GIMA
Geographical Information Management and Applications

Map supported point cloud registration

a method for creation of a smart point cloud

Falco Joosten (5770653)

fjoosten@cyclomedia.com

Edward Verbree (TU Delft)

Prof. Peter van Oosterom (TU Delft)

Arjen Swart (Cyclomedia)

Bas Boom (Cyclomedia)

Master thesis

17 August 2018

cyclomedia
Smart Imagery Solutions

Map supported point cloud registration

As a method for creation of a smart point cloud

By

Falco Joosten

in partial fulfilment of the requirements for the degree of

Master of Science

in Geographical Information Management and Applications (GIMA)

at the Delft University of Technology, Utrecht University, Wageningen University and University of Twente,
to be defended publicly on Friday September 14, 2018

Supervisors:	Ir. Edward Verbree
	Prof. dr. ir. Peter van Oosterom
External Supervisors:	Arjen Swart, MSc.
	Dr. ing. Bas Boom

TU Delft

cyclomedia
Smart Imagery Solutions

ABSTRACT

Rich urban point clouds are becoming more and more important in 3D city modelling. While significant progress has been made in classification, segmentation, registration and extraction of buildings, little research has been conducted on the creation of smart point clouds. This thesis discusses to which extent topographic maps can be used to improve the positioning of point clouds while at the same time enriching them with meaningful information. The transfer of knowledge from map to point cloud is conducted in two steps. First, the positioning of mobile laser scanner data is geographically aligned to high accuracy maps. In this way, the positioning of the point clouds is improved, enabling them to be combined with other sources of geographical information. Second, building attributes from the map are attached to the corresponding buildings in the point cloud. This results in a smart point cloud with unique object identifiers and attributes for each building. Through the map supported registration and enrichment of point clouds, the usability of LiDAR data is improved. More accurate registration enables the transfer of attributes and thus the amount of extractable information is increased. In this way, point clouds can become better structured and more meaningful sources of information.

ACKNOWLEDGEMENTS

This thesis could not have been written without the academical support from Delft University of Technology. It was a pleasure to work together with Edward Verbree and Peter van Oosterom. It was only possible to write this thesis thanks to their guidance.

Thanks to all colleagues at Cyclomedia who expressed their interest in the topic. Their questions and constructive feedback kept me motivated and enthusiastic about the research. Special thanks to my supervisor Arjen Swart, who possesses an astonishing amount of knowledge and is always willing to share it with others. Also, I would like to thank Bas Boom who has always been able to address critical topics during our weekly meetings.

Last but not least, I would like to thank my parents, sister and girlfriend for their inexhaustible support.

TABLE OF CONTENTS

List of Figures

List of Abbreviations

2D	Two-dimensional
3D	Three-dimensional
AHN	Actueel Hoogtebestand Nederland (*Height model of the Netherlands)*
BAG	Basisregistraties Adressen en Gebouwen (*Dutch key registries of addresses and buildings)*
BGT	Basisregistratie Grootschalige Topografie (*Dutch key registry for large-scale topography)*
BAGGT	Self-appointed new map which is a combination of the BGT and BAG
CPU	Central Processing Unit
CSV	Comma Separated Values file
DCR	Digital Cyclorama Recorder system
FFT	Fast Fourier Transform
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GPU	Graphics Processing Unit
ICP	Iterative Closest Point algorithm – method for alignment of point clouds
IMU	Internal Measurement Unit
LAS	Laser file format for LiDAR point data
LAZ	Compressed laser file format
LiDAR	Light Detection and Ranging
MMS	Mobile Mapping System
MLS	Mobile Laser Scanner
NCC	Normalized Cross Correlation
OpenCV	Open Source Computer Vision Library
PGO	Pose Graph Optimization
RANSAC	Random Sample Consensus algorithm – method for outlier reduction
RGB	Red Green Blue additive colour model
RGBD	Red Green Blue Depth
SLAM	Simultaneous Localization and Mapping
TLS	Terrestrial laser scanners
UTC	Universal Time Coordinates

1 Introduction

This first chapter will introduce the topic and its associated definitions. Primarily, definitions such as Cyclomedia's recording system, point clouds and positioning are explained to provide context in **section 1.1**. Thereafter, issues with positioning errors and the usability of the recorded data are explained in **section 1.2** on the problem statement. Subsequently, the research questions are identified in **section 1.3**. The content and scope of this research is delineated in **section 1.4**. Finally, the chapter concludes with a brief overview of the entire thesis in **section 1.5**.

