An automatic recognition method for building floor plans

Jain, P.

Award date:
2009

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mathematics and Computer Science

An Automatic Recognition Method for Building Floor Plans

Poojith Jain
(0666444)
Master Thesis

Supervisors:
dr. M. A. Westenberg
dr. B. Speckmann

Referee :
prof. dr. K. M. van Hee

Eindhoven, The Netherlands
July 2009

Abstract

The cadastral organization in the Netherlands, called the Kadaster, *takes responsibility for the registration of land and property. In the current system used by the Kadaster, a piece of land, called a parcel is associated with legal information such as ownership right and ownership boundaries. A parcel may be associated either with a single ownership right or multiple ownership rights. For example, a parcel having a house owned by a single person indicates single ownership right and a parcel having a building with several apartments, each owned by a different person indicate multiple ownership rights. On selecting a parcel, its legal information is produced to the user. However in the case of multiple ownership rights, it is difficult to associate which part of the parcel is owned by which person. To address this issue, the current system produces a scanned image of the floor plans of the given building. These floor plans indicate the ownership rights of each floors. But the information from the scanned images of the floor plans cannot be interpreted by a computer for queries. Hence, there is a need for an alternative method to deal with parcels having multiple ownership rights.*

This thesis describes the recognition method for extraction of relevant information such as ownership boundary and ownership rights for legal purpose from the scanned building floor plans.

The recognition method consist of two stages, first extracting and representing the information contained in an image in a graph representation. Then processing this graph representation to extract ownership boundary, ownership rights and representing them in computer processable format.

The information related queries can be answered as data in the image is represented in computer processable format. The extracted information also represented in a suitable format for visualization.

Acknowledgments

I am indebted and grateful to my supervisors Dr. M. A. Westenberg and Dr. B. Speckmann for their continuous support, stimulating suggestions and encouragement during the course of the project. I thank them for their suggestions for the improvement of the thesis report, both in terms of grammar and content. This thesis could not have been completed without their support. I sincerely thank ir. M. Sterenberg, The Kadaster for providing all the support and necessary information for the completion of the thesis. I would like to express my gratitude to Capgemini for giving me the opportunity to do this project. I thank the Kadaster for providing all the necessary support and data to complete my thesis. I greatly appreciate the suggestions given by my colleagues M. G. DSlva and X. L. Ernest Mithun. Finally, I would like to thank all those who directly or indirectly involved in this project.

Poojith Jain
Eindhoven
July 31, 2009

Contents

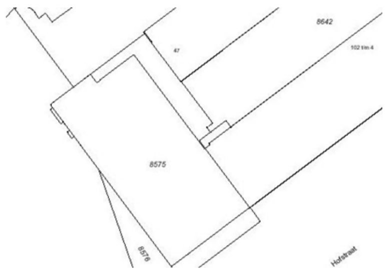
Chapter 1

Introduction

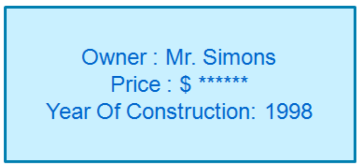
A correct, consistent and complete registration of land and property plays a very important role for legal purposes. A *cadastre* is the organization that takes care of the land and property registration. A cadastre (Stoter 2004) is defined by FIG (International Federation of Surveyors) as *"Cadastre is normally a parcel based, and up-to-date land information system containing a record of interests in land (e.g. rights, restrictions and responsibilities). It usually includes a geometric description of land parcels linked to other records describing the nature of the interests, the ownership or control of those interests, and often the value of the parcel and its improvements. It may be established for fiscal purposes (e.g. valuation and equitable taxation), legal purposes (conveyancing), to assist in the management of land and land use (e.g. for planning and other administrative purposes), and enables sustainable development and environmental protection."* In the Netherlands, the *Kadaster* takes the responsibility of the registrations of the land and property ownership.

The Kadaster describes information regarding land use in cadastral maps. These maps are *parcel* based where a parcel is an individual lot of land with its own legal description. Each parcel is associated with a unique identifier known as *parcel number*. A parcel may also be associated with legal information such as, ownership right, ownership type, address etc. An *ownership right* is the real right entitled to a person (or persons) for a parcel. A parcel may have a single ownership right or multiple ownership right. For example, a parcel having a house owned by a single person indicates single ownership right and a parcel having a building with several apartments, each owned by a different person indicate multiple ownership rights.

The cadastral map of a city used by the Kadaster comprises all parcels of that city. Generally, a parcel has a single ownership right associated to it. On selecting one such parcel, information on legal details about that parcel is provided to the user. Figure 1.1 shows one such example of a parcel with single ownership right where details about that parcel are provided. In case of a parcel with multiple ownership rights, the current system at the Kadaster provides a scanned image of the floor plans containing the ownership rights of that building. The building having multiple ownership rights may be either apartments of a building or stores of a commercial building,



(a) A parcel.


Owner : Mr. Simons
Price : $ ******
Year Of Construction: 1998

(b) Legal information about the parcel.

Figure 1.1: *A parcel with single ownership right.*

each with many owners. We generalize both these cases as apartment buildings. The floor plans associate each apartment of that building with an ownership right. Figure 1.2 shows one such scanned image of a floor plan. In the figure we observe that there are multiple floors. Each floor has a floor plan associated to it. The numbers in the floor plans indicate the ownership right. The thick lines in the image indicate the ownership boundary. The thin lines indicate the interior partitions of the ownership boundary. The texts in the image give details about the usage type of the regions. For example, the text 'kamer' indicates room, 'badkamer' indicates bath room, etc.

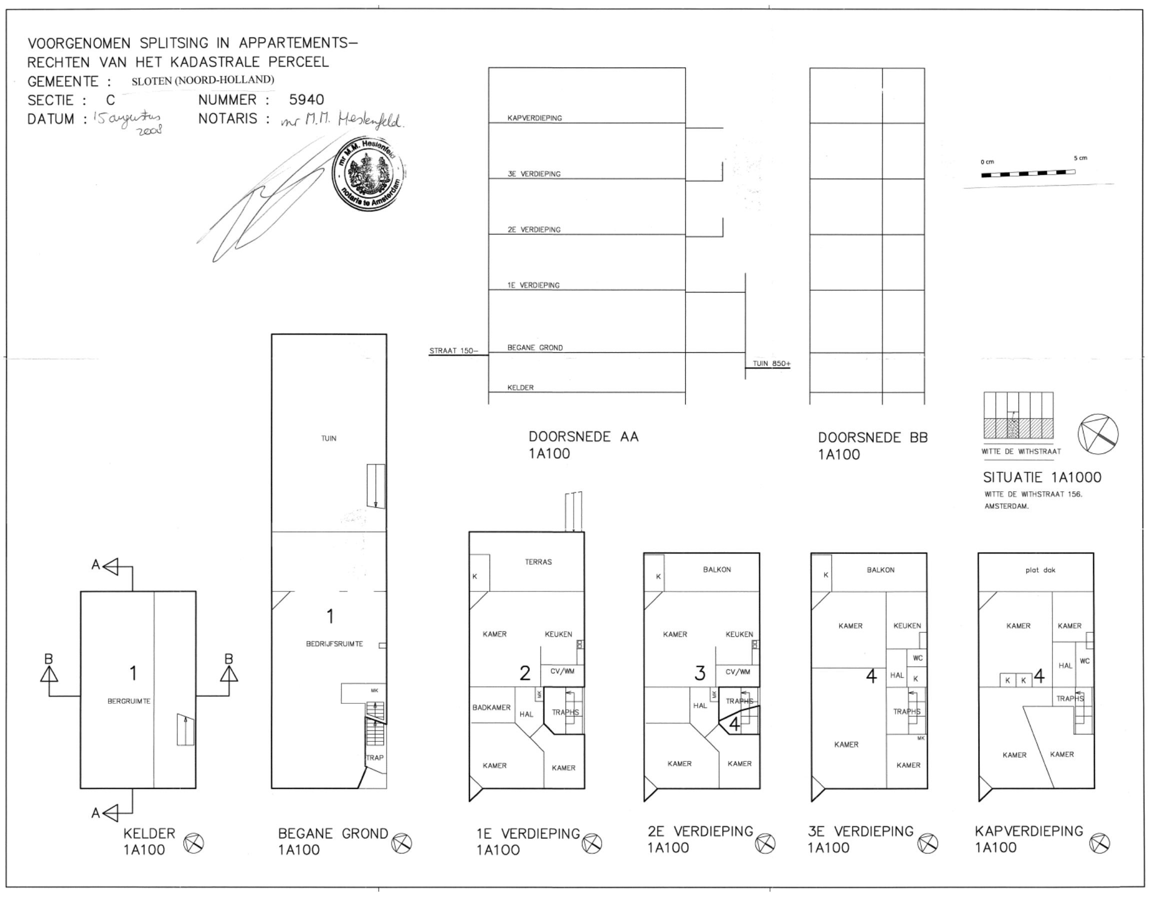


Figure 1.2: *Scanned image of a building containing floor plans including ownership rights (shown as large number) of each floor. The ownership boundaries are indicated with thick lines.*

The method used by the current system to deal with parcels having multiple ownership rights has a limitation. A scanned image of the floor plans is only human readable. Queries such as, 'who owns the 4th floor in a building with parcel number 5940?', or 'who is the neighbour of Mr. Simons?' cannot be answered by a computer, as the system will have to interpret the information from the scanned floor plans.

A solution to the above problem is by extracting the information from these scanned images. Information such as ownership boundary, ownership rights, floor number etc. are extracted. In order to extract the information from the scanned images, an automatic recognition system should be developed. This method recognizes and then extracts the information from the scanned images and stores it in a computer processable format. This will enable it to answer all the related queries.

Once the information is extracted from the scanned images, it has to be visualized. A possible solution is to visualize the extracted information in 2D. However, while using 2D, the visualization is cluttered when used for parcels with multiple ownership rights. Therefore, instead of visualizing in 2D, a 3D visualization could help to avoid the clutter. Figure 1.3 shows the 3D visualization of the data. Here, the floors are stacked on top of each other and placed on the parcel. This way, the information of each ownership right is visualized separately.

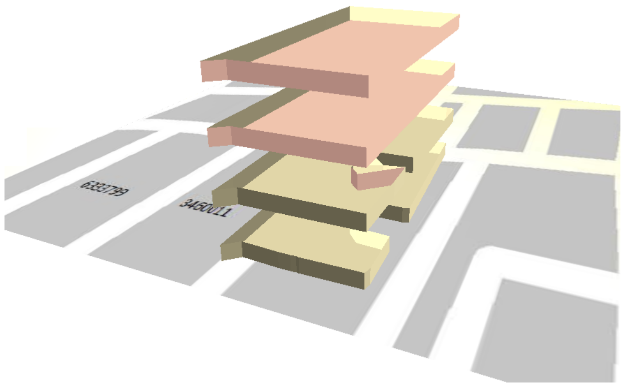


Figure 1.3: *3D modeled building with distinguishable floors, placed on a parcel.*

In order to visualize in 3D, the data must be represented in a suitable format. There are several formats available that can be used to visualize the extracted information. But most of the formats define models purely based on the geometric aspects (use of lines and polygons to represent the shape of a desired part). A format that adds semantic (meaning and relationship between objects) and topological (spatial properties) aspects into the models, along with geometrical aspects is *CityGML*. CityGML (G. Gröger 2008) is a common semantic information model for the representation of 3D urban objects that can be shared over different applications. It is an XML-based format for the storage and exchange of virtual 3D city models. By representing the data in CityGML format, a 3D city model is obtained, which can be visualized. CityGML is also extendable to adapt to the requirements of specific applications. Hence CityGML is a potential format to represent ownership rights information from the scanned images of floor plans.

1.1 Aim of the project

The aim of the project is to:

- Conduct a feasibility study on CityGML to represent legal information like ownership rights, ownership boundaries.
- Develop a method to automatically extract the information from building floor plans and represent it in computer processable format.

Figure 1.4 shows an overview of the project.

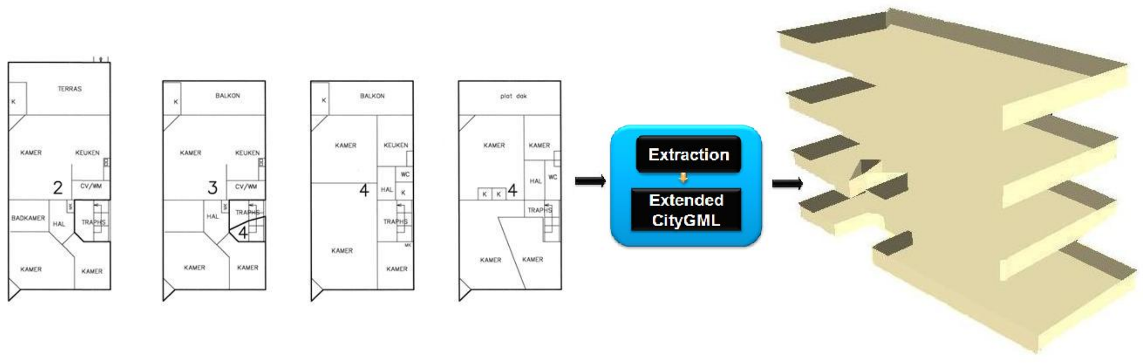


Figure 1.4: *Overview of the project, which shows a scanned image of the floor plans as an input to the system and an extended CityGML model as an output.*

This project is divided into two parts. This thesis contains the study on developing a method to develop a method to automatically extract the information from building floor plans and represent it in computer processable format. Then representing the extracted information in CityGML(D'Silva 2009) to visualize is conducted as another part of the project.

1.2 Problem definition

The aim is to develop a method to automatically extract ownership rights, ownership boundary, floor number from scanned floor plans of apartment buildings. The floor plans also contain details which are not relevant such as text describing room type and thin lines to indicate the subdivision of an apartments into rooms. These irrelevant details should be discarded. The extracted information should be represented in a suitable format that can be easily converted into CityGML for visualization purposes.

