TDK-dolgozat

Mátyás Pitlik

Contour-based building recognition and change detection from LiDAR data

EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

DEPARTMENT OF SOFTWARE TECHNOLOGY AND METHODOLOGY

Author:

Mátyás Pitlik

Computer Science MSc

V. grade

Supervisor:

Máté Cserép

Assistant Lecturer

Budapest, 2019

Contents

Chapter 1

Introduction

With the continuous improvement in remote sensing technology and IT possibilities, nowadays the available solutions aiming at the observation of our environment play a greater role in our everyday life than ever.

LiDAR (*Light Detection And Ranging*) is an emerging remote sensing technology which uses active sensors based on laser pulses to generate a three-dimensional model of the surveyed area. It is widely used on several scientific fields from terrain modeling, to indoor mapping, to meteorology and recently for autonomous driving purposes. Moreover, its airborne variant makes it possible to cost-efficiently scan larger surface areas with high resolution even at regular intervals, which provides an alternative data source for automated analyses about the surface and its changes. The results can be directly applied in various fields related to GIS challenges, e.g. in case of land use inspections, urban planning, civil engineering, monitoring of environmental disasters (extent of flood and waterlogging, earthquake damages), afforestation and maintaining GIS databases. However, the data volume acquired from the laser scans requires efficient processing or even the support of distributed computations.

The goal of this research was to create a novel, robust approach for contour-based building recognition and change detection from airborne LiDAR data. The proposed algorithm favorable computer resource (especially memory and storage space) requirements compared to other studies through reducing the number of required input data sets. To create a general solution which not only recognizes ideal, closed contours, but which is also capable of working with different contour fragments, a series of in-depth contour analysis and transformation steps should be integrated into the method preparing a reliable and accurate contour classification phase. When comparing epochs of multitemporal data sets to detect changes regarding the buildings of the scene, a potential and promising approach could be built upon the possible contour pairings of the identified buildings with special emphasis on the possibility to reconstruct lack or loss of information from the previous steps.

In this study, a method fitting these goals and expectations will be presented in detail. The technological background and an assessment for the results of related articles will be summarized by Chapter 2. After that, I will present in Chapter 3 the contour-based building recognition method and the change detection algorithm. Chapter 4 provides some details of the implementation focusing on the operation-based design of the solution. Finally, Chapter 5 shows the performance and validation results for the method followed by the discussion of notable challenges regarding the research methodology.

Chapter 2

Background of the study

2.1 General information

2.1.1 Areas of application

Change detection is the process of comparing multitemporal data and drawing conclusions from them. This can be performed in various ways, but fully automated methods have the greatest value and potential. The data to be examined could be scans of the land surface acquired by remote sensing techniques. In this case, several areas of application can benefit from the results, e.g. supervising permits for land use, monitoring changes in natural disaster management (flood level, earthquake damage, etc.), biomass estimation and afforestation monitoring in forestry and maintaining GIS databases of urban areas.

2.1.2 LiDAR technology

The methods for change detection have evolved much in the last decades. This is related to the improvements of the LiDAR (Light Detection And Ranging) systems. LiDAR is a remote sensing technique, which uses laser pulses to measure distances to a target being surveyed. This distance is determined based on the elapsed time between the emission of the original pulse and the perception of the reflected one. The technology even allows detecting multiple returns of the same pulse, which most frequently occurs when surveying areas of vegetation as it is shown in Figure 2.1. Laser beams have a shape of a cone (not a line), therefore multiple return in case of vegetation also occurs because the cone can hit the tree (and its leaves) at different heights. Based on the position of the scanner and the angle corresponding to each emitted pulse, a three-dimensional point cloud can be constructed, which represents the target surface.

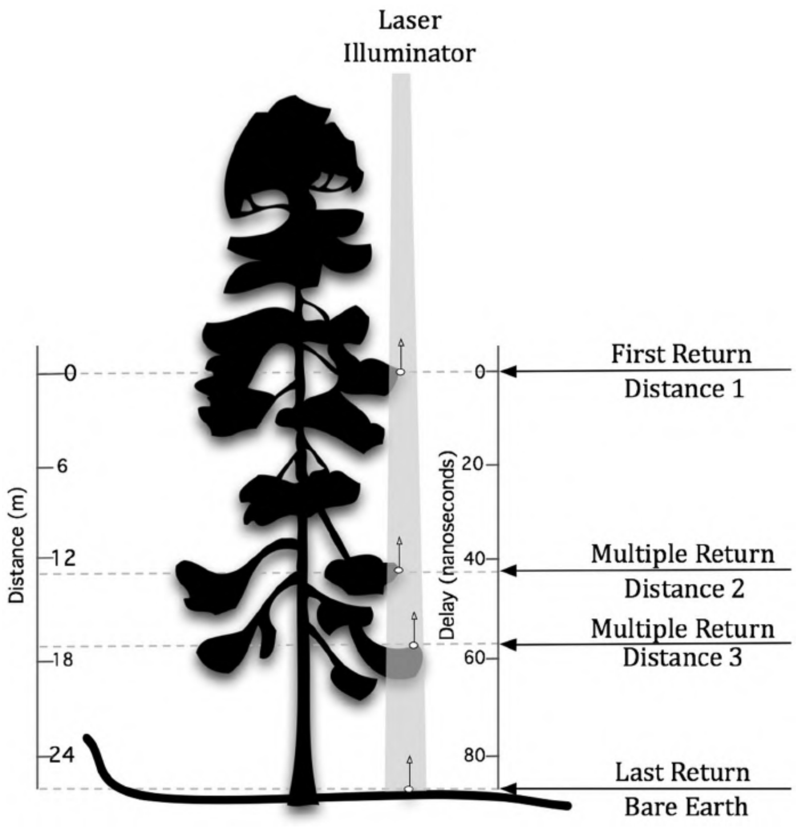

Figure 2.1: Demonstration of multiple return in LiDAR technology. (Source: [1])

The main advantages associated with LiDAR technology are the high spatial accuracy, the fast delivery of data (so the cost-effectiveness) and the quality being less dependent on optically unfavorable weather conditions. These characteristics make LiDAR system a more suitable choice for data acquisition than using the traditional satellite and aerial images. It is also important to mention, that even if the scanner position is fixed, repeated scans will not result in the same point cloud, there are no guarantees for fixed points. Considering different standpoints and hence the possible occlusion effects as well as different incidence angles, it is even more difficult to deal with the false changes coming from the varying point density and quality.

2.2 Data acquisition

Considering the platform equipped with a LiDAR sensor, terrestrial, mobile and airborne types can be distinguished. The application types require different specifications for the device.

2.2.1 Terrestrial scanning

Terrestrial LiDAR means a stationary device being able to create a 3D model of the scene from the fixed location of the scanner. It is often used in conventional topography application, in geomorphology and for deformation analysis.

2.2.2 Mobile scanning

In this case, the scanners are attached to a moving vehicle. Autonomous driving is an emerging use case for this technology. It scans the surroundings (facades, cars, power lines, etc.) along the way, but only from one direction, therefore the entire 3D model of the object will not be available.

2.2.3 Airborne LiDAR

A LiDAR system can be located on an aircraft (e.g. drone, helicopter, plane) as well, depending on the expectations for the flight altitude, the extent of the area covered and point density of the data acquired.

It has to be acknowledged that aerial LiDAR scans in cities with skyscrapers are suffering from shadowing effects. However, the point cloud will contain the building roof points even though the walls are occluded, therefore identifying changes in the scene should be possible. The alternatives have their limitations too (multiple scan registration, high viewing angles, etc.) so airborne LiDAR seems to be the best option in urban environments for building recognition.

2.3 Data formats

2.3.1 Point cloud

Considering the operating principles of LiDAR it is natural to think of the data as a set of spatial points (eventually with additional information, e.g. intensity) corresponding to the emitted laser pulses. There are file formats following this logic, which are supported by the common geographic information system softwares. Evidently, LAS files have an exact specification [2], containing various header information beside the point data. The compressed version of it is called LAZ. Either way, the file size can easily reach gigabyte scale [3] which makes it difficult to keep all data in the memory of an average PC for processing. Besides, the points in the file do not have any topology or connectivity information, only point-wise data (e.g. coordinates, intensity), so identifying neighboring points poses a challenge too.

2.3.2 Grid

If the individual points are interpolated on a regular grid the data is in DEM (Digital Elevation Model) format. This can be treated as a raster image, where the grid values denote height information. DEM is the general name, but depending on the real world scene represented by the data there are other notions. A DSM (Digital Surface Model) means the actual surface, including the terrain and every other object on it, as shown in Figure 2.2. However, a DTM (Digital Terrain Model) only contains the ground points, which either leads to "holes" (so called *nodata* points) in the grid or uses interpolated values. Finally, a nDSM (normalized DSM) is simply the difference between the previous elevation models, which can be interpreted as above-surface objects.

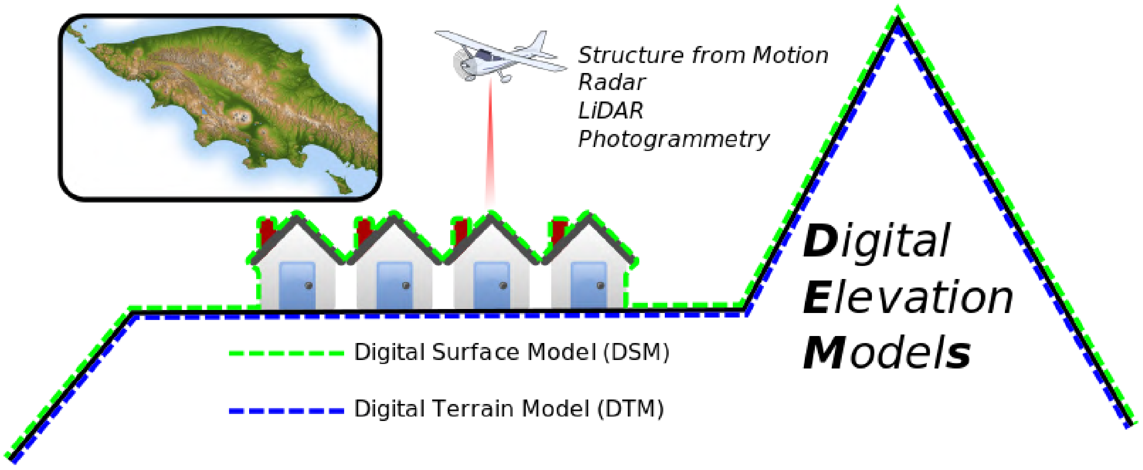

Figure 2.2: The difference between DSM and DTM.
(Source: Wikimedia Commons, `https://commons.wikimedia.org`)

2.4 Data set

As the data set for this study, the *Current Dutch Elevation* (*Actueel Hoogtebestand Nederland*, AHN) was chosen, which contains detailed altitude data for the whole Netherlands both in point cloud and grid format. The project is in its third iteration (AHN-3 [4]) since 2014 which is expected to finish in 2019. The previous acquisitions were performed between 2007 and 2012 (AHN-2 [5]) as well as between 1996 and 2003 (AHN-1).

When working with any kind of data, one should be aware of the reliability of the input and computed values. The accuracy of a point cloud means the difference between the measured coordinates of a point and its actual position. These are present in the data set both in the form of individual measurement (random) errors and systematic errors (the latter caused by e.g. the merge process of the scan swaths into a common coordinate system). After comparing point clouds, the uncertain changes fall into the range of the sum of the separate accuracy error values.