1.1. Context

Whilst three-dimensional datasets are being used for eye striking visualizations, their practical purposes are not fully exploited yet. Where maps are the result of accurate measurements and georeferencing, 3D models often use local coordinates and lack accurate world coordinates. Since reference points are based on a **Global Navigation Satellite System** (GNSS) with occasional positioning errors, it remains difficult to determine the exact world coordinates. Maps can act as trustworthy landmarks to improve the positioning of 3D models and at the same time provide benefits for the creation of semantically richer point clouds.

Cyclomedia is providing spherical imagery of the public space in the Netherlands. Their own patented Digital Cyclorama Recorder (DCR) system is mounted on top of their cars (**Figure 1**). In this way, Cyclomedia provides systematic large-scale visualizations at street level. Their Mobile Mapping System (MMS) creates 360 degrees street level images which give an extended overview of the outdoors public space (Cyclomedia, 2015). The spherical imagery is competing with traditional surveys. Both time and money are saved since inspections can be carried out from behind a desk.

The imagery is combined with a **Light Detection and Ranging** (LiDAR) system. This system emits laser beams to illuminate its surroundings and cause reflections. The return time and strength of the reflections is measured to estimate distances with an average accuracy of two centimetres. Each distance is then combined with the scan

angle to calculate the measured positions as points in a three-dimensional coordinate system. Those coordinates (x, y, z) are the most essential result of a point cloud survey. While the LiDAR scanner is measuring its surroundings, certain other attributes are provided for each point in space as well. For example, additional information is gathered concerning the source location, scan direction and number of returns for each laser beam.

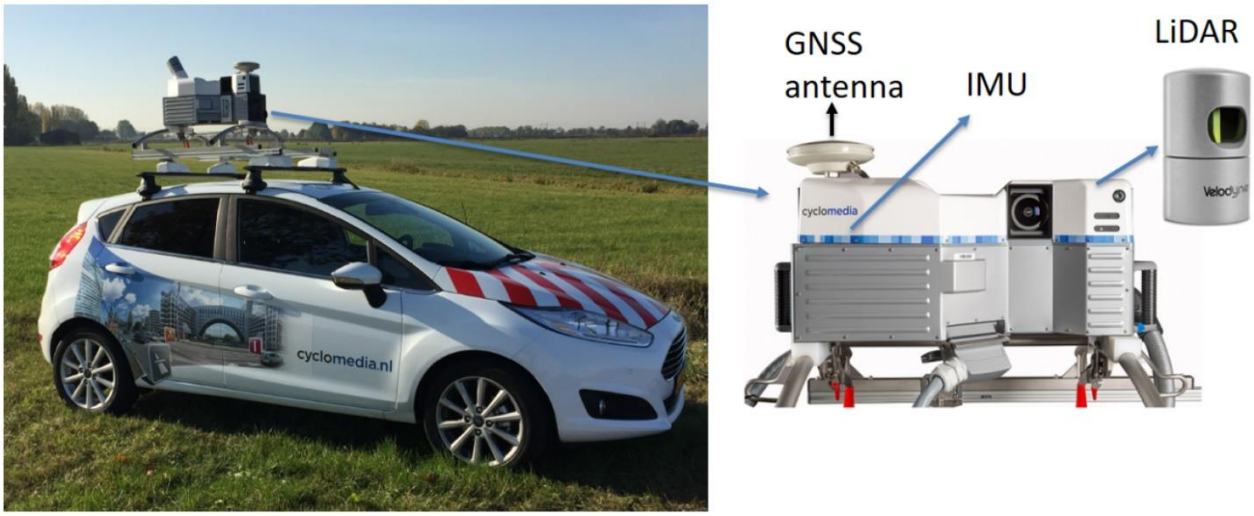


Figure 1: Cyclomedia's Mobile Mapping System with the DCR10 system includes a Global Navigation Satellite System, Inertial Measurement Unit, Light Detection And Ranging system, and five cameras in order to create parallax-free panoramas. The resulting images and point clouds are used in tools for geo-referenced viewing and measuring.

The resulting scene is called a **point cloud**. In Cyclomedia's solutions, the point clouds are combined with images to add colours to the model. The LiDAR system itself is not able to capture colours while scanning its surroundings. Colours are extracted from captured images which are being taken simultaneously by the recording system. Colour values from the overlapping images can be attached to the point cloud to create an RGB (red, green, blue) point cloud, such as visible in **Figure 2**. On the other hand, depth-information from the point cloud can be added to the pictures. The resulting image is an RGB-D image, where the "D" stands for depth. The main purpose of linking depth information is to the images is to enhance measurements in "distance-value-added" panoramic images. On the other hand, the coloured point clouds provide visual feedback to the scanned area (Verbree, Zlatanova, & Dijkman, 2005).