1.3 Kadaster floor plan standards

The Kadaster follows some standards to draw floor plans of the buildings. These standards are considered as the requirements to be followed in the floor plans. The requirements are:

- Floor plans scanned with high resolution
- The images are either binary or grey scale images
- Thick lines in the image indicate ownership boundary
- The numbers within the ownership boundary indicate the ownership rights
- Only one ownership right associated with an ownership boundary
- Numbers do not overlap with graphics such as lines and symbols.

1.4 Related work

Some work has been done in the field of extracting information from cadastral maps(den Hartog et al. 1996, Nakajima 1984, Lawrence et al. 1996, Ogier et al. 1998, Cofer and Tou 1972, Freeman and Pieroni 1980), but little work has been done in the field of extracting information from building floor plans. Cadastral maps and floor plans both are line drawings with similar properties having lines to represent features of the object. The work done on the cadastral maps may be possibly used in extracting information from the floor plans. One of such works done by Chen (Chen et al. 1996) gives some insight in to problems involved in automatic information extraction such as identification of numbers, text in the image and gives a solution for handling, then the paper describes a map interpretation system for automatic extraction of high level information (such as parcels and their attributes) from scanned images of Chinese cadastral maps. The paper describes three main components needed for automatic information extraction. They are text/graphics separation, parcel extraction and rotated character recognition. The idea of the text extraction method can be extended to automatic extraction of information from the floor plans in our problem.

Musavi *et al.* (Musavi et al. 1988) present automatic methods for processing digitized images of cadastral maps. The paper explains a method for scanning, thresholding, salt and pepper noise filtering, thinning, chain coding and vector reduction methods used to automate the conversion process. The paper explains algorithms for raster to vector conversion and segmentation. It also gives efficient ways to remove noise from the image. Boatto *et al.* (Boatto et al. 1995) describe the process to be followed to develop an interpretation system for Italian land register maps and also describes a graph based representation of the extracted information. The paper proposes an idea of representing corners as nodes and lines linking these corners as edges in the graph representation. This gives insight to develop a graph based system, as graph processing is much easier and simpler to do compared to image processing to identify the features of the objects in an image.

The research identifies that parameters for the automatic extraction of information from cadastral maps depends on standards adopted by each country. Janssen *et al* (Janssen et al. 1993) discuss the simple and basic system for automatic interpretation of Dutch Cadastral maps. The paper proposes to incorporate knowledge of the rules for drawing these maps in a system to automate the interpretation of the cadastral maps. This work identifies that rule based processing techniques will have much better accuracy and efficiency compared to other processing techniques such as identifying the parcel based on region growing segmentation of the objects. The rule based system can easily identify the relevant and irrelevant information and concentrating processing on relevant data results in an efficient system. This work motivates us to develop a rule based system.

1.5 Results

As a result of our study, we propose an extension to CityGML(D'Silva 2009), developed for cadastral purpose. This extension shows ownership rights and other details associated to each apartment building, which provides a solution for the representation of multiple ownership rights. We have also developed a method for automatic extraction of information from scanned floor plans of the building. This method extracts relevant information such as ownership rights, ownership boundaries, floor numbers from these images. The extracted information from these scanned images can be represented in the proposed extension to CityGML. However, the method of extracting the information does have some limitations. For example, when the number that represents ownership rights overlaps with lines that represent ownership boundaries, our method cannot currently handle this situation. The thesis presents a *proof of concept* for an image processing pipeline and way to represent the multiple ownership rights associated with a parcel.

1.6 Organization of the thesis

This thesis focuses on the feasibility study on development of a method to automatically extract the information from building floor plans and representing the information in computer processable format.

The remainder of this thesis is organized as follows. Basic concepts of image processing and graphs are discussed in Chapter 2. Chapter 3 introduces our image processing approach for feature extraction from scanned floor plans. The techniques used in our approach include thresholding, morphological operations and corner detection. This chapter also describes the method to convert the essential part of the floor plan to a graph representation. In Chapter 4, we describe the various graph processing techniques used to extract the required information such as ownership rights and its boundary, from the graph. The method to represent this information in computer processable format is also discussed. The results are given in Chapter 5, where we show a performance evaluation of our approach on various input floor plans. How our approach handles exceptions is also discussed. Finally, we draw conclusions in Chapter 6.

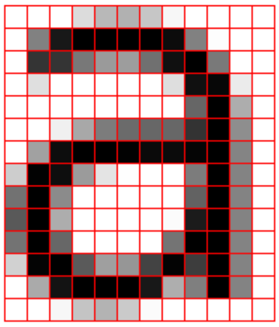
Chapter 2

Basic Concepts

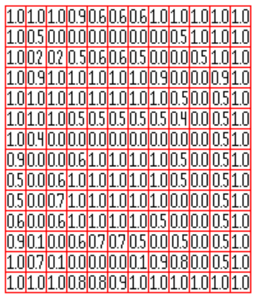
The building floor plans are given as scanned images and we need to extract information about the features of the objects in the image. This can be done using image processing and image analysis. Once the extraction process is completed, the extracted information is represented in a format in which it can be easily processed further. One simple and efficient way to do this is to represent it by a graph. This chapter describes the basic concepts used in this thesis.

2.1 Computer Representation of Raster Images

A raster image is defined as a rectangular array (two-dimensional matrix) of sampled values. Each element in the array represents the smallest element in the image called *pixel*.
These pixels are of uniform size and shape. Mostly, they are assumed to be square, do not overlap and there are no gaps in between them. The pixels are a digital representation of the portion of the image they correspond to. The pixels are associated with a number known as pixel value corresponding to the color and intensity of the element in that position of the image. The higher the values the brighter the element and the lower the values the darker the picture element.
The number of pixels in the horizontal (width) and vertical directions (height) represents the dimensions of the raster image. Each horizontal pixel row in the image is called a scan line. The quality of a raster image is determined by the total number of pixels called *resolution* and the amount of information in each pixel. Fig.2.1 shows the computer representation of a raster image containing the letter "a". The matrix on the right indicates the computer representation of the image in the left. A black pixel correspond to the value zero, a white pixel to the value one and shades of grey corresponds to intensities in between zero and one. The dimension of the image is 12×14.



(a) Raster image



1.0	1.0	1.0	0.9	0.6	0.6	0.6	1.0	1.0	1.0	1.0	1.0
1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0	1.0	1.0
1.0	0.2	0.2	0.5	0.6	0.6	0.5	0.0	0.0	0.5	1.0	1.0
1.0	0.9	1.0	1.0	1.0	1.0	1.0	0.9	0.0	0.0	0.9	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
1.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5	0.4	0.0	0.5	1.0
1.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0
0.9	0.0	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.6	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.7	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.5	1.0
0.6	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.0	0.5	1.0
0.9	0.1	0.0	0.6	0.7	0.7	0.5	0.0	0.5	0.0	0.5	1.0
1.0	0.7	0.1	0.0	0.0	0.0	0.1	0.9	0.8	0.0	0.5	1.0
1.0	1.0	1.0	0.8	0.8	0.9	1.0	1.0	1.0	1.0	1.0	1.0

(b) Pixel representation.

Figure 2.1: *Representation of raster image.*

There are three types of raster images: binary images, grey scale images and color images. Binary images can have only two colors black and white which represent background and foreground respectively. The advantages of binary images are low storage space, simple and easy processing. An example of a binary image is shown in Fig.2.2(a).

Grey scale images can have more intensity values usually up to 255 for an 8-bit representation. In that case, zero usually represents black (weakest) and the maximum value (255 for 8-bits) represents white (strongest). An example of an 8-bit grey scale image is shown in Fig.2.2(b).

Color images include both color and intensity information for each pixel. Each pixel provides this information, which will be interpreted in a suitable color space. An example of a color image is shown in Fig.2.2(c).

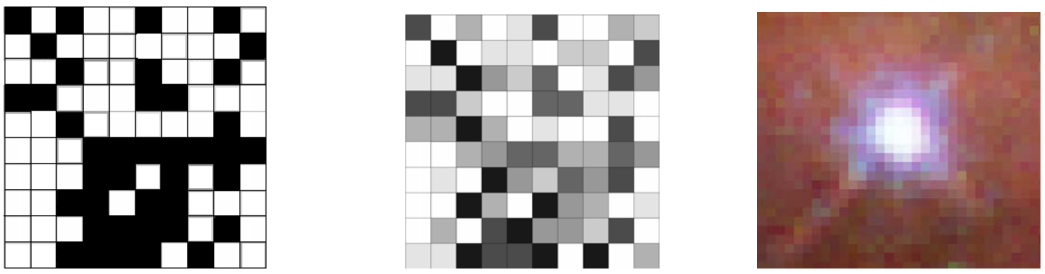

(a) Binary image. (b) Grey scale image. (c) Color image

Figure 2.2: *Different types of images.*

2.2 Image Processing and Analysis

This section explains about some image processing terminology and describes the methods that are used in this thesis.

2.2.1 Neighborhood

A pixel p with coordinates (x, y) has four horizontal, vertical neighbor pixels and four diagonal neighbor pixels.
The horizontal (B, C) and vertical (A, D) neighbor pixels are shown in Fig. 2.3(a) and their coordinates are given by:

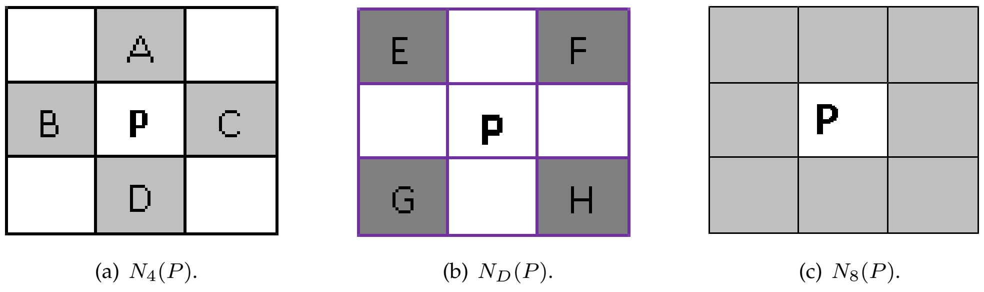
A=$(x, y-1)$, B=$(x-1, y)$, C=$(x+1, y)$, D=$(x, y+1)$.

The set of these neighbors is denoted by $N_4(P)$.
The diagonal pixels of p named E, F, G and H in Fig.2.3(b) have coordinates

E =$(x-1, y-1)$, F = $(x+1, y-1)$, G=$(x-1, y+1)$, H=$(x+1, y+1)$.

The set of these diagonal neighbors is denoted by $N_D(p)$.

Finally, the combination of sets $N_4(p)$ and $N_D(p)$ yields the 8-neighbors of p denoted as $N_8(p)$, which is shown in Fig.2.2 (c).



(a) $N_4(P)$. (b) $N_D(P)$. (c) $N_8(P)$.

Figure 2.3: *Representation of Raster Image.*

2.2.2 Connectivity and Adjacency

Connectivity is the fundamental concept used to identify the borders and regions. We say two pixels are connected, if one of the pixels belongs to a neighborhood set of the other one.
We use both 4-adjacency and 8-adjacency in the following. They are defined as below.

- *4-adjacency*: Two foreground pixels p and q are 4-adjacent if q is in the set $N_4(p)$.
- *8-adjacency*: Two foreground pixels p and q are 8-adjacent if q is in the set $N_8(p)$.

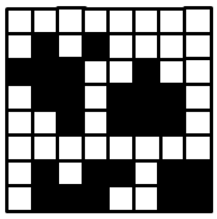
A path between two pixels $p(x, y)$ and $q(s, t)$ is defined as a sequence of distinct pixels with coordinates

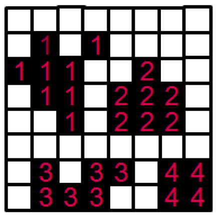
$$(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots\ (x_i, y_i), \ldots\ (x_n, y_n)$$

Where $(x_0, y_0) = (x, y), (x_n, y_n) = (s, t)$ and pixels (x_i, y_i) and (x_{i-1}, y_{i-1}) are adjacent for $1 \leq i \leq n$.

2.2.3 Connected Component Labeling

Let S be an object (component) representing a subset of pixels in an image. Consider any two pixels in the component S, we say that these pixels are connected in S if there exists a path between these pixels and all the pixels in this path belong to S. For any pixel p in S, the set of pixels that are connected to it in S is called a connected component of S. If it has only one connected component then set S is called a connected set.
The process of identifying the connected component and labeling their pixels with unique number or color is called connected component labeling. The efficient implementation of the connected component labeling algorithm (Stefano and Bulgarelli 1999) runs in linear time. The result of connected component labeling is shown in Fig.2.4 where it shows the connected components identified and assigned a unique value for the pixels in each connected component.


(a) Input Image having four connected components.



(b) Output: connected components are identified and assigned a unique value for the pixels in each connected component.

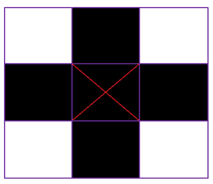
Figure 2.4: *Connected component labeling.*

2.2.4 Morphological Operations

Morphological operations are used to understand the structure and the form of an image. The basis for morphological processing originates from set theory. Most of the morphological operations are based on union, intersection and not operations from set theory which transforms the images according to rules of it. Morphological operations give solutions for numerous image processing problems like salt and pepper noise removal, identifying the topology of the object etc. Topology defines the spatial property of an object.

Structuring Element

The exact effect of any morphological operation on an image is determined by a structuring element. A structuring element is a rectangular grid of patterns. The structuring element consists of a pattern specified as the coordinates of a number of discrete points relative to some origin. The two most common structuring elements (given a Cartesian grid) are the 4-connected and 8-connected sets, N_4 and N_8 respectively as shown in Fig.2.5. The structuring element (SE) decides the neighborhood of each pixel in the image.


(a) N_4.