The systematic error is guaranteed to be no more than 5cm for all of the three time intervals. The random error is at most 15cm (except for the AHN-1 which has 45cm) and it is even less in the DEM format because of the interpolation [6]. The point density increased from 0.06-1 point / m^2 to 6-10 point / m^2, respectively. Both the point accuracy and density suggest that the AHN-2 and AHN-3 data should be compared to be able to identify changes in building scale.

2.5 Related work

When performing the change detection task, it is reasonable to incorporate a building recognition step in the process. This decision can be confirmed by multiple aspects.

- Buildings have mostly geometrical shapes, therefore the possible changes can be constrained (e.g. new floors and wings can be built, but a building hardly ever shows pointwise changes).
- These systematic changes mean that all sampled points from a building (part) should change in the same way, and this kind of redundancy in the data enables to eliminate individual erroneous differences.

- The building recognition procedure can also improve the overall performance because only a subset of the data has to be analyzed at a time.

The application of the building recognition step can be done either before or after producing the change set of the two scans. The chosen approach can depend on the nature and variety of the original data, the methods planned for the study or expectations regarding performance.

After reading the related studies described later, the following distinction seemed appropriate. First, I will present multiple existing methods for building recognition in 2.5.1. Most of them use DEM-like data as their input, but there are point cloud based solutions too. After that, publications related to change detection will be shown in 2.5.2.

The referred articles will be discussed in a way that evaluates each of them based on its potential to be able to contribute to this study after being adapted to our specific use case if necessary.

2.5.1 Building recognition

DEM-based approach

Researchers quickly recognized the possibilities arising from the use of LiDAR. Already in 1995 an article was published aiming at automatic building recognition [7]. The authors even took one step further, namely using different models for building reconstruction based on geometric constraints and domain knowledge (but this capability does not really provide additional value for this study).

The approach starts with a preprocessing phase to generate the DEM from the measured 3D points of the surface. As the AHN data are already available both in LAZ and in DSM format, there is no need for such a procedure in our method. Moreover, an approximation of the surface without buildings and other above-ground objects can be avoided as well because DTM data are also available.

Starting from this point, the difference data set of the DTM and DSM is computed, which consists of all the above-ground elements. After that, the differences are filtered with a threshold value adjusted to the expected building dimensions. In order to manage buildings individually a segmentation step is carried out, which results in connected components. Based on an experimental value the small segments get excluded. Furthermore, segment with a bounding box not completely contained in the data set are rejected too. In order to determine the buildings more accurately, the former global thresholding is refined by deriving a local threshold value based on the height information in the bounding box of each segment.

The presented method only uses size and height information when marking a segment as valid, consequently, false changes can be found where trees are located too close to a building (or even partially cover its roof). As a possible solution for this problem, intensity and color information could be taken into account.

Weidner addressed this issue in a later publication as well [8] by proposing a method which still relies solely on height information. The vegetation areas are associated with differential geometric quantities, e.g. the variance of surface normals derived from the neighboring height data from the DSM. After a refinement process, the vegetation areas are also given in the form of segments, which are excluded from the existing segments of the buildings. The paper mentions other differential geometric properties (roughness, gradients and curvatures), but their role and usefulness were not detailed enough in the study, so this topic should be further investigated for appropriate evaluation.

The basic idea to work with the difference data set of the DSM and DTM (with the so-called normalized DSM, nDSM) appears in [9] as well. Although the generation of an approximated DTM is a separate step in the study, the possibility of inaccurate surface feature detection is worth considering in every other application too. In this particular case, the cause of this phenomenon is the distortion of the computed reference surface near object boundaries, which prevents to recognize the exact outlines. To mitigate this effect, on the one hand, a buffer zone was added with an extent adjusted to the grid resolution, on the other hand, areas in the original DSM with high slope value (indicating tall buildings where the distortion effect is the strongest) were marked.

The use case of the study (flood inundation) did not require individual surface objects to be identified, but the authors suggested an artificial intelligence related technique possibly fitting for the purpose. The combination of topographic characteristics along with the available spatial information should be used as input in a neural network to distinguish between building and vegetation, which already proved to be a challenge in the above-mentioned publications.

Another method using elevation data was presented in [10]. The core concept is to use the information from LiDAR data to generate an elevation image for further processing. First, the edges on the image are determined, which mostly represent surface objects, e.g. building and tree outlines.

During edge analysis, quantitative measures are computed to aid the edge classification procedure. Among them, there are general attributes (size, height, closure), geometric properties (orthogonality, parallelism) and shape related information (symmetry, circularity). For all of these values, there are expectations which should separate the buildings from vegetation areas. Closed edges are a requirement for shape examination and this does not contradict the domain knowledge about buildings either. Circularity is used to express the simplicity being present in most building contours. Symmetries based on parallel and orthogonal sections are also typical for buildings.

Experiences showed that circularity is effective in filtering out non-building contours from candidates. However, the geometric properties proved to be decisive at the end, so it must be pointed out that it poses a limitation to the method. Neither buildings with a more complex shape, nor those melted together with trees can be recognized easily this way. The prior identification and exclusion of vegetation areas based on multiple returns in LiDAR data could be helpful though.

It is possible to use an entirely different toolset for the LiDAR data classification, e.g. statistics based on geomorphometric parameters [11]. At the beginning of the procedure, a segmentation process is performed. The seed cells for this are determined by computing the difference between the original LiDAR DEM and the median filtered one. The median filtering is capable of removing very high and very low elevation values compared to its neighbors as long as the window size used for this operation is small enough to prevent the effects originating from the local terrain changes.

The authors found that a specific range in the positive differences seemed to contain mostly points close to building outlines. The iterative segmentation algorithm uses experimental intervals of slope and aspect values coming from local elevation differences as the region growing criteria. After that, the segmented building points and the remaining background points are divided into connected components. These objects serve as input for the upcoming statistical analysis.

Each object is represented by a set of parameters (namely mean elevation, standard deviation of elevation, mean slope and standard deviation of slope) to carry out an unsupervised classification. The K-means clustering algorithm identified ten clusters and based on their centroids all objects were assigned to a cluster. The given clusters can be examined to refine the results. Compactness is a measure which can imply possible subclasses. Variance analysis (e.g. ANOVA) can be used to assess the cluster differences and so to group them together.

The main disadvantage of the method is the required experimental user interaction at multiple points of the process. The key parameters of the described algorithms were set manually and the automation might not be possible for general use. Also, the interpretation of the clusters was a manual step. Visualization indicates best the correlation between clusters and real surface classes, which later can be confirmed by reasoning about the meaning of the geomorphometric parameters. However, the method has the potential to identify building subclasses in the form of different clusters and thus giving a more detailed classification.

Point cloud based approach

From a computational aspect, processing the LiDAR data in a DEM format is the most effective. Using any other source of information requires more resources, but it can potentially increase the quality of the results.

Ma [12] used the original LiDAR points in a gridded format for local analysis of the surface, to prevent error-effects and to achieve higher accuracy. Although this method also generates its own DEM, some of the initial steps are used later again to recognize the buildings in the area.

For each LiDAR point, a planar regression surface is derived based on the neighboring points located in a small window of the grid. Then the distance is measured related to the examined points and the derived surface. Either the root mean square is computed compared to the regression surface, or the height difference is calculated between the actual point and its projection to the surface, but both cases require a suitable threshold value aligning with the vertical accuracy of the data. This leads to a differentiation between points falling onto planar surfaces (including ground, building roofs and even cars) and other points of e.g. trees.

After identifying the connected regions, the assumption that the ground segment is bigger than the largest building in the area is used to select the large regions which will serve as the basis of the DTM. In this method, the holes marking building segments were interpolated (unlike in the AHN data set). Using a triangulated irregular network (TIN), artificial features are introduced into the generated DTM. The building class consists of the intersection of a height filtered nDSM and the planar points mentioned before. To refine the results in the visualization, false positive small features can be discarded, and with a morphology dilation operator building boundary points can be recovered.

The author mentions a few limitations regarding the method. First, only buildings with a planar roof can be detected which is too restrictive. Additionally, smaller buildings being partially covered by trees are easily missed.

Hybrid approach

Another study uses LiDAR data together with aerial images [13]. The motivation is to use complementary sources to achieve high resolution along all dimensions. Aerial images usually have higher resolution, while laser scanning provides accurate elevation information.

The LiDAR DEM is initially used to get a vague approximation of the building areas by applying a height filter. But the main information for further processing comes from the areal image edges recognized by the Canny detector, which are stored as vectors to speed up the procedure.

The method relies on perceptual grouping to organize the individual edge segments into higher level features. In this case, the goal is to determine the outer contours of the buildings. The method supports both straight lines and arcs to be able to describe more general shapes. The grouping is based on geometric properties, which are proximity, similarity, colinearity and overlap. A closure algorithm is needed to create a chain of these edge segments which represent the building outline. As the buildings have mostly regular shapes in the real world, a regularization algorithm can be applied as refinement.

The edge detection of the optical image will mix edges by shadows, trees and buildings together, so the main challenge is to separate them correctly.

2.5.2 Change detection

DEM-based approach

As it is mentioned in [14], an important prerequisite for change detection is to extract the common area of the scans so that the corresponding parts can be compared. This study also requires a spatial database of a building inventory, which makes it harder to apply the method in arbitrary situations.

Like often before, the DEM format is preferred to speed up the computations. Then the histogram of the difference grid is analyzed to get the potential constructions and demolitions of buildings. The further a difference value is located from the mean of the histogram, the more significant that change is. The authors found that using the standard deviation of the difference grid as the histogram threshold compared to its mean is sufficient to detect changes of entire buildings. However, smaller differences, e.g. the construction of a new floor cannot be detected as easily. This statistical approach alone cannot differentiate between changes of buildings and other objects, that is why the building database is needed as well.

The study also refers to the possible shadowing effect caused by skyscrapers, which means that there can be areas with no LiDAR points and therefore no conclusion can be drawn for these artifacts. As a solution, density thresholding was used. All polygons were filtered out, where the laser point density was below the average of all marked areas. The result consists of demolished, new and unchanged labels, but to avoid errors, the database should fit the older survey flight.

Another approach uses only the LiDAR data to classify the surface before the change detection procedure [15]. First, normalized DSM is computed to be able to focus on the surface objects. Then a region growing segmentation algorithm is parameterized to detect potential buildings. Segments originate from neighboring points exceed the minimal building height and they can expand based on a maximal slope value. The latter criterion serves for homogeneity to exclude vegetation objects, but other attributes (e.g. pulse intensity, shape, height texture) are also necessary to correctly classify the segments.

The change detection process sequentially categorizes the segments. A preprocessing step removes those segments that are not eligible for further analysis (cf. shadowing effect). If a segment from either scan has no corresponding segment from the other date and the difference is significant as well, then a new or demolished building is detected. For the remaining segments, after applying a morphological opening filter, the difference grid is calculated to derive heightened, decreased and unchanged labels based on the frequencies of positive and negative changes of height values. In special cases, a change can be contradictory, which indicates either a wrong segmentation or partial reconstruction of the building.

The biggest risk of the method influencing the results is the possibility to merge separate buildings into one segment. To avoid that, a segment splitting step could be introduced, where the data from both scans are simultaneously analyzed. Another challenge is when the almost identically rebuilt buildings need to be detected too, but for this task, the LiDAR data only is not necessarily sufficient.

