constant horizontal distance.

Another advantage of the Hough Transformation is the ability to overcome missing data. An example can be seen in figure 2.16. Due to the data caption process, many building boundary pixels were not detected. By means of the Hough Transformation however it is possible to reconstruct the missing building corner. A small amount of points located along a line answers the purpose, despite a lot of incorrect data exist in the neighborhood.

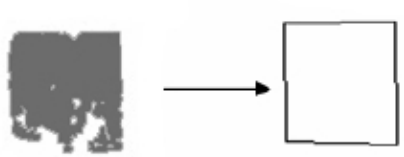


Figure 2.16: Example of the strength of the Hough Transformation in overcoming missing data. A small amount of points lying on a line is enough, to detect the correct edge. From: (Vozikis 2005).

Chapter 3

Proposed building generalization method

3.1 Overview

The proposed generalization method was developed facing the need of generalized building outlines, that could serve as a base for various 2D and 3D visualizations. Therefore, the generalization process should result in simple polygons with restriction to rectangular angles and to a certain degree of detail. The geometry of the raw buildings is analyzed in a sort of "top down-way", meaning that a building polygon is first approximated by a simple rectangle. If the rectangular model does not fit well enough, it is refined by cutting out corners or by replacing an edge by more edges. A model is fitted to the data by filtering the essential points (the so-called split points) out of all building polygon points. If e.g. a rectangle is going to be fitted, four split points are searched, which must be located in the corners of the building. As soon as the split points are found, the points between the split points are used to calculate the final location of the building edges.
The level of detail of the generalization can be changed by variation of two input parameters, which have to be specified by the user.

3.1.1 Prerequisites of input data

The input data must comply with the following requirements:

1. The input polygons must be represented by a list of points, ordered in the manner they are encountered by traversing the polygon in one direction.
2. They must be simple closed polygons, thus they must not contain holes.
3. A polygon has to be composed of a minimum of four points.
4. The building points must be roughly equally distributed along the building outline (i.e. the euclidean distance between every two successive points should be similar).

3.1.2 Proposed generalization algorithm

The proposed generalization algorithm can be divided into five major steps:

1. Checking, if the point density along the building boundary is constant.
2. Calculation of the major orientation of the building. It is taken from the orientation of the longer edge of the minimal bounding rectangle (MBR). The edges of the generalized polygon are designed to be parallel to one of the edges of the MBR.
3. Search for split points in three different levels of detail. This fixes the geometry of the generalized polygon.
    - Level 1: Rectangular model
    - Level 2: "L-", "T-" or "Z-model"
    - Level 3: "U-model"
4. Calculation of the position of the generalized edges in relation to the MBR-edges. The distance of an edge from the according MBR-edge is the median of the distances of all points assigned to this edge from the MBR-edge. The vertices of the generalized polygon are calculated by a intersection of all successive edges.

5. Elimination of too short edges.

Details on the algorithm will be presented below.

3.1.3 User defined parameters

Two parameters are required for the execution of the generalization tool. They have to be specified by the user. Through both parameters the generalization level can be controlled:

Variationparameter $\mathbf{V_{spec}}$: This term is used for the standard deviation of the distances of the building points, which are assigned to one single edge, to the appropriate MBR-edge. The calculation will be described in detail later on. The user specified variationparameter influences the level of detail of the generalization. After the generalization process of each level, the actual values for the variationparameter of all edges are calculated. By comparing the values with the user specified variationparameter, the quality of the generalization is examined. If the actual values V are below V_{spec}, the generalization quality will be good enough and the computation will be stopped. Otherwise the next level generalization is started.

Minimal length of edge $\mathbf{M_{spec}}$: As the name implies it is the minimal length of an edge of the final generalization. This term also affects the level of detail of the generalization.

3.2 Method

In this section the proposed generalization method is described in detail. All buildings are processed independently one after the other.

3.2.1 Minimum bounding rectangle (MBR)

The first processing step is the determination of the main orientation of the building. For this purpose the minimum bounding rectangle (MBR) is

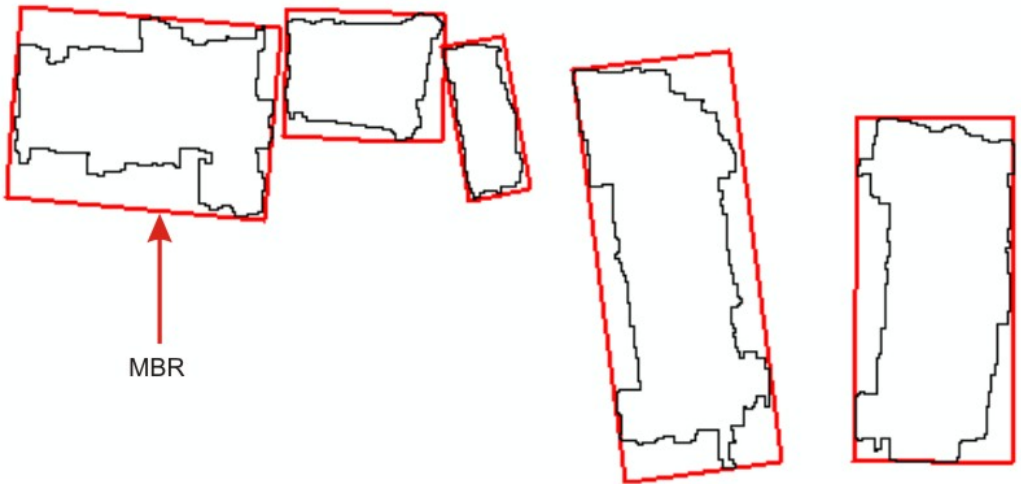


Figure 3.1: Minimal bounding rectangles (MBR) of some buildings. The orientation of an MBR is an adequate measure for the main orientation of a building polygon.

calculated. The orientation of the longer edge of an MBR of a building polygon corresponds in the majority of cases with the orientation of the building, as it is perceived by a human operator. If a building polygon, that has a rectangular shape but very noisy edges, should be approximated by a rectangle, it will be well represented by a rectangle that has all edges parallel to the MBR. The evaluation of a set of test buildings shows, that the orientation of all rectangular shaped buildings are detected correctly by computing the MBR.
The algorithm described in section 2.2.2 is used for this task. The calculation of the MBR is implemented in the following way (see figure 3.2):

- Calculate the convex hull of the polygon (a built-in ArcMap function was used).
- From all encasing rectangles having one side collinear with the convex hull's edges, take that one with the smallest area.
  Details:
  - Go through all points of the convex hull.
  - Create a local coordinate system with origin in the actual point

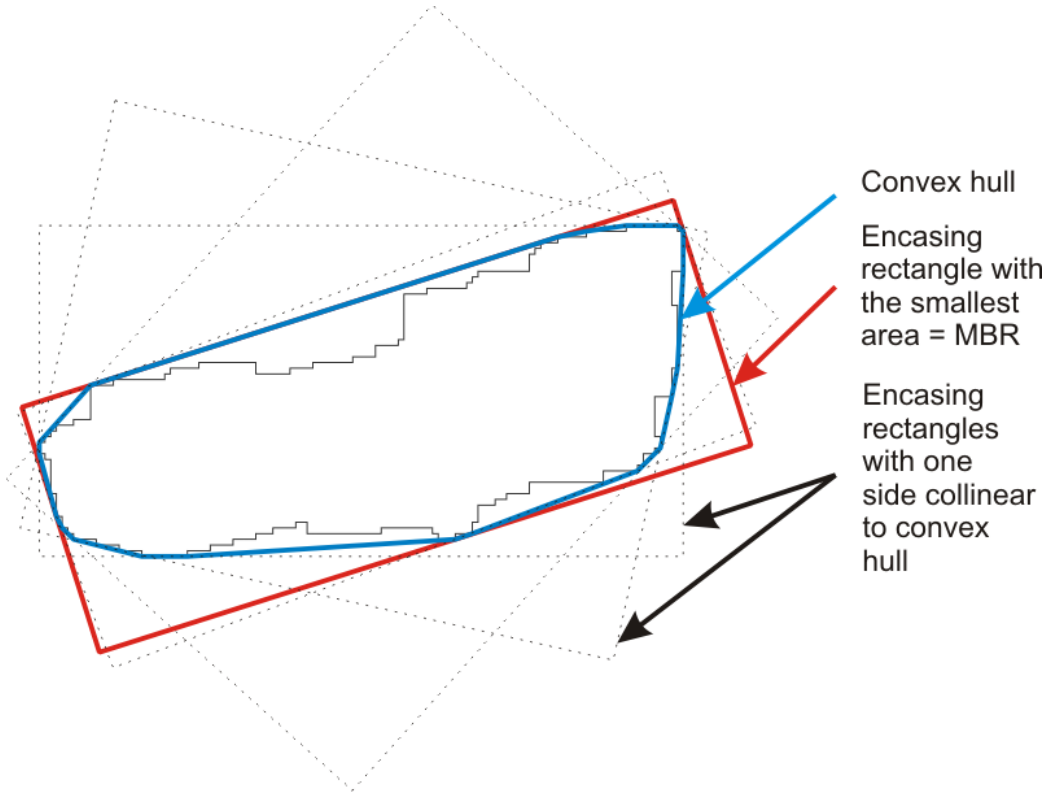


Figure 3.2: Illustration of the calculation steps for the determination of the MBR.

and positive x axis directed toward the successive point.

- Calculate local coordinates of all points of the convex hull.
- Get the minimum and maximum x coordinates and the minimum y coordinate. The maximum y coordinate is zero.
- Test for the least area rectangle resulting from the minimum and maximum points.
- Retransform the local point coordinates of the rectangle to get the real coordinates.

See figure 3.1 for some example buildings with corresponding MBR. All edges of the final generalization will be parallel to the edges of the MBR.
Sometimes problems occur with distinct "L-", "T-" or "Z-Shapes". In this cases, the MBR is no satisfying method to compute the orientation of the building. Alternative methods for computing the main orientation of a building are thinkable. This will be discussed in section 6.

3.2.2 Level 1

Now that the MBR is known, the best fitting rectangle with edges parallel to the MBR is calculated. The task is to find four split points $S = \{s_1, s_2, s_3, s_4\}$ from the set of the building points $P = \{p_1, \ldots, p_n\}$ (n is the count of building points), which best represent the corners of a rectangle. In other words, a split point should be an *essential* point of the polygon (figure 3.3, s_1 to s_4), whereas all other points can be seen as *redundant*[1] points. Let's $T = \{t_1, t_2, t_3, t_4\}$ be the set of MBR corner points and $d(p_i, p_j)$ be the euclidean distance between two points p_i and p_j. The split points s_1 to s_4 are defined as those building points that have the minimum distance to the MBR-corners:

$$s_i = p_j \in P \quad \text{with} \quad d(p_j, t_i) < d(p_k, t_i) \qquad \begin{matrix} k = 1, \ldots, j-1, j+1, \ldots, n \\ i = 1, 2, 3, 4 \end{matrix} \tag{3.1}$$

[1]The term *redundant* is used, because the points hold no new information about the form of the building.

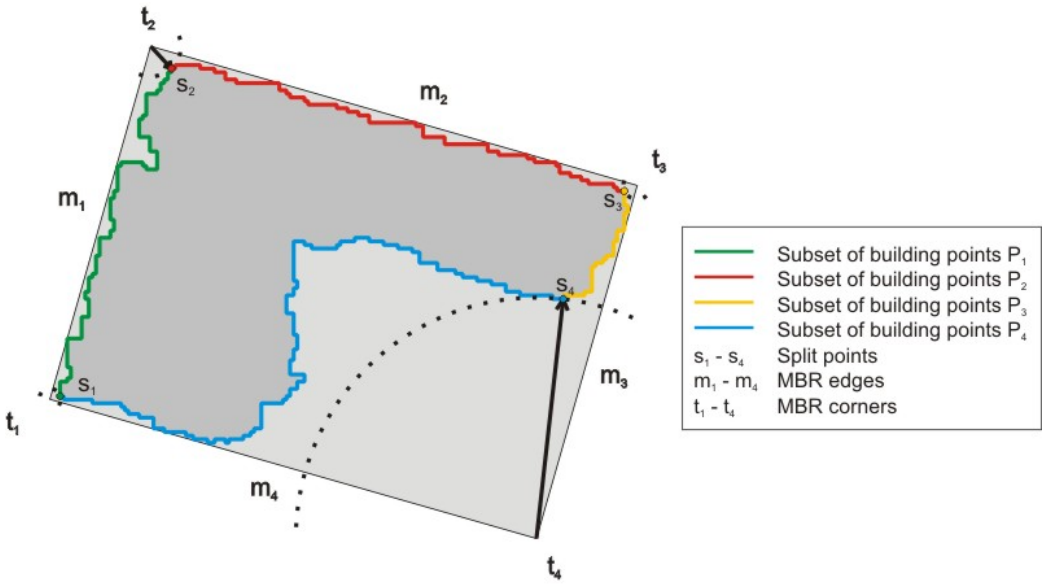


Figure 3.3: Construction of the split points at level 1: The split points are those building points which have smallest distance from the according MBR-corners.

where n is the count of building points in P. In that way the set of building points can be split into four subsets $P = \{P_1, P_2, P_3, P_4\}$. The order of the building points is not changed and the split point is the first point in each subset: $P_i = \{p_{s_i}, p_{s_i+1}, \ldots, p_{s_{i+1}-1}\}$. For example, if the first split point in subset P_1 is building point 5 and the second split point is building point 26, the subset P_1 contains the building points $\{5, 6, \ldots, 25\}$.
The assumption is, that all building points being an element of a single subset, can be well generalized by a straight line, which is parallel to the corresponding MBR-edge. If this is true, the model, that is defined by the characteristics of the split points (and which is a rectangle in level 1), is suitable for the actual building. As a criterion serves the variationparameter V_i. Let m_i be the MBR-edge, that is defined by the MBR-corners t_i, t_{i+1} and $d(p_j, m_i)$ be the nearest distance (euclidean norm) from a point $p_j \in P_i$ to m_i. The set of distances measured from all points $p_j \in P_i$ to m_i is D_i. The

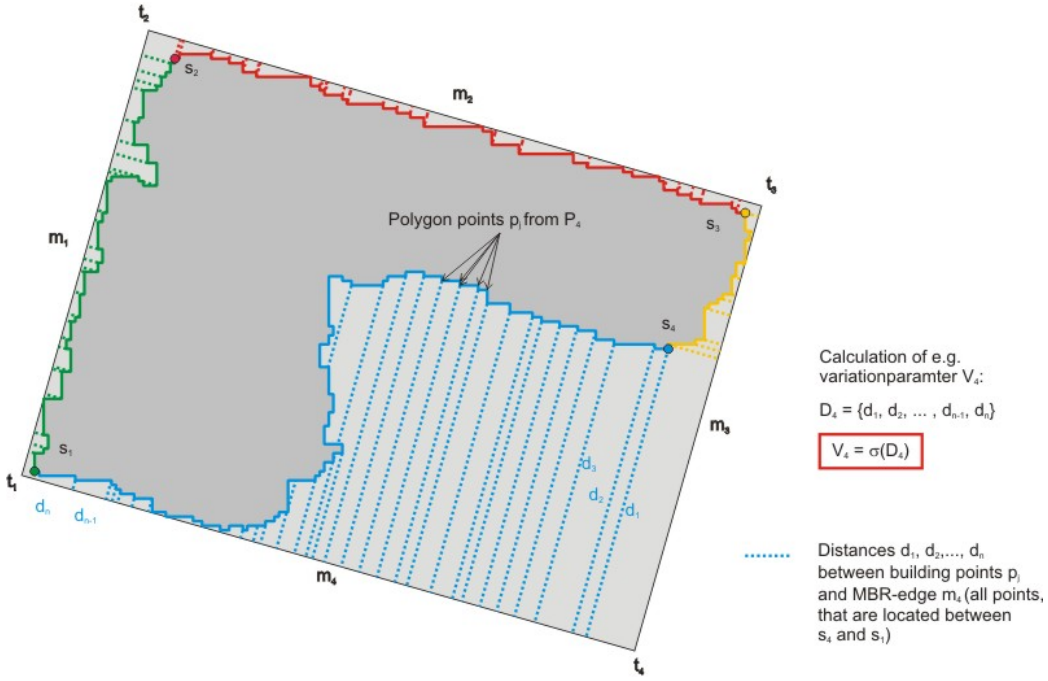


Figure 3.4: Calculation of variationparameter V_4: D_4 is the set of distances from all points $p_j \in P_4$ to m_4. The variationparameter V_4 is the standard deviation of the elements in D_4.

variationparameter V_i is the standard deviation of the elements in D_i:

$$V_i = \sigma(D_i) = \sqrt{\frac{1}{|P_i| - 1} \sum_{j=1}^{|P_1|} (d(p_j, m_i) - \overline{d})^2} \tag{3.2}$$

where $\overline{d}$ is the arithmetic mean of the distances in D_i. In figure 3.4 the calculation of the variationparameter is illustrated.
If for all subsets V_i is smaller than the variationparameter specified by the user V_{spec}, the rectangular model will be accepted. If V_i is larger than V_{spec} for an individual subset, the rectangular model will be replaced by a more complex model as described in the following section.