The mobile laser scanner collects the points through illumination of its surroundings. The resulting point cloud scene already contains relative positions of the points with respect to each other. However, to accurately place the points

in a coordinate system, the scene must be georeferenced by supplying it with ground truth coordinates (Talaya et al., 2004). To retrieve their absolute global positions, the laser internal time system is synchronized to a global positioning time system by using a **Global Navigation Satellite System** (GNSS) in the first place.

Figure 2: Laser scanners are not capable of recording colours. The RGB (Red, Green, Blue) colour values are derived from pictures taken simultaneously with the Digital Cyclorama Recording system. The point cloud scene includes the recording trajectory of the mobile mapping system in red. Since the LiDAR system is continuously measuring, the points of origin appear in the scene as a solid line.

The GNSS is combined with an **Internal Measurement Unit** (IMU) to place the scene more accurately in the coordinate system. The IMU is an additional unit, mounted on the car to improve positioning when GNSS-signals are weak. The IMU is a very sensitive piece of hardware, which can measure acceleration and angular velocity to estimate new positions independently from any external systems. It only requires a reliable initial location with periodic recalibration to occur drift and calculate its current position. Therefore, it is a powerful positioning tool as soon as GNSS is lacking (Kain & Yates, 1996). In situations as such, the IMU can estimate navigation errors and provide additional information to the GNSS data.

1.2. Problem statement

Relative positional accuracy rates of LiDAR systems are rather high. The LiDAR sensor mounted on cars from Cyclomedia has a typical accuracy of +/- 2 centimetres (Someren, 2017). This results in a low amount of inconsistencies between mutual points in one scene. However, the LiDAR system also makes use of an external GNSS receiver to determine its own absolute position, as was explained in the previous chapter. Discrepancies emerge in built up urban environments (see **Figure 3**) or under tree cover, where the GPS signal is weaker.

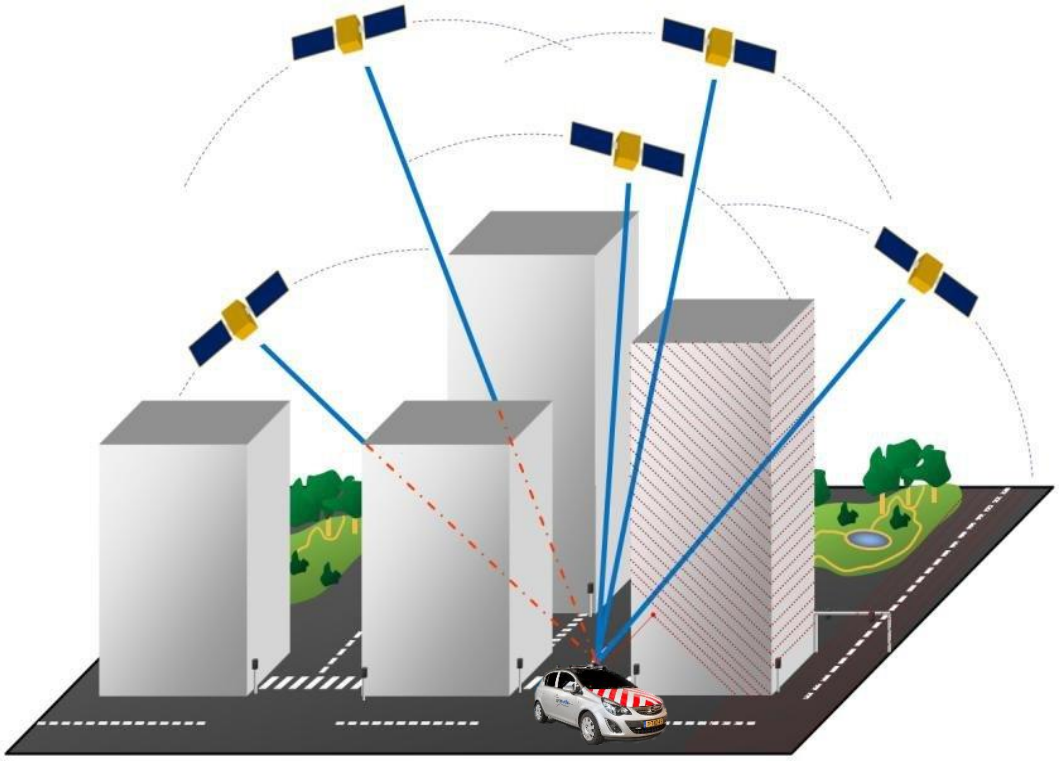


Figure 3: Positioning errors occur in urban areas where the GNSS is inaccurate due to occlusion. The distorted satellite signals are causing trouble for the Mobile Laser Scanner, which then acquires wrong positions. Eventually, the point cloud will be stored and projected at false positions in the coordinate system (image modified from [Kukko Et Al., 2012]).