Point cloud based approach

Butkiewicz et al. [16] chose to use raw point clouds in order to achieve the highest accuracy possible. The study points out, that many steps in a DEM-based approach can lead to errors in measurements, such as fitting LiDAR points to a regular grid or applying morphological operators.

To efficiently detect neighborhood relations an irregular triangulation is performed on the points. For each point of the newer scan, it is determined whether it can be a change or not. The point is projected onto the older scan, then it is only considered a change if the point exceeds the allowed height variation based on interpolation from the nearest point of the old scan.

The obtained set of marked point will contain false positive changes at this phase because of e.g. pulses returned from antennas or road lights, therefore noise filtering is highly recommended. In a very simple form, all marked points having no other marked point as a neighbor could be removed. After that, starting from arbitrary marked points, the triangulation can be used to extract the 3D model for individual connected changes. The change models can be filtered as well e.g. based on the area of the 2D projection, the number of the containing marked points or other measures.

The presented method has another interesting aspect. It uses the horizontal and vertical accuracy values of the point cloud throughout the computation process. That way, it is less possible that the detected changes come from some measurement error. However, higher accuracy can only be accomplished by computationally more demanding procedures. Therefore, it is worth considering, that a high resolution DEM can guarantee the expected accuracy for as large objects as buildings.

Chapter 3

Methodology

3.1 Study area

In order to properly showcase the proposed approach, a suitable test area had to be selected. The most important aspect was to find an area where building constructions have been carried out between the two epochs of data acquisitions, therefore the building recognition and change detection capabilities can be demonstrated on the same scene. The AHN data set provides laser scans for the entire Netherlands, ultimately a smaller extent of the town Drachten from the northern part of the country was chosen, as shown in Figure 3.1.

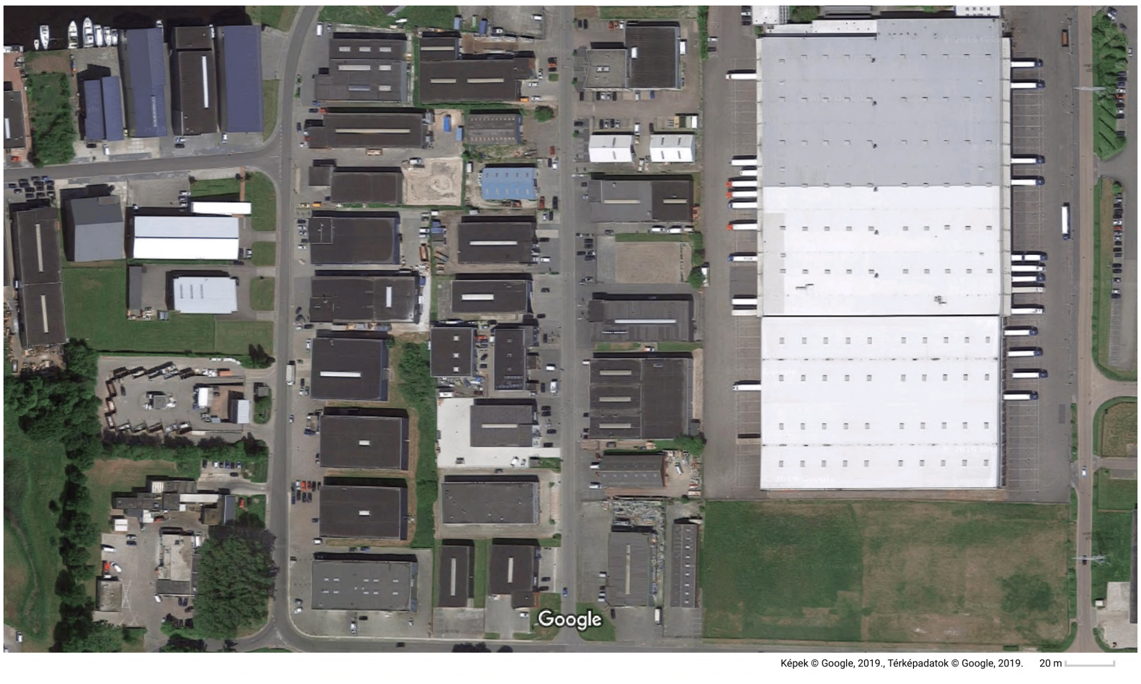

Figure 3.1: Satellite image of the study area. (Source: Google Maps)

The size of the test area is approximately 0.15 km^2. Obviously, this extent with a smaller amount of data alone is not eligible for performance measurements, but it is ideal to demonstrate every step of the methodology in detailed images throughout this chapter. Some additional advantages can be associated with this test area as well. While the scene mostly contains buildings with regular shapes, there are some special cases which point out the challenges and difficulties of this study.

The AHN data set is available both in raw point cloud format and as preprocessed DSM files which are offered both in 5 m and 0.5 m grid size resolution. Among those, the 0.5 m DSM was chosen, so the computational complexity is reduced (compared to raw point clouds), yet a high level of precision can be achieved in building recognition due to the expected surface object dimensions. Figure 3.2 visualizes the chosen DSM data set.

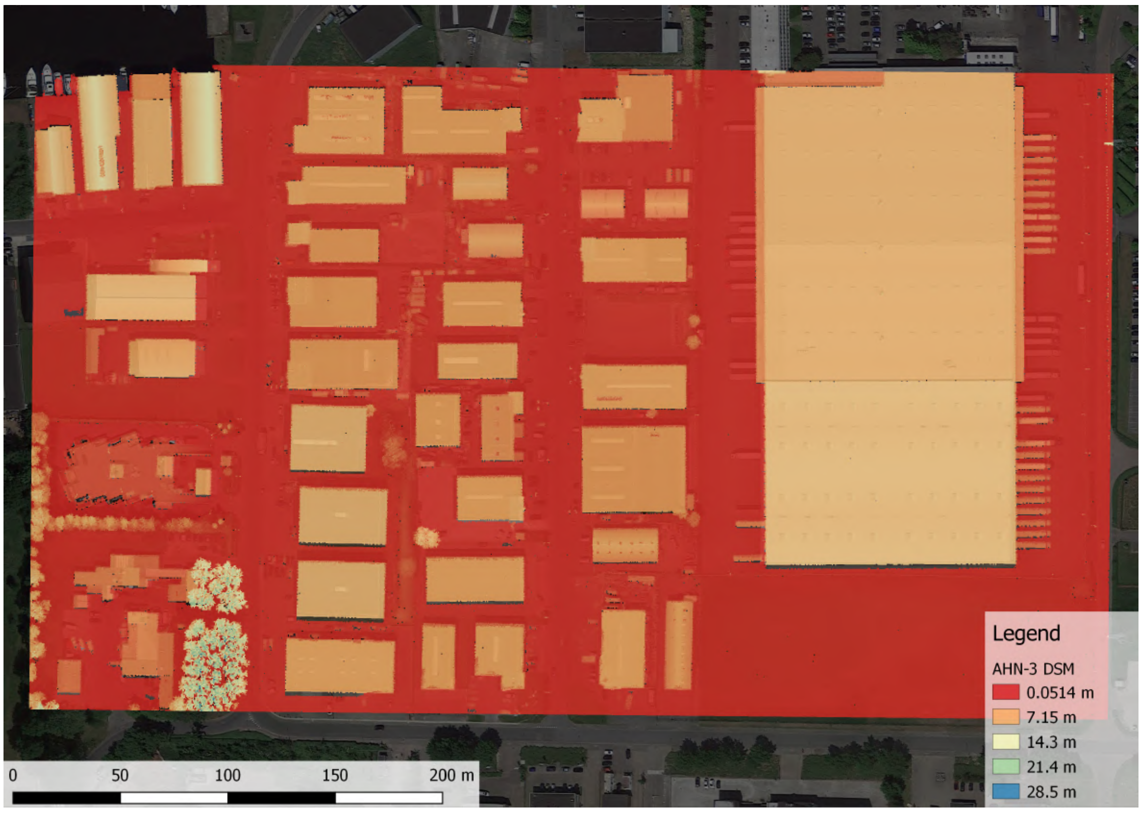

Figure 3.2: Visualization of the AHN-3 DSM data set colored by height.

3.2 Building recognition

The fundamental idea originates from the related studies of Wang and Schenk [10] presented in Section 2.5. Unlike many other approaches, it depended only on the DSM data to detect buildings, which is definitely an advantageous characteristic in terms of required computer resources and makes it worth trying to elaborate a complex approach with change detection capabilities atop of it. However, while the test data used in the before mentioned study resulted in mostly closed contours of surface objects for further processing, this favorable assumption rarely holds in a more general environment[1]. Therefore a more robust contour-based approach had to be designed capable of building edge detection based on partial contour-lines. In the following sections, I will present the methodological background knowledge relevant to this study and the detailed analysis of the steps making up our object recognition and change detection procedure. The proposed solution consist of the following steps:

1. Detect edges and generate contours.
2. Filter out extremely curved contours.
3. Split contours into consecutive straight segments.
4. Remove possible redundant segments.
5. Identify building contours.
6. Compare multitemporal data and classify changes.

3.2.1 Edge detection

Traditionally, edge detection denotes various mathematical methods in digital image processing. In greyscale images, edges can be extracted from abrupt intensity changes in the image pixels. There are different mathematical approaches (e.g. discrete approximations for the derivative of the image function) that enable detecting edges by examining the local environment of each pixel.

Based on the exact mathematical computations taking place when applying an edge detector, different characteristics will describe the resulting edge map. However, some fundamental behavior will hold for the most commonly used convolutional detectors ensuring the desired quality for further processing.

- never indicating edge if there is no change in the image at all;
- minimal number of false positives and false negatives;
- the indicated edge falling close to the physical one;
- edges with arbitrary orientation can be detected;

[1]No sufficing test location could be found within the AHN data sets.

- the physical edge is indicated only once.

Considering the summary of edge detection above, it is clear that the method is not restricted to digital greyscale images only. In our case, the DSM data from the laser scans (or with other words, the height map of the area) represents a suitable analogy to digital images, as the DSM is just another raster data type. The only difference is that it contains height information instead of intensity values, but this does not affect that the sharp changes identified by the edge detector will contain the building boundaries among others and that is the initial step required for a contour-based object recognition method.

Canny edge detector

The *Canny edge detector* [17] uses a multi-stage algorithm based on gradient edge detection (meaning the utilization of the gradient operator's magnitude), including steps for smoothing the image to reduce noise, filtering the edges to achieve clean, thin contours and applying threshold values to eliminate weak edges. All that makes it a complex and flexible method to effectively search for building edges.

In image processing literature, the terms *edge* and *contour* stand for the same phenomenon with the conceptual difference that contours group edge pixels together to denote object boundaries. In the following, the word contour will be consequently used, as it describes better the end goal of the method.

3.2.2 Contour characteristics

In this research a general contour representation was assumed, where the detected contours should be available as a collection of individual connected sequences of 2-dimensional points. The neighboring points are presumed 8-connected (meaning that horizontally, vertically, and diagonally connected pixels are all considered to be neighbors). It extends the neighborhood relation of 4-connected points, so the designed algorithms will be general and capable of operating on specialized data type as well.

Since a contour detecting procedure should be able to produce closed contours, all contour lines are assumed to be ideal contours, therefore they are treated as if they were closed contours and the contour parts will be stored for both directions as a result of trying to perform closure on e.g. a curved line.