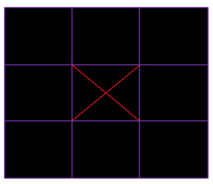

(b) N_8.

Figure 2.5: *Structuring Element.*

Dilation

The two basic operations in morphology are dilation and erosion. These two operations define most of the other morphological operations. The basic effect of the dilation operation on a binary image is to enlarge the boundaries of the foreground region of the image and fill the isolated background pixels in the foreground region. Thus areas of foreground pixels will grow in size while the holes within the foreground regions become smaller.
Let A represent the image pixels and B be the structuring element which belongs to the set Z^2. The reflection set of B denoted as B̂ is defined as

$$\hat{B}=\{w|w = -b, \text{for } b \in B\}$$

Then $(\hat{B})_z$ denotes translation of set B by vector z= $(z_1,\ z_2)$ defined as

$$(\hat{B})_z = \{c|c = b + z, \text{for } b \in B\}$$

The dilation of A by B denoted as $A \oplus B$ is defined as

$$A \oplus B = \{z|(\hat{B})_z \cap A \neq \emptyset\}$$

In our case the black pixels represent the foreground and white pixels represent the background pixels. The exact dilation operation is described as for every foreground pixel, all pixels in the neighborhood defined by the structuring element are made to foreground pixels.

Erosion

Erosion is another basic morphological operation. The basic effect of erosion on a binary image is to shrink the boundaries of the foreground region of the image and enlarge the isolated background pixels in the foreground region. In the erosion operation, all pixels in the foreground ground region which have background pixels in the neighborhood defined by the structuring element are made in to background region pixels. This operation can be used to remove isolated patches of the foreground region pixels.

The erosion of A by B denoted as $A \ominus B$ and is defined as

$$A \ominus B = \{z|(\hat{B})_z \subseteq A\}$$

The exact operation is described as, for each background pixel, all pixels in the neighborhood defined by the structuring element are made in to background pixels. In the boundaries, if we do not have neighboring pixels since they go out of the boundary then we consider them as background pixels.
Dilation and erosion are dual to each other with respect to the set complementation and the reflection. That is:

$$(A \ominus B)^c = A^c \oplus \hat{B}$$

Closing

The closing operation is defined as the dilation operation followed by the erosion operation. Let set A represent the image pixels and set B represent a structuring element then the closing of A by B denoted as $A \bullet B$ is defined as

$$A \bullet B = (A \oplus B) \ominus B$$

Closing operation is used to:

- Smooth sections of contours
- Fuse narrow breaks and long thin gulfs
- Eliminate small holes
- Fill gaps in contours

Closing operation removes islands and thin filaments of background pixels. This operation is useful for handling noisy images where some foreground pixels are made as background pixels, i.e. missing pixels.

Opening

The opening operation is defined as the erosion operation followed by the dilation operation. The opening of A by B denoted as $A \circ B$ is defined as:

$$A \circ B = (A \ominus B) \oplus B$$

Opening operation is used to:

- Smooth sections of contours
- Break narrow bridges
- Eliminate thin protrusions

Opening removes islands and thin filaments of foreground pixels. Opening and closing operations are duals of each other with respect to set complementation and reflection. That is

$$(A \bullet B)^c = (A^c \circ \hat{B})$$

We can visualize the effect of each morphological operation in Fig.2.6.

Skeletonization

Skeletonization is the process of computing skeletons of the objects. The skeletons describe the geometric shape of an object. They capture the topology and geometry of the shape in a compact manner. It is possible to reconstruct the original object from its skeleton. The main features of the skeletons are:

- Thin
- Preserve connectivity
- Preserve topology

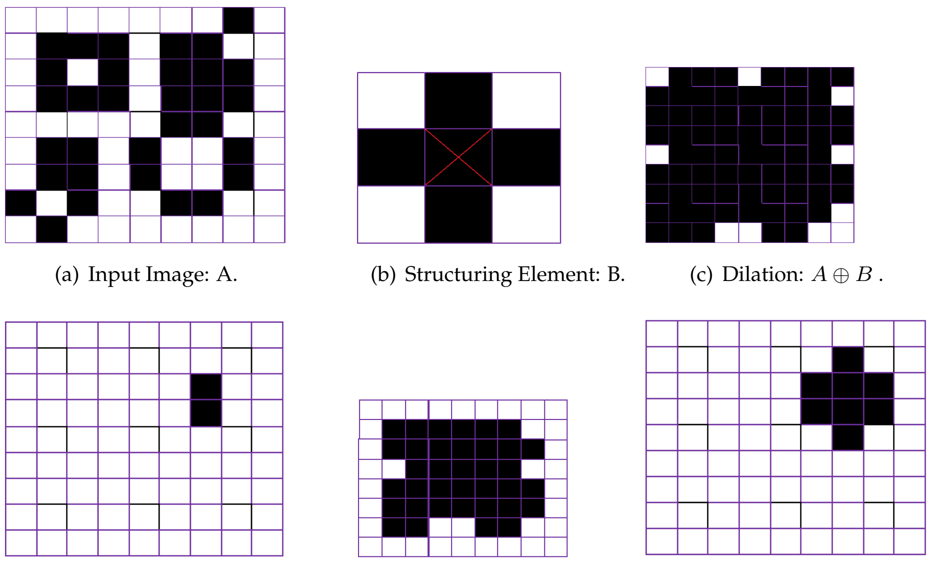

(a) Input Image: A. (b) Structuring Element: B. (c) Dilation: $A \oplus B$. (d) Erosion : $A \ominus B$. (e) Closing : $A \bullet B$. (f) Opening : $A \circ B$.

Figure 2.6: *Morphological operations.*

- Centered within the shape

Thin: The skeleton is a thin representation of the original object. Ideally, the skeleton should be one pixel wide.

Preserves connectivity: The skeletons should preserve the connectivity as in the original shape even in the presence of the noise. Our approach uses contour detection to identify the object boundary, hence it is important to preserve connectivity.

Preserves topology: The skeletons must preserve the topology and geometry of the object. This is important as in our implementation we are interested in only the topology of an object.

Centered: The skeletons should be somehow centered within the object.

The skeletonization can be done using various methods like thinning, voronoi-diagram based, distance field etc.(Reiners 2009), but in our implementation we use the thinning operation for the skeletonization process. Thinning is used to obtain the thin version of the digital objects in the image while maintaining the topology. Thinning is done by erosions.
The thinning operation on a set A by structuring element B denoted as $A \otimes B$, is defined as

$$A \otimes B = A - (A \circledast B)$$

Where $(A \circledast B)$ indicates the hit-or-miss transform (Gonzalez and Woods 2001), defined as

$(A \circledast B) = (A \ominus X) \cap [A^c \ominus (W - X)]$.
Where X indicates a shape, W indicates a window which encloses X .

The thinning operation is done by shifting structuring element B to each pixel position in the image A and in each pixel position we compare B with pixels in image A at that place. If the pixels exactly match with the structuring element then pixel at the origin of the structuring element is set to background pixel, otherwise it is left unchanged. Note that the structuring element must always have a zero (black) or a blank at its origin if it is to have any effect.

Thinning is done by iteratively removing the simple points from the set of object points. The simple points are pixels whose removal does not change the topology of the object and will be always located in the boundary of the object. The removal of the simple points introduces new simple points. After removing all the simple points the thin version of the object remains and it is called a skeleton. During the thinning process, we should ensure that center points are not removed and connectivity is preserved to obtain a centered skeleton. The skeletonization using thinning is computationally efficient as the thinning process is simple and easy to implement. The thinning process should be done in parallel with all simple points to ensure the centeredness of the skeleton.
As the thinning operation can be defined with erosion, we can define the skeletonization process in terms of erosion and opening operations.
The skeleton of a set A is defined as

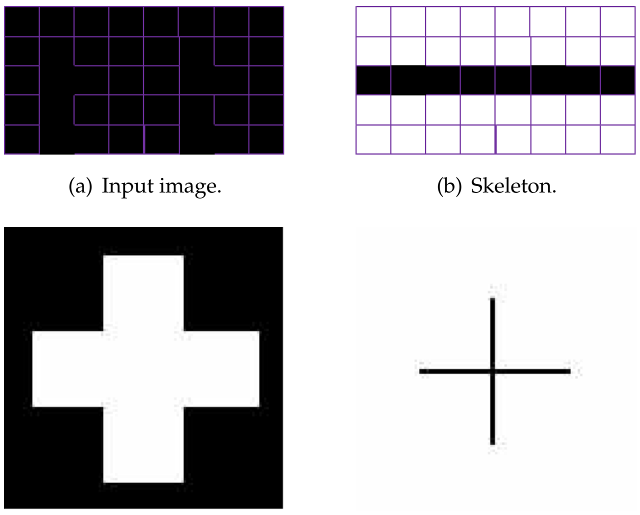
$$S(A) = \bigcup_{k=0}{}^{K} S_k(A)$$

Where $S_k(A) = (A \ominus kB) - (A \ominus kB) \circ B$

Where B is a structuring element and $(A \ominus kB)$ indicates k successive erosions of A. That is:

$$(A \ominus kB) = (.....((A \ominus B) \ominus B)....) \ominus B$$

Where K is the last iterative step before A erodes to an empty set. The result of the skeletonization process is shown in Fig.2.7 and we can observe that skeletonization works as expected by preserving the topology and connectivity of the objects in the image.


(a) Input image. (b) Skeleton.

(c) Input image. (d) Skeleton.

Figure 2.7: *Skeletonization operation.*

2.3 Graphs

Graphs are useful to store topological and geometrical information and they are also computationally efficient to perform most of the operations like identifying cycles, connectivity etc. Graphs represent connectivity information in a simple and efficient way.
A graph G is defined as an ordered pair of vertices (nodes) and edges (links). Which is represented as $G = (V, E)$, where V denotes the set of vertices and E denotes set of edges present in the graph G.
Vertex : Vertex (node) V of a graph is either an end point or intersection point of a graph. The vertices are usually visually represented by circles.

Edge : Edge shows link between two nodes. A link is the abstraction of the connectivity between nodes. An edge *e* is represented as an ordered pair of vertices (i, j) in a directed graph and an unordered pair of vertices in an undirected graph. In a visualization directed graphs will have the direction shown by an arrow. The undirected graphs are bi-directional and their edges are represented by a straight line.

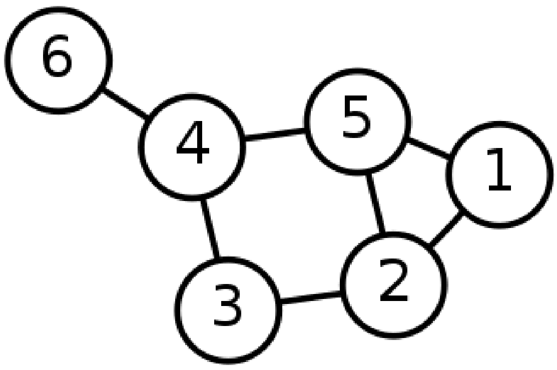


Figure 2.8: *Graph.*

Fig.2.8 shows an example of an undirected graph G (V, E). Where vertices V are $\{1, 2, 3, 4, 5, 6\}$ and edges E are $\{(1, 2), (1, 5), (2, 5), (2, 3), (3, 4), (4, 5), (4, 6)\}$.

Sub graph : Sub graph is a subset of a graph $G(V, E)$ represented by $G^{'}(V^{'}, E^{'})$. Where $V^{'}$ is a subset of V and $E^{'}$ is a subset of E.

Path : Path is a sequence of adjacent edges that are traveled in the same direction. A path exists between two vertices (A, B) when there is a sequence of uninterrupted edges while traveling from vertex A to vertex B.

Cycle: A path is called cycle when initial and terminal nodes are the same and there are no edges that are traveled more than once.

Circuit: A circuit is a cycle where all the edges are traveled in the same direction.

Chapter 3

Building Reconstruction from Floor Plans

The aim is to automatically extract the information such as the ownership rights and their boundaries etc. from the scanned building floor plans. Fig.3.1 shows the process pipeline to extract the information. The automatic recognition system takes the scanned building floor plans as the input. Once we acquire the image, we perform a series of processing on this image, as shown in the figure, which is later converted into a graph representation. This representation contains all the necessary information in the original image. The process involved is:

- Step 1: Preprocessing
  - preprocessing is done on these images to convert it into binary and to remove the noise from the image
- Step 2: Data reduction
  - Optical character recognition is used to identify the meaning of text and numbers in the image
  - Removal of the unnecessary information present in the image
- Step 3: Graph construction
  - Extracting the topology
  - Identifying the corners of the object in the image
  - Constructing a graph from corner detected object

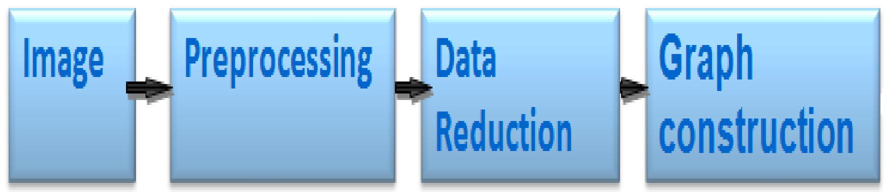


Figure 3.1: *Feature extraction of raster images.*

The graph representation will have all the information regarding apartment boundaries, ownership rights and the coordinates of their boundaries. This chapter explains the process pipeline used for the information extraction.

3.1 Building Floor Plans

The scanned images are floor plans of buildings. These images are scanned black and white line drawings which describe the geometry and topology of the buildings. These drawings are scanned with high resolution, which results in high quality images. These are stored in TIFF(Tagged image file format) format to maintain the high quality. In these drawings, the boundary of each ownership right is represented by a set of polygons. Each such a polygon represents ownership rights of one legal person and it is identified with a unique number. There can also be labels (in Dutch) describing usage of the property, e.g. labels such as keukens, kamer, balkon, etc. The drawings contain continuous thick lines, continuous thin lines, and symbols (alphanumeric characters and cadastral symbols).