3.2.3 Level 2

In the example building in figure 3.4, V_4 might be larger than V_{spec}. Therefore level 2 of the generalization process is started.
In this level, the rectangular model is refined by cutting out one or more corners. Depending on how many and which corners are eliminated, the resulting polygon has "L"-, "T-", "Z-"- or a even more complicated form.
The idea is, that if a split point s_i (that has been found in level 1) has a very large distance from its corresponding MBR-corner t_i, it will be suitable to divide the combination of subsets P_{i-1} and P_i into four new subsets. Again the assumption is, that all points being elements of a single subset of the new subsets can be well generalized by a straight line, which is parallel to any of the MBR-edges. If this is true, a suitable model for the generalization of building points in P_{i-1} and P_i will be found.
As a first step the distances from the split points to the correspondent MBR-corners are examined. The user specified minimal length of an edge M_{spec} serves as a reference value. If the distance from a split point to the MBR-corner is larger than M_{spec}, the split point will be replaced by three new split points. If the distance is smaller or equal M_{spec}, the split point will remain in the set of split points and no change will be applied to that corner of the rectangular model:

$$\{s_i\} \Rightarrow \begin{cases} \{s_{i_b}, s_{i_m}, s_{i_f}\} & \text{for} \quad d(s_i, t_i) > M_{spec} \\ \{s_i\} & \text{for} \quad d(s_i, t_i) \leq M_{spec} \end{cases} \qquad i = 1, 2, 3, 4 \qquad (3.3)$$

The new split points s_{i_b}, s_{i_m} and s_{i_f}[2] are determined in the following way:

- The split point s_{i_b} (s_{i_f}) is the first building point being element of P_{i-1} (P_i), that has a smaller distance from the MBR-edge m_{i-1} (m_1) than a specific value (a value of 2 m seems to be adequate with the test data), when each point is tested subsequently starting from s_i going backward

[2] b ... backward, m... middle, f ... forward

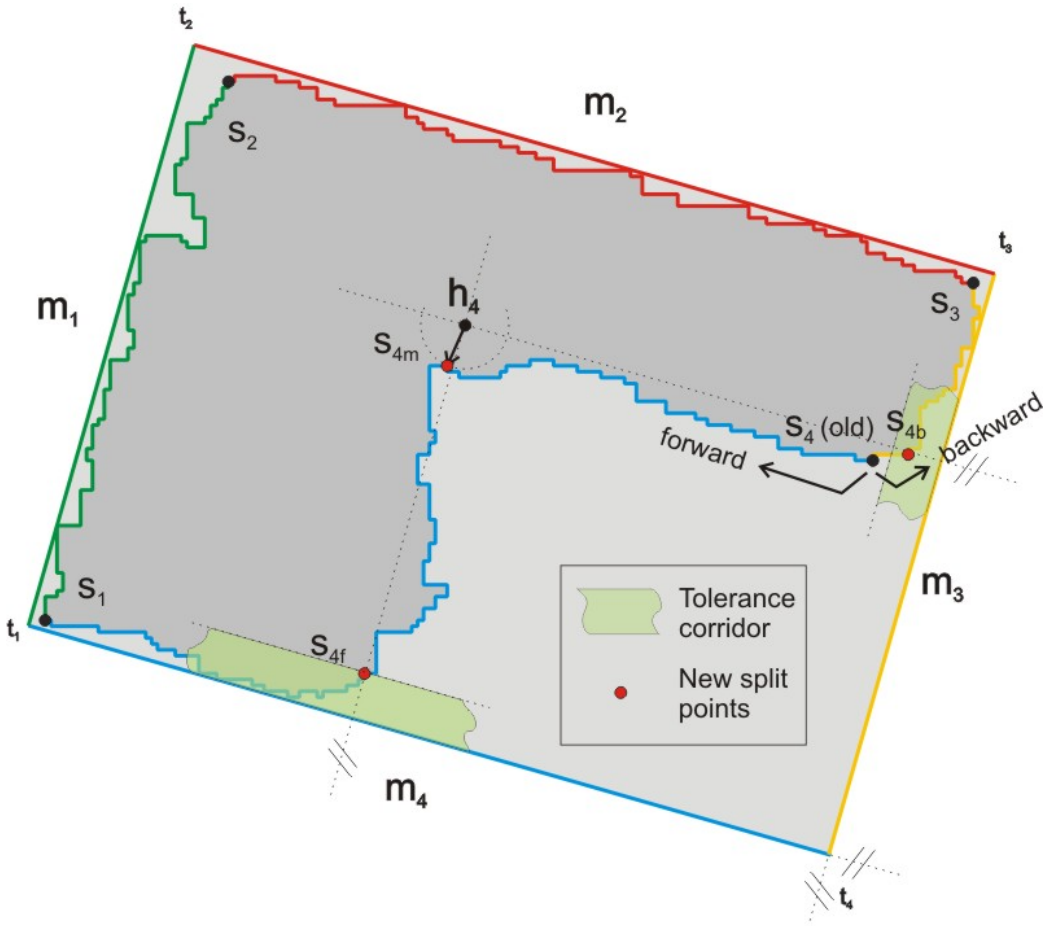


Figure 3.5: Computation of new split points at level 2 (“L”-model).

(forward).

$$\begin{array}{rcllll} s_{i_b} & = & p_j \in P_{i-1} & \text{with} & d(p_j, m_{i-1}) < const & \wedge \quad j = \max \\ s_{i_f} & = & p_j \in P_i & \text{with} & d(p_j, m_i) < const & \wedge \quad j = \min \end{array} \tag{3.4}$$

- Intersection of the line that goes through s_{i_b} and is parallel to m_i and the line that goes through s_{i_f} and is parallel to m_{i-1} results in the intersection point h_i. This can easily be formalized by defining a coordinate system which has its origin in t_i, the positive x-axis through t_{i-1} and the positive y-axis going through t_{i+1}. If $(x(s_{i_b}), y(s_{i_b}))$ are the coordinates of s_{i_b} and $(x(s_{i_f}), y(s_{i_f}))$ are the coordinates of s_{i_f}, then h_i will be defined as follows:

$$h_i = (x(s_{i_b}), y(s_{i_f})) \tag{3.5}$$

- The split point s_{i_m} is defined as the point being element of P_{i-1} or P_i, and having the smallest distance to h_i:

$$s_{i_m} = p_j \in \{P_{i-1}, P_i\} \quad \text{with} \quad d(p_j, h_i) < d(p_k, h_i) \qquad k = 1, \ldots, j-1, j+1, \ldots, n \tag{3.6}$$

  where n is the count of points in the union of the subsets P_{i-1} and P_i.

After iterating through all s_i with $i = 1, 2, 3, 4$, either no new split point, or 3, 6, 9 or 12 new split points are found. They are inserted into the set of split points considering the correct order whereas the corresponding s_i is eliminated from the set of split points. The count of split points is now $4 + 2 \cdot v$, where v is the count of corners, that have been cut out from the rectangular model.
Once the level 2 split points are found, the procedure is the same as described in level 1. By means of the split points, the set of building points P is divided into subsets P_i. It is assumed, that points being element from a single P_i, can be well generalized by a straight line. This is verified by calculating the variationparameter V_i of the subset P_i and by comparing it with V_{spec}. If one

Figure 3.6: Four split points have been found in level 1. After running through level 2 no split points have been added, because they are located very close to the MBR-edges. The variationparameter V_2 is still larger than V_{spec}. A "U-"-model will be fitted in level 3.

of the V_i is larger than V_{spec}, the straight line is not a suitable generalization model for that subset and it will be further refined in the next level. If all V_i are smaller than V_{spec}, the model, which is defined by the split points, will be accepted as a suitable model for the actual building and no further refinements of the model will be needed. In this case, the level 3 is skipped. A visualization of the calculation steps, described so far, can be seen in figure 3.5. In the example building, the split point s_4 is replaced by three new split points. The count of split points is now 6 and they determine a "L"-form (see figure 3.5). All subsets pass the variationparameter criterion. For this reason the process of finding a suitable model is terminated at this level and the algorithm goes on with the calculation of the position of the final generalized edges (described in section 3.2.5).

3.2.4 Level 3

Figure 3.6 illustrates another example building and shows the location of the split points after running through level 1 and level 2. Obviously the split points s_1 to s_4 have been found already in level 1. From the result of the variationparameter test ($V_2 > V_{spec}$) in level 1 it followed, that the rectangular model had to be refined and for this reason level 2 was started.

But all split points are located close to the corresponding MBR-corner (they were not detected by the test in level 2) and therefore, the corners of the model should not be changed. Since V_2 is still larger than V_{spec}, level 3 is started.

The basic idea of level 3 generalization is described in the following. If two successive corners of a polygon are essential points (justified by level 2) and if there is large variation in the raw building, there might be two reasons: Either the large V_i can be attributed to large error in the raw data, or the geometry of the concerning building edge is more complex than assumed in level 2. In this case the generalization should be enhanced by adding more line segments. Since only rectangular angels are allowed, the simplest refinement is cutting out a rectangle, which means replacing the straight line by 5 shorter line segments. If other angle values besides 90° occur in this part of the building, it will show up as a very noisy building polygon to the algorithm and will not be generalized satisfactorily.

This idea is realized by a search for five new sets of building points that will replace a single subset from level 2. That means, if V_i of subset P_i (from level 2) is too large, the aim of level 3 is to find and add four new split points $s_{i_1}, s_{i_2}, s_{i_3}, s_{i_4}$ out of this set of building points:

$$\{s_i\} \Rightarrow \begin{cases} \{s_i, s_{i_1}, s_{i_2}, s_{i_2}, s_{i_4}\} & \text{for} \quad V_i > V_{spec} \\ \{s_i\} & \text{for} \quad V_i \leq V_{spec} \end{cases} \qquad i = 1, \ldots, n \qquad (3.7)$$

where n is the count of split points resulting from level 2.

Again, all building points from a single subset will be generalized by a straight line. In that way, the straight line, that was the model for P_i in level 2, is in level 3 replaced by 5 lines, which form a "U".

The first step is the determining of a value U_i, that gives evidence, wether the model described above is appropriate for the refining of P_i. For this purpose the points of P_i, which have the largest distance from the MBR-edge, are observed. If the proportion of those points in relation to the totality of the points in P_i is greater than a reference value U_{ref}, it will be assumed, that one of the new segments is shifted to the interior of the MBR. In this case

the new split points will be calculated. In the following the calculation of U_i is described.

D_i has already been defined as the set of distances $d(p_j, m_i)$, where p_j are all points in subset P_i and $i = 1, \ldots, 4 + 2 \cdot v$ (see section 3.2.3). However there are only four MBR-edges. For this reason the notation has to be changed. The distances in D_i are now measured between a point p_j and MBR-edge m_1 or m_2. So D_i is now the set of distances $d(p_j, m_w)$, where p_j are all points in subset P_i and $w = 2 - i \cdot \bmod 2$. Let n be the count of points in P_i. The distances in D_i are transformed into a histogram. Therefore the elements of D_i are sorted in ascending order and afterwards divided into a certain amount of classes (C_1 to C_c). Typically the count of classes c of a histogram is chosen in dependency of the count of samples (Staudinger 2003):

$$c = \begin{cases} 5 & \text{für} \quad n < 25 \\ \sqrt{n} & \text{für} \quad 25 \leq n \leq 100 \\ 1 + 4.5 \cdot \lg n & \text{für} \quad n > 100 \end{cases} \tag{3.8}$$

The smaller the count of classes, the clearer the histogram, but the more information gets lost. If, on the other hand, the count of classes is large, an accumulation of values in the last class (which is an indication for a "U") might remain undetected. For this application the samples are the distances in D_i. With the available test data the count of distances in D_i varies from approximately 50 to 150 points. Consequently 7 to 10 classes should be created. In consideration that an eventual accumulation of values in the last class should necessarily be detected, a very small number of classes is chosen, namely 6. The class width is calculated by dividing the range of D_i by the required count of classes: $(d_{max} - d_{min})/6$.

The elements of the last class C_6 (which contains the largest values of D_i) are counted. The ratio of the count of elements of C_6 to n (the count of D_i) is the value that leads to an estimation of the geometrical structure of the subset of building points P_i:

$$U_i = \frac{|C_6|}{n} \tag{3.9}$$

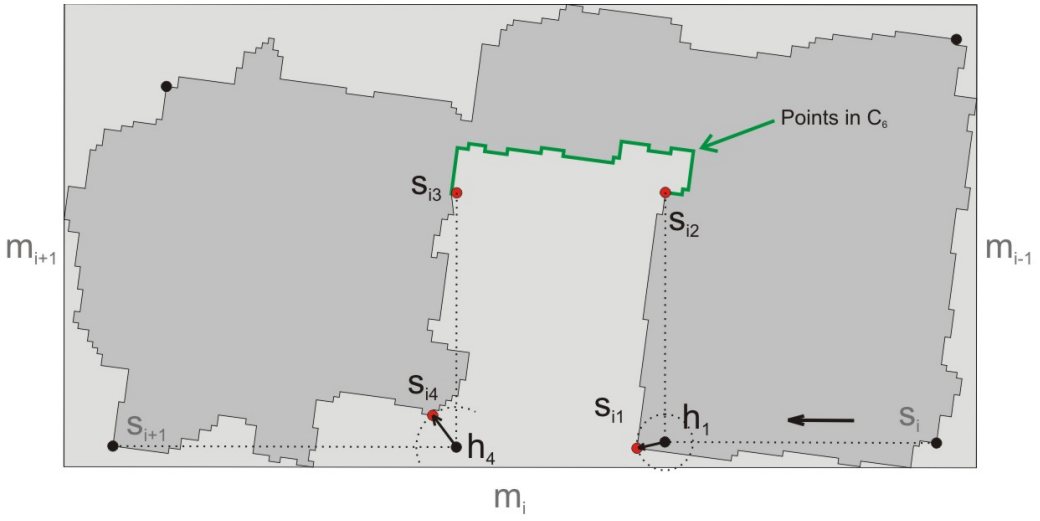


Figure 3.7: Construction of the third level split points

If the calculated ratio is less than a certain constant U_{ref} (with the available data a value of $U_{ref} = 11\%$ turns out to be an adequate value to detect an accumulation of distances in this class), either the data is of poor quality (meaning that there is too much variation in the elements of D) or the geometry of the building is more complex than expected. In that case no satisfying generalization can be processed by means of this method. For this reason the building will be labeled for manual editing.
However, if U_i is larger than U_{ref}, it is assumed, that the building points can be modeled by edges forming a "U" and the search for the new split points is started. This process is described in the following.