Although the contours meeting the above expectations are not eligible in that form for further processing, the existing grouping into connected subsets and the internal point order enables a deep structural analysis for a detailed evaluation.

This contour model was elaborated based on the popular, open-source image processing library *OpenCV*, but the output of any other library or implementation could be used instead as long as it is compatible with these restrictions.

3.2.3 Prefiltering

This step is based on the assumption that buildings tend not to have tortuous outlines. Using not too restrictive conditions is important to prevent from discarding building contours. At the same time, the more non-building contours can be filtered out at the beginning of the process, the more efficient will be the method later at the computationally more demanding steps.

The main idea is the detect extremely curved contour parts while not excluding circular walls. To do so, a sliding approach is used on each fixed-size subsequence of a contour. If it is true for the majority of the examined point sequences that they fit into the bounding box determined by their endpoints, then the contour in question will be kept. It is foreseeable that corner segments of the building cannot always comply with this condition, but this is balanced through the percentage value. Alternatively, if any contour contains a larger straight segment, it will be kept automatically.

It is clear that both aspects can only work with large enough contours, consequently isolated smaller contour pieces get discarded at this phase. Fortunately, it is not a huge loss because it is indeterminable at this point whether such small contours fall to a building boundary or not. Moreover, the smaller a missing contour part is, the higher the chance that it will not prevent from detecting the entire building.

In Figure 3.3 all of the detected contours can be seen, while Figure 3.4 shows the remaining contours after applying the prefilter. It excellently demonstrates the elimination of small contours and some simplification in vegetation areas.

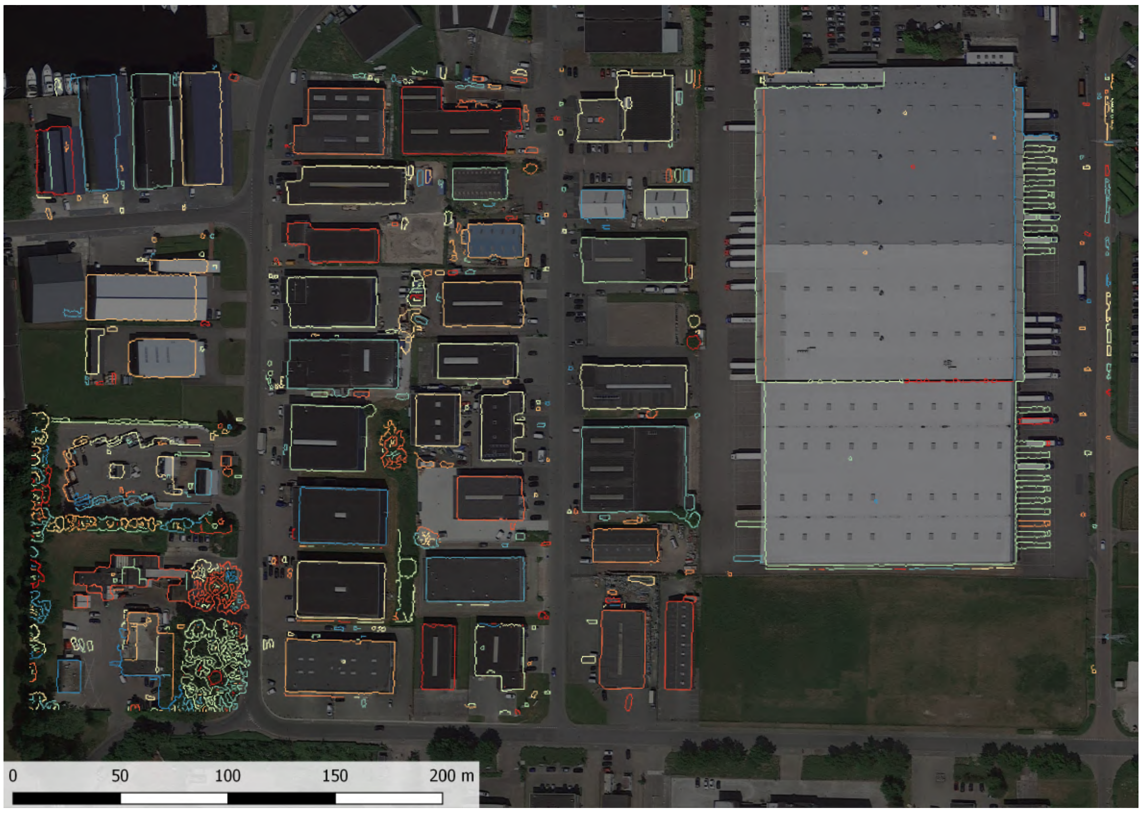

Figure 3.3: The detected contours for the AHN-3 data set using randomly generated colors to distinguish them.

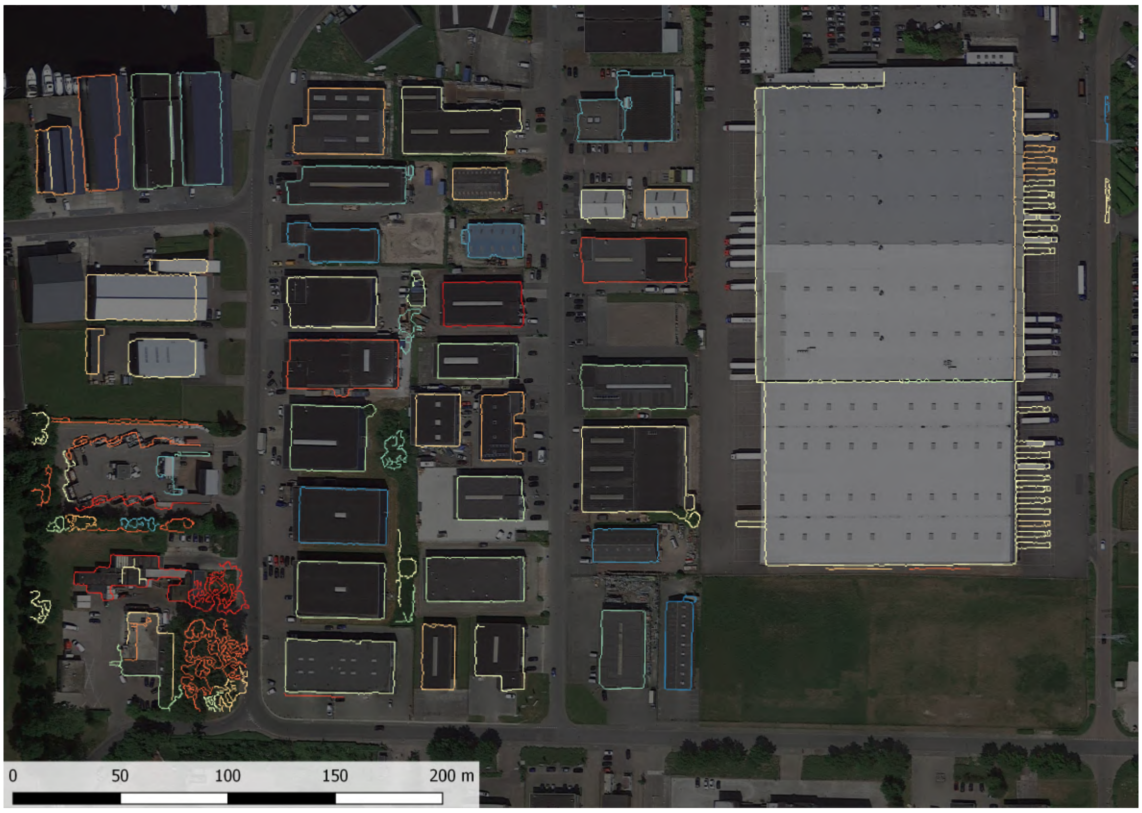

Figure 3.4: The prefiltered contours using randomly generated colors.

3.2.4 Contour splitting

It is an essential preparation to be able to identify the redundant contour segments because otherwise it would be nearly impossible (and computationally expensive) to mark the equivalent subsequences.

The goal is to split each contour into consecutive straight segments, as shown in Figure 3.5. The challenging part originates from the fact that not all 8-connected pixels are also 4-connected, and handling these two cases together could not be done on an easy and clean solution. If there is a difference along both coordinate axes at the beginning of a segment, the allowed directions can be determined in advance for the segment. However, a rasterized slanting line will contain neighboring point differences where only one coordinate axis value changes, so it would not be useful to separate segments for every distinct relation type between 8-connected raster pixels. On the other hand, starting with a partially 0 difference vector for a segment does not mean that it cannot be also a slanting line as a whole. There should be a limitation of course, that in the same segment there cannot be deviation to both directions along the axis showing initially 0 difference.

Even after thorough consideration, one issue remains to be resolved. Namely, always

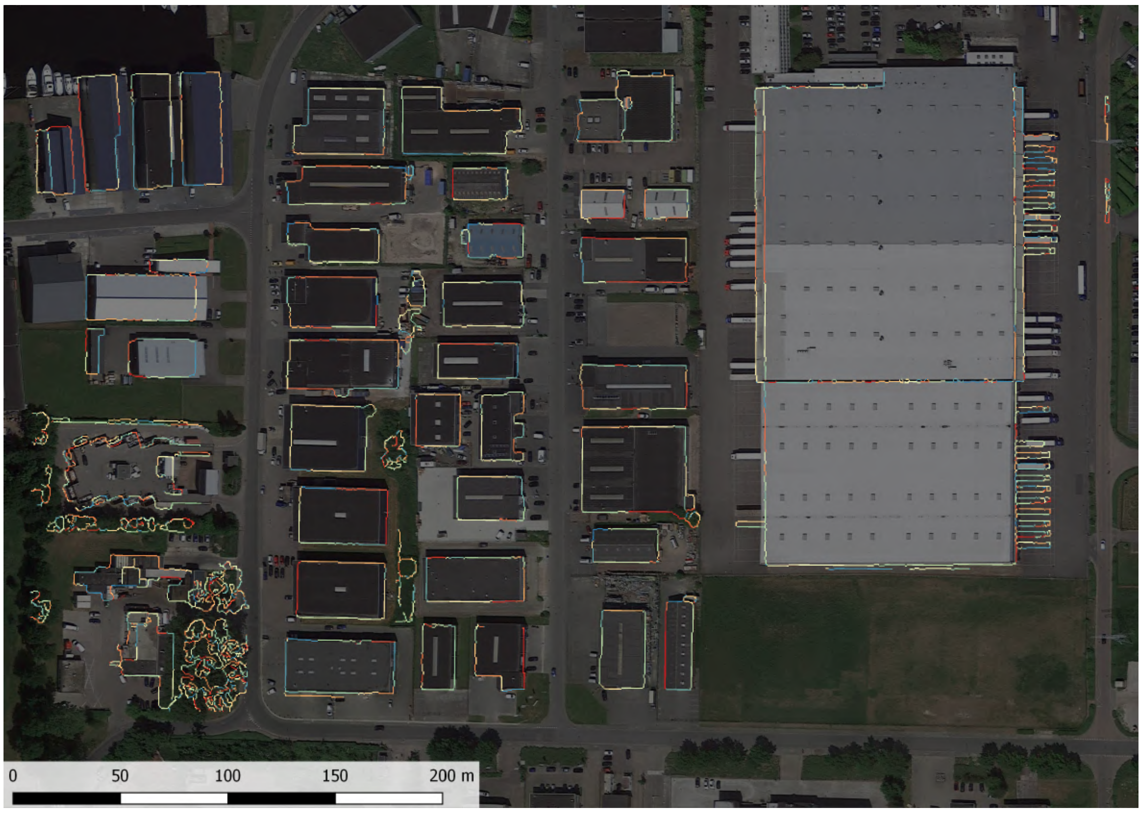

Figure 3.5: The identified straight contour segments using randomly generated colors.

looking at the difference of only two neighboring points locally, the method will not be able to differentiate among segments with reorganized point-differences, e.g. between a perfect perpendicular corner and a rasterized slanting line. To avoid that, an extra post-processing step was embedded split these segments further into axis-parallel parts.