If the GNSS is providing unreliable positions, the IMU can minimize the positioning errors. Still in some cases, even a combination of GNSS and IMU does not guarantee accurate navigation. For example, speed bumps can cause strong vibrations or shocks that have a bad influence on the IMU's accuracy. Errors may magnify seriously due to strong vibration and shock interferences (Peng, Zhi, Wang, Liu, & Zhang, 2014). Also, GNSS errors can become worse due to atmospheric distortions, multipath errors or jamming (Joosten, 2016). Such conditions are constantly

changing through time. As a result, inconsistencies also arise if the mobile laser scanner is arriving at the same location multiple times, such as is the case at intersections. For all these reasons, large positioning errors will continue to appear in LiDAR datasets. Intensive positioning processing and calibration are required to determine the actual recording positions. The trajectory is computed again from behind a desk using the raw measurements collected by the digital recording system.

Distortions in the positioning are resulting in wrongly projected coordinates for the point cloud. The errors are not clearly visible when looking at a single point cloud scene alone. The differences will only become visible as soon as the 3D model is combined with other geographical data in one view. If the positioning accuracy of those LiDAR datasets would be better, they could be used for multiple other purposes as well. Through the alignment of point clouds with existing sources of geographical information, such as maps with high accuracy on a large scale, it becomes easier to combine point clouds with relevant information concerning the visible objects. For the effective transfer of map features to the point cloud, it is essential that the two datasets have almost no registration errors (Murali, 2018).

Besides registration errors, point clouds suffer from several other limitations as well. For instance, their visualisation relies on hardware with very powerful graphic processing units. A medium sized point cloud scene demands too much internal memory from the average point cloud user (Houshiar & Winkler, 2017). The large size of point cloud data has been an impeding factor. Therefore, point clouds are mostly being used indirectly through their reduced interpretations (e.g. meshes), which require less processing power. In addition, specialized software, knowledge, skills and insight are required for explorative processing and analysis (Verbree & Van Oosterom, 2015). Thus far, the data is mainly seen as a source material for derivation of other more manageable products, after which much of point cloud data is hardly being used again.

Besides that, the raw point clouds are unstructured and lack classification. To enhance the direct usability of LiDAR datasets, it should become easier to combine additional sources of information with the points. Underlying tasks known as registration, georeferencing, segmentation, and structuration have a major influence to attach knowledge onto points (Billen, Poux, & Ruymbeke, 2017). Segmentation is the process of grouping point clouds into homogeneous regions with similar properties. Consecutively, classification is the process of labelling those groups

of points (Grilli, Menna, & Remondino, 2017). The final step is object detection and identification, in which objects within a class are identified as separate entities. Through improving the positional accuracy and enhancing the amount of available information in the point cloud, the direct usability will increase.

The positioning of point clouds can be adapted to different types of maps. For example, registries of buildings and addresses could be used to match the positioning of a building in the point cloud with its corresponding footprint on the map. Also, maps containing the topography of objects in public space can be used to move point cloud trees and street lights to their actual georeferenced positions. In most examples of point cloud registration, different versions of the same point cloud are aligned with each other to match their locations and orientation. The registration in this thesis however, is conducted through making the objects in the LiDAR dataset fit with their corresponding locations in a map. The map functions as a reference template over which the point cloud will shift to find the best match. The challenge here is to shift the points to the accurate locations in the map while at the same time preventing inconsistencies in the point cloud.

In the above approach, the map is accepted as a ground truth. It is thereby assumed that the map always corresponds to reality. This is an important assumption which does not always apply. The Dutch key registries which are being used in this thesis, show typical accuracy rates of approximately 5 centimetres. Exceptions exist, since Dutch legislation allows for error margins up to 30 centimetres (Rossem, Eekelen, & Reuvers, 2012). Therefore, in some cases, the point cloud might outperform the map in its resemblance to the truth. However, the eventual goal is to enable a combination of point cloud data with different sources. The focus will be on point cloud enrichment, whereby existing points are assigned corresponding attributes from the map. Therefore, it is important to make the point cloud fit the map as accurately as possible.