- Looping through the elements of P_i, the first and last occurrence of a point being element of subset C_6 is stored as new split point s_{i_2} and s_{i_3}:

$$\begin{aligned} s_{i_2} &= p_j \in P_i \quad \text{with} \quad p_j \in C_6 \quad \wedge \quad j = \max \\ s_{i_3} &= p_j \in P_i \quad \text{with} \quad p_j \in C_6 \quad \wedge \quad j = \min \end{aligned} \tag{3.10}$$

  If a very noisy building polygon is computed, it might happen that the first detected point being an element of C_6 will be located far away from the essential corner point of the "U"-form. But this happens very seldom and is accepted in the face of a preferably simple algorithm.

- Intersection of the line that is parallel to MBR-edge m_i and goes through s_i (s_{i+1}) with the line that is parallel to MBR-edge m_{i+1} and goes through s_{i_2} (s_{i_3}) results in the intersection point h_1 (h_4).

- Searching through all building points located between s_i and s_{i_2} (which is the point subset $\{p_{s_i}, p_{s_i+1}, \ldots, p_{s_{i_2}-1}\}$) for the nearest point relative to the intersection point. This gives the new split point s_{i_1}:

$$\begin{aligned} s_{i_1} = p_j \quad \in \quad & \{p_{s_i}, p_{s_i+1}, \ldots, p_{s_{i_2}-1}\} \quad \text{with} \quad d(p_j, h_2) < d(p_k, h_2) \\ & k = 1, \ldots, j-1, j+1, \ldots, n \end{aligned} \tag{3.11}$$

  where n is the count of points in subset $\{p_{s_i}, p_{s_i+1}, \ldots, p_{s_{i_2}-1}\}$. In the analogous way s_4 is found:

$$\begin{aligned} s_{i_4} = p_j \quad \in \quad & \{p_{s_{i_3}}, p_{s_{i_3}+1}, \ldots, p_{s_{i+1}-1}\} \quad \text{with} \quad d(p_j, h_3) < d(p_k, h_3) \\ & k = 1, \ldots, j-1, j+1, \ldots, n \end{aligned} \tag{3.12}$$

  where n is the count of points in $\{p_{s_{i_3}}, p_{s_{i_3}+1}, \ldots, p_{s_{i+1}-1}\}$.

After all new split points are detected, they are added to the set of split points from level 2 (considering the correct order). In the same way as in level 1 and 2, it has to be checked if the subsets P_i (defined by the split points s_i) can be well generalized by straight lines. If V_i is smaller than V_{spec} for all P_i, than the refined model will be accepted and the location of the generalized edges will be calculated (see next section). However, if V_i is larger for any subset P_i, the quality of a generalization by means of this model will be assumed to be not good enough. Since no further refinement of the model is implemented, no enhancement of the generalization is possible by means of this algorithm.

3.2.5 Computation of the building footprint

So far the best fitting model has been selected (in level 1, 2 or 3) by assigning each point of the building to a subset of points P_i and by assuming, that the points of one subset can be well generalized by a straight line, that is parallel

to one of the MBR-edges. Now the position of the straight line segment is determined by the distances e_i (the "offset") to the corresponding MBR-edge (figure 3.8). For e_i, the median of the distances in D_i is taken:

$$e_i = \begin{cases} d_{(n_d+1)/2} & d_{(n_d+1)/2} \in D_{i_{ord}} \quad n_d \text{ odd} \\ \frac{1}{2}(d_{n_d/2} + d_{n_d/2+1}) & d_{n_d/2}, d_{n_d/2+1} \in D_{i_{ord}} \quad n_d \text{ even} \end{cases} \qquad i = 1, \ldots, n \tag{3.13}$$

where $D_{i_{ord}}$ are the distances in D_i ordered by value, n is the count of split points and n_d is the count of points in P_i or rather the count of distances in D_i. This is the offset of the generalized line in relation to the correspondent MBR-edge. Other descriptive values could be taken instead of the median. But it has to be taken into consideration that the distribution of the distances is not normal. Only in case that the building polygon is an exact rectangle and the "raster"-structure can be handled as random errors, the distances represent a normal distribution. The task is to determine an accumulation of values, that represents the generalized position of the edge. The median is a more "robust" measure and is useful to detect an existent accumulation. The final vertices u_i of the generalized polygon are the intersection points of two successive generalized lines. The coordinates of the vertices u_i can easily be calculated in a local coordinate system with origin in t_2 and the positive x-axis oriented through t_1:

$$u_i = (e_{i-i \bmod 2}, e_{i-1+i \bmod 2}) \qquad i = 1, \ldots, n \tag{3.14}$$

where n is the count of split points. The result of the intersection is shown in figure 3.9.

Elimination of too short edges

Some edges of the generalized polygon may be very short, as shown in figure 3.10 (a). They must be eliminated in order to provide meaningful results. Therefore, the edges of the generalized polygon are tested for their length, starting with the first edge. Those edges, that are shorter than the value

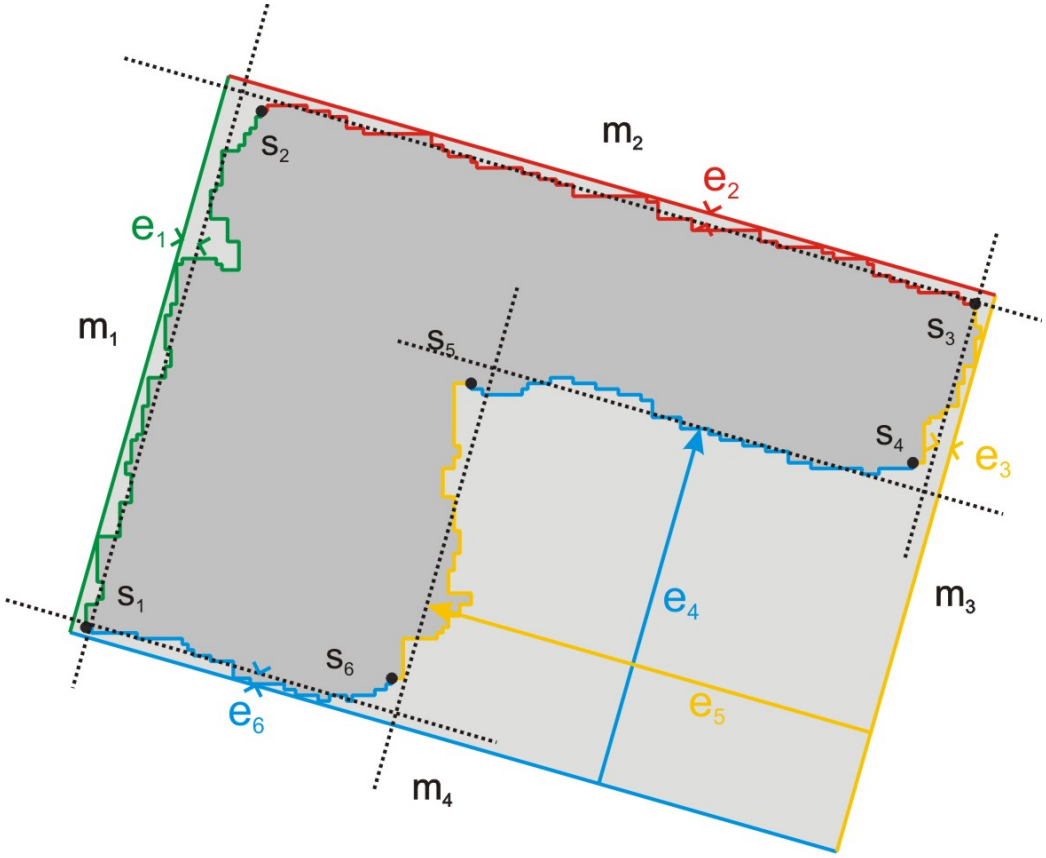
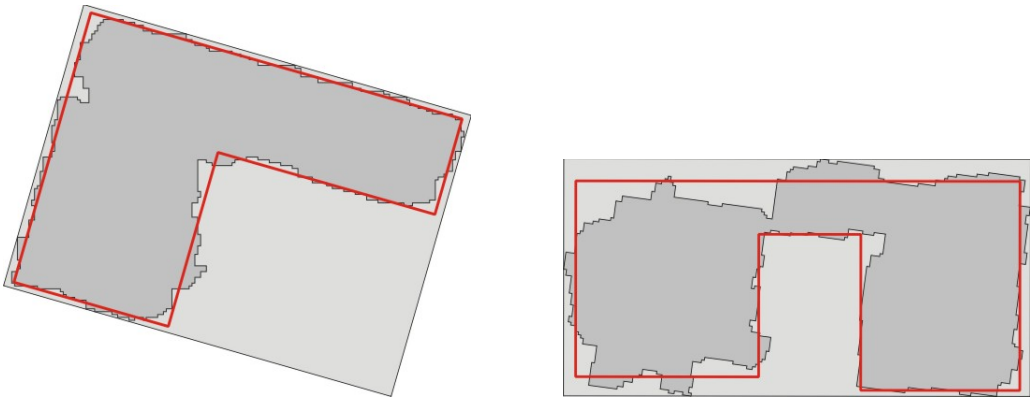
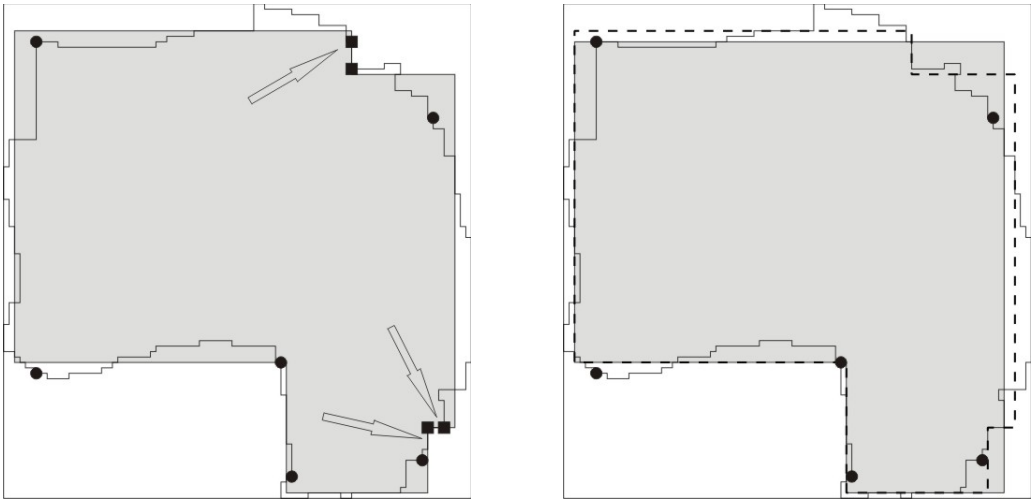


Figure 3.8: Computation of the final location of the edges



(a) Building from figure 3.5 (b) Building from figure 3.7

Figure 3.9: Result of generalization



(a) Before elimination: all edges, which will be eliminated are indicated by arrows. For this purpose the split point symbolized by rectangles are removed from the split point list

(b) After elimination

Figure 3.10: Elimination of short edges

specified by the user (minimal length of edge M_{spec}), are eliminated. This is realized in the following way: If edge i (which belongs to split point s_i) is too short, the split points s_i and s_{i+1} will be removed from the set of split points. If two successive edges, which belong to s_i and s_{i+1}, are too short, the same action will be taken: s_i and s_{i+1} are eliminated. The point subsets P_i, $i = 1, \ldots, n$ are reorganized and the position of the generalized polygon edges is recomputed. The result of the edge elimination is illustrated in figure 3.10 (b).

3.2.6 Built-in validation

The generalization process is finished after the eliminiation of too short edges. Through the elimination of split points in the last step, the variation parameter values V_i may have changed. For that reason another quality test is accomplished by recalculating the V_i. The generalized polygons are classified into two groups: the valid generalized polygons (V_i is smaller than V_{spec}

for all i) and the polygons, which require further manual processing (V_i is larger than V_{spec} for at least one i). If the quality criterion is not met, either the building structure will be too complex, the building edges will be too noisy or the orientation of the building was not detected correctly. Reasons for the problems and possible improvements of the method are presented in section 6. The poorly generalized buildings are labeled as "insufficiently generalized". They have to be edited manually by the user.

Chapter 4

Implementation and practical application

The building generalization algorithm presented in the previous chapter was embedded in a work flow, including a graphical user interface, data input, data output, a data preprocessing tool and a tool for facilitating the manual post processing of the buildings. The so-called *generalization tool* was implemented as a Makro in ArcGIS[1]. It is a semi-automatic generalization tool, because those buildings, which can not be automatically generalized by the algorithm in a satisfying manner, must be edited manually by the user. The advantages of implementing the tool in ArcGIS are the widespread use of the software on the one hand and the availability of an integrated VBA (Visual Basics for Applications) software development environment on the other hand.

The tool takes a polygon shape file containing the building polygons as input data and produces a new polygon shape file containing the generalized building footprints.

The command buttons necessary to start the tool are located on a separate toolbar in the main application window of ArcGIS.

The generalization tool consists of two parts. First an automatic generalization is processed by means of the presented generalization method. The

[1]ESRI ArcGIS Desktop 9.1 (`http://www.esri.com`)

generalized buildings are saved as a new polygon shape file. The second part is the manual editing of the buildings, which could not be generalized with satisfying results.

In this chapter the application of the generalization tool is demonstrated. First of all, the study area and data is presented. Afterwards the procedure of generalizing this set of building polygons from the user's point of view is described.

4.1 Study area and data

The data used for the practical demonstration is a set of 315 vectorized building polygons. It is a sub area of the data, that was used during the development of the tool. The test area covers an area of about one square kilometer and is located in Hall in Tirol, Austria (see figure 4.1).
The building outlines are a result of an object oriented classification of a true-colored orthophoto followed by a vectorization. The classification was accomplished by means of the software Definiens Professional[2]. No height information was introduced into the classification process. The ground resolution of the orthophoto was 0.25 m. The regions classified as buildings where vectorized and exported into a shapefile. The smallest distance between two neighboring points is therefore 0.25 m and the segments of the building polygons are oriented in direction of the image raster.