To sum up, at this phase three important aspects were contradicting to each other: i) the computational efficiency, ii) not having limitations to the minimal segment size, and iii) the additional difficulty in achieving order independent partitioning for noisy, branching contours too. As these goals could be optimized against each other, meeting all three expectations cannot be achieved.

3.2.5 Redundancy removal

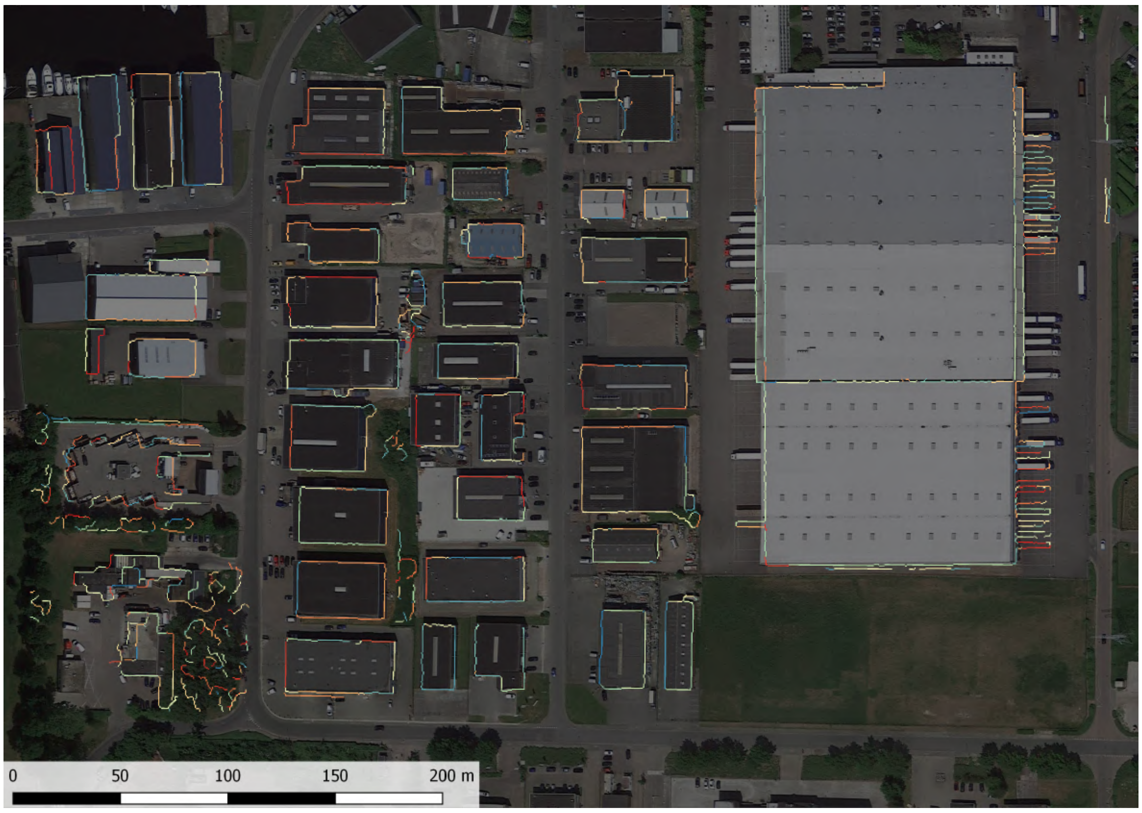

Figure 3.6: The identified contour segments grouped by direction using randomly generated colors.

Using the exact and clear output of the contour splitting phase, there is not much complexity left at this step. For efficiency reasons, the similarity between segment end-points along with segment distance is tested first (independently from the point order of the segment of course). Knowing that noises can confuse segment boundaries from different directions, a point-wise comparison serves as a fallback solution to delete as many redundant contour segment as possible. After that, the consecutive segments having similar direction are grouped together to reduce the overall number of segments. Figure 3.6 illustrates the intermediate state of the process after this step.

3.2.6 Building filter

After the necessary contour manipulations, the most important part regarding the results is to collect the building contours and to discard everything else.

At first, contours containing fewer points than a threshold are filtered out. It is reasonable to expect the buildings to have a minimal size in all dimensions. Furthermore, it would be risky to draw established conclusions from such few data. After that, multiple conditions are tested and if a contour meets more than one of them, it will not be categorized as a building. Each condition is just a potential indicator for a contour not denoting building boundary, therefore multiple must be fulfilled for the final conclusion.

- The distribution of the directions determined by the contour segments is calculated. It is suspicious if the majority of the equally divided angle partitions is present.
- The proportion of the complex segments is calculated. A segment is considered complex if the number of points contained in it is significantly more than the size of a straight line between the segment endpoints.
- The proportion of the longer segments is also used as a condition.
- The last evaluation aspect is the average segment size.

Despite the greatest effort to reliably create closed, complete contours without restriction to building shapes, it could not be achieved in noisy or complex areas (see Figure 3.7), therefore an approximating solution was used in the form of computing the convex hull from the contour points. Unfortunately, this can introduce easily some false positive raster pixels in the result, e.g. in case of buildings with concave floor area, or nearby vegetation contours attached to the building outline.

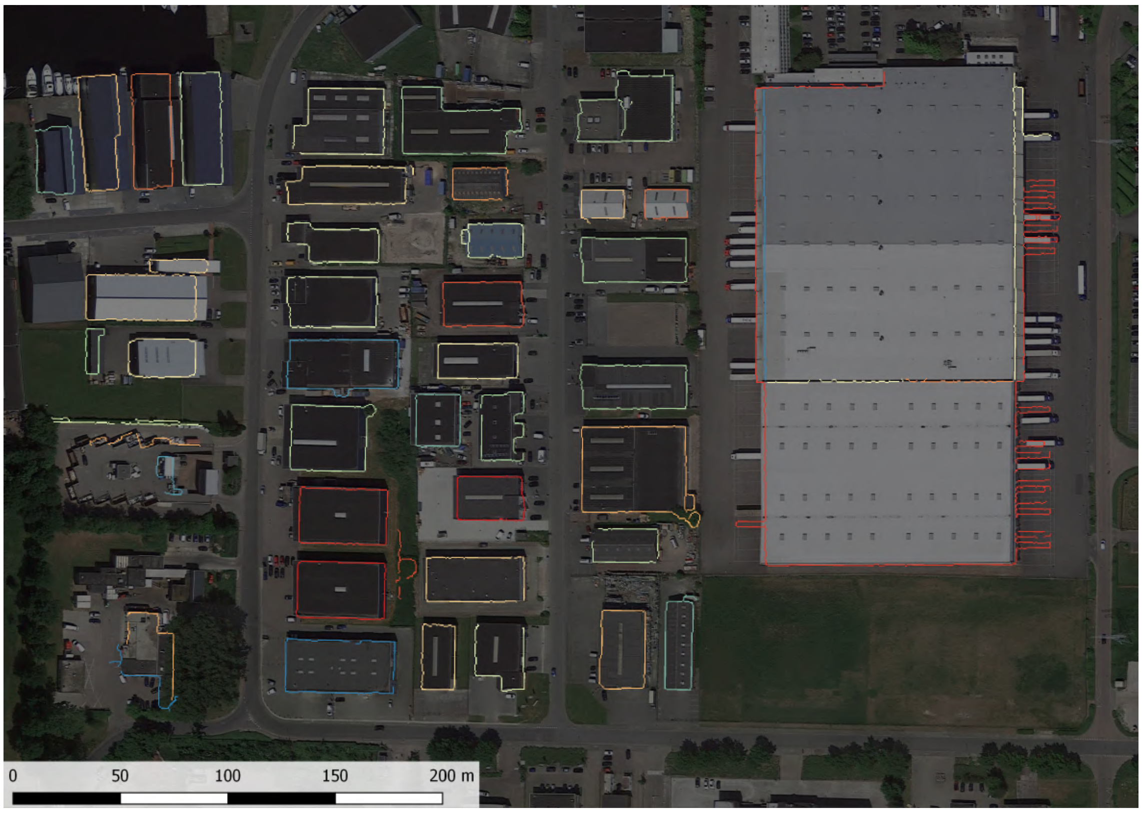

Figure 3.7: The final contours using randomly generated colors.

To reduce the impact of these false positive parts as much as possible, the goal is to eliminate some of the noise in a contour segment chaining procedure. If most of the segments can form a roughly closed contour then only the successfully chained segments are kept, otherwise all contour segments are used to create the convex hull.

In Figure 3.8 and 3.9 the output of the building recognition part of the method can be seen for both the AHN-2 and the AHN-3 data sets, respectively.

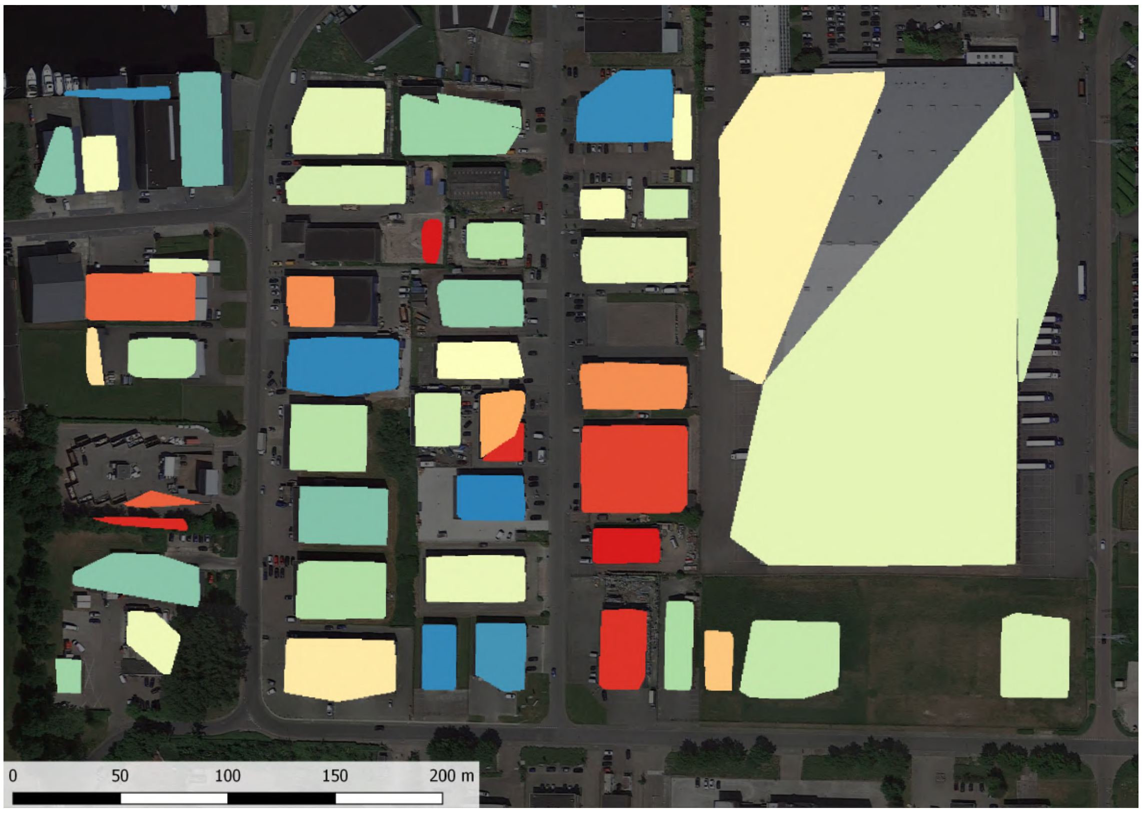

Figure 3.8: The generated convex hull polygons of the AHN-2 data set using randomly generated colors.