As soon as the point cloud is aligned to the map, it should be possible to automatically identify certain objects in the point cloud. For example, footprints of buildings can be used as a bounding box in combination with a small buffer to group all points belonging to a certain building in the map. The resulting collections of points will enable possibilities to attach new knowledge to the point cloud. Of course, the bounding boxes of buildings will contain more relevant corresponding points as soon as the map supported registration has been conducted. This transfer of knowledge makes the point cloud more “smart”. It allows for smart filtering to reduce the required amount of

processing power. Above that, the new attributes can be visualised in a point cloud scene where different buildings are projected in different colours representing their year of construction, type of ownership or surface area.

1.3. Research identification

The objective of this thesis is to align point clouds with key registries and thereby allow for the attachment of new attributes to the point cloud. At the same time, this map supported registration will improve the positioning of point clouds as well. To solve the afore mentioned problems and find the correct answers, several questions must be asked first. The main research question is:

- In which way can **map supported registration** of mobile laser scanner data be used to create a **smart point cloud** of buildings?

This specific question raises a list of other questions regarding the automated registration and enrichment of point clouds. To answer the main research question, those questions must be tackled first:

1. Which registration techniques can be combined in an algorithm to improve the efficiency and robustness of the registration?
2. How can template matching be used to optimize existing point cloud registration techniques?
3. To what extent can the alignment support the transfer of semantics to the point cloud?

The three questions stated above are asked in a chronological sequence. Different existing registration techniques are explored and explained first. As soon as the most suitable techniques have been identified, they will be applied on the data. Maps are being used to align the point clouds to existing sources of geographical information. The map supported registration will be compared to existing methods of point cloud registration. Thereafter, the aligned point cloud is used to create a smart point cloud. Separate collections of points are being created for every matched map object. Attributes from the topographic map are then linked to the corresponding point cloud objects. Eventually, a smart point cloud will visualize buildings in a range of different colours representing their corresponding map attributes.

1.4. Scope

With the stated research questions, several challenges can be distinguished. Unfortunately, not all those challenges can be resolved within this thesis. Time constraints are forcing the research to deal with a limited amount of issues only. Therefore, the limitations on the scope are identified below.

Firstly, the reliability of the used methods must be considered during every research step. Different methods are being used with varying degrees of reliability. In this research, the point cloud will be adjusted to align with the map. In this way, the map remains unchanged in one fixed position while the point cloud uses the map's georeferenced topography as a reference point for the acquisition of accurate coordinates. At some stage, discrepancies between the two will arise due to errors in either map or point cloud. Differences as such could indicate misplaced or missing objects in maps. Likewise, they could be an indication of distortions or noise in the point cloud as well. Such restrictions on the data quality are taken into consideration, but the detection of map errors and removal of noise is considered out of scope. Possible map errors are taken for granted, even if they decrease the absolute positioning accuracy of the point cloud.

Differences in the time of acquisition could create problems as well. Both versions of reality, as well the map as the point cloud, can contain errors in relation to each other and the real world. Such errors threaten the reliability of the results. In case of a dissimilarity, it is often hard to know which representation better resembles reality. On the contrary, the locations of such errors could become one of the most interesting self-contained outcomes. A map with locations of discrepancies between the LiDAR data and topographical maps can be very valuable input for as well Cyclomedia as mapping agencies or governments.

In this research, errors in the elevational axis (z) cannot be considered since the 3D point clouds are being matched to 2D maps. Registration will be conducted for two dimensions only, namely the x and y axis. Comparing two-dimensional versions of point clouds saves time and effort, but the downside is that valuable height information is not being used. However, since 2D building buffers are sufficient to transfer attributes to the point cloud, usage of simplified point clouds will meet the registration requirements. It would still be valuable to include another registration technique for improvement of the elevational values as well. This would be possible since a detailed height model of the ground level of the Netherlands is publicly available. The national height model (AHN) of the

Netherlands could be used as a template for registration of point clouds in future research. However, conducting three-dimensional corrections on the longitude, latitude and altitude is out of the scope of this research.

Besides elevational errors, rotational errors are left out of consideration as well. The problem here is that rotational errors do not simply result in an incorrectly orientated point cloud. On the contrary, heading issues are leading to a very slight deformation of the point cloud. Solving such irregularities is out of the scope of this research. Above all, rotational errors cannot be solved through adjusting the orientation of the map. Also, rotational errors rarely occur in the dataset which makes it unnecessary to make such adjustments. The same goes for scaling errors. They are not taken into consideration since both map as well as point cloud are assumed to be correctly scaled. Such characteristics of the datasets are important to mention since they have further implications for the available registration methods.