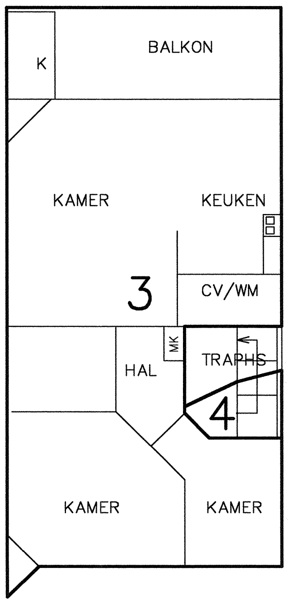
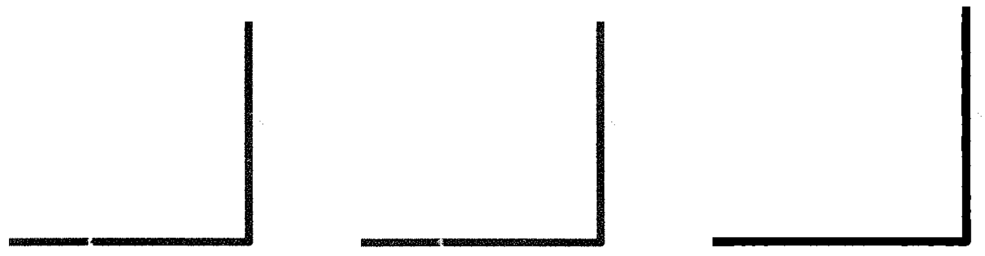


Figure 3.2: *Building floor plan: The large numbers indicate ownership rights. The ownership boundaries are visible as thick lines. The thin lines and text labels provide a subdivision of the property.*

One of the building floor plans is shown in Fig 3.2. In this diagram thick lines indicate the boundary of the ownership. Each of such a region formed by a thick line will always be associated with a number to indicate the ownership rights. There are no regions enclosed by the thick lines which will have more than one number associated with them and there will be no region without such a number. The text associated with this region indicates usage type of that ownership right. Examples are *Kamer* indicating it is a room, *Keuken* indicating kitchen, *Hal* indicating hall etc. The numbers indicating the ownership rights are usually in the middle of the polygon representing the apartment boundary. The numbers usually do not overlap with the lines or symbols. But the texts describing the usage do overlap with the lines and symbols and that makes it difficult to perform character recognition. The Dutch land register authority issues a set of guidelines for drawing the floor plans such as thick lines for ownership boundary and the numbers in large font to indicate ownership rights etc. These rules form a graphic language and allow the reader to understand the drawing. However, it is not possible to rely entirely on these guidelines for the automatic recognition of floor plans because of the reason that the rules are not always followed strictly.

3.2 Preprocessing

The scanned building floor plan images are stored in gray scale format and will have some noise associated to them. The floor plans were originally black and white drawings but as result of scanning, these drawings will be stored in gray scale format. The gray scale format does not provide any additional information for our purpose, so the first step is to convert the images into binary images. The noise in the image affects the extraction of the actual information from the image. Hence the next step is noise removal. The preprocessing step converts images back into binary format and performs noise removal. The details of these operations are described in the following.



(a) Noisy gray scale input image. (b) Result image after thresholding. (c) Result image after closing: gaps are filled.

Figure 3.3: *Thresholding and closing operation.*

3.2.1 Thresholding

The gray scale images are converted to binary images (i.e. black and white) using a *Thresholding* operation. Thresholding classifies each pixel as either belonging to the foreground (black) or the background (white) according to a condition indicating if the gray level is larger or smaller than a suitable value. Through the analysis of various input images, we found that the value 128 is the best suited value to be considered as the threshold, as it emphasizes the noise as minimal as possible. It is important that the thresholded image reproduces the original image is as accurate as possible. Otherwise it will affect the graph construction phase, which is discussed later. The binary images obtained as a result of thresholding will simplify the further processing.

The Fig.3.3(b) shows the result of the thresholding operation and we can observe that the output image is binary. Thresholding may emphasize pre-existing noise. For example, some pixels belonging to the foreground region may have pixel value less than 128 and will be changed into a background pixel (missing pixel) as a result of the thresholding operation. As a result some of the gray pixels of the foreground region in the input image are turned into white pixels. This is shown in Fig.3.3(b). Hence the system needs to have some process which will remove the noise introduced due to thresholding.

3.2.2 Noise Removal

The scanned floor plans are almost noise free, but there are small irregularities in the image such as some missing pixels and gaps in the foreground region. The gaps in the image are caused due to the irregularities while drawing the floor plans. This will be inherited even in the image while scanning. There will be some missing pixels due to the noise and as a result of the thresholding operation. Having such irregularities and discontinuities is the main concern for *contour detection* which plays a major role in the detection of the object boundary. The contour detection requires continuity in the object pixels; hence it is very crucial to eliminate the discontinuities such as missing pixels and gaps.

Noise removal (denoising) is the process of removing unwanted noise from an image. A denoised image is an approximation to the underlying true image before it was contaminated. A good denoising process must simultaneously preserve structure and remove noise. Our approach considers these missing pixels and gaps as the noise in the image. By denoising, we refer to filling the missing pixels and gaps. The missing pixels and gaps can be filled using a mathematical morphological operation called *closing*, see section 2.2.4.

The result of the closing operation is shown in Fig.3.3(c); where we can observe that the closing operation filled small holes and gaps in foreground region. The noise in the image is mainly pepper noise (i.e. missing pixels), which can be effectively removed using closing operation resulting in a smooth continuous image which can be used for further analysis.

3.3 Data Reduction

Once preprocessing is done on an image, the next step is to remove all the irrelevant information from the image. Now we need to identify which information is relevant and which is not relevant. As we are interested only in the ownership rights and its boundaries, we conclude that the text representing usage type and thin lines indicating interior boundaries of the ownership rights are the irrelevant information. However, the numbers indicating the ownership rights are relevant. Hence the first step in data reduction is to identify the numbers and store the number information. Later we can remove these numbers from the image along with other irrelevant information. The steps involved in this process are as follows.

3.3.1 Label Identification

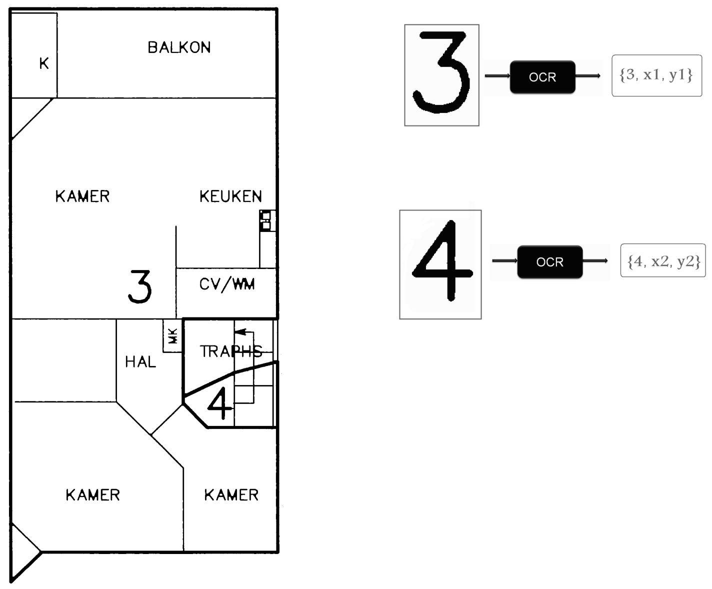
Label identification is one of the most important steps in the recognition system for building floor plans. Without label identification it is impossible to assign semantics to the objects identified in an image. Hence label identification plays a major role to have a complete and meaningful recognition system. It is also important to have an accurate and efficient label identification as even a small error in the identification process results in a flaw in the accuracy of the system such as assigning wrong ownership for an object in an image.

With label identification, we can identify usage type, numbers representing ownership rights and also detect total number of ownership rights in the floor plan (Apartment). We use the term *apartment* to refer a region owned by one legal person in a floor of the building. The labels in the image are numbers and text, having different font size. The recognition system concentrates on reading numbers, as we are only interested in identifying the ownership rights. In the current implementation label identification is limited to number detection. However, the same approach can be used for the text identification.

The label identification process involves three steps:

- Identifying the location of the labels within an image
- Extracting these labels from the image
- Recognizing the labels using OCR.

The entire Label identification process is illustrated in Fig.3.4.



(a) Input image which has numbers to indicate the ownership rights.

(b) Extraction and identification: Extracting the region of numbers from the image to a new image, which are send to optical character recognition for the identification.

Figure 3.4: *Label identification process to identify the labels in an image.*

There are some assumptions made regarding ownership right representation in the image. They are as follows

- Numbers do not overlap with any symbols or lines.
- Numbers are not rotated in the image.

- All the numbers have the same font and size

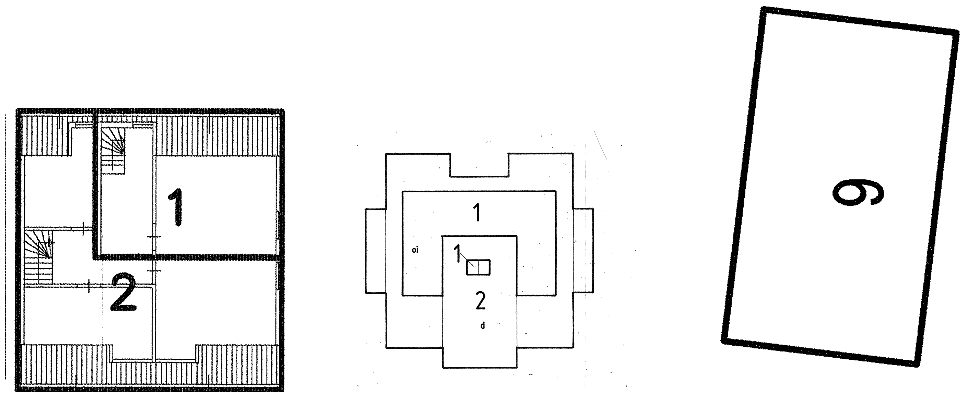
Considering the first assumption, the segmentation is performed based on the size of the objects in the image. In our approach we identify the connected components using *connected component labeling* and the number of pixels in each component, described as in section 2.2.3. This information will be used for the segmentation of the numbers. Once the connected component labeling process is done, pixels of each connected component will have a unique number. Then we can find the size of each connected component in terms of number of pixels in each component by counting the unique labels. The size property of a connected component can be used for segmentation of numbers and text from the image. Since the numbers and the text have different font size, it is possible to differentiate the numbers and the text. Through the analysis of the various input data we found that all the numbers which represent ownership rights of the apartment have size between 2000 and 4000 pixels. We mark all the components satisfying this size criterion as potential numbers. There may also be other small objects that satisfy this criterion, so not all of these objects are numbers. This poses no problem, since components that cannot be recognized as numbers in the following step will be discarded. Once the labels are recognized, the locations of these labels are identified.

The system needs to extract the labels from the scanned image and copy them to another image. Once these numbers are extracted and copied to a new image, the system needs to identify these numbers. The recognition system uses *Optical Character Recognition* (OCR) techniques to identify these numbers. OCR is a system that provides a full alphanumeric recognition of printed or handwritten characters in the images by processing them. The OCR system to recognize these numbers should be powerful, since it is very important to accurately identify these labels and assign the right meaning to the apartments in the floor plans. The recognition system uses *Google's Tesseract OCR*(Smith 2007) libraries, one of the powerful OCR libraries available as open source software.

The OCR library takes the image as input and returns all the text in it, but it does not return their positional information. Due to this limitation, each number in an image needs to be extracted to a new image one at a time to maintain its positional information. These images are sent for OCR to identify the numbers. The identified numbers are stored with their positional information.

Exceptions

Our recognition method is based on the assumption that numbers representing the ownership rights are always inside the corresponding region. But there are some scenarios which these numbers are outside the ownership boundary and a line is drawn to indicate the region it belongs to as shown in Fig.3.5(b). In such scenarios, the system results in a wrong recognition. Our method also assumes that numbers are not rotated, but Fig.3.5(c) shows a scenario in which number "6" is rotated over an angle of 90 degrees. So we would need to rotate the numbers before sending them to the OCR, but this feature is not implemented as part of the system. Another assumption is that there should not be any graphics such as lines overlapping with the numbers. But in Fig.3.5(a) there is a scenario in which number "2" overlaps with a line. In such scenarios, our method fails to work as we use size criterion for number identification. When a number overlaps with the graphics, the size criterion will not hold anymore. So clustering based approach needs to be used for number identification (Chen et al. 1996). In such scenarios we need to extract these numbers resulting in gaps in the line, which was overlapping with the number. These gaps can be filled considering the slopes of the partitioned lines, the width of the gap and the width of



(a) Number overlapping with line. (b) Ownership represented outside the boundary. (c) Number rotated.