4.2 Work flow

4.2.1 Automatic generalization

All command buttons of the generalization tool are located on a separate toolbar (figure 4.2 (a)), which must be activated in the current project. The shapefile with the building polygons is loaded as a layer into the active data

[2]formerly known as eCognition (`http://www.definiens.com/`)

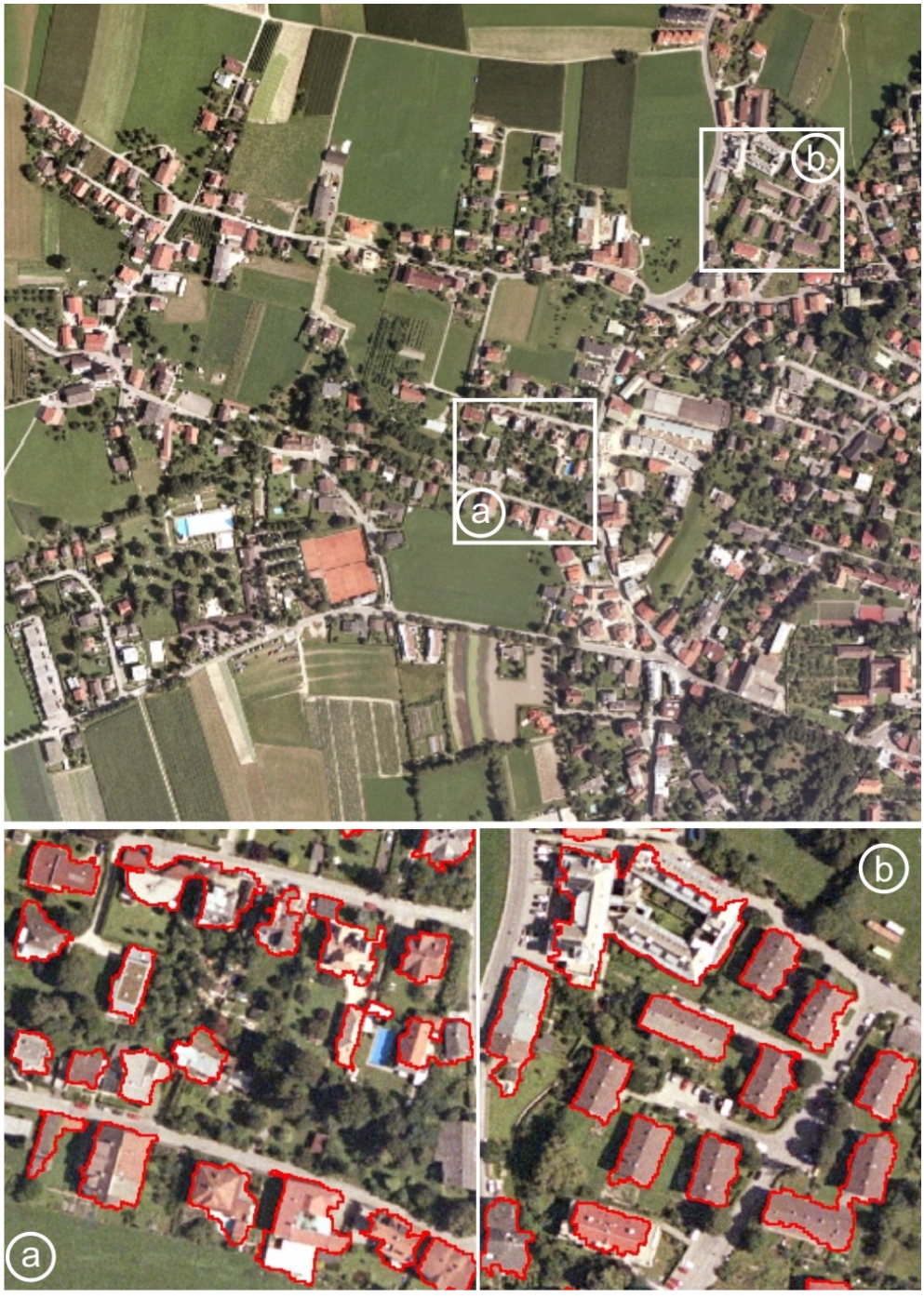
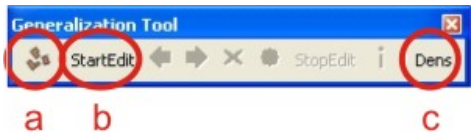


Figure 4.1: Test area Hall in Tirol. The ground resolution of the true-colored orthophoto is 0.25 m. In the more detailed figures a and b the building polygon data is added.



(a) All commands are located on a separate toolbar (a - start of the automatic generalization, b - editing tool, c - densification function).

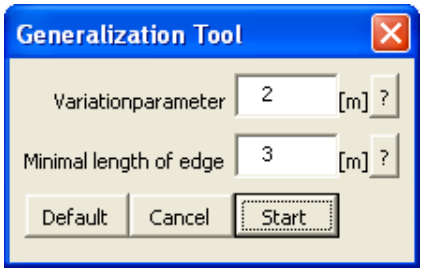
(b) The shapefile with the building polygons is loaded into the data frame. The layer is selected.

Figure 4.2: Toolbar and work flow in ArcGIS

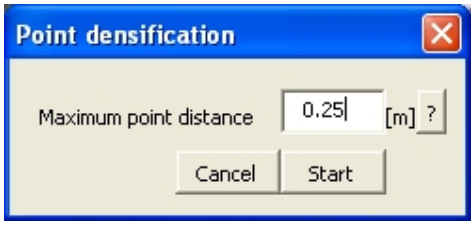
frame and the layer is selected.

In order to facilitate the fulfilling of the constant point density prerequisite, a point densification function is integrated in the generalization toolbar. The function interpolates points linearly into the boundary in such a manner, that approximate equidistance between neighboring points is achieved. The user must specify a maximal point distance. As a maximal point distance, the least point distance occurring in the data should be chosen. If the value is chosen higher, there will be irregularities in the point distribution. If the value is chosen smaller, the data volume will be unnecessarily large and therefore the processing time of the generalization that follows will raise. In order to achieve equidistance between neighboring points along the polygon boundary (which is a prerequisite of the data), the distance function is executed with the data. It is started by clicking on the "Dens"-button in the toolbar (figure 4.2 (a) - button c). The maximal accepted point distance has to be entered in the appearing input window (figure 4.3 (b)). As a maximal point distance 0.25 m is chosen because the polygon data is derived from 0.25 m raster pixels. The function produces a new shape file containing the densified polygons. The shape file is automatically loaded into the data frame by the function. The count of polygon points triples approximately due to the densification.

The new shapefile must be selected. The input window for the user specified



(a) The user interface of the generalization tool.



(b) Input window of the densification function.

Figure 4.3: User interfaces of the generalization tool and the densification function

values is activated by clicking the button in the toolbar (figure 4.2 (a) - button a). The window appears, where the variationparameter and the minimal length of an edge must be specified (figure 4.3 (a)). The following values are chosen:

Variationparameter:	2 m
Minimal length of an edge:	3 m

The automatic generalization is started by activating the "Start"-button. The building polygons are generalized by means of the algorithm described in chapter 3. First the MBRs are calculated and afterwards the three generalization levels are run through. A quality check is done using the two user specified values. Finally the generalized polygons are computed and saved in a new shapefile. Two columns are added to the table of the new shapefile (figure 4.4). The content of both columns is identical. The first column ("Critical") is generated for documentation purposes. The content of this column will not be changed during the editing process. The second column ("NotEdited") is required by the editing tool. A "1" in this column indicates, that the building is not satisfyingly generalized. The built-in validation of the tool (which works with the variationparameter) flags 37 buildings "NotEdited". These buildings have to be generalized manually. The new shapefile is automatically loaded into the data frame. Processing time of the automatic generalization is approximately one minute with a standard system (AMD Athlon(TM)XP1800+, 1.54 GHz, 1 GB of RAM).
Examples of the generalization result are shown in figure 4.5 and figure 4.6.

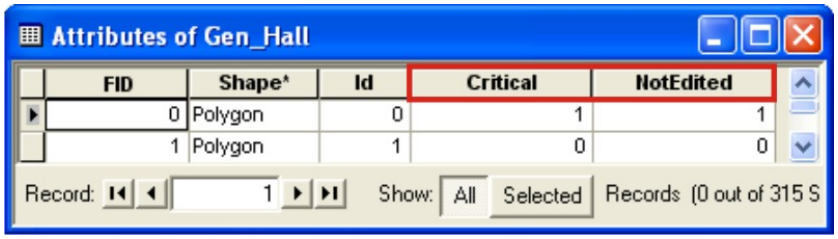


Figure 4.4: Two columns are added to the table of the new shapefile by the algorithm.

The complete result is documented in the appendix (figure A.3 to figure A.13). The figures cover the whole test area. The generalized polygons are combined with the original polygons and the orthophoto. Figure A.1 gives an overview about the location of the tiles.

4.2.2 Manual editing

The buildings which are marked in the "NotEdited"-column, have to be edited manually (some example buildings from the test data can be seen in figure 4.7).
In order to facilitate the manual post processing, an editing tool was integrated into the generalization toolbar. After selecting the shapefile containing the generalized buildings, the edit session is started by activating the "StartEdit"-Button (figure 4.2 (a) - button b). Now the editing state is checked by the tool. If there exists no columns "NotEdited" in the shapefile-table, the session is canceled. If no values "1" are left in the columns, the edit session is terminated - there are no buildings left that have to be edited. If there are values "1" in the "NotEdited"-column, the first building that has a "1" is selected and additionally a centering and zooming operation is applied in the active view on that building. Now the built-in editing functionality of ArcGIS can be used in order to adapt the geometry of the generalization. The vertices of the polygon can be moved or deleted. New vertices can be inserted.

While the user edits one building after the other, guided by the tool, the

Figure 4.5: Detail of the generalization result with and without orthophoto. Blue polygons are valid generalizations, the ruled polygon has to be edited manually.

Figure 4.6: Detail of the generalization result. Blue polygons are valid generalization, the ruled polygon has to be edited manually.

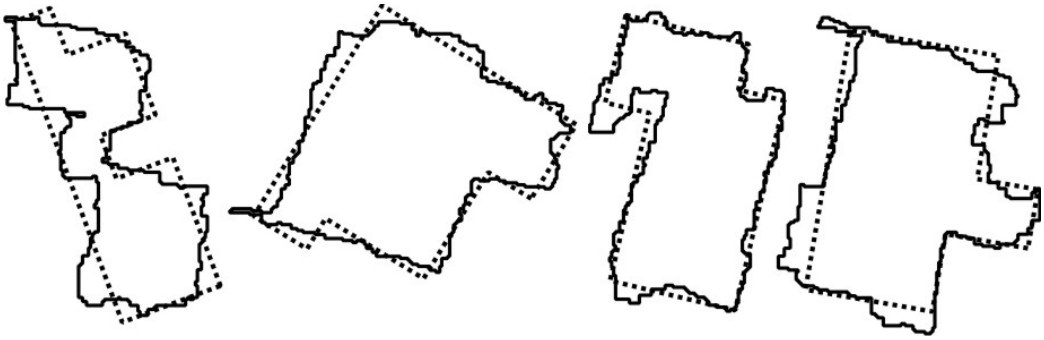


Figure 4.7: Example of buildings that are marked by the tool as insufficiently generalized.

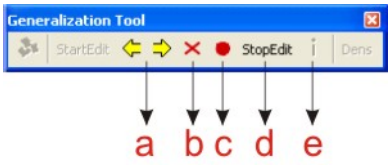


Figure 4.8: Functionality of the edit tool

following editing functionality is available:

- *Selecting the next or previous polygon*: By a click on the "Next" or "Previous"-button (figure 4.8 - a), the active view is automatically centered on the next or previous building, that is flagged in the "NotEdited"-column. A mouse-over tool tip provides information about the ongoing process.

- *Saving the intermediated editing state*: If the "Save"-button is activated (figure 4.8 - b), all changes in the geometry are saved (the tool executes the ArcGIS Editor command "Save Edits") and the value in the "NotEdited"-column of all buildings, that have been shown until this moment, is changed to "0". The edit session remains open and the editing process is continued. In order to prevent loss of changed data and loss of the progress information in the "NotEdited"-column, it is recommended to save the work from time to time, especially if a large dataset is processed.

- *Saving the intermediated editing state and closing the edit session*: If

the user wants to interrupt the editing process, he should save the work and also save the progress by flushing "0" into the "NotEdited"-column. In this way it is possible to continue the edit session with the building after the last building, that has been edited so far. This is exactly what the "StopEdit"-button does (figure 4.8 - d). The edits and the progress are saved, and the edit session is closed. To start a new edit session, the "StartEdit"-button must be activated again.

- *Deleting polygons*: Furthermore, a function is available, which deletes the actually selected polygon (figure 4.8 - b). It is important to use that function (and not to simply delete the object by pressing the "Del"-key), because it undertakes the updating of the list of buildings, that must be edited.
  It has to be considered, that if a building is deleted, the FIDs of the shapefile will of course be updated by the GIS. The FIDs of the original building polygons and the corresponding generalized polygons are not identical any more. If the user is interested in preserving the connection between original and generalized polygon, he should create a column in the shapefile of the generalized buildings (before the editing is started!), containing copies of the IDs in the original building shapefile.

- *Displaying the FID of the actual polygon*: When pointing to the "i"-field on the toolbar (figure 4.8 - e), a tool tip provides information about the FID of the actual selected building.

After editing all buildings, the changes are saved by pressing the "StopEdit"-button. No "1" are left in the "NotEdited"-column.

The edit tool provides a useful help for the manual editing of the buildings. But some more functionality would be helpful, like for example the insertion of new polygons: it is more time-consuming to move all vertices of a polygon than to create a new one. For that it must be guaranteed, that the new building gets the same FID as the deleted one. This functionality would require a more comprehensive strategy for the editing tool. Another problem

is an eventual inconsistency in the data, when the functionality of the ArcGIS Editor and the editor tool described here interfere. This may lead to software crashes, if the user is not aware of the difficulties.

Chapter 5

Validation

5.1 Quality measurement

After completing the automatic generalization and the manual editing process with the test data, the quality of the generalized building footprints is validated. Quality parameters are needed, in order to make a statement about the quality of the generalization.
In the following section, three quality criteria are presented.

5.1.1 Point distance measures

The distances from equally distributed points along the original polygon to the generalized polygon give a useful measure for the distance of both polygons. From the set of calculated distances, statistical quantities like the the maximum distance, the mean distance, the standard deviation of the distances, etc. can be calculated. They serve as scalar descriptors for the similarity of original and generalized polygon.
Let A be a polygon that should be generalized. It is an infinite bounded subset of $\mathbb{R}^2$. Let a_n be a finite, ordered subset of points of A. The points should be equally distributed along the boundary, i.e. of the length of all line segments between two neighboring points must be equal.
Let G (also an infinite bounded subset of $\mathbb{R}^2$) be the generalized polygon.
The distance $d(a_k, G)$ from the point a_k to the boundary G is defined as the

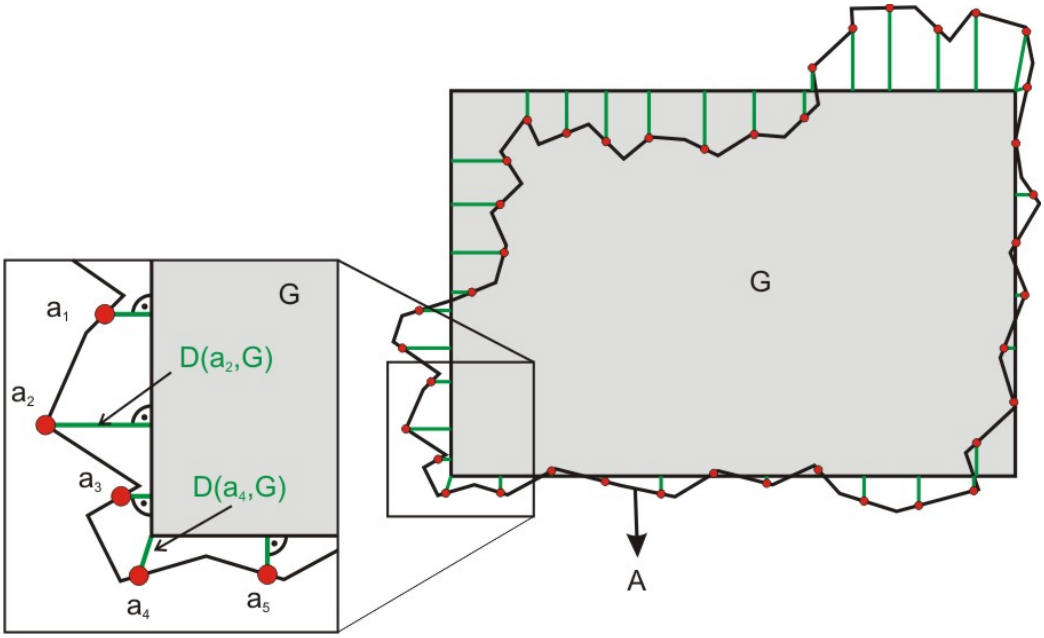


Figure 5.1: Point distance measure: The nearest distances from equally distributed points a_k (along the original polygon boundary A) to the generalized polygon G are calculated. The smaller the mean distance, the higher the similarity between original polygon and generalized polygon.

euclidean distance between the point and the nearest point (in terms of the euclidean norm) which is part of the boundary.
All distances are calculated. These values serve as a quality measure for the similarity and distance of the generalized building in relation to the original data. We can for example calculate the maximum distance, the mean disthgggnance, the standard deviation etc. The smaller the mean distance, the greater the "similarity" between the original and the generalized polygon.

5.1.2 Hausdorff distance

The Hausdorff distance is a measure of the degree of mismatch between two sets. It is used in the field of pattern recognition and computer vision for determining the difference or similarity of shapes (Huttenlocher, Klanderman, and Rucklidge 1993).