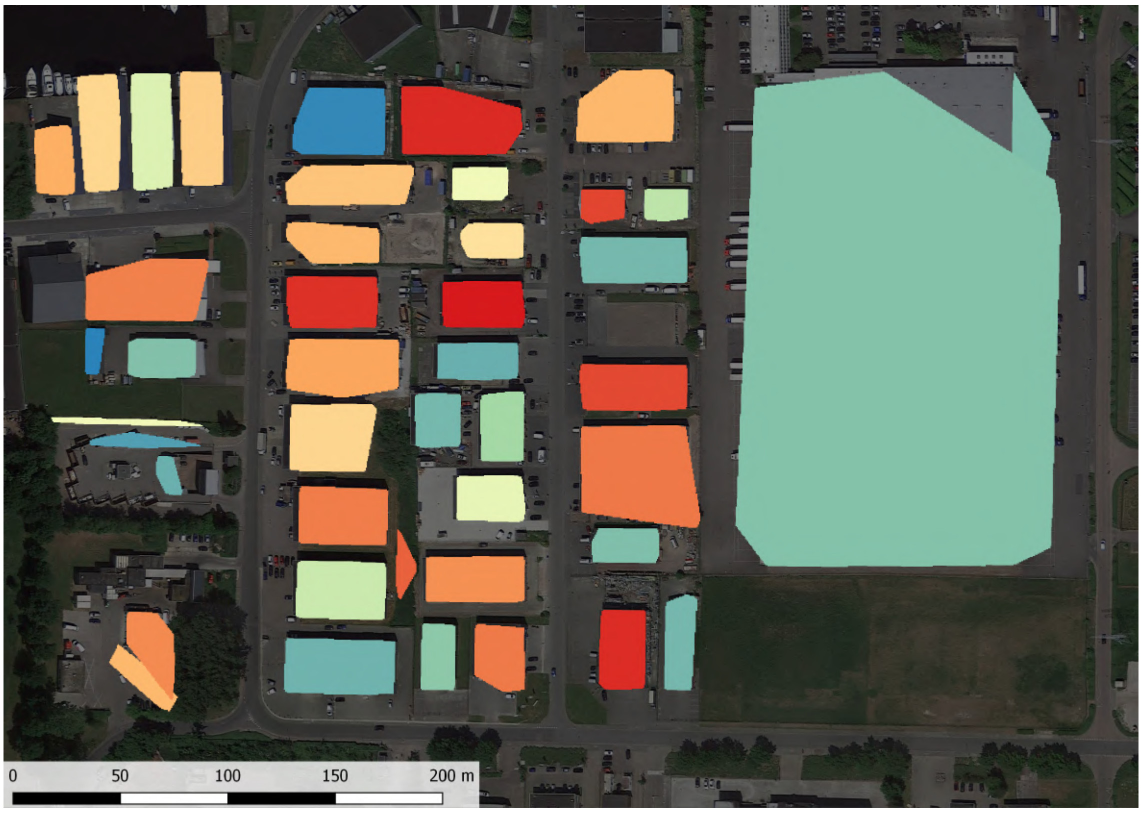

Figure 3.9: The generated convex hull polygons of the AHN-3 data set using randomly generated colors.

3.3 Change detection

The whole building recognition process serves as the basis for the change detection phase where the raster pixels of the generated convex hull polygons will be compared to determine changes regarding the buildings of the scene.

3.3.1 Tessellation statistics

The basic concept is to describe each existing overlapping polygon pair through the available height data and thus be able to draw conclusions with regard to other features of the overlapping segment as well.

This way, a simple statistics consisting of the extent of a segment (given in raster pixels) supplemented with the average height difference for the same area can be computed. To achieve valid results, preparing for outlier values is essential, and it occurs frequently in raster pixels of scattered spots with no original laser scan data.

The procedure produces a variety of building and non-building polygon pairs identified by their unique identifier generated for each convex hull. For efficiency and consistency reasons, topology information is also stored, meaning each building polygon from one epoch is associated with every overlapping polygon from the other epoch. It makes it possible to ignore the insignificant overlapping areas from the other epoch, as it is not realistic to carry out a reconstruction on less than 10% of a building.

3.3.2 Classification

Obviously, the previous statistics will result in the following four cases which means a tessellation with no overlaps and no gaps for the test area.

1. building segment with another building segment
2. building segment with non-building segment
3. non-building segment with building segment
4. non-building segment and non-building segment

Among them, the category with two non-building segments does not really provide additional valuable information because it aggregates height differences for the largest segment of the area by far, where the underlying real-world changes are unknown and can be varying.

The rest of the pairings contains at least one presumed building area, which makes it possible to conclude for demolitions, new building and reconstructions.

When two building segments are compared it is considered unchanged unless the average height difference exceeds 2 meters. The convex hull approach can cause false positive building raster pixels as already mentioned before, but these have a relatively small impact on the average over the entire building area. Furthermore, in many cases, no significant change takes place in these false positive raster pixels either.

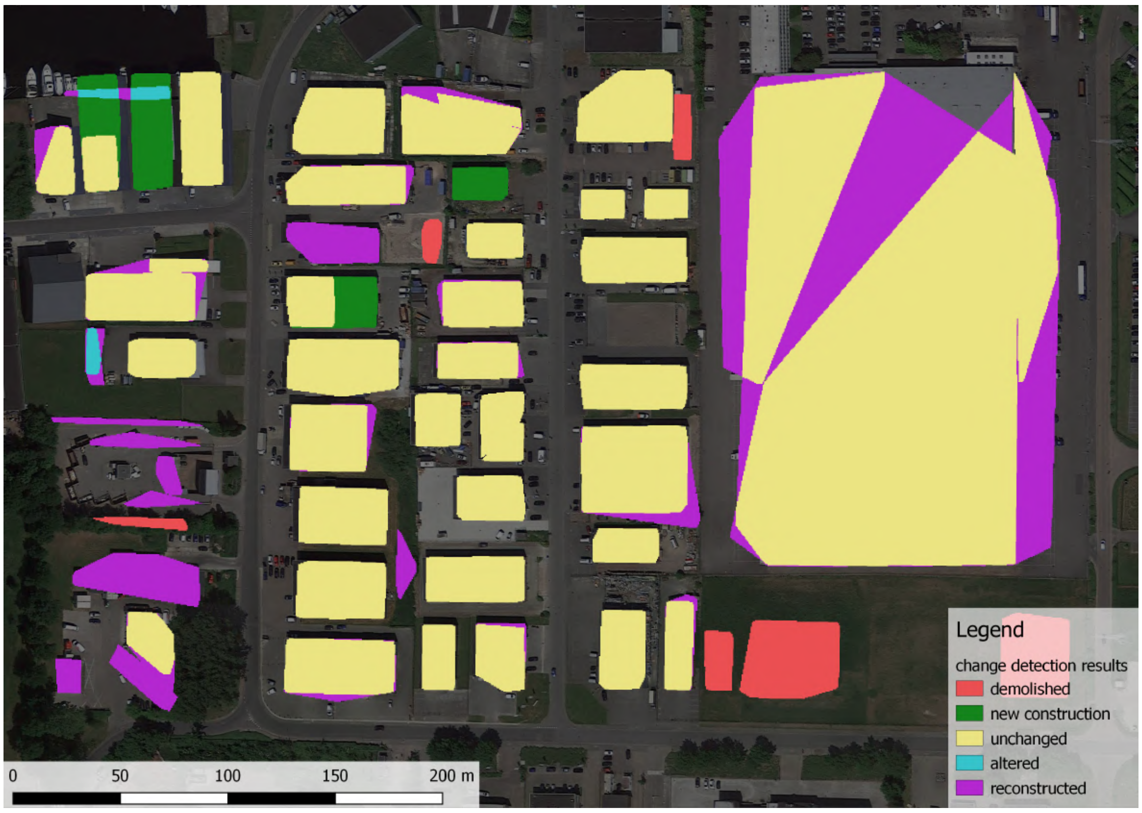

Figure 3.10: The classification result of the building change detection.

If a building and non-building segment form a pair together, that not necessarily means there is no building relating the detected one. Complex automated methods usually cannot achieve 100% accuracy on an arbitrary data set. Several decisions can be responsible for false negative outcomes, e.g. $i)$ the edge detector missing a building contour, $ii)$ accidentally discarding crucial contour segments, or $iii)$ the misclassification of buildings with unusual characteristics. However analyzing the height difference compared to the detected building is an effective tool to correct previous mistakes, therefore to reconstruct missed buildings. Again, the 2 meters threshold was chosen to denote the significant changes. The final result of the method is visualized in Figure 3.10.

Chapter 4

Implementation

The prototype implementation for the methodology described in Chapter 3 was carried out in standard C++11 as part of the *CloudTools* geospatial framework. CloudTools – developed at Eötvös Loránd University – aims to create an easily reusable, high-abstraction level, operation-based software library for raw point cloud and DEM processing.

As DEM files (preprocessed from the original point cloud) were selected as the input for the proposed algorithm as discussed in Section 3.1 due to computational and storage space efficiency, the implementation strongly depends on the *GDAL/OGR*[2] geospatial and geoprocessing software library for the input and output management of the spatial data. The *OpenCV*[3] (Open-source Computer Vision) image processing library was utilized for contour extraction, filtering and processing through the provided C++ interface. The images in this study used for the visualization of the results were exported from *QGIS*[4], an open-source geographic information system application.

The Visual Studio 2017 IDE served as the development environment on a 64-bit Windows operating system.

4.1 Architecture

The implementation follows the design principles and architecture of the CloudTools library. The complete algorithm is divided into a sequence of smaller operations, which yields multiple advantages:

[2]GDAL/OGR: `https://www.gdal.org/`
[3]OpenCV: `https://opencv.org/`
[4]QGIS: `https://www.qgis.org/`

- it provides a higher abstraction level and reusability of the parts;
- it enables to perform flexible changes on the workflow;
- it makes the intermediate results of the process easily retrievable.

The UML class diagram shown in Figure 4.1 demonstrates how the implementation fits into the frames of the CloudTools library. Each class is responsible for a corresponding task of the method described in Chapter 3. and they are derived from the abstract base class with the appropriate abstraction level.

On the most abstract level, the `Operation` class indicates a general transformation from arbitrary data type to any other ($Any \rightarrow Any$). Processing of specifically DEM data sets is represented by the `Calculation` class ($\{DEM\} \rightarrow Any$). Finally, the `Transformation` class can be used as a superclass for mapping a collection of DEM files to an output DEM file ($\{DEM\} \rightarrow DEM$).