Figure 3.5: *Exceptions in the floor plan drawings, makes difficult for automatic extraction of the information.*

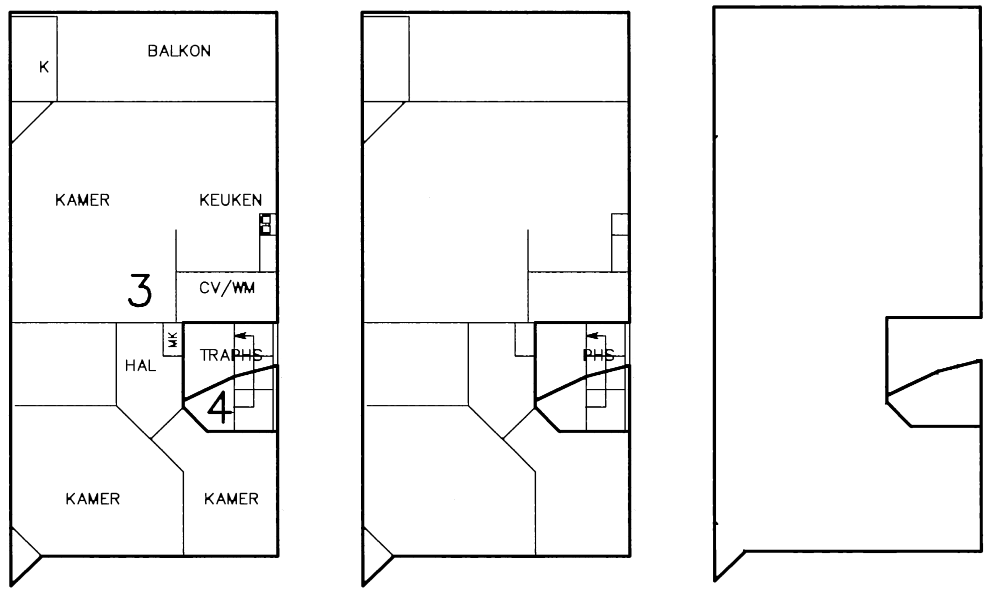
the number represented in the image. If the slopes of the partitioned lines are the same and the width matches with the width of the number in the image, then the system can fill these gaps on the line to make a complete continuous line. This is not yet implemented as part of this system.

3.3.2 Removing Labels and Thin lines

We are only concerned with identification of the ownership rights and its boundaries. All the information regarding the usage and type of the apartment, represented by the labels such as BALKON, KAMER are shown in Fig.3.6(a) is irrelevant. Also the interior information of the apartment represented by the thin lines is not relevant. Therefore we remove the text and thin lines from the scanned image in this step, so that the image can be used for further processing. We can also remove small objects such as isolated islands of black patches from the image.

We can remove all the small objects and labels from the image using their size property as we did as in the label identification step. In the previous step we identified all the connected components and their size using connected component labeling. In this step we assume that all the connected components (objects) with size less than 4000 pixels are either labels or small objects which have no significant meaning in the floor plans. The systems finds these pixels belonging to an object satisfying the size criterion and assigns a new pixel value i.e. white (1) which will make these objects part of the background. This is equivalent to removing objects from the image, which can be observed in Fig.3.6(b).

The next step is to remove thin lines which define the interiors of the apartments. The method used for this process is the morphological operation called *opening*, see section 2.2.4. The opening operation eliminates some of the pixels in the boundaries of the foreground region. The thin lines are one or two pixels wide and the opening operation completely removes these thin lines, leaving only thick lines in the images. In Fig.3.6(c) we can observe that all the thin lines are removed from the image, leaving only the thick lines. The opening operation also affects the thick lines by reducing their thickness. It will also turn some pixels in the foreground region to background region pixels. A closing operation restores the width of the thick lines and it is an optional operation based on the resolution of the scanning process. If the image has high resolution, then we can avoid this closing operation since the effect of opening makes no significant changes to thick



(a) Scanned floor plan: Having Text and Thin lines.

(b) Labels removed: Removing irrelevant text from the image.

(c) Thin lines removed: Removing thin lines in the interior of the boundary.

Figure 3.6: *Label and thin line removal removes all the irrelevant information such as thin lines and text from the image.*

lines.

3.4 Graph Construction

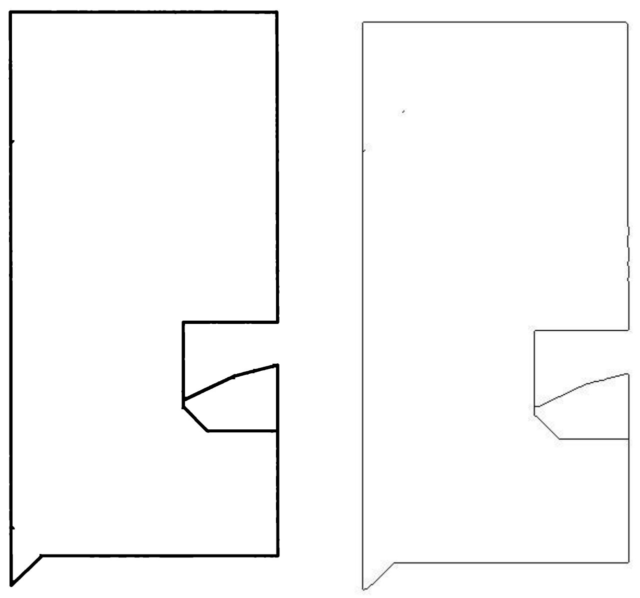
The image is now prepared to extract topology and geometry of objects. The extraction of the topology and geometry of objects can be done by shifting from the pixel world to some representation in which we can store this information in computer processable format. It is important to do this because to answer questions such as " Who is the neighbor of Mr. Simon (having ownership right 3)?", "who owns the fourth floor of building A?" is very hard with image data. One of the suitable formats to represent topology, connectivity and geometry of the information is a *graph representation*. The advantage of a graph representation is that it is easy to process the information stored, computationally efficient and also saves storage space.

As an extension to this thesis, the aim is to represent the information in CityGML format(D'Silva 2009). In CityGML all the objects are represented as polygons and these are specified by the coordinates of their corners. A graph representation will make a conversion to CityGML straightforward. The following section describes the steps involved in graph construction from the image data.

3.4.1 Skeletonization

After the data reduction step, the image is left with only thick lines indicating the boundaries of the apartments. In our approach we are only concerned with geometry and topology of the object. The further processing with these thick and high resolution images adds up the complexity of the system as an operation such as corner detection becomes very difficult and also computationally expensive. Fig.3.7(a) shows an image of boundaries of an object in thick lines. These thick lines do not give any additional useful information. However, all the subsequent operations needed to extract the information can be efficiently done on one pixel wide objects. So the objects in an image are converted into an object containing one pixel wide lines while preserving the features of the topology and the connectivity of the original object to improve the efficiency. This conversion can be done by the process called *skeletonization* described in section 2.2.4.

As a result of the skeletonization process we will get a skeleton of the object and this is use-


(a) Input image: Having thick lines.

(b) Skeleton: One pixel wide lines preserving geometry, connectivity and topology.

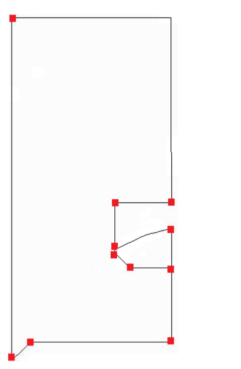
Figure 3.7: *Skeletonization process applied to an image.*

ful in providing a simple and compact representation of the object. The skeleton contains all the important information present in the original image such as topology and geometry which is required by the recognition system. The result of the skeletonization is shown in Fig.3.7(b) in which it is clear that this process preserves the topology and connectivity of the objects in the original image and the lines are just one pixel wide.

3.4.2 Corner Detection

Once the skeleton of the objects in the image is extracted, the next step is to identify the *corners* of the objects. A corner is defined as a point at which two or more edges intersect. Corner detection is the last step before the system converts an image into a graph representation. Each corner becomes a node in the graph, making it important to find all the corners accurately. Missing to detect a single corner may result in changing the topology and geometry of the object in the graph representation. One such scenario is shown in Fig.3.8, where the system misses to identify the topmost right corner. The resultant graph generated is shown Fig.3.8 (b) and we can observe that the entire geometry of the object is changed and resulting in a wrong interpretation of boundary of ownership rights by the system. Hence it is important to have a powerful and accurate corner detection algorithm which will identify all the corners of the object at any cost of computation.

There exist many corner detection algorithms which are either very accurate with high com-


(a) Input image: Error in corner detection algorithm as top left most corner not being detected.

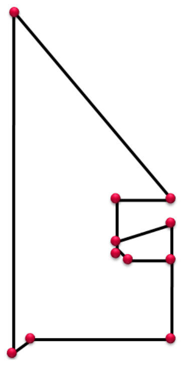

(b) Resultant graph: The geometry of the object is changed due to error in corner detection algorithm.

Figure 3.8: *The effect of error in corner detection algorithm resulting change in the geometry of the object.*

putational cost or computationally efficient with lower accuracy and there are a few that balance both the aspects. The computationally efficient algorithms are fast but may miss some corners, which makes them useless for this application. The algorithms which balance computational cost and accuracy will have reasonable speed and accuracy. The stringent requirement to identify all the corners makes these algorithms useless for our approach. In our approach we use the *Harris corner detection algorithm*(Harris and Stephens 1988) which is computationally expensive.

We choose Harris corner detection algorithm because of its invariance to:

- rotation
- scaling
- illumination variation
- image noise

The Harris corner detector is based on the local auto-correlation function of a signal, where it measures the local changes of the signal with patches shifted by a small amount in different directions.
Harris uses corner measures to determine whether a pixel is a corner or not. The corner measure is given by

$$\mathbf{C}(x, y) = \mathbf{det}(M) - k(\mathbf{trace}(M))^2$$

$$\mathbf{det}(M) = \lambda_1 * \lambda_2$$

trace$(M) = \lambda_1 + \lambda_2$

where k is a constant whose value is in between 0.04 and 0.06 to balance the contribution of the trace.

λ_1 and λ_2 are eigenvalues.

M is the Harris matrix (or Tensor) which is defined as

$$M(x, y) = w_G(r; \sigma) \left[\begin{array}{cc} {I_x}^2 & I_x I_y \\ I_x I_y & {I_y}^2 \end{array} \right].$$

Where I_x, I_y indicate the partial derivatives of the intensity function in the X and Y directions respectively.

- The derivatives of the intensity function *I(x, y)* are first calculated in each pixel point
- Then the entries of matrix M $({I_x}^2, {I_y}^2, I_x I_y)$ are obtained
- Finally, the entries are smoothed by a Gaussian filter $w_G(r; \sigma)$ of selected size σ

After smoothing and diagonalizing, the diagonal entries will be the two eigenvalues λ_1 and λ_2:

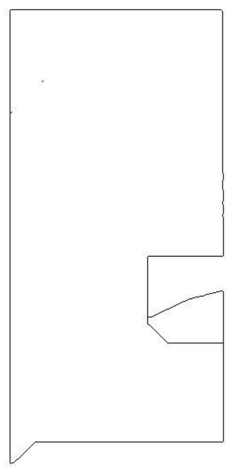
$$M_d = \left[\begin{array}{cc} \lambda_1 & 0 \\ 0 & \lambda_2 \end{array} \right].$$

The eigenvalues define the curvature and based on these values we determine whether the region is a corner or not:

- If both eigenvalues are small, then the windowed image region is of approximately constant intensity.
- If one is high and the other is low, then there is an edge.
- If both are high, then there is a corner.

The search for eigenvalues of the matrix is computationally expensive but here we can directly compute the trace and determinant of *M* to find the corners. A *corner is detected when* $C(x, y) > C_{thr}$, where C_{thr} is the threshold on corner strength and the value of C_{thr} is decided by the user. The smaller the threshold value, the larger the number of non corners (false negatives) being detected. The larger the value, the higher the false positives, i.e. corner detector may miss some real corners. In our application we choose a value between three and eight to identify the corners. These values are determined after analyzing the various input data and standard values preferred in (Harris and Stephens 1988).

The output of the Harris corner detection algorithm on an image is shown in Fig.3.9. In this figure corners are shown in red colored blocks. As we can see, all the corners of the objects in the image have been identified. We can also notice that there are some extra corners being detected, which are not really the corners of the objects. We can now either remove these extra corners in


(a) Input image : Having one pixel wide lines.

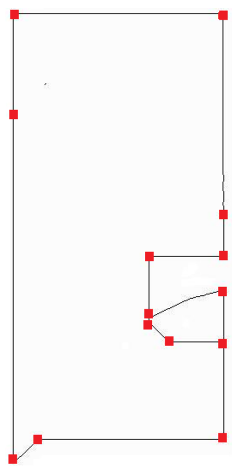

(b) Image after corner detection: All the corners of the object has been identified, including some extra corners.

Figure 3.9: *Harris corner detector applied to an image to identify the corners of the object.*

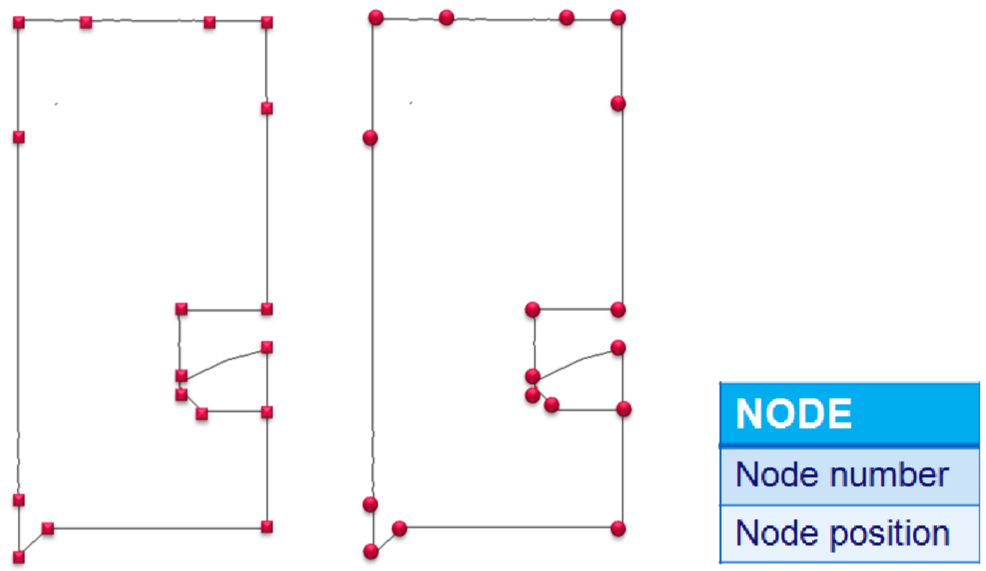
this step itself or it can be done after graph generation from this image. To remove these extra corners in this step, we consider the neighbor (N_8) pixels of the corner. The next step is to check whether these pixels intersect with the lines forming the corner. If there are only two such pixels and they are at 180 degrees, then we will remove these extra corners and join the line. In our approach this is done later as graph processing is easier and computationally efficient compared to doing in the same step using image processing. Doing this using graph processing helps in reducing the complexity of the system. Once all these corners of the objects in an image are detected, the graph construction of an image can be started, where each of these corners becomes a nodes in the graph.