Besides that, the point cloud is shifted in such a way that it will fit the map more accurately. Translations are applied on the LiDAR dataset rather than the map which remains fixed in its initial position. LiDAR scanners have high accuracy rates. In some cases, LiDAR data might even outperform the absolute positional accuracy of maps. This might threaten the quality of the point clouds. It may seem impractical to decrease the data quality of point clouds for the sole purpose of making it fit the shape of the map. However, this is beneficial for the integration of the two datasets and the transfer of attributes. Besides that, it is assumed that the overall positioning of point clouds will be improved through their alignment to maps. Since the true ground level coordinates remain unknown, it would be hard to detect registration errors as well. Therefore, some version of reality must be trusted and discovering the true coordinates of the point cloud is considered out of scope.

To conclude, this thesis will focus on map supported point cloud registration in two dimensions, namely x and y. Local translations are being calculated whereby the point cloud is fitted to existing Dutch key registries of buildings. With those calculated shifts, the LiDAR dataset is aligned through adjustment of the mobile laser scanner's trajectory. Hereafter, the positioning of the point cloud is improved with the main objective to transfer attributes from the key registries to the point cloud. This data integration results in a smart point cloud with linked map attributes.

1.5. Thesis structure

This paragraph can be considered as a reading guide for the next chapters. In total, the thesis is divided into six chapters. The first chapter was an introduction to map supported point cloud registration and the creation of smart point clouds. In **chapter 2**, existing literature concerning registration and smart point clouds is discussed. **Chapter 3** describes prerequisites such as the research location, datasets and hard- and software that are being used. In **chapter 4**, the processing approach is explained. Different methods and techniques are combined to produce the results, which are shown in **chapter 5**. Finally, the conclusions are elaborated in **chapter 6** with a critical attitude and room for discussion.

2 Related work

Over the past few decades, many different techniques to solve localization issues and increase the usability of point clouds have been proposed. These techniques include the use of registration, template matching, classification and annotation. In this chapter, the different existing methods will be explained. First, existing literature on geographical alignment of data is elaborated in **section 2.1**. Then, different viewpoints on the substantive enrichment of point clouds are discussed in **section 2.2**.

2.1. Registration techniques

The combined pairwise reading and referencing of various 3D point cloud data views is also known as registration. A lot of research on point cloud registration has already been done. Many scientists are focusing on point cloud to point cloud registration, where datasets acquired from different views are being aligned in such a way that the intersecting areas show perfect overlap (see **Figure 4**). They develop methods to align point clouds obtained from different points of origin in different time frames. Such methods mostly simplify the data by reducing the number of points. Thereafter, key points are extracted from both datasets to match the different versions and determine the

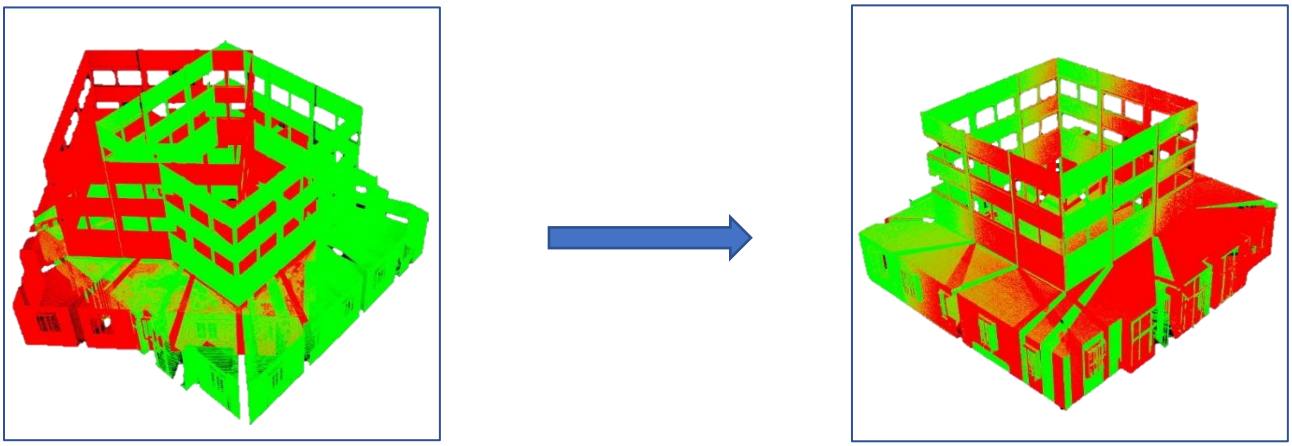


Figure 4: With point cloud to point cloud matching, two datasets recorded from different points of origin are being matched. Deviations between different versions of the same point cloud can be caused by multiple environment-dependable limitations, such as differences in the points of origin or differences in the time of acquisition (image taken from [Ge, 2017]).

required transformations (Ge, 2017). The usage of descriptive key points can ensure higher matching accuracies for the registration process.