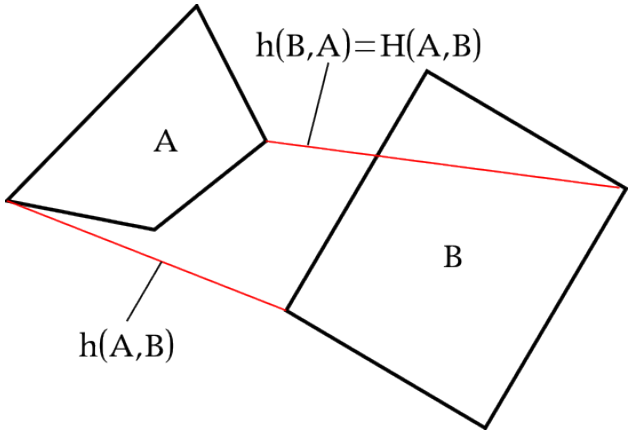


Figure 5.2: Hausdorff distance

Given two finite point sets $A = \{a_1, \ldots, a_p\}$ and $B = \{b_1, \ldots, b_q\}$, the Hausdorff distance is defined as:

$$H(A, B) = \max(h(A, B), h(B, A)) \tag{5.1}$$

where

$$h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\| \tag{5.2}$$

The term $\|a - b\|$ is a distance function, e.g. the euclidean distance. The function $h(A, B)$ is called the *directed* Hausdorff distance from A to B. It identifies a point $a \in A$, that is farthest from any point of B, and returns the distance from a to its nearest neighbor in B using the distance function $\|a - b\|$. If $h(A, B) = d$, then each point of A must be within distance d of a point of B. The Hausdorff distance $H(A, B)$ is the maximum of $h(A, B)$ and $h(B, A)$. It measures the distance of the point of A that is farthest from any point of B and vice versa.
Let K be the infinite set of all closed, bounded set of the $\mathbb{R}^n$ (e.g. a set of building polygons in $\mathbb{R}^2$). Analog to the euclidean distance, which is a metric over the elements of $\mathbb{R}^n$, the Hausdorff distance is a metric over K. The Hausdorff distance maps an element of $\mathbb{R}$ to every pair of sets that are elements of K.

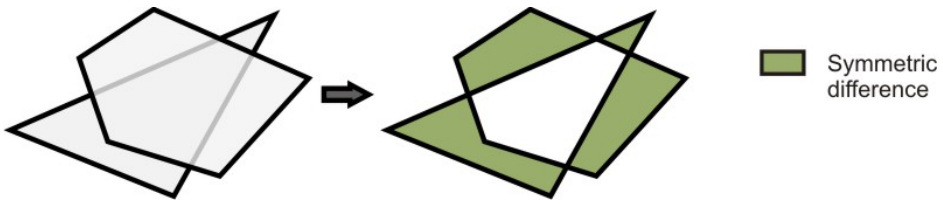


Figure 5.3: Symmetric difference of two sets.

5.1.3 Area of symmetric difference

The symmetric difference, coming from set theory, is employed e.g. for spatial analysis in GIS. The symmetric difference $A \triangle B$ of two sets A and B is defined as:

$$A \triangle B = (A - B) \cup (B - A) \tag{5.3}$$

It is the set of elements, which are in one of the sets but not in both (see figure 5.3).
The area of symmetric difference serves as useful distance measure and thus as a measure for the amount of generalization. The greater the area of symmetric difference of the original building and the generalized building, the greater is the effect of generalization.

5.2 Validation of the presented generalization algorithm

In this section the results of the validation process are presented. The quality of the automatic generalization is first inspected manually. Furthermore, distance respectively similarity values of original and generalized buildings are validated. The results of the built-in validation are reported and afterwards analysed with respect to its significance.

(a) Example of merged buildings due to segmentation errors.

(b) Another example of merged buildings.

(c) Noisy building outlines due to classification inaccuracy

Figure 5.4: Error in building classification

5.2.1 Errors in the building classification

Most of the buildings located in this part of Hall are detached houses and have a simple geometry. These houses enable a good classification result. Some buildings are merged together and are therefore classified as a single building. Sometimes there is no good result achieved in these cases, because merged buildings have a complex structure, that can not be modeled by the algorithm. Some actually detached houses are merged together into a single building polygon because of classification errors (e.g. a farm building in figure 5.4(a), or two actually separated family houses in figure 5.4(b)). This causes problems for the generalization algorithm, because every building polygon is expected to be a single building. Complex building structures, as they result from merging errors, can not be handled appropriately. In many cases the vectorized polygon is noisy because of classification inaccuracy (see figure 5.4(c)). This can be corrected by the algorithm in many cases.

5.2.2 Evaluation of automatic generalization

It must be differentiated between a comparison of the generalized buildings with the original building polygons coming from the image classification process and a comparison of the generalized buildings with the original orthophoto or the cadastre. Since the algorithm processes information only

from single building polygons and does not use additional information about neighboring buildings or the surrounding area (e.g. the street network), the evaluation is limited to a comparison between the generalized building and the raw vectorized building polygon. The following criteria are tested:

- Manual evaluation (intuitive evaluation)
- Mean point distance
- Area of symmetrical difference
- Variationparameter (built-in evaluation)

The mean point distance and the area of symmetrical difference are computed only for those buildings, that fulfilled the built-in quality test.

Manual evaluation (intuitive evaluation)

All generalized buildings are visually compared with the original building polygon and inspected in respect to their suitability to serve as base data for a 3D visualization.
From 315 generalized buildings (using a variationparameter of 2 m and a minimal edge length of 3 m), 29 buildings are manually classified as "insufficient generalized". This is a percentage of 9.2%. All other buildings are, from the user's point of view, qualified to serve e.g. as a base for a modelling of 3D-building-models. The count and the percentage of bad generalized buildings

Variationparameter [m]	**1**	**1.5**	**2**	**2.5**	**3**	**3.5**	**4**
Count of buildings	42	48	29	55	66	42	47
Percentage of buildings %	13.3	15.2	9.2	17.5	21.0	13.3	14.9

Table 5.1: Count and percentage of buildings, that are manually classified as "insufficient generalized" (validated with different values for the variationparamter).

with all tested values for the variationparameter are listed in table 5.1 and a chart of the percentage can be seen in figure 5.5.

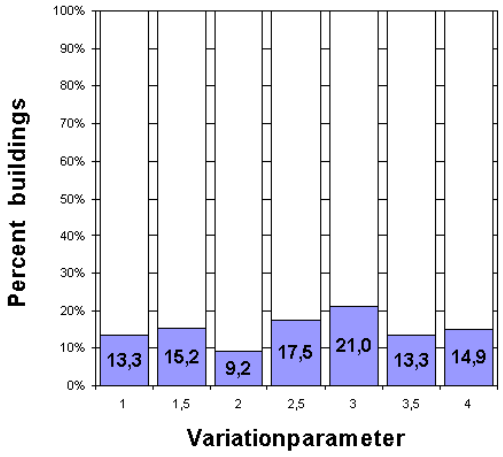
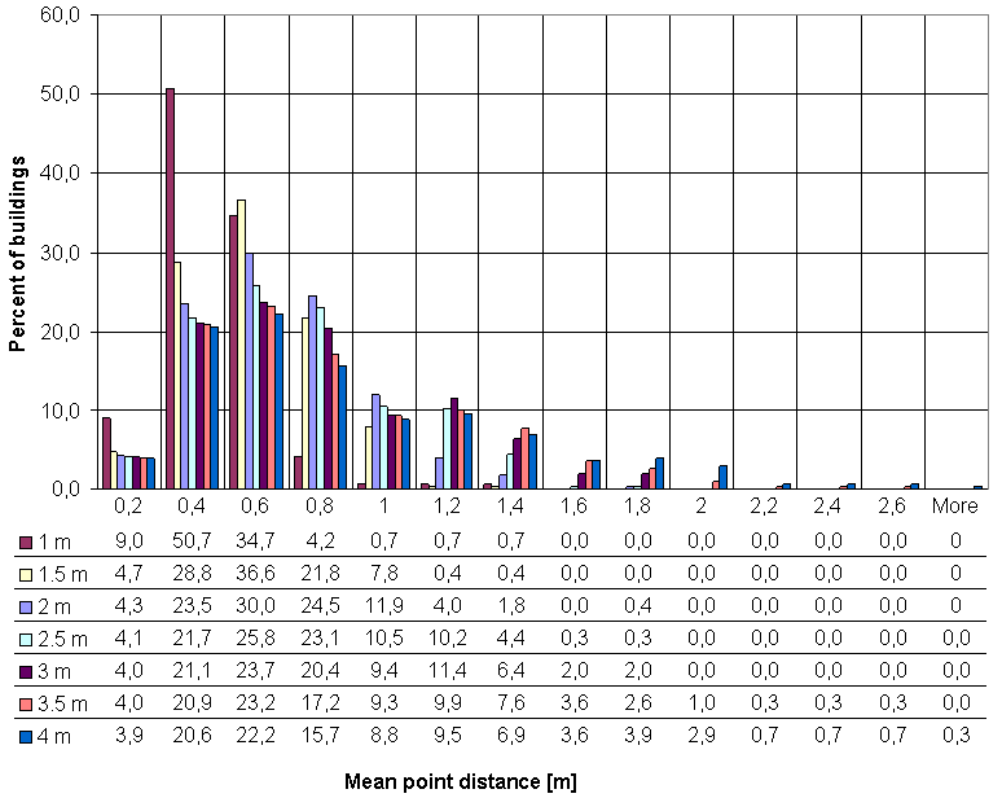


Figure 5.5: Percentage of buildings, that are manually classified as "insufficient generalized" (validated with different values for the variationparamter).

Reducing the variationparameter leads to a more detailed generalized polygon. If changing the variationparameter to 1 m, one might think that a "better" (from an intuitive approach) generalization result should be achieved. But in this case 42 buildings (13.3%) fail the manual quality test, thus more than with a variationparameter of 2 m. By reducing the variationparameter the degree of generalization is changed. This shows that there is a correlation between the degree of generalization and the robustness of the algorithm concerning gross errors in the building outline. This means that the generalized polygon fits better the building outline, but, at the same time, that it is a better modeling of the errors, that came from the former classification process. A human operator interprets the more detailed generalization as not being useful to serve as input data for a 3D-building-modelling.
From 30 buildings, manually classified as "insufficient generalized", 15 are derived from building polygons with complex structure (merged buildings, mainly due to wrong segmentation). At 15 buildings the orientation of the MBR does not correspond to the main orientation of the buildings, and therefore the orientation of the generalized building polygon is wrong, too.

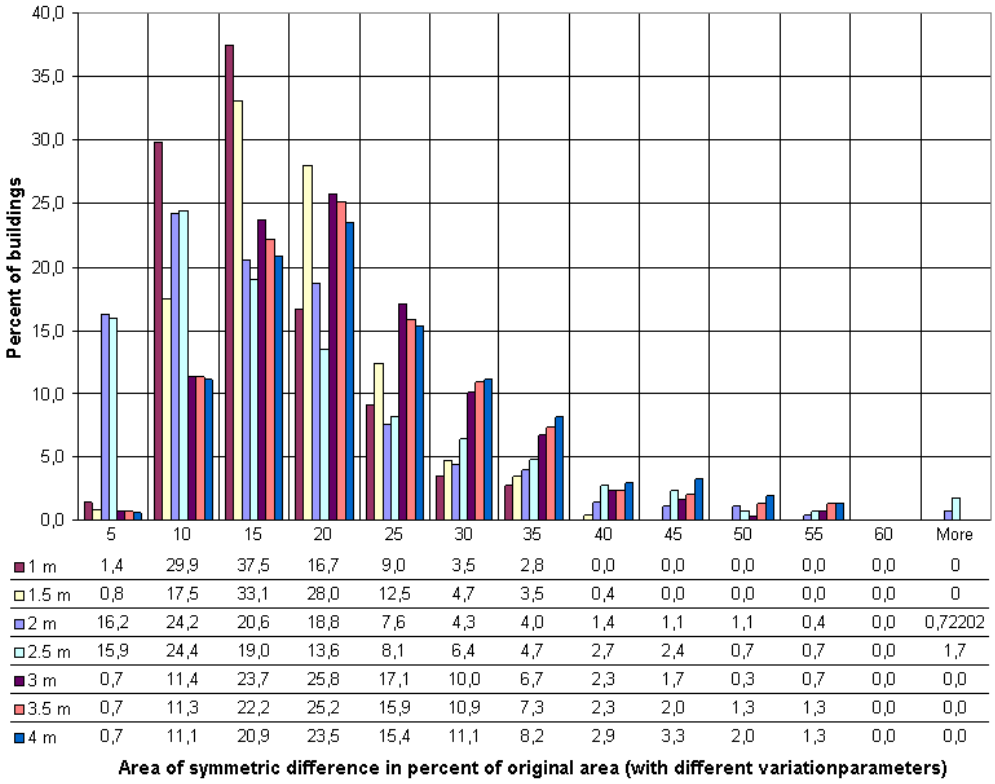


	0,2	0,4	0,6	0,8	1	1,2	1,4	1,6	1,8	2	2,2	2,4	2,6	More
1 m	9,0	50,7	34,7	4,2	0,7	0,7	0,7	0,0	0,0	0,0	0,0	0,0	0,0	0
1.5 m	4,7	28,8	36,6	21,8	7,8	0,4	0,4	0,0	0,0	0,0	0,0	0,0	0,0	0
2 m	4,3	23,5	30,0	24,5	11,9	4,0	1,8	0,0	0,4	0,0	0,0	0,0	0,0	0
2.5 m	4,1	21,7	25,8	23,1	10,5	10,2	4,4	0,3	0,3	0,0	0,0	0,0	0,0	0,0
3 m	4,0	21,1	23,7	20,4	9,4	11,4	6,4	2,0	2,0	0,0	0,0	0,0	0,0	0,0
3.5 m	4,0	20,9	23,2	17,2	9,3	9,9	7,6	3,6	2,6	1,0	0,3	0,3	0,3	0,0
4 m	3,9	20,6	22,2	15,7	8,8	9,5	6,9	3,6	3,9	2,9	0,7	0,7	0,7	0,3

Figure 5.6: Distribution of mean point distances for different variation parameters (only buildings, that passed the built-in quality test).

Mean point distance

In order to calculate the mean point distance between the raw vectorized building and the generalized building, additional points are added into the polygon boundary of the raw vectorized polygon in such a way that rough equality of the distances of neighboring points is achieved (see section 5.1.1). After that, the distances between every point and the generalized polygon are calculated. This gives a set of values, from which statistical qualities (e.g. minimum, maximum, mean, standard deviation, etc.) are calculated.
Figure 5.6 shows a histogram of the mean point distance, computed for different variationparameters.



	5	10	15	20	25	30	35	40	45	50	55	60	More
1 m	1,4	29,9	37,5	16,7	9,0	3,5	2,8	0,0	0,0	0,0	0,0	0,0	0
1.5 m	0,8	17,5	33,1	28,0	12,5	4,7	3,5	0,4	0,0	0,0	0,0	0,0	0
2 m	16,2	24,2	20,6	18,8	7,6	4,3	4,0	1,4	1,1	1,1	0,4	0,0	0,72202
2.5 m	15,9	24,4	19,0	13,6	8,1	6,4	4,7	2,7	2,4	0,7	0,7	0,0	1,7
3 m	0,7	11,4	23,7	25,8	17,1	10,0	6,7	2,3	1,7	0,3	0,7	0,0	0,0
3.5 m	0,7	11,3	22,2	25,2	15,9	10,9	7,3	2,3	2,0	1,3	1,3	0,0	0,0
4 m	0,7	11,1	20,9	23,5	15,4	11,1	8,2	2,9	3,3	2,0	1,3	0,0	0,0

Figure 5.7: Area of symmetric difference in percent of original area, computed with different variationparameter (only buildings, that passed the built-in quality test).