3.4.3 Graph Construction

An essential step in the recognition system is the conversion of scanned floor plans to a *graph* representation. Floor plans are nothing but line drawings. In the graph representation the lines are decomposed into *edges* and corners which connect two or more lines, into *nodes*. During graph construction, the objects in the binary raster image are partitioned into a set of a line segments and corners, each corresponding to the edges and the nodes respectively. The corners of the objects in the image are identified in the previous step. As a result, the objects are represented by set of lines and corners, which can be converted into a suitable graph representation. Each connected object in the image will form a graph. If there are multiple isolated components then there will be many isolated graphs, called a *forest*. All the information required such as node position, edge length, angle etc. to reconstruct the image from the graph is identified and stored. As a result, whatever processing needs to be done on the image can be done on the graph and it results in an efficient system. The steps involved in the graph construction are identification of the nodes, edges, and also their attributes.

Node Identification

In node identification, all the identified corners of the objects in the image become the nodes. Once we identify these nodes, we store their attributes such as node number and their position. These attributes are nothing but the number and its position in the image. The image after node identification is shown in Fig.3.10(b), where bubbles show the nodes and Fig.3.10(c) shows the corresponding information stored.



(a) Input image: All the corners of the object been identified.

(b) Node identification: corners of the object turns in to nodes the graph.

(c) Node storage: The node number and position of the corner in the image is stored as the node position.

Figure 3.10: *Node identification and storage.*

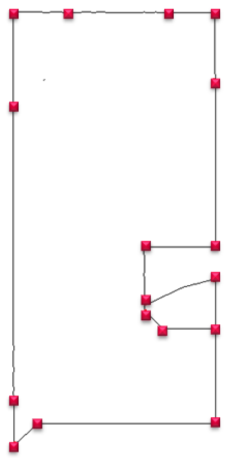
Edge Identification

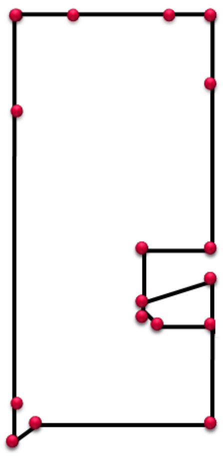
A graph is used to describe the connectivity between the nodes. The connectivity in the graph is shown in the form of edges. If the nodes are connected by a set of connected pixels in the foreground region, we identify this set of pixels to be an edge. Once we identify the edge, then the length of the edge is computed by counting the number of pixels between the nodes. The angle made by these edges is calculated as:

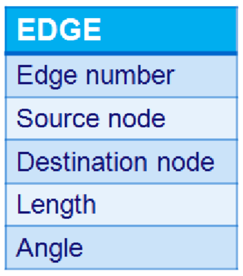
$$\text{Angle} = \tan^{-1}(|\tfrac{dy}{dx}|)$$

Where dx= (x-coordinate of destination node - x-coordinate of source node), dy= (y-coordinate of destination node - y-coordinate of source node) and | | represents the absolute value

We find the correct quadrant based on dx and dy and map them to absolute angle. Then we store source node and destination node, from which we can identify the position of the edge. Fig.3.11 shows the graph constructed from the image and its corresponding information stored. Once we construct the graph then it will have all the topological and structural properties needed for further processing.


(a) Corner detected image: Shows the connectivity between the corners.


(b) Edge identification: Connectivity between the corners are shown as edge in a graph.



EDGE
Edge number
Source node
Destination node
Length
Angle

(c) Edge storage: Information regarding edge is stored.

Figure 3.11: *Edge identification and storage.*

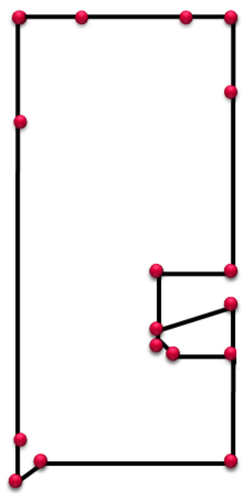
3.4.4 Graph Optimization

As a result of corner detection some non corners (not a real corner) of the objects were detected as corners. These corners are turned into nodes in the graph representation and they are removed in this step. The removal process of non corners is as follows:

- Identify all the nodes having only one incoming and outgoing edge
- check whether these edges makes an angle approximately 180 degrees (175 - 185)

Now the obtained nodes are considered as extra nodes and are deleted. Then delete the edges that connect this node and replace them by a single edge. This step improves the efficiency of the system. Fig.3.12 shows the effect of optimization and its corresponding result.

The further processing such as identifying boundary of ownership and floor number can be done using the graph we obtained and it does not involve processing on the image.


(a) Input graph: Having some extra non corner nodes in the graph.

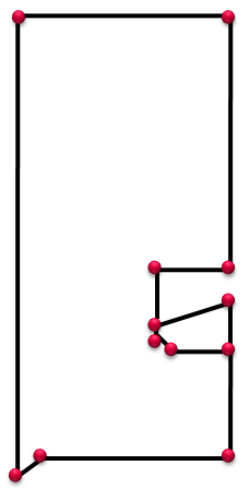

(b) Optimized graph: Extra non corner nodes are removed from the input graph.

Figure 3.12: *Optimization.*

Chapter 4

Graph Processing

An essential step in the automatic recognition system of the floor plans is the conversion from a raster image to its graph representation as explained in the previous chapter. The subsequent steps include identification of apartments and associating semantics like ownership rights, floor numbers etc., to the apartments in the drawing. The graph representation of the drawing is used for further processing. This chapter explains the process used to identify all the required information from the graph, as shown in Fig.4.1. This process takes the graph as the input and returns all the required information in the graph in an XML file format as the output. The information such as coordinates of the ownership boundary, the floor number and the ownership rights needs to be extracted. This process identifies all the required information by processing the input graph.

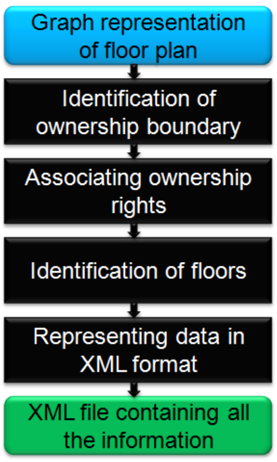

Graph representation of floor plan
Identification of ownership boundary
Associating ownership rights
Identification of floors
Representing data in XML format
XML file containing all the information

Figure 4.1: *Graph processing.*

4.1 Ownership boundary identification

The ownership boundary is the key information to be extracted from the image. This information is extracted by identifying faces in the graph. The current graph representation does not contain enough information for boundary identification. So we need to have a suitable representation for face identification by extracting some additional information from the graph. Hence the additional information required should be identified.

To identify this additional information, each edge is decomposed into two half edges, called *twin edges*. Since there may be many incoming and outgoing edges for an edge, the next and the previous edge for each edge needs to be defined to traverse through the graph. Once the next field denoted as e_{next}, is defined, the graph traversal can be performed. While traversing e_{next} decides on the next edge to be considered. The apartment boundary is defined by faces in the graph, as shown in Fig.4.2. The e_{next} is defined in such a way that while traversing through the graph always faces are identified.

To identify e_{next} for each edge, all the outgoing edges of the destination node are found. Then

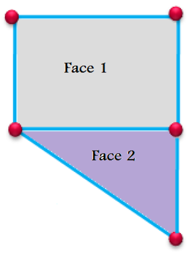


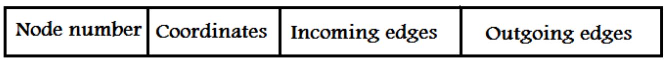
Figure 4.2: *Faces defines ownership boundary.*

the differences in the angle between the current edge and the outgoing edges are calculated. Once these differences are found, the edge having minimum angle difference is considered as the e_{next} for the current edge.

The calculation of e_{next} plays a very important role in this approach, as it defines the boundary of the ownership rights. To identify e_{next} for each edge the minimum angle difference is considered to ensure that the smallest possible region (face) is obtained while traversing in clock wise direction. The face obtained defines the boundary of the apartment. In a similar way previous field can also be calculated.

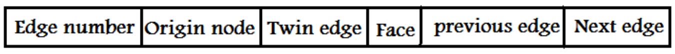
Once the identification of the apartment boundaries has been done, the next step is to identify the apartments connected to each other. This is done by examining all the edges defining the apartment boundaries and looking for their twin edges. If these twin edges belong to any other apartment boundary, then these apartments are considered as connected or neighbors. This information is needed for further processing.

A suitable representation for storing this information is a *doubly connected edge list* (DCEL)



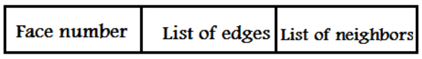
Node number	Coordinates	Incoming edges	Outgoing edges

(a) Node structure.



Edge number	Origin node	Twin edge	Face	previous edge	Next edge

(b) Edge structure.

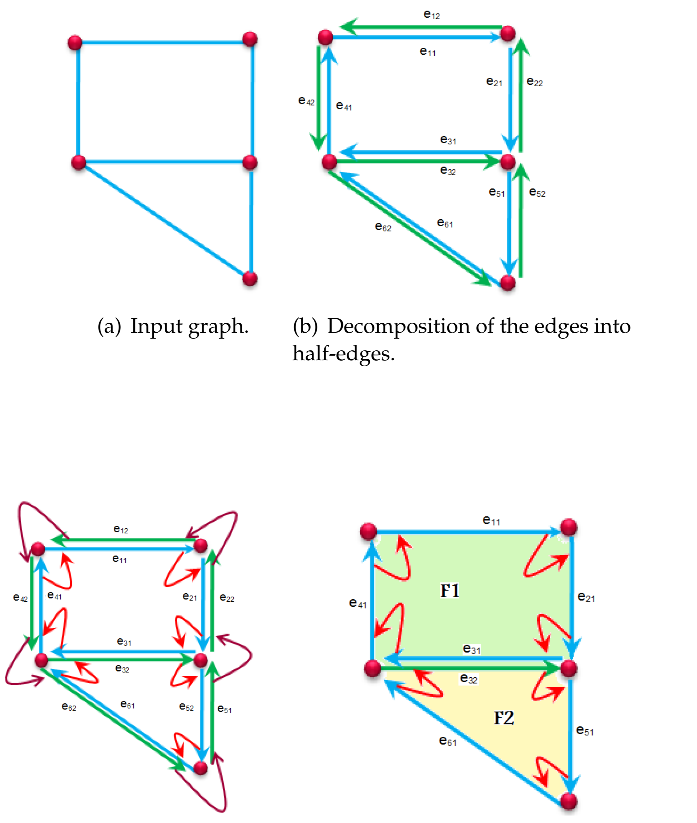


Face number	List of edges	List of neighbors

(c) Apartment data structure.

Figure 4.3: *Doubly connected edge list (DCEL).*

(de Berg et al. 2000). DCEL stores the information regarding nodes, edges and faces. The structure



(a) Input graph. (b) Decomposition of the edges into half-edges.

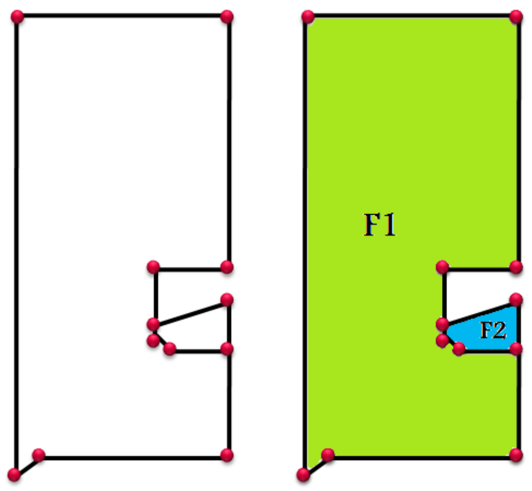
(c) Defining next of each edge. (d) Identifying the ownership boundary.

Figure 4.4: *Recognition process of ownership boundary in the graph representation.*

of the node data representation is shown in Fig.4.3(a) which shows the information regarding a node. This information includes node number (a unique number to identify the node), the co-ordinates of the positions of a node and also all the incoming edges and outgoing edges of that node. The information regarding edges is also stored, which includes edge number (a unique number to identify the edge), a pointer to the source node of the edge, a pointer to its twin edge, a face to which it belongs to, a pointer to the previous edge and a pointer to the next edge. The structure of an edge is shown in Fig.4.3(b). Fig. 5.3(c) shows information stored regarding the apartment boundary or the face. This information includes a face number which is used to identify the boundary, a pointer to the list of edges which forms the boundary and pointer to the list faces which are connected to this face.

Fig.4.4(a) shows a simple scenario which illustrates the identification of the ownership boundaries. Fig.4.4 shows the stages that the input graph undergoes in the identification process. The input graph has two faces. The twin edges are shown in Fig.4.4(b). After the decomposition of each edge, the e_{next} for each edge is defined. This is shown in Fig.4.4(c). The arrow shows the next edge, which is obtained by taking the minimum angle between the edges. The successor of e_{21} in the figure can either be e_{31}, or e_{51}. In order to find a smallest possible region, e_{31} has to be picked. After defining e_{next} for each edge, the faces are identified. This is shown in Fig.4.3(d). On traversing in the clockwise direction, two faces F1 and F2 are found. This face information is updated in the edge representation.