Many different methods for point cloud registration are available. Registration can be used to improve both relative and absolute positioning. In relative positioning, deviations between different versions of the same point cloud scene are being matched. When trying to improve the absolute positioning, the point cloud is compared to existing landmarks with exact known positions and placed in a projected coordinate system as accurately as possible. Most researchers are focussing on the relative positioning of multiple versions of 3D point clouds. This thesis differs from most existing research since it tries to match the 3D point cloud with a 2D map. The attributes of 3D point clouds can be used to generate and match 2D projections. Computing only two dimensions saves time and is therefore more efficient. 2D projections can easier be used in cross correlation techniques for image registration purposes. Since this method also estimates redundant solutions, the confidence level of the result increases as well (Christodoulou & Van Oosterom, 2018).

Template matching is the process of locating the position of a subimage (template) inside a larger image or search area. It involves shifting the template over the search area and storing the similarities for each position of the search window. The shifted position with the largest similarity is most likely the correct location of the image in the template (Goshtasby, Gage, & Bartholic, 1984). The **Normalized Cross Correlation** (NCC) is a familiar and widely applied measure of similarity for template matching, recognition and detection in the spatial domain. It provides a fast method to detect the correlation between two images based on the grayscale of pixels. Some limitations of the technique are its inability to deal with scaling, rotation and perspective distortions (Lewis, 1995). However, those limitations do not hold for the problem described in this thesis.

The accuracy of matching techniques can be improved through outlier elimination. The **Random Sample Consensus**, known as RANSAC (Fischler & Bolles, 1987), is commonly used for the removal of outliers that do not bring any valuable information for the registration. In point cloud registration, it provides the basis for an automatic system to match key points between images. For example, outlier elimination can be used to remove incorrect key point correspondences from images. By only selecting and comparing “inliers”, it is possible to search for similarities between different pictures. The performance of RANSAC is highly dependent on the size of the point cloud and the

percentage of inliers. Outlier removal methods such as RANSAC are important for accuracy improvement of point cloud registration.

Besides techniques for noise reduction, many different methods are available for the process of registration. The most common method for alignment of two point clouds might be the **Iterative Closest Point** (ICP) algorithm (Besl & McKay, 1992). It also requires the selection of key points, after which it makes use of a target and a source point cloud. The source point cloud is fitted to the fixed positions of the target points. The source model is iteratively transformed until the distance between corresponding points is minimized (C. Zhang et al., 2016). A risk of the ICP algorithm however, is that it gets stuck in a local minimum if the two point clouds are outside an acceptable range in terms of their position and orientation (Jung, Song, Chang, & Park, 2018). In other words, it needs sufficient overlap to achieve good registration. Therefore, coarse registration is often first applied before the ICP algorithm completes the fine registration.

Registration algorithms are responsible for finding the transformation that best aligns a geometric space (Mendes et al., 2017). Sometimes however, the exact positioning is unknown due to the absence of a referencing system. This problem is known as the **Simultaneous Localization And Mapping** (SLAM) problem (Leonard & Durrant-Whyte, 1991). It can be approached as a graph with nodes representing different poses. By linking different landmarks or points of view to each other, the edges are matched in a consistent way. Since the final result always includes noise, the method typically relies on least-square error minimization techniques (Grisetti, Kummerle, Stachniss, & Burgard, 2010). The challenge is to find a configuration of the nodes that shows the least possible amount of errors. Normally, SLAM is used by autonomous robots for real-time mapping of unknown locations. In this thesis, the settings are less complicated. Still, the theory of SLAM can very well be used to solve the more static idea of referencing point clouds to landmarks in maps.

A schematic overview of the SLAM problem is shown in **Figure 5**. Every node in the graph corresponds to a robot pose. Through pose graph optimization, the estimated locations are measured to determine the most likely positions (Bailey & Durrant-Whyte, 2006). An intuitive way of formulating SLAM is to use a graph whose nodes correspond to the poses of the car, the edges are representing constraints between the poses. A map is computed by finding the spatial configuration of the nodes that is maximally consistent with the measurements (Grisetti et al., 2010).

Such measurements can originate from different sources. Usually, the distance between the robot and landmark locations is used since it is assumed that the current position of the robot is unknown.