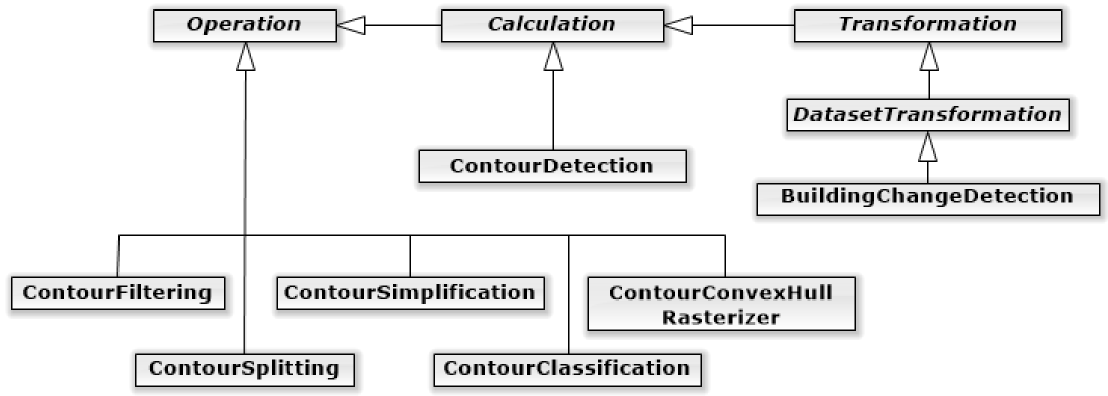

Figure 4.1: Class diagram of the prototype implementation.

In the prototype implementation of this research, custom `Operation` subclasses were created:

- From the initial DEM files, the `ContourDetection` class extracts a collection of contours.
- The contour filtering, splitting, classification and further processing are implemented in the `ContourFiltering`, `ContourSplitting`, `ContourSimplification` and `ContourClassification` classes.
- Finally, for conversion between raster and vector data types, the `ContourConvexHullRasterizer` maps the contour collection to polygons in a raster format. The `BuildingChangeDetection` class generates

the final DEM output by processing the original AHN DSM input files and the rasterized building polygons.

The workflow diagram in Figure 4.2 is a concise representation of the entire method. As the diagram shows, the tasks preceding the change detection of buildings (`BuildingChangeDetection`) have to be executed twice, because there are two input sources (AHN-2 and AHN-3). Since there is no data dependency between them, the two sources can be easily processed in a parallel manner, improving performance.

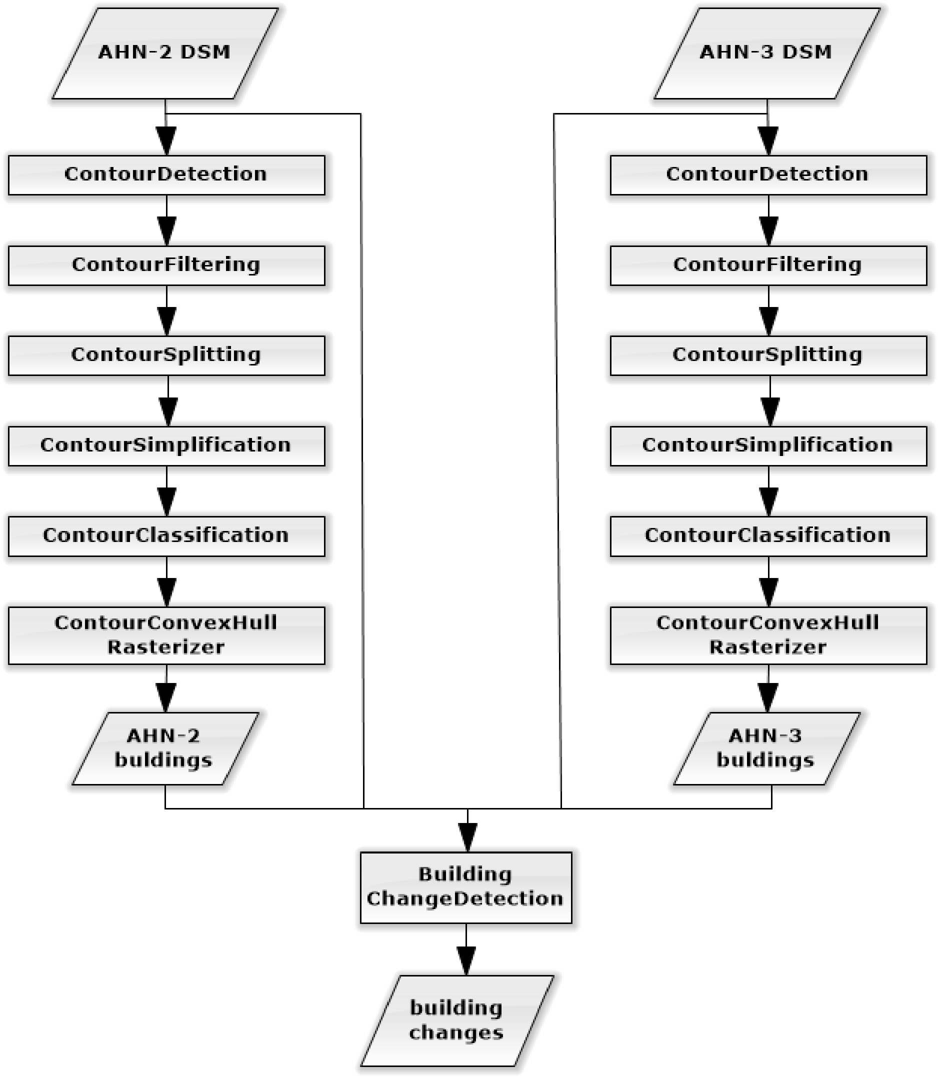

Figure 4.2: Workflow diagram for the proposed algorithm.

4.2 Implementation remarks

As for the implementation of the Canny edge detector, the recreation of this well-known algorithm did not seem necessary, as the widely used image processing library OpenCV could be utilized. Even if using its built-in functions prevents us to access all of the partial results of the edge detection procedure, the provided parameters to customize the behavior can ensure the required flexibility. Although a general solution would deduce the parameter values, in this particular case the main goal is to detect all significant edges and there is not enough information for a sophisticated distinction between edges based on solely the height data of arbitrary areas. Besides, the needed computational investment is not ideal either for a small possible quality gain compared to rational reasoning.

All parameters for customization are related to the capability of the edge detector to filter out weak edges. The threshold values set up a condition for the calculated gradient values, and the flag defines the computation used for that gradient value. The parameters are adjusted to reliably detect at least 3 meters tall buildings after the height map data is transformed into an 8-bit greyscale image. Parameters[5]:

- *threshold1*: the lower threshold for the hysteresis procedure
- *threshold2*: the upper threshold for the hysteresis procedure
- *L2gradient*: a flag, indicating an option for more accurate computation

As a result, a binary image is generated, which can be processed by a procedure for finding contours also provided by OpenCV. According to the documentation, it is based on the algorithm described by Suzuki [18]. Unless proven otherwise later, keeping highly optimized solutions for typical challenges seem to be the right decision again.

The optionally generated hierarchy information is not helpful with noisy contours. Nevertheless, using the retrieval mode do discard nested contour is useful to reduce the number of all detected contours. The approximation method serves as a tool for storage efficiency by simplifying the contours in terms of the number of stored points, but this would prevent contour analysis necessary for later steps of the method. Parameters[6]:

- *hierarchy*: storing hierarchy information
- *mode*: contour retrieval mode

[5]`https://docs.opencv.org/3.3.0/dd/d1a/group__imgproc__feature.html#ga04723e007ed888ddf11d9ba04e2232de`

[6]`https://docs.opencv.org/3.3.0/d3/dc0/group__imgproc__shape.html#ga17ed9f5d79ae97bd4c7cf18403e1689a`

- *method*: contour approximation method

It generally holds that each detected contour starts with a point with minimum Y-coordinate value which is followed by 8-connected points from the counterclockwise direction. After that, there are many possibilities for the actual ordering of the included points depending on the exact shape of the contour. It is worth mentioning that the redundant contour segments do not always consist of the same set of points, presumably because sometimes more points are part of the contour than it is necessary for the minimal connectivity among them.

Based on these observations and the assumed contour representation, the biggest inconvenience was that the contour parts almost always had the same starting and ending point because of the redundancy (the ideal, truly closed contours were the only exception). Consequently, no conclusions could be drawn at first glance, the computational cost was always proportional to the size of the contour.

Chapter 5

Results

The results from a functional point of view were already showcased on a study area through Chapter 3. This chapter focuses on the performance results of the prototype implementation in Section 5.1 and on the verification of the building recognition in Section 5.2. Finally, the results and interesting issues are discussed in Section 5.3.

5.1 Performance

To carry out the performance testing, the following test areas with increasing size both in area and file size were chosen:

- *t1*: the study area in Drachten — 0.15 km^2 — 2MB
- *t2*: the university campus in Delft — 2.14 km^2 — 33MB
- *t3*: the city center in Delft — 8.64 km^2 — 90MB

Task	Execution time		
	t1	t2	t3
building recognition	7.6 sec	273 sec	970 sec
change detection	18 sec	451 sec	1760 sec

Table 5.1: Execution time in seconds for test areas with increasing extent. The column headers are referring to the test areas described above.

Table 5.1 contains the performance results for the different test areas executed on a computer with a 3rd generation Intel mobile processor having 2.5 GHz base frequency, on a single CPU core. Each value represents the average of five individual measurements.

The row for building detection shows the results when the process is stopped right after creating the building polygons for both epochs, i.e., no change detection takes places. The second row with change detection stands for the entire procedure, including the generation of the intermediate results for building recognition.

According to the results, more than half of the computation time is needed for the building recognition part in case of a test area with a considerable extent. For the study area *t1*, the program executes so quickly that the communication with the hard drive could have a significant impact on the proportions among the measurements.

The most important aspect to be discussed is the time complexity of the algorithm. Evidently, it takes more than linear time, as the execution time increases faster than linear with the size of the input. Looking into the details, the Canny edge detector algorithm has $\mathcal{O}(n \log n)$ complexity where the n denotes the number of raster pixels in the DEM file. Besides, the implementation of the proposed algorithms is of polynomial time ($\mathcal{O}(n^k)$) in the number of contours or building polygons at worst, because those are always independently processed from each other and the computationally more demanding parts take place for the inner structure of the individual elements.

As already mentioned in Section 4.1, having no data dependency between the multitemporal data sets in the building recognition process makes simultaneous calculations possible using parallelism, obviously at the cost of doubled memory usage.

5.2 Verification

The governmental *TOP10NL*[7] public data sets contain building records for the Netherlands. The verification process compares the official building polygons with the ones detected by the presented method. However, it is not easy to guarantee that the data acquisition date for the building records and the laser scanning are essentially identical, therefore mistakenly inaccurate results might be possible. Moreover, the misregistration of buildings can also lead to systematic translation errors. The TOP10NL building layer over the study area *t1* is displayed in Figure 5.1.