The apartments neighbor or the connected faces are identified next. In Fig.4.4(d) the face F1 is defined by the edges $\{e_{11}, e_{21}, e_{31}, e_{41}\}$. Similarly the face F2 is defined by the edges $\{e_{32}, e_{52}, e_{62}\}$. Now we can observe in the figure that the edge e_{31} belongs to F1 and its twin edge e_{32} belongs to F2. So we can say that F1 and F2 are neighbors (connected). This information is updated in the apartment data representation. In a similar manner we do the operations on the floor plan graphs. The input graph of the floor plan and the ownership boundaries identified are shown in Fig.4.5.



(a) Input graph. (b) Boundaries identified.

Figure 4.5: *Ownership boundary identification.*

4.2 Associating ownership rights

After, the apartments' boundaries are identified, the next step is to assign the ownership rights to their apartments. The ownership boundaries are represented as polygons. To associate the extracted ownership rights to their respective ownership boundary, a *point in polygon test* is used . Based on this test, appropriate rights are associated to each ownership boundary (apartment) as shown in Fig.4.6.

4.3 Identifying the floor numbers

So far we have discussed the method for extracting information from a single floor plan. However, buildings may have multiple floors. Hence there will be floor plans for each floor. These

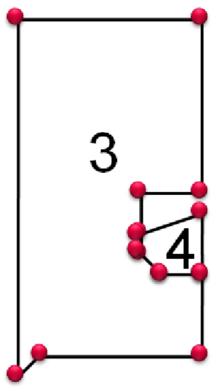


Figure 4.6: *Associating ownership rights*

floor plans are stored in single scanned image as shown in Fig.4.7. The figure shows a building which has four floors and the corresponding floor plans are shown from left to right.

The processing is done on each floor plan is as discussed before, resulting in a group of iso-

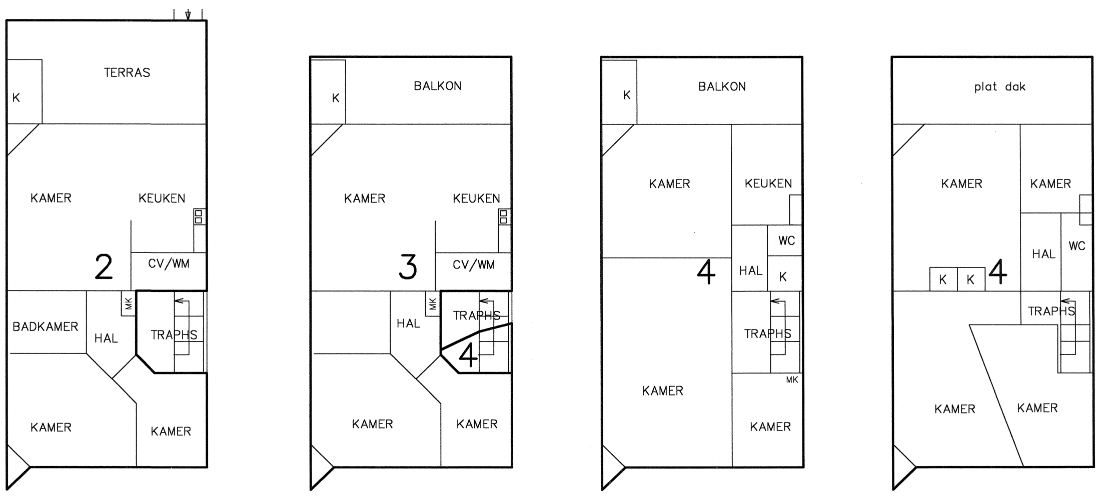


Figure 4.7: *Floor plans of a building : Having four floors.*

lated graphs as shown in Fig.4.8. The figure shows the four isolated graphs that represent each floor, each having all the necessary information associated with them. But the information regarding the floor numbers is not yet recognized. This is needed to stack the floors on top of each other. Hence in this step the floor numbers of the ownership boundaries are identified and associated to the apartments.

The information regarding the faces that are connected to each other is already identified. Using this information all the ownership boundaries belonging to the same floor are identified and grouped together.

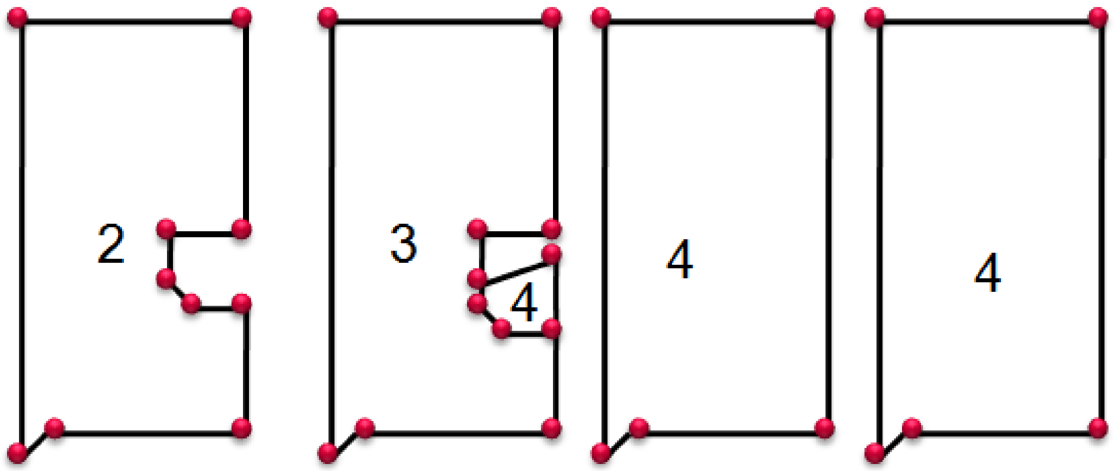


Figure 4.8: *Graph representation: Each isolated graph represents a floor.*

The floor plans of all floors are drawn from left to right which means the left most floor plan represents the lower most floor and the rightmost floor plan represents the top most floor as shown in Fig.4.7. This property is used to identify the floor numbers, as x-coordinates of the first floor will be smaller than those of the second floor and so on. Hence the smallest x-coordinates in the groups are identified and sorted in ascending order to find the floor numbers. The order in the sorted list represents corresponding floor numbers and these floor numbers are assigned to each group and then to each apartment. This information is updated in the apartment data structure. Thus all the semantics such as ownership rights, floor number, and coordinates of the boundary are assigned to the apartments. The floor number of each ownership boundary is shown in the Fig.4.9.

This method works fine only if each floor is drawn from left to right in the scanned floor plan to indicate ground, first floor and so on. In other cases our method fails to provide correct floor numbers. One possible solution to solve this is to use OCR on the text labels describing the floors for identifying the floor numbers.

4.4 Representing data in XML format

After applying all the image processing techniques, optical character recognition and processing the graph representation of the objects in the image, all the relevant information in the image, like apartment boundaries, ownership rights, the floor number etc are extracted. A suitable format to store this information needs to be identified so that the computer can process the relevant information. The final goal is to represent the extracted data in extended CityGML format (D'Silva 2009). Hence a format from which it could be easy to convert into CityGML format is considered. CityGML is an application domain extension (ADE) of the XML data model, specifically designed for city models. This motivates to represent the data in XML format. As XML is the open data model, which allows the user to define own tags and XML is very flexible, so it may be used for writing any data and exchanging information. The XML schema used for our application is shown in Fig.4.10.

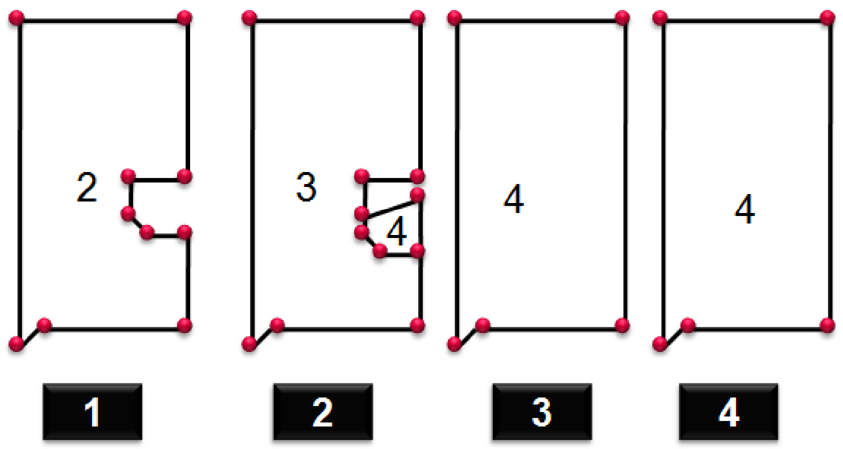


Figure 4.9: *Floor identification: Floor number associated with each ownership right.*

Our requirement is to group the information regarding buildings, this can be done using XML as it supports nesting of tags. The root tag of our schema is <KadasterOwnership>, just to give insight about the information in the document. All the information regarding apartment and apartment rights will be enclosed with in this tag. We have many buildings, hence we use <Building> and </Building> tag to indicate information regarding each building. In the figure we have multiple <Building> tags nested inside the root tag to indicate multiple buildings.

Each building will have many apartments, each apartment information is enclosed between <Apartment> and </Apartment> tags. There are multiple <Apartment> tags nested inside <Building> to store information regarding multiple apartments in a building, as shown in the figure.

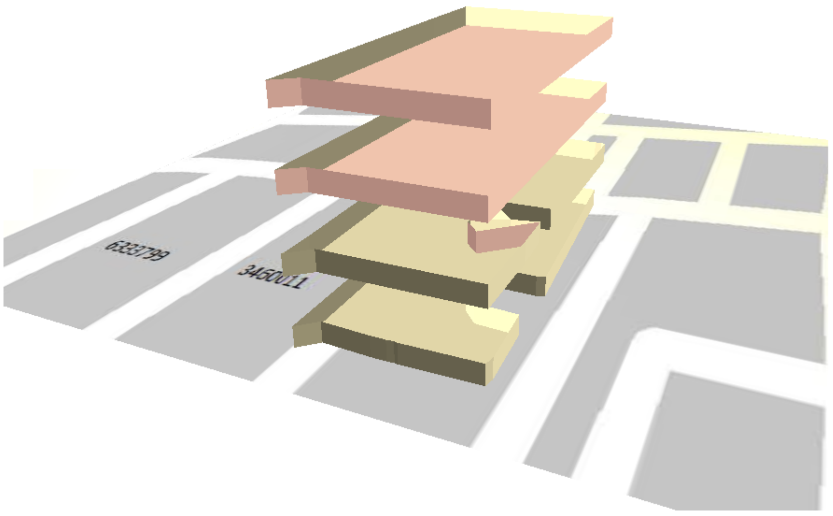
Each apartment has many properties like floor number, ownership rights, boundary etc, enclosed between <Apartment> tags . Extracted apartment rights are specified between <ApartmentRight> and </ApartmentRight> tags. Similarly the floor number of the apartment is stored between <FloorNumber> and </FloorNumber> tags. The apartment rights and floor number are indicated by numbers as in the figure. The boundary of the apartment is specified between the tags <ApartmentBoundary> and </ApartmentBoundary>. The boundaries are represented as polygon, hence <Polygon> and </Polygon> tags are used to represent it. The polygons are specified by their corners, hence all the corner coordinates are enclosed inside <CornerList> and </CornerList> tags. The x, y, z coordinates of each corner is specified as shown in the figure. This XML format can be easily mapped into CityGML format. A visualization of the CityGML representation of the extracted data is shown in Fig.4.11.

```
<?xml version="1.0" encoding="UTF-8"?>
    <KadasterOwnership>
        <Building>
            <Apartment>
                <ApartmentRight>
                    3
                </ApartmentRight>
                <FloorNumber>
                    4
                </FloorNumber>
                <ApartmentBoundary>
                    <Polygon>
                        <CornerList>
                            x1    y1    z1
                            . . .
                        </CornerList>
                    </Polygon>
                </ApartmentBoundary>
            </Apartment>
            <Apartment>
                ....
            <Apartment>
        </Building>

        <Building>
            ..........
        </Building>
    </KadasterOwnership>
```

(a) XML schema used to represent the information extracted from the building floor plans.

Figure 4.10: *Output data representation.*



(a) CityGML representation of extracted data from the image.

Figure 4.11: *CityGML data representation.*

Chapter 5

Results and Discussion

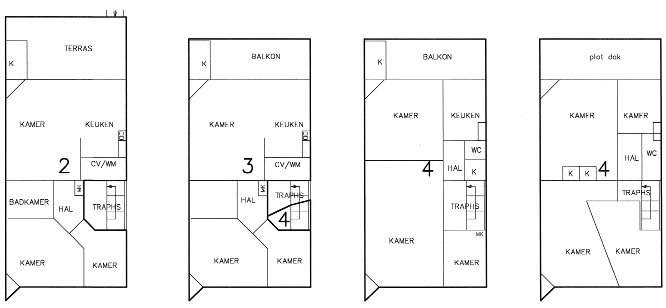
This chapter discusses some of the input files used to evaluate the performance of our approach. The input files are floor plans of apartment buildings, provided by the Kadaster. We use LandXplorer, a CityGML viewer to display the output files. Each scenario shows how our approach handles the various input files.