Area of symmetrical difference

ArcGIS provides a tool to calculate the symmetrical difference between the raw vectorized polygon and the generalized polygon. Statistical data can be collected about the area of the symmetrical difference. In order to compare values for different building sizes, the area of symmetrical difference is computed in percent of the raw vectorized building area.
In figure 5.7 the result for different variationparameter is shown. The following can be observed: the higher the variationparameter is chosen, the higher is the percentage where the maximum of the distribution is located. High percentage of the area means small congruence between original polygon and generalized polygon.

Variationparameter (built-in evaluation)

The built-in evaluation method is the variationparameter, as described in chapter 3. The variation of the distance between every point of a building edge and the MBR edge serves as a measure for the degree of generalization. The count of buildings that are marked as insufficiently generalized by the tool itself (computed with different values for the variationparameter) can be seen in figure 5.8.
When using a variationparameter of 1 m, 54 % of all buildings fail the quality check. Since the variationparameter is used to control the generalization process and, at the same time, to estimate the generalization result afterwards, this may not be expected. But the reason for this high percentage can be found either in the limitations of the algorithm (the level of detail is not unlimited) or in the minimal length of an edge of 3 m. If shorter edges were allowed, a smaller value for the variationparameter would be achieved.
The larger the variationparameter is set by the user, the less buildings fail the test. For the same reason as described above, this can not directly be seen as a quality statement about the correctness of the building representation, but it tells something about the degree of generalization. A high value for the variationparameter leads to a high tolerance in the calculation and therefore eventually hides inadequate generalization.

5.2.3 Evaluation of the built-in classification

After running through all generalization levels, the variationparameter is again calculated and used as a quality criterion of the resulting generalization. If the value exceeds the user defined value, the building is marked as critical building. It has to be evaluated how much the estimation by the algorithm corresponds to a manual classification. Several sets of parameters are tested. Figure 5.9 shows graphically the amount of buildings marked as critical by the tool and how the selection corresponds to manual evaluation. The first subset is the set of bad generalized buildings recognized by the tool, but which are not marked manually as critical. The second subset is the set of buildings recognized as critical by the tool and also by manual validation.

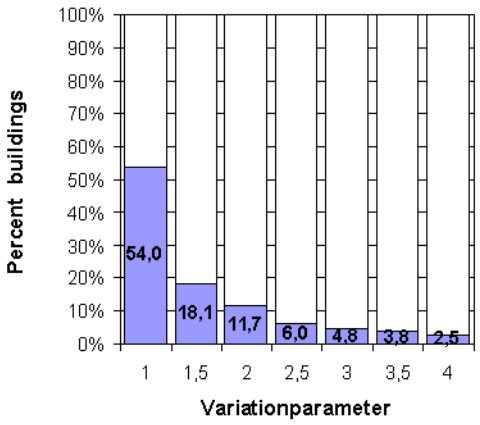


Figure 5.8: Percent of buildings, marked as insufficiently generalized by the tool, tested with different variationparameter.

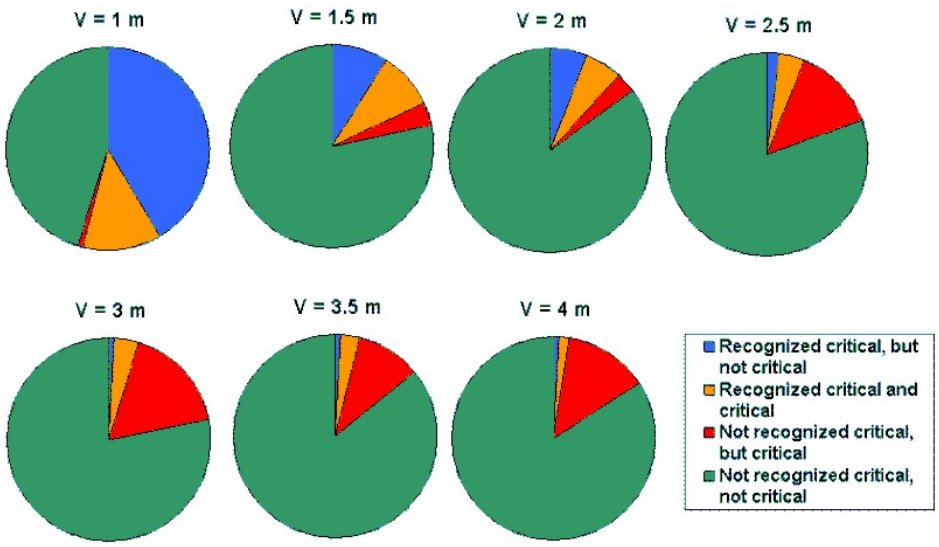
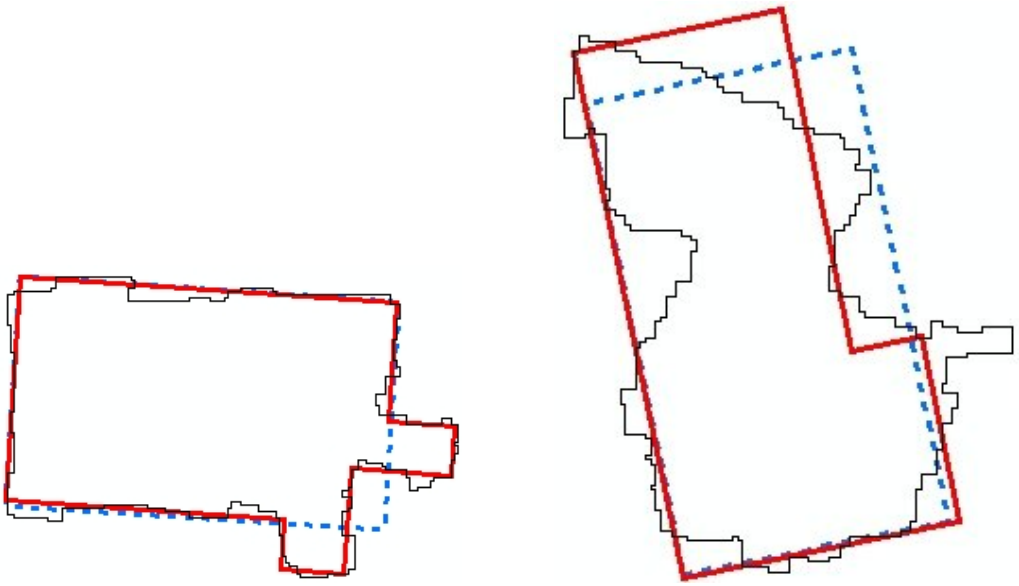


Figure 5.9: Comparison of the built-in and the manual validation. A pie chart is the set of all buildings. "Recognized" means, that the building is marked by the tool as not valid.



(a) Satisfying classification: Both generalized polygons are accepted by the tool. Red polygon: V=2.5 m; blue, dashed polygon: V=3 m

(b) Unsuccessful classification: The red polygon (V=2 m) is marked as critical by the tool, the blue, dashed one (V=2.5 m) is accepted.

Figure 5.10: Examples of the built-in evaluation

Variationparameter [m]	**1**	**1.5**	**2**	**2.5**	**3**	**3.5**	**4**
Recognized critical	170	57	37	19	15	12	8
Recognized critical %	54.0	18.1	11.7	6.0	4.8	3.8	2.5
Recognized critical true	40	29	19	13	12	9	5
Recognized critical true %	12.7	9.2	6.0	4.1	3.8	2.9	1.6
Not recognized but critical	2	11	10	42	54	33	42
Not recognized but critical %	0.6	3.5	3.2	13.3	17.1	10.5	13.3
Not recognized but critical (% from recognized critical)	1.4	4.2	3.6	14.2	18.0	10.9	13.7
Recognized critical wrong	130	28	18	6	3	3	3
Recognized critical wrong %	41.3	8.9	5.7	1.9	1.0	1.0	1.0

Table 5.2: Classification of generalization quality by the algorithm, using different values for the variationparameter.

The third subset is the set of buildings not marked as critical by the tool but very well by manual validation. This set is fortunately very small with a low varitionparameter (0.6 %), but grows when using a greater variationparameter (17.1 % with V=3 m). The last subset is the set of the remaining buildings, which are neither recognized critical by the tool nor by hand.
An example for different generalization results achieved by different values for the variationparameter can be seen in Figure 5.10(a). Both generalizations are classified as valid by the tool. This corresponds to a manual classification.
An example, where the classification by the tool is not successful, follows in figure 5.10(b). The orientation of the building has been calculated wrong. In the first case (V = 2 m, blue dashed polygon), the generalization is classified as critical and this corresponds to the visual evaluation. In the second case (V = 2.5 m, red polygon) the generalized polygon is not detected as critical by the tool.

Coarse orientation error

A manual analysis of the result of the automatic generalization shows, that 16 buildings (5 %) have a coarse orientation error. The reason for that kind of error is, that the orientation of the MBR is not equal to the orientation of the

building (see section 6.1). The count of wrongly orientated generalized buildings, that are not detected by the tool, depends on the variationparameter and is shown in table 5.3.

Variationparameter [m]	**1**	**1.5**	**2**	**2.5**	**3**	**3.5**	**4**
Recognized critical	0	6	8	12	12	13	15
Recognized critical in %	0	37.5	50	75	75	81.25	93.75

Table 5.3: Count of wrong orientated generalizations, that are not detected by the built-in classification.

Chapter 6

Discussion and conclusion

The last chapter of this work starts with a brief description of restrictions of the presented algorithm, and makes proposals on possible enhancements of some aspects.

6.1 Limitations of the presented algorithm

Limited geometry for the generalized building

Through the presented method it is not possible to achieve an arbitrarily geometry of generalization. The level of detail is restricted to a certain number of defined geometries, which are listed below:

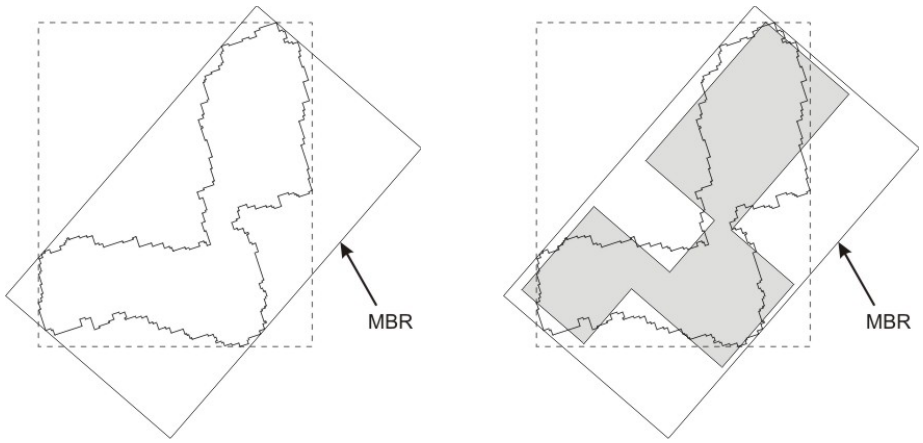
1. Simple rectangles.
2. Rectangles, where one or more corners are replaced by two new edges respectiveldy (see figure 6.1(a), (b) and (c)).
3. Rectangles, where one or more edges are replaced by five new edges each (formed like an "U", see figure 6.1(d)).
4. The most complex geometry is a combination of all so far mentioned geometries: A rectangle, where corners are cut out and additionally one or more edges are replaced by five new edges respectively (see figure 6.1(e) and (f)).

Figure 6.1: Restriction in geometry

An enhancement of the algorithm aiming in a greater flexibility in the building structure is possible up to a certain extent. However, it has to be considered, that the higher the complexity of a building structure, the lower is the probability of a strictly rectangular structure.

Determining the orientation by means of the MBR

In some cases, depending on the shape of the building polygon, the calculation of the MBR leads to a rectangle, that is wrong orientated regarding the orientation of the building edges. An example is shown in figure 6.2. The rectangle in the left image is the calculated MBR, whereas the dashed rectangle is the desired bounding rectangle (from which the orientation of the building is derived). If the MBR is taken as a measure of the future orientation of the generalized polygon (as it is in the presented algorithm), this leads to a wrong generalization (figure 6.2(b)).
One improvement to get this problem under control would be an adaptation of the algorithm by H.Freeman and R.Shapira (see section 2.2.2) used to cal-



(a) Minimal bounding rectangle (bold rectangle) has a different orientation than the building polygon ...

(b) ... this results in a completely wrong generalization.

Figure 6.2: Incorrect orientation of the final generalization due to the MBR orientation

culate the MBR.: Originally an iteration through all points of the convex hull is done. The area of the encasing rectangle, that has one edge through the actual point of the iteration and the following point of the complex hull, is calculated. There are five points of contact between the convex hull and the actual rectangle. If the smallest rectangle is found, one could calculate the distances of all building points (that are located between two of these points of contact) to one single rectangle edge (see figure 6.3). If the orientation of the rectangle differs much from the orientation of the axis of the building, the distances will vary strongly (dashed lines in figure 6.3). If the orientation is similar, there will be no significant variation in the distances (drawn through lines in figure 6.3). The abrupt change in variation and rapid growth of the distances afterward must be interpreted correctly as a corner of the building. If the variance of the distances increases a certain value, the orientation of the MBR is considered to be very different from the orientation of the axes of the building polygon. In this case, it should not be taken as a measure for the orientation of the building, although it is the MBR. Rather the rectangle, that has the second smallest area, might be a better measure for the

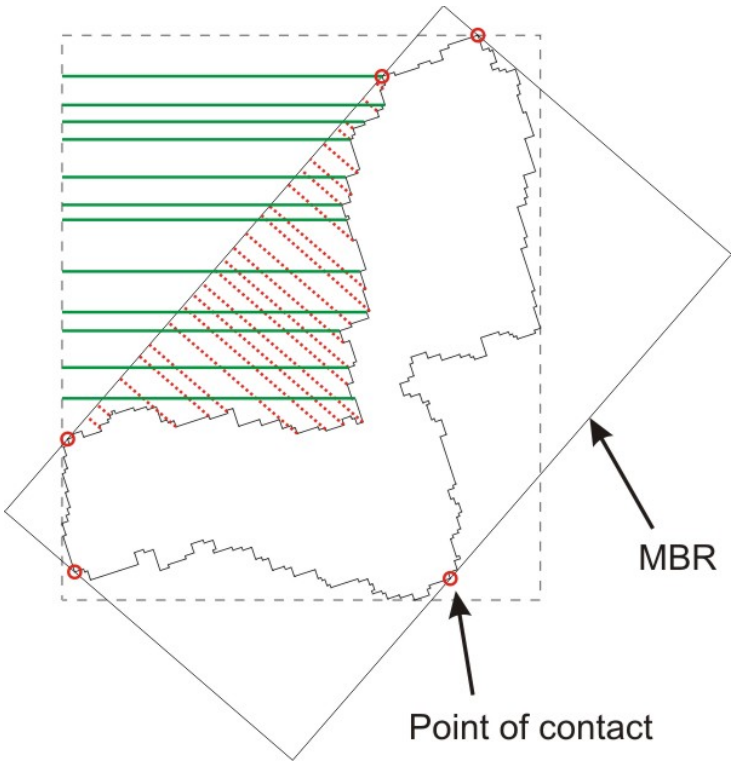


Figure 6.3: The length of the dotted lines vary strongly - this is an indication for a wrong detection of the building axis. This measure could be integrated into the implemented MBR-algorithm, in order to guarantee a correct orientation of the generalized polygon.