The verification tool of the CloudTools library produced the following results displayed in Table 5.2 for the AHN-3 data set from around 2014 and the TOP10NL building

[7]TOP10NL: `https://zakelijk.kadaster.nl/-/top10nl`

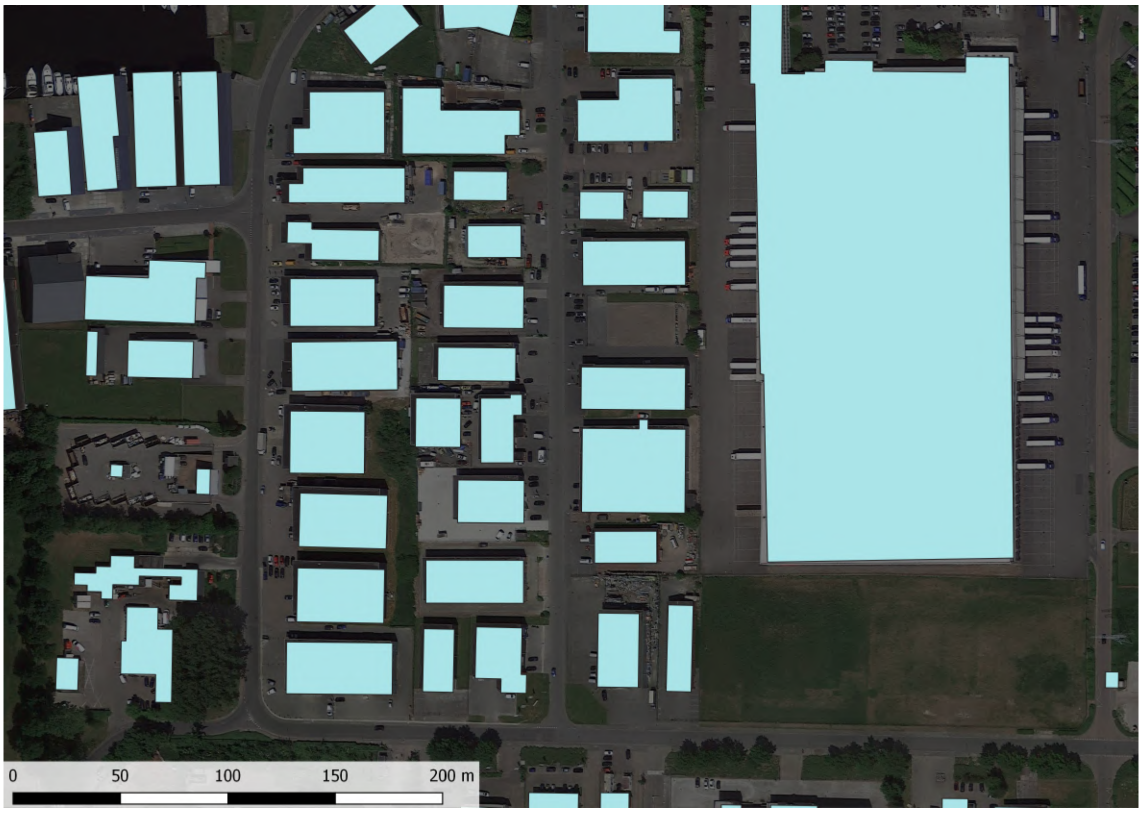

Figure 5.1: The building polygons of the TOP10NL building records on the study area *t1*.

records from November of 2015.

The basic verification option uses a simple raster pixel-wise comparison. Each marked pixel in the output is categorized as either approved or rejected based on the reference building raster points. The percentage value is calculated from the number of approved pixels divided by the number of all marked pixels.

The corrected variant tries to eliminate the effect of the systematic translation errors, therefore misclassifications near to building boundaries are reevaluated.

Mode	Accuracy		
	approved	rejected	ratio
Basic	211 608 px	47 324 px	81.72%
Corrected	231 240 px	27 692 px	89.31%

Table 5.2: Verification results for the building recognition output on the AHN-3 data set of the study area *t1*.

The results, 81.72% and 89.31% respectively, are acceptable considering the fact that the contour of the large warehouse building on the right side of the study area could not be well approximated because of the nearby parking trucks, which lead to a significant amount of false positive raster pixels. The secondary source of the false positive results is the convex hull approach for representing building contours further increased by vegetation contours merged together with actual building contours.

Note. *The verification process could be further improved by calculating also the false negative results indicating missed buildings.*

5.3 Discussion

After examining various spacial surface objects on the selected test areas and considering other possible corner cases, some limitations for the method in its current form could be determined.

Limited information content. Using only the surface data sets as the input for the method can sometimes lead to difficult situations. The height of a detected surface object compared to its surroundings cannot be determined easily, that is a reason why many studies use the DSM and a computed DTM together. It is hard to detect if a building contour is distorted by nearby vegetation without introducing limitations to the shape of the building. In this case, the multiple return information from the original point cloud (or from some rasterized version) could help. But as always, more data requires more computational resources.

Equal contours. Each contour from the height map image represents the same local sharp height difference, independent from its meaning in the scene. Nested contours (e.g. chimneys, inner courts), separator lines in multi-level buildings and connecting corridors between buildings are equal to real boundary contours. Therefore, deciding whether a building complex is only one building or contains multiple buildings is a challenge.

Other surface objects with regular shape. Large trucks, containers, ships or even a heap of earth on construction sites can have a rectangular shape resembling a building and resulting in false positive detections.

Artificial contours. The DSM can contain scattered spots with no data acquisition from the laser scanning. Not using a morphological operator or interpolation techniques leads to artificial contours in the scene.

Chapter 6

Conclusion

I presented a method in this study for contour-based building recognition and change detection from airborne LiDAR which could be directly applicable for various tasks facing certain GIS challenges. Automated and efficient processing on rapidly increasing data sets, e.g. for object recognition purposes is an active and important topic nowadays.

Utilizing contours generated from the height map of the area for building recognition made it possible to work out a method providing favorable computer resource (especially memory and storage space) requirements compared to other studies through reducing the number of required input data sets. Such optimization could be particularly relevant considering the vast amount of laser scan data provided by state-of-the-art LiDAR technology.

The performance analysis of the prototype implementation showed an advantageous computational complexity having great practical importance for the proposed algorithm.

To compare epochs of multitemporal data sets for change detection purposes regarding the buildings of the scene, I worked out an approach based on possible pairings of identified building contours represented by their convex hull polygon, and successfully achieved a kind of reconstruction capability in case of lack or loss of information. These characteristics together with the validation results of the test area are promising signs for the robustness of the method especially with improvement possibilities already in mind.

6.1 Future work

During the implementation phase, multiple ideas came up to further improve the existing solutions.

- More complex procedures could probably result in a more robust method. Trying to merge incomplete contour fragments should reduce the missed buildings in the area. The building filtering phase could be supplemented by other conditions based on global shape analysis e.g. using principal axis determination.
- Atop of the refined building recognition process, a multi-level output system with confidence values could be elaborated to express the reliability and successfulness of the previous steps.
- The convex hull representation of the contours should be refined by partitioning the polygons into convex segments. This could have a positive effect on the validation result because it would eliminate a significant source of the false positive regions.
- Finally, the computational efficiency should be examined too. It is straightforward, that the epochs could be processed parallel, but also smaller parts of the method could be potential candidates for parallel computing.

Acknowledgements

EFOP-3.6.3-VEKOP-16-2017-00001: Talent Management in Autonomous Vehicle Control Technologies – The Project is supported by the Hungarian Government and co-financed by the European Social Fund.

Bibliograhpy

[1] Celia García-Feced, Douglas J Tempel, and Maggi Kelly. "LiDAR as a tool to characterize wildlife habitat: California spotted owl nesting habitat as an example". In: *Journal of Forestry* 109.8 (2011), pp. 436–443.

[2] ASPRS Board of Directors. *LAS Specification*. Tech. rep. 1.4 – R13. American Society for Photogrammetry and Remote Sensing, July 2013. URL: `https://www.asprs.org/wp-content/uploads/2010/12/LAS_1_4_r13.pdf`.

[3] L Swart. "How the Up-to-date Height Model of the Netherlands (AHN) became a massive point data cloud". In: *NCG KNAW* 17 (2010).

[4] PDOK. *Besteksvoorwaarden inwinning landsdekkende dataset AHN2014-2019*. Tech. rep. 2.0 final. Dutch National Spatial Data Infrastructure, May 2015. URL: `http://www.ahn.nl/`.

[5] PDOK. *Kwaliteitsdocument AHN2*. Tech. rep. 1.3 final. Dutch National Spatial Data Infrastructure, May 2013. URL: `http://www.ahn.nl/`.

[6] Corné Van der Sande, Sylvie Soudarissanane, and Kourosh Khoshelham. "Assessment of relative accuracy of AHN-2 laser scanning data using planar features". In: *Sensors* 10.9 (2010), pp. 8198–8214.

[7] Uwe Weidner and Wolfgang Förstner. "Towards automatic building extraction from high-resolution digital elevation models". In: *ISPRS journal of Photogrammetry and Remote Sensing* 50.4 (1995), pp. 38–49.

[8] Uwe Weidner. "Digital surface models for building extraction". In: *Automatic extraction of man-made objects from aerial and space images (II)*. Springer, 1997, pp. 193–202.

[9] G Priestnall, J Jaafar, and A Duncan. “Extracting urban features from LiDAR digital surface models”. In: *Computers, Environment and Urban Systems* 24.2 (2000), pp. 65–78.

[10] Zheng Wang and Tony Schenk. “Building extraction and reconstruction from lidar data”. In: *International Archives of Photogrammetry and Remote Sensing* 33.B3/2; PART 3 (2000), pp. 958–964.

[11] George Miliaresis and Nikolaos Kokkas. “Segmentation and object-based classification for the extraction of the building class from LIDAR DEMs”. In: *Computers & Geosciences* 33.8 (2007), pp. 1076–1087.

[12] Ruijin Ma. “DEM generation and building detection from lidar data”. In: *Photogrammetric Engineering & Remote Sensing* 71.7 (2005), pp. 847–854.

[13] Minghong Xie, Kun Fu, and Yirong Wu. “Building recognition and reconstruction from aerial imagery and lidar data”. In: *Radar, 2006. CIE’06. International Conference on*. IEEE. 2006, pp. 1–4.

[14] Tuong Thuy Vu, Masashi Matsuoka, and Fumio Yamazaki. “LIDAR-based change detection of buildings in dense urban areas”. In: *IGARSS 2004. 2004 IEEE International Geoscience and Remote Sensing Symposium*. Vol. 5. IEEE. 2004, pp. 3413–3416.

[15] T Vögtle and E Steinle. “Detection and recognition of changes in building geometry derived from multitemporal laserscanning data”. In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 35.B2 (2004), pp. 428–433.

[16] Thomas Butkiewicz et al. “Visual analysis and semantic exploration of urban lidar change detection”. In: *Computer Graphics Forum*. Vol. 27. 3. Wiley Online Library. 2008, pp. 903–910.

[17] John Canny. “A Computational Approach to Edge Detection”. In: *Readings in Computer Vision*. Ed. by Martin A. Fischler and Oscar Firschein. San Francisco (CA): Morgan Kaufmann, 1987, pp. 184 –203. ISBN: 978-0-08-051581-6. DOI: `https://doi.org/10.1016/B978-0-08-051581-6.50024-6`. URL: `http://www.sciencedirect.com/science/article/pii/B9780080515816500246`.

[18] Satoshi Suzuki et al. “Topological structural analysis of digitized binary images by border following”. In: *Computer vision, graphics, and image processing* 30.1 (1985), pp. 32–46.

List of Figures