5.1 Scenario 1: Four storied building floor plan

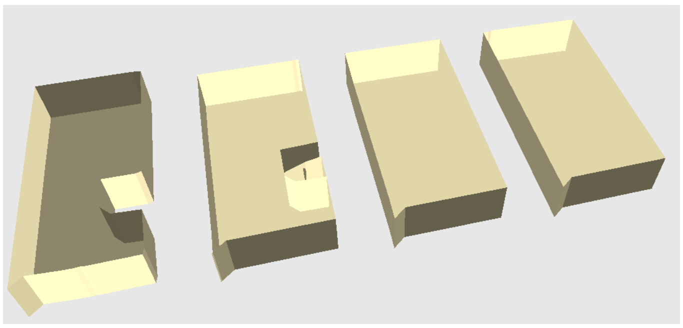
The input file is shown in Fig.5.1(a) and concerns a building in Amsterdam. The image dimensions are 5488 $\times$ 2366 pixels. The building has four floors and its floor plans are shown in the figure. From left to right, these are ground floor, first floor and so on. There are no exceptions as described in section 3.3.1 in these floor plans. There is only some text overlapping with the thin lines. However, both text and thin lines are irrelevant for our application and do not impose any problems in our approach. The output file generated is shown in Fig.5.1(b) which shows the extracted apartment boundary of the input file in CityGML. The ground floor has only single ownership right i.e. ″2″. It has rights for the entire ground floor except for the common area. In the output file, it is clearly visible that except the common area the whole floor is owned by ″2″. The first floor has two ownership right ″3″ and ″4″. Their boundaries are extracted and shown in the output file where we can see one large division owned by ″3″ and a small region owned by ″4″. The second and the third floor have only a single ownership right each, which can be seen in the output file. The entire extraction process and representation in CityGML takes approximately two minutes.

5.2 Scenario 2: Numbers overlapping with lines in floor plans

The input file is shown in Fig.5.2(a) and concerns a building in Eindhoven. The image dimensions are 3228 $\times$ 1639 pixels. The input file contains a multi storied floor plan. The floor plans of three floors are shown in the input file. In the input file, we can observe the numbers representing the ownership rights overlapping with the thin lines. This scenario shows how our approach handles the numbers overlapping with the thin lines. These thin lines will be removed by the opening operation in our application, since they are not relevant. This results in the numbers being isolated. These numbers are sent for OCR to identify them. Fig.5.2(b) shows the information extracted from the input file and displayed in CityGML format. The entire process takes approximately 1 minute and 30 seconds. The numbers overlapping with thin lines is a rare scenario, but this scenario can be handled using our approach. Our approach fails if the numbers are overlapping with thick lines and the solution for this is described in section 3.3.1.



(a) The image showing floor plans of four floors in a building.



(b) The image shows the extracted information represented in CityGML: Here floors are not stacked on top of the other and just shows the geometry of the apartments.

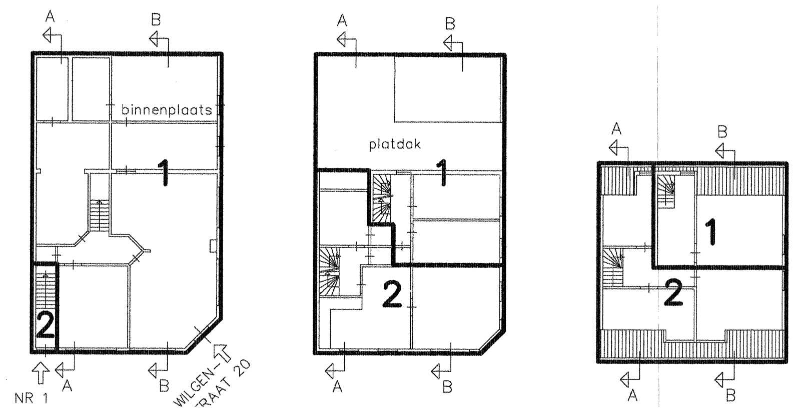
Figure 5.1: *Scenario 1: Our method works exactly as expected*

5.3 Scenario 3 : Ownership rights represented using text

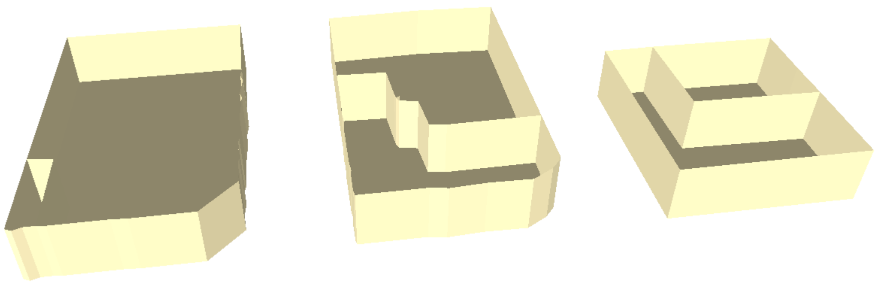
Fig.5.3(a) shows the floor plan of the ground floor of a building. The image dimensions are 2832×3992 pixels. This scenario contradicts one of our assumptions which states that each ownership boundary must have a number that indicates ownership rights associated to it. In Fig.5.3(a), we can observe an area without any ownership right and is shown in grey color. Instead, the usage type of the region is given on that area which says ”gemeenschappelijke gang” to indicate it to be a common area. Our method will extract only the ownership boundary but will not be able to associate the ownership rights. The output file generated for this scenario is shown in Fig.5.3(b). The approximate time it takes to process this image is one minute.

5.4 Scenario 4: Handling curved apartment boundaries

Fig.5.4(a) shows the floor plan of the ground floor of a building. The image dimensions are 3048× 4616 pixels. This scenario shows how curved lines in the floor plans are handled. The grey colored region has a curved line to indicate its boundary but these curved lines will be represented by a series of straight lines in our approach. This results in an approximate representation of the curved lines. The output file is shown in Fig.5.4(b) where the grey boundary shows how the curved lines are extracted and represented in CityGML. The approximate time it takes to process this image is one minute 30 seconds.



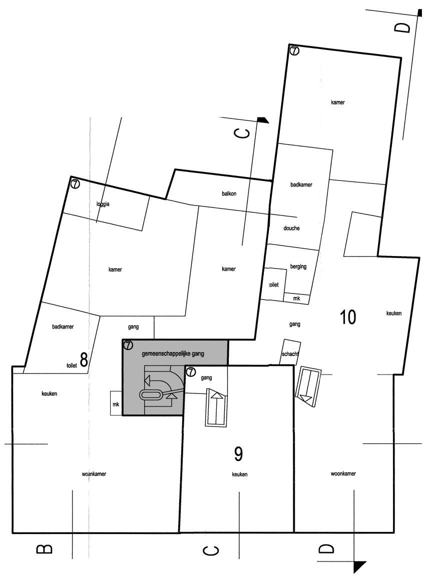
(a) The input file showing floor plans of three storied building.



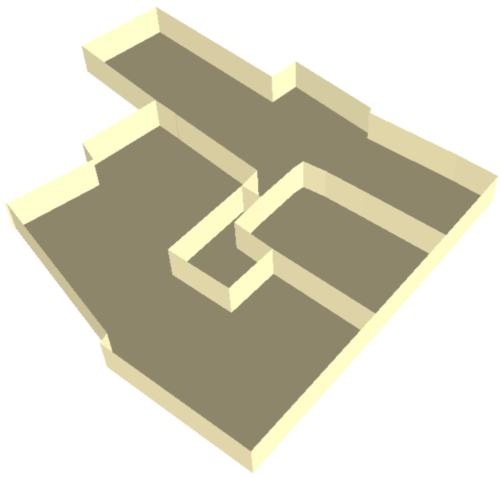
(b) Handling ownership rights overlapping with thin lines.

Figure 5.2: *Scenario 2: Our method handles some of the exception.*

These scenarios show our approach works fine with all the input files having no exceptions. It also handles some of the exceptions such as in scenario 2, 3 and 4. The errors occur only regarding ownership rights. The errors such as not finding the ownership rights are caused only when the numbers overlap with the apartment boundary they belong to. The method for finding ownership rights presumes them being isolated and inside the apartment boundary. It will fail to find them if they are not. The solution for this is described in section 3.3.1, however not yet implemented.

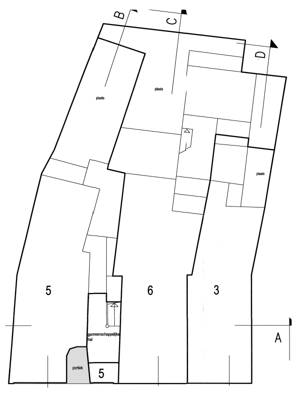


(a) The ground floor plan of a building having no ownership rights associated with an apartment, shown in grey color.

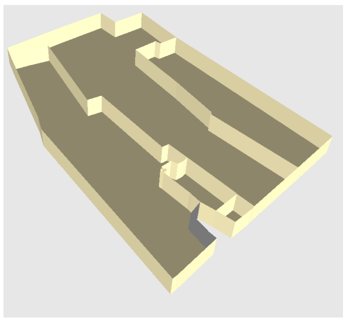


(b) CityGML representation of the floor plan, the ownership rights are not associated with the region corresponding to grey color in the input image.

Figure 5.3: *Scenario 3: How our method handles the exception of having multiple ownership rights inside an apartment.*



(a) The ground floor plan of a building having curved apartment boundaries.



(b) CityGML representation of the floor plan, handling curved apartment boundaries.

Figure 5.4: *Scenario 4: Handling curved apartment boundaries.*

Chapter 6

Summary and Future work

6.1 Summary

A method for automatically extracting the relevant information from scanned building floor plans has been presented by proposing a process pipeline. This process pipeline takes scanned floor plans of an apartment building as an input and outputs the data extracted in XML format. Each image is initially preprocessed to convert the image into a binary image and to remove the noise in the image. The resulting image is sent for a data reduction stage. In this stage, our method extracts numbers representing ownership rights and removes the irrelevant information from the image, such as thin lines and texts. Later the image is sent to a graph construction phase, where we extract the skeleton of the objects in the image and convert it into a graph. This is done by detecting the corners in the skeleton and using them as nodes. The edges of the graph are formed from the lines connecting the corners. The graphs obtained from the previous stage are processed to identify the ownership boundaries, to associate the ownership rights and the floor number. The extracted information is represented in XML format, which can in turn be converted into CityGML format.
The method as described is an inexpensive and computationally efficient method that can be employed to automatic recognition of building floor plans and to store it in a computer processable format.

6.2 Future work

In our approach we assume that the numbers representing ownership rights do not overlap with the boundary lines. But in rare cases, numbers do overlap with the lines. This is yet to be handled.

We assume that the floor plans are drawn from left to right, with the left most floor being the ground floor, and so on. Based on this assumption we find the floor numbers. But this may not be true in all the cases. Hence we need to have a completely new approach to handle this problem. One of the solutions could be to use the labels that describe the floor numbers

We do not extract information on the height of the floor. Instead, we assume a standard height for each floor. We stack the floors one top of each other without considering the actual geometry of the building, but we need to consider the geometry to have areal representation of the building. All these aspects need to be considered and solved in future work.

Bibliography

Boatto, L., Consorti, V., Del Buono, M., Di Zenzo, S., Eramo, V., Esposito, A., Melcarne, F., Meucci, M., Morelli, A., Mosciatti, M., Scarci, S. and Tucci, M.: 1995, An interpretation system for land register maps, *Document image analysis*, IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 474–482.

Chen, L.-H., Liao, H.-Y., Wang, J.-Y., Fan, K.-C., Hsieh and C-C.: 1996, An interpretation system for cadastral maps, *ICPR '96: Proceedings of the International Conference on Pattern Recognition (ICPR '96) Volume III-Volume 7276*, IEEE Computer Society, Washington, DC, USA, p. 711.

Cofer, R. and Tou, J.: 1972, Automated map reading and analysis by computers, *FJCC '72: Fall Joint Computer Conference*, Vol. 41, pp. 135–145.

de Berg, M., Cheong, O., van Kreveld, M. and Overmars, M.: 2000, *Computational Geometry: Algorithms and Applications*, second edn, Springer, Berlin.

den Hartog, J., ten Kate, T. and Gerbrands, J.: 1996, Knowledge-based interpretation of utility maps, *CVIU '96 :Computer Vision and Image Understanding*, Vol. 63, pp. 105–117.

D'Silva, M. G.: 2009, *A feasibility study on CityGML for cadastral purposes*, Master's thesis, Technische Universiteit Eindhoven, Eindhoven, The Netherlands.

Freeman, H. and Pieroni, G.: 1980, *Map Data Processing*, Academic Press.

G. Gröger, T. H. Kolbe, A. C. C. N.: 2008, OpenGIS City Geography Markup Language (CityGML) Encoding Standard, Open Geospatial Consortium Inc.

Gonzalez, R. C. and Woods, R. E.: 2001, *Digital Image Processing*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Harris, C. and Stephens, M.: 1988, A combined corner and edge detector, *Proc. Fourth Alvey Vision Conference*, pp. 147–151.

Janssen, R., Duin, R. and Vossepoel, A.: 1993, Evaluation method for an automatic map interpretation system for cadastral maps, *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*, pp. 125–128.

Lawrence, R., Means, J. and Ripple, W.: 1996, An automated-method for digitizing color thematic maps, *PhEngRS '96: Photogrammetric Engineering and Remote Sensing*, Vol. 62, pp. 1245–1248.

Musavi, M. T., Shirvaiker, M. V., Ramanathan, E. and Nekovei, A. R.: 1988, A vision based method to automate map processing, *Pattern Recogn.*, Vol. 21, Elsevier Science Inc., New York, NY, USA, pp. 319–326.

Nakajima, M.: 1984, A graphical structure extracting method from an urban map using parallel vector tracers, *IECE*, Vol. J67-D, pp. 1419–1426.

Ogier, J., Mullot, R., Labiche, J. and Lecourtier, Y.: 1998, Multilevel approach and distributed consistency for technical map interpretation: Application to cadastral maps, *CVIU :Computer Vision and Image Understanding*, Vol. 70, pp. 438–451.

Reiners, D., T. A.: 2009, *Skeleton-based Hierarchical Shape Segmentation*, Eindhoven, The Netherlands.

Smith, R.: 2007, An Overview of the Tesseract OCR Engine, *ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition*, IEEE Computer Society, Washington, DC, USA, pp. 629–633.

Stefano, L. and Bulgarelli, A.: 1999, A simple and efficient connected components labeling algorithm, *10th International Conference on Image Analysis and Processing (ICIAP'99)*, Vol. 41, p. 322.

Stoter, J. E.: 2004, *3D Cadastre*, PhD thesis, TU Delft, Delft, The Netherlands.