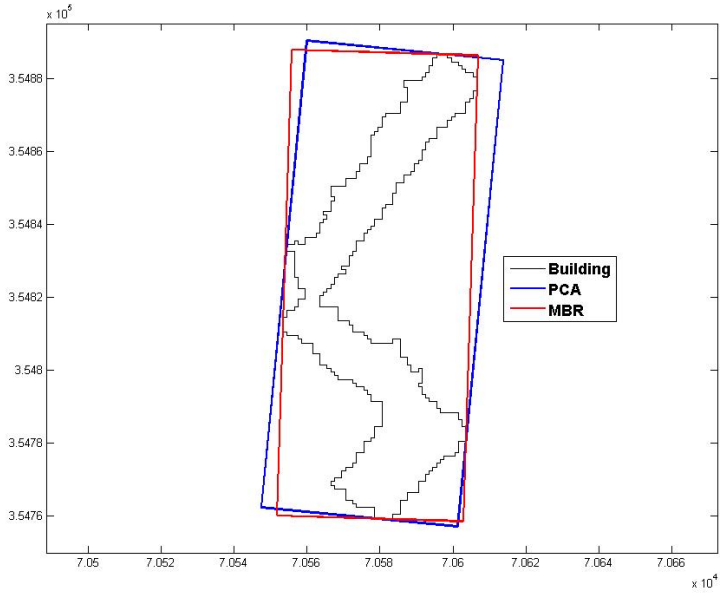


Figure 6.4: Comparison MBR versus principal axis

orientation (which must be tested) and should be taken instead of the MBR. Another point of departure might be the determination of the principal axis of the polygon (see section 2.2.3) and to use the orientation of the axis as the orientation of the generalized polygon. However, as experiments show, the orientation of the MBR and the principal axis of a polygon are similar. Unfortunately, this is the case in situations, where the MBR is not a suitable measure for the orientation of the building (see figure 6.4). From this it follows, that this measure is not more adequate for solving the building orientation task than the MBR.

Distance of parallel edges

Figure 6.5 shows a group of buildings that has been classified incorrectly as a single building. By means of the presented generalization algorithm it is not possible to detect the composition of buildings. But the algorithm could be enhanced by testing for the distance of collinear edges of the generalized polygon. Parallel edges, that are very close together, and have not been elim-

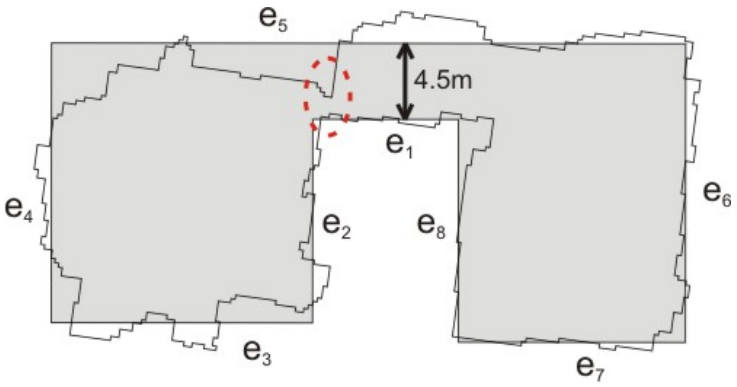


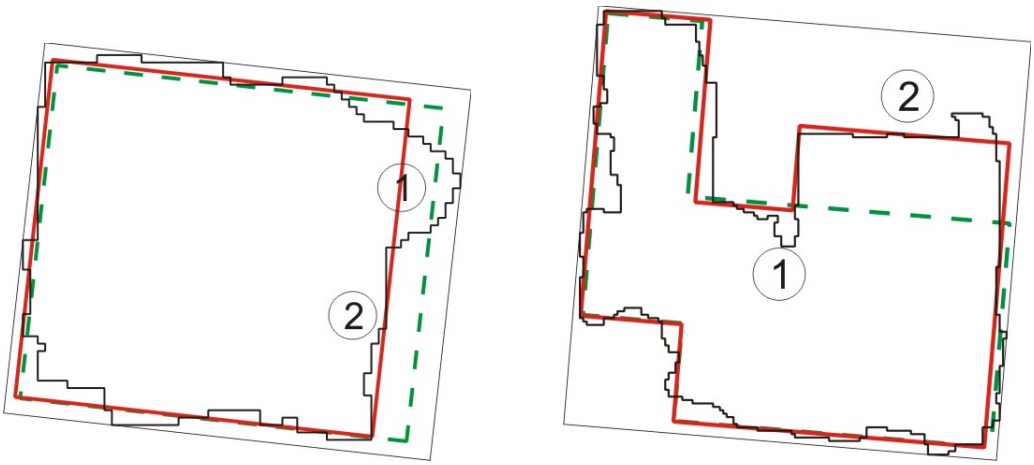
Figure 6.5: Distance between edges is 4.5m

inated in the course of the test for short edges (section 3.2.5), can be a sign of an incorrectly classified composition of multiple buildings. Figure 6.5 shows an example generalization that consists of three major parts. Among them is a narrow part of only $4.5\,m$ of broadness. Consider the subset of building points belonging to edge e_1 and the subset of building points belonging to edge e_5. e_5 must not be adjacent to e_1. Calculate the euclidean distance between the points of all combinations of the elements from both subsets. The building polygon could be separated into two polygons at those points with the least distance. The generalization algorithm should be applied once again to the new polygons.

Point density of input data

The position of an edge of the final generalization is defined through the median of the distances from the building points to the MBR-edge. Thinking of a local coordinate system (where the x-axis goes through an MBR-edge and is oriented counterclockwise), only the y-values of a building point have an impact on the generalization result. Considering as well the x-values (thus the distribution of the points along the MBR-edge) would however improve the result.

An alternative way leading to the same effect is to insert points into the raw boundary without changing the boundary's shape in such a manner, that the euclidean distance between neighboring points of the boundary is similar



(a) (b)

Figure 6.6: Effect of point densification: The green, dashed polygons are derived from the original building polygons, whereas the red polygons are calculated from building polygons with equidistant points. Obviously the latter fit better.

for all points. This was formulated as a prerequisite for the input data in section 3.1.1. For that reason a densification function, which carries out the interpolation of the required count of points, was integrated into the generalization tool.

Figure 6.6 shows two examples of a generalization with different input polygons. The green dashed polygons are a generalization derived from the original polygons. The minimum point distance in the original building polygon is 0.25 m. After interpolating more points, the maximum distance between points is 0.25 m. The red line is a generalization derived from this densified polygon.

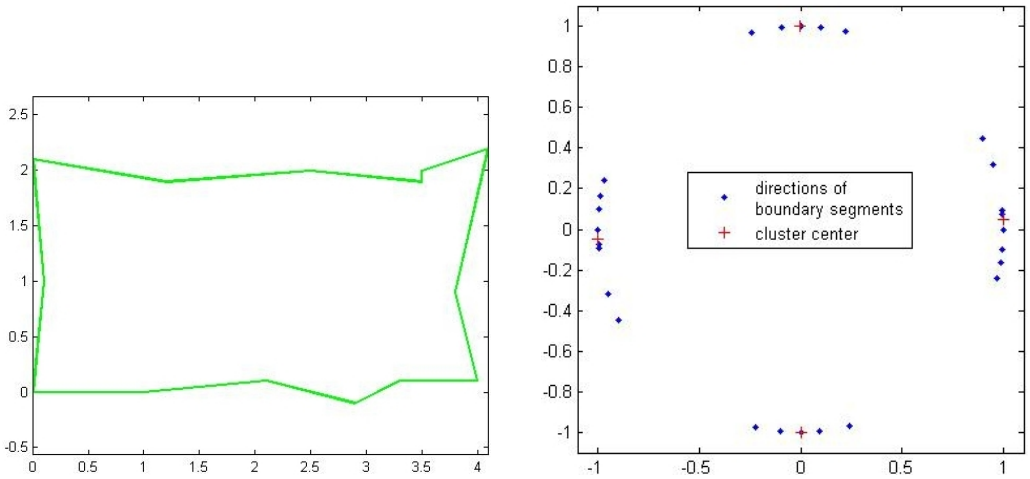
As it can be seen in figure 6.6(a), the position of the right vertical edge is affected by the great amount of building points with little distance from the MBR-edge (region marked with 1). The lower part of the building points (see area 2) seems to represent more likely the real position of the building edge. However, the point density is much lower than in the upper part. Expectedly, the red edge, that has been generalized from the densified polygon, goes now through the points marked with 2.

6.2 Outlook

Other interesting methods, also from other disciplines, are adaptable to these problems mentioned above. Some ideas should be mentioned at the end of the thesis. *Data clustering* methods can be used to detect the orientation of the building edges. One data clustering algorithm is the LBG[1]-algorithm (k-Means). Given a set of vectors (in our case these vectors are points in $\mathbb{R}^2$), the application of a LBG-algorithm determines a *codebook*. The codebook is a set of reference vectors. Each reference vector will be located in the center of mass of the vectors of its voronoi-set after the execution of the algorithm. Details on the method can be found in (Fritzke 1998). The application of a data clustering method is shown in figure 6.7: The direction of each line segment of a building polygon (figure 6.7(a)) is represented by a point located on the unit circle. The result are the points in figure 6.7 (b) (marked in blue). Since the building is composed from four edges, the directions of the line segments will be concentrated around four values. In other words, the created points accumulate in four clusters. The center of the clusters are detected by a k-Means algorithm and are marked in red in figure 6.7 (b). In this way four main directions of the building polygon are determined. The next step would be to allocate every point to an edge, and to fit a straight line with the direction calculated above to every group of points. In this example, the generalization model is determined by the count of clusters, the algorithm should search for. Unfortunately, the count of clusters can not be discovered by a k-Means algorithm.

It could be useful to introduce additional information, like investigation of the *neighborhood* of a building polygon, into the generalization process. The street network could give useful information about the orientation of nearby houses. Assumptions about the "type" of a building could be made on the basis of the "type" of buildings in the neighborhood. Furthermore, information from the cadaster could be integrated into the generalization process.

[1] a method named after its developers Linde, Buzo and Gray.



(a) The original building polygon.

(b) Points on the unit circle, representing the direction of the line segments. The centers of the four clusters are the result of a k-Means clustering.

Figure 6.7: Example of an application of k-Means clustering in building generalization.

Another method should be mentioned: *RANSAC* (Random Sample Consensus) is a method for fitting a model to data, that contains a significant percentage of gross errors (Fischler and Bolles 1981). The method takes randomly the minimum count of observations from the data, and computes a realization of the model. Afterwards it determines the count of points, that are close enough to the calculated realization. If there are enough compatible points, an improved estimation would be calculated from all of the compatible points (e.g. by means of a least squares estimation). If there are not enough compatible points, another set of observations is randomly selected. This procedure is repeated, until a realization is found or until a predefined number of trials is reached. RANSAC could be used to detect lines in building polygon data or to directly fit a building model.

6.3 Conclusion

The aim of this work was the development of a generalization method and the implementation of the method in a ready-to-use tool embedded in a GIS-software. The level of generalization should be chosen in the face of 3D-visualization purposes, thus the generalization process should result in simple building structures like rectangles or a composition of rectangles. The focus should be set on a high level of automation.

A model based generalization method proofed to be accurate and reliable to solve the task of providing simplified building outlines for visualization purposes. The rectangularity-constraint is realized by fitting lines parallel to an MBR of a building polygon. Simplicity in the generalized geometry is achieved by running through three levels of detail. The computation starts with a fitting of the simplest geometry, a rectangle, and will go on with a composition of rectangles, if the result is not satisfying. The concept of split points turned out to be accurate to find a suitable model for an arbitrary building polygon, and on the other hand to fit the model to the building polygon.

The generalization method, that has been developed, meets the requirements mentioned in the objectives in section 1.1. The validation of the results, achieved with the data from Hall, demonstrates, that the algorithm is successfully applied and moreover handles building polygons with noisy edges. The degree of automation is high (88.3 % of the buildings from the test data are automatically generalized), considering the limited amount of building primitives, that are fitted to the data. The buildings, which didn't pass the built-in quality test, have a more complex geometric structure and can not be generalized by means of the provided models. The manual inspection shows, that from the set of all input polygons, 9.2 % can not be automatically generalized in a satisfying way. From all polygons 3.2 % have a bad generalization quality, but remain undetected by the built-in quality check. Due to the coarse orientation error, 5 % of the buildings are generalized inaccurately. By enhancing the method of detecting the orientation of a building, a the overall result could even be improved.

List of Figures

Bibliography

Chan, T. (1996). Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete and Computational Geometry 16/4*, 361–368.

Douglas, D. and T. Peucker (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cancadian Cartographer 10/2*, 112–122.

ESRI, E. S. R. I. (1996). Automation of map generalization - the cutting-edge technology.

ESRI, Environmental Systems Research Institute. *ArcGIS Desktop Help: How Simplify Building (Coverage) works*. ESRI, Environmental Systems Research Institute.

Fischler, M. A. and R. C. Bolles (1981). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM 24*, 381 – 395.

Freeman, H. and R. Shapira (July 1975). Determining the minimum-area encasing rectangle for an arbitrary closed curve. *Communications of the ACM 18 (7)*, 409 – 413.

Fritzke, B. (1998). *Vektorbasierte Neuronale Netze*. Shaker Verlag.

Gonzalez, R. and R. Woods (2003). *Digital Image Processing* (2nd ed.). Prentice Hall.

Gross, H., U. Thoennessen, and W. Hansen (2005). 3d-modeling of urban structures. *International Archives of Photogrammetry and Remote Sensing 36*.

Gruber, F. J. (2001). *Formelsammlung für das Vermessungswesen*. Konrad Wittwer, Stuttgart.

Huttenlocher, D. P., G. A. Klanderman, and W. J. Rucklidge (1993). Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence 15*, 850–863.

Jähne, B. (2004). *Practical Handbook on Image Processing for Scientific and Technical Applications*. CRC Press.

Kanani, E. (2000). *Robust estimators for geodetic transformations and GIS*. Ph. D. thesis, ETH Zurich.

Kelnhofer, F. (2001). Grundzüge der Kartographie. Script from the lecture at Vienna University of Technology.

Keyes, L. and A. C. Winstanley (2001). Using moment invariants for classifying shapes on large scale maps. *Computers, Environment and Urban Systems 25*.

Maas, H.-G. (1999). Closed solutions for the determination of parametric building models from invariant moments of airborne laserscanner data. *International Archives of Photogrammetry and Remote Sensing 32*, 193–199.

McMaster, R. (1987). Automated line generalization. *Cartographica 24/2*, 74–111.

Meng, L. (2001). Heutiger Stand von Theorie und Methodik der Generalisierung. *Kartographische Bausteine 19*.

Sampath, A. and J. Shan (2004). Urban modeling based on segmentation and regularization of airborne lidar point clouds. In *XXth ISPRS Congress, Istanbul, Turkey, Commission III*, pp. 937 ff.

Sester, M. (2000). Generalization based on least squares adjustment. *International Archives of Photogrammetry and Remote Sensing 33*.

Shan, J. and A. Sampath (2006). *Urban Terrain and Building Extraction from Arborne Lidar Data*, Chapter 2. CRC Press/Taylor & Francis Group.

Staudinger, M. (2003). Ausgleichungsrechnung I. Script from the lecture at Vienna University of Technology.

Staufenbiel, W. (1973). *Zur Automation der Generalisierung topographischer Karten mit besonderer Berücksichtigung großmaßstäbiger Gebäudedarstellungen*. Ph. D. thesis, Technische Universität Hannover.

The MathWorks. Matlab documentation. `http://www.mathworks.com/access/helpdesk/help/toolbox/map/index.html?/access/helpdesk/help/toolbox/map/reducem.html&http://www.mathworks.com/support/functions/alpha_list.html?sec=7`; accessed March 2007.

Vinson, S., L. Cohen, and F. Perlant (2001). Extraction of rectangular buildings in aerial images. *Proc. Scandinavian Conference on Image Analysis (SCIA'01)*.

Visvalingam, M. and J. Whyatt (1993). Line generalisation by repeated elimination of points. *Cartographic Journal 30(1)*, 46–51.

Vozikis, G. (2005). *Automated generation and updating of digital city models using high-resolution line scanning systems*. Ph. D. thesis, Vienna University of Technology.

Weibel, R. and C. Jones (1998). Computational perspectives on map generalization. *GeoInformatica 2/4*, 307–314.

Appendix A

Test area

For completeness the generalization result achieved with the presented algorithm is documented in figure A.3 to figure A.13. The test area is divided into 12 tiles. Figure A.1 gives an overview about the location of the tiles. The generalization result is shown together with the original polygons and the orthophoto.

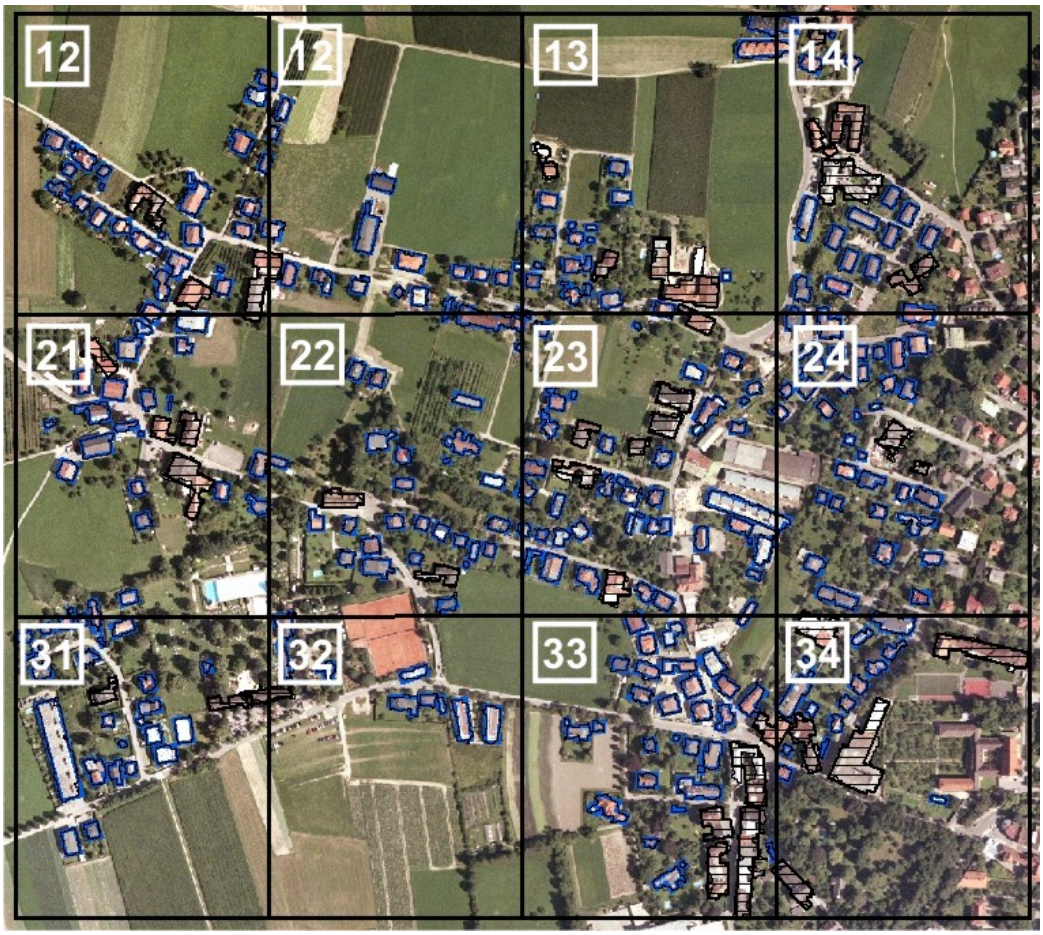


Figure A.1: Test area with generalization result - overview

Figure A.2: Result of the automatic generalization, tile 11

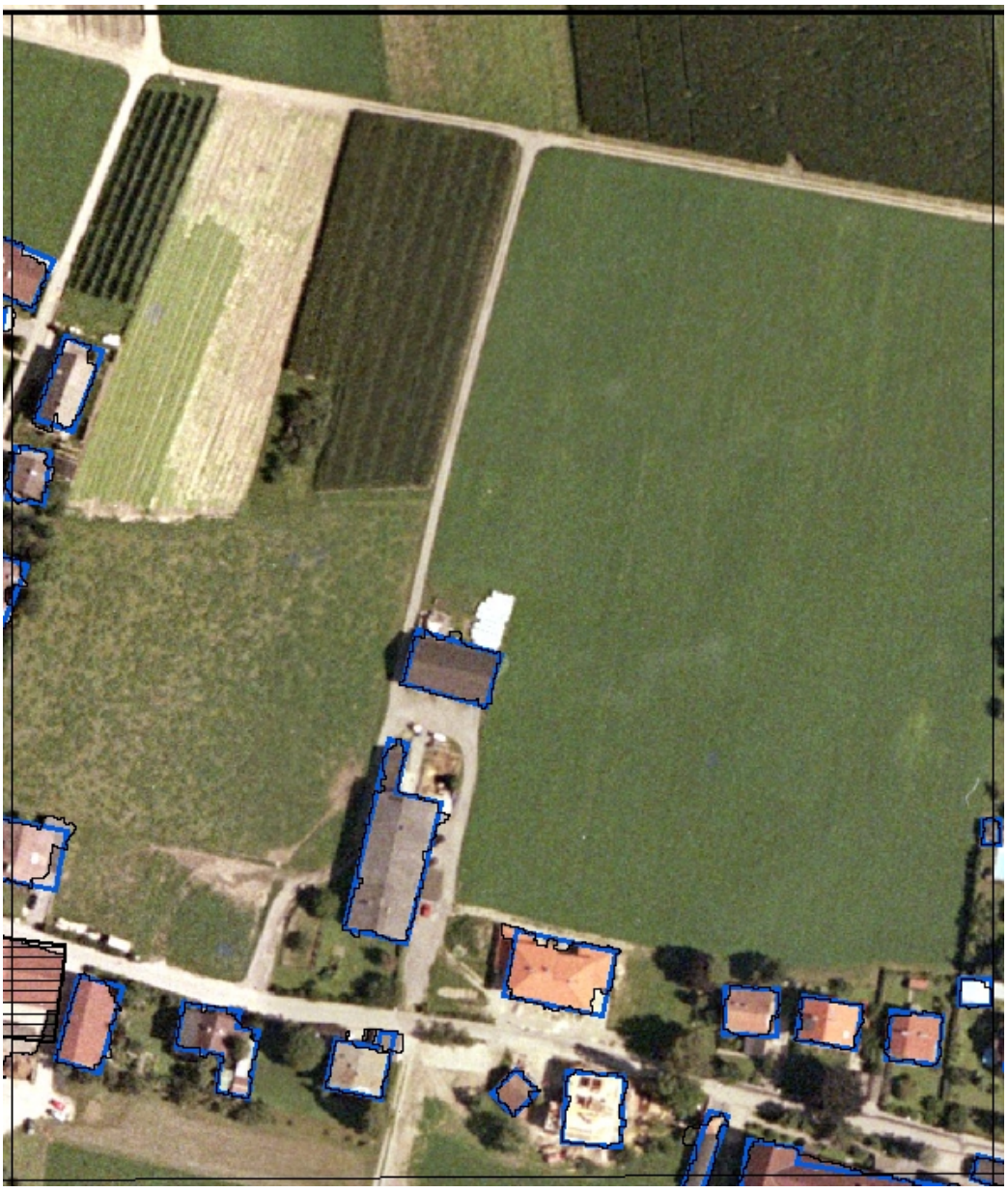


Figure A.3: Result of the automatic generalization, tile 12

Figure A.4: Result of the automatic generalization, tile 13

Figure A.5: Result of the automatic generalization, tile 14

Figure A.6: Result of the automatic generalization, tile 21

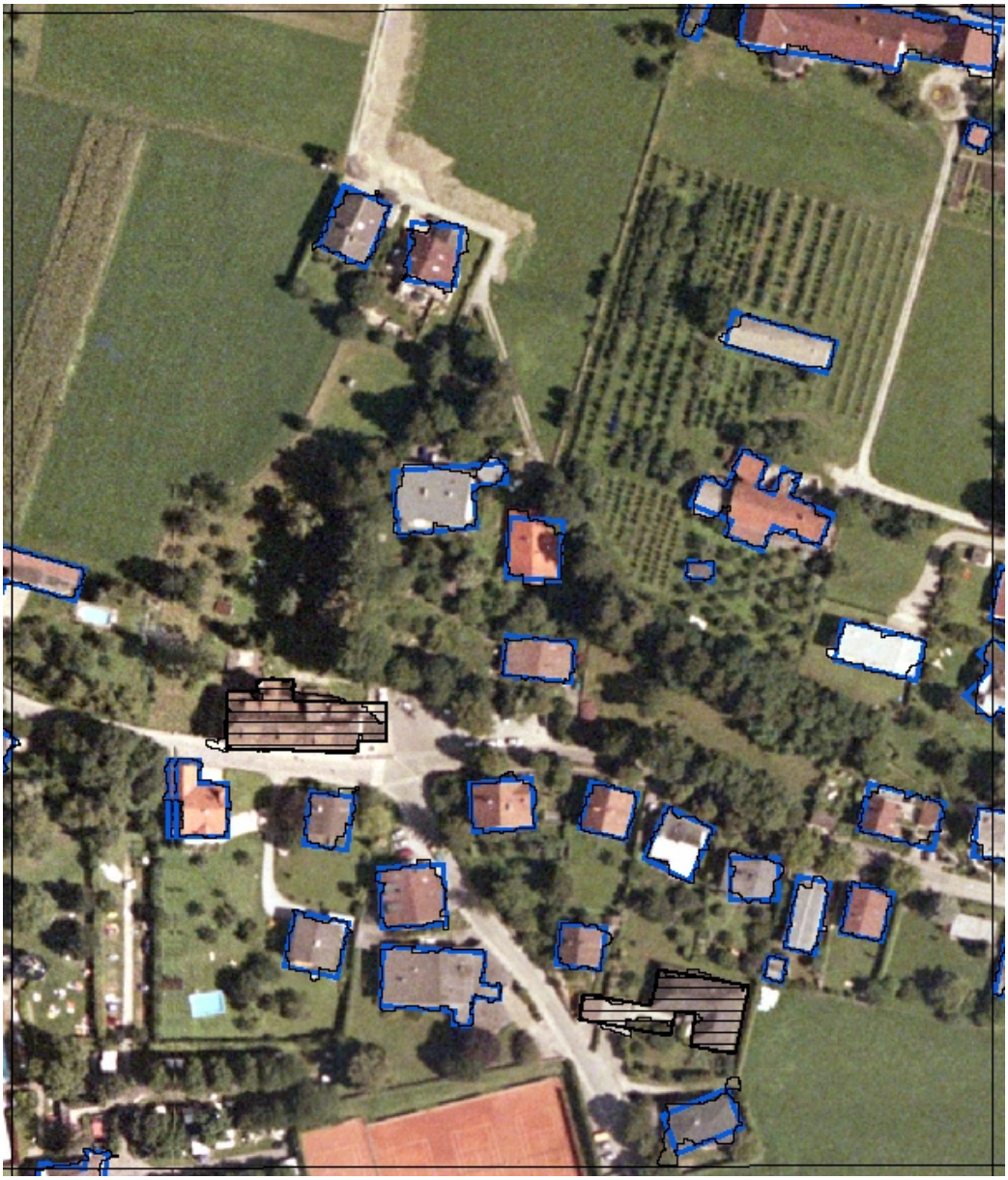


Figure A.7: Result of the automatic generalization, tile 22

Figure A.8: Result of the automatic generalization, tile 23

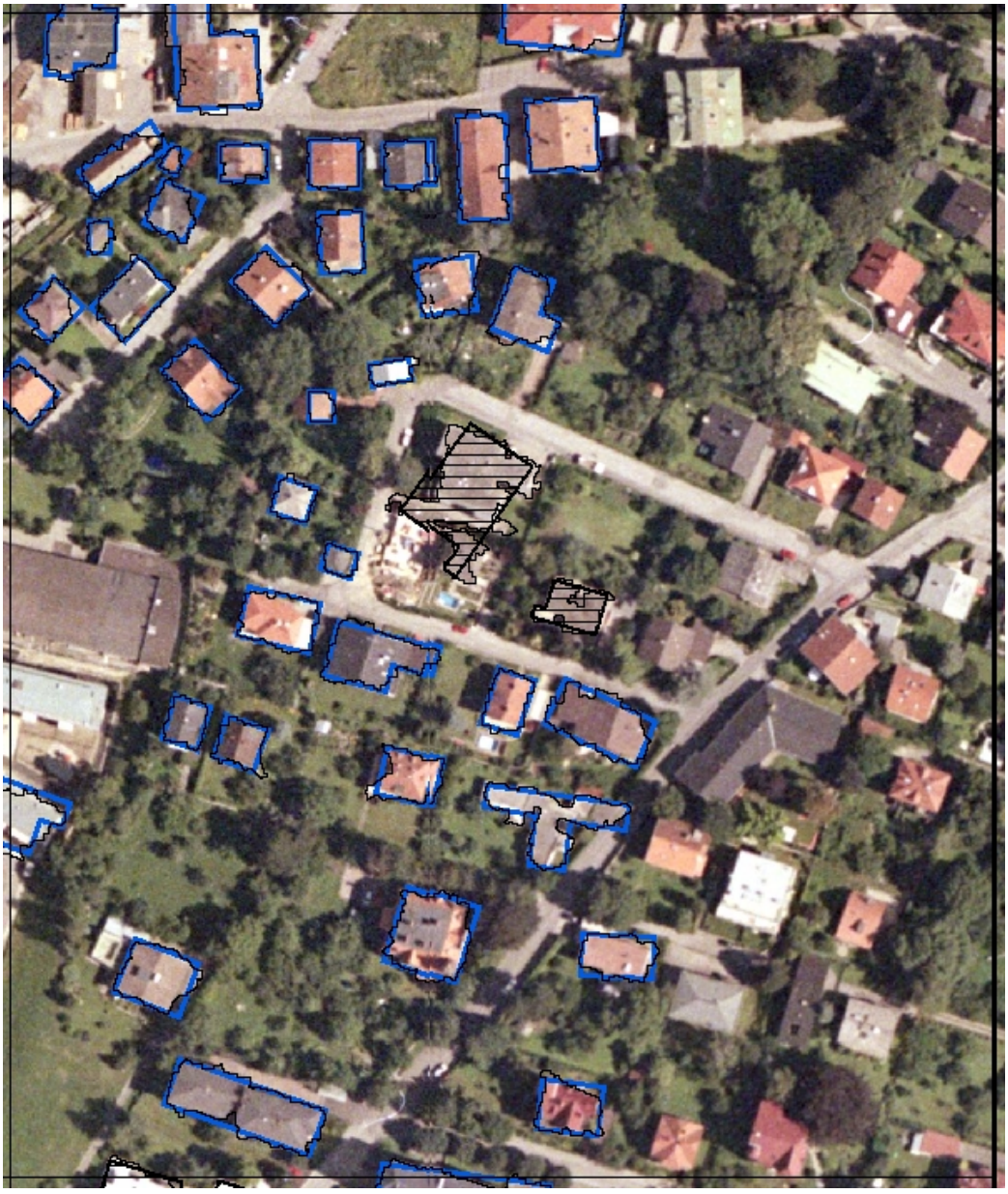


Figure A.9: Result of the automatic generalization, tile 24

Figure A.10: Result of the automatic generalization, tile 31

Figure A.11: Result of the automatic generalization, tile 32

Figure A.12: Result of the automatic generalization, tile 33

Figure A.13: Result of the automatic generalization, tile 34

Appendix B

Curriculum vitae

Name	Marieke Dutter
Date of birth	March 15, 1978
Place of birth	Graz, Austria

1984 – 1988	Elementary school in Graz and Vienna
1988 – 1996	Grammar school in Vienna
1996	School leaving examination (Matura)

1996 – 2000	Studies of "Instrumentalpädagogik" at Vienna University of Music and Performing Arts
2000	First diploma examination (teaching qualification examination)
2000 – 2001	Two exchange semesters at Sibelius Academy in Helsinki, Finland
Since 2001	Studies of "Vermessung und Geoinformation" at Vienna University of Technology