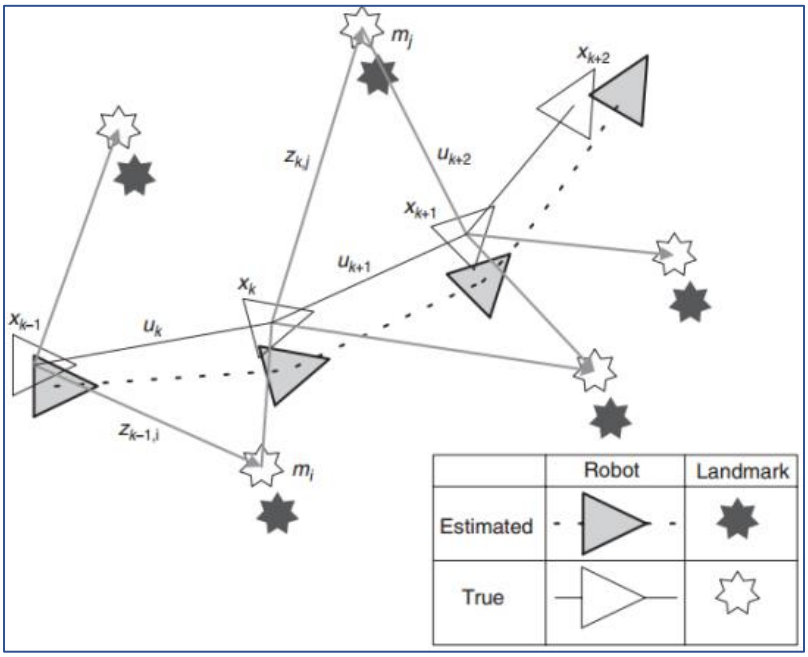


Figure 5: In the Simultaneous Localization And Mapping (SLAM) problem, the location and trajectory of a robot are estimated simultaneously. Measurements are based on the distance between the robot and landmark locations (image taken from [Bailey & Durrant-Whyte, 2006]).

In this thesis, pose graph optimization is applied as well. The main difference however, is that the position of the mobile laser scanner is known through GNSS and IMU. Therefore, measurements from the GNSS and IMU are combined with distances from map land marks in the pose graph optimization. Currently, Cyclomedia is already using traffic intersections to calculate possible positioning differences between two versions of the same point cloud. This point cloud to point cloud registration will be supplemented with the to be conducted map supported registration. In this way, a new positioning indicator is added to the current pose graph optimization. This could have major influence on improvements of the estimated GPS positions of the car and sensor. Also, it will gradually translate the LiDAR dataset into its new positions making sure the local translations are synchronised with each other. Thereafter, the point cloud which is derived from newly calculated sensor positions better matches the map and paves the way for transferring map attributes to the point cloud.

2.2. Smart point clouds

Improvements in the positioning enlarge the opportunities to combine point clouds with other data sources. However, most other spatial datasets are very different from point clouds since they contain objects rather than separate points. Vector maps for example, are defined by edges between different spatial features and therefore show strict divisions between objects. Raster data, on the other hand, has no such strict boundaries. Raster data shows more resemblance to the real world which is not made up of distinct lines (Maffini, 1987). Point clouds show many similarities with raster data since the data is continuous and gradually changing as well. Due to the large number of overlapping objects and multi interpretable categories, segmentation of point cloud remains an open problem (L. Zhang, Li, Li, & Liu, 2018).

Different methods have been developed to derive maps from point clouds, for example in vegetation mapping (Höfle, Hollaus, & Hagenauer, 2012) and 3D building model generation (Zolanvari, Laefer, & Natanzi, 2018). However, fewer examples exist of research in which information from these maps is projected back into the point cloud. As a result, LiDAR datasets suffer from several structural limitations. This is stimulating the more indirect exploitation of point clouds through meshes (Poux, Hallot, Neuville, & Billen, 2016). The raw data is now often being transformed into polygon meshes or other derived information products. Point clouds could become more intelligent structures if more domain information would be attached through semantic variables.

A polygonal mesh is a collection of vertices, edges and faces that define the shape of a point cloud. Triangulation, which generates a network of triangles, is one of the most common methods (Catalucci, Marsili, Moretti, & Rossi, 2018). For any type of mesh, interpolation of the original data is required. Interpolation implies that intermediate values are rounded or estimated. This does not always reduce the overall accuracy of the point cloud. If the amount of noise is limited and the nodes in the mesh are accurate, the interpolation might resemble the real world very well. However, without noise reduction, outliers will appear as strange peaks in the interpolated versions of the point cloud. While polygon meshes might be a suitable representation for applications in computer graphics, in the world of geomatics it is lacking desirable features. For example, a dense polygon mesh of a building is not dividable into meaningful entities. It is unable to distinguish architectural structures such as walls and roofs (Böhm & Haala, 2005).

Automatic point cloud classification is a widely researched field of knowledge. Complex and incomplete scenes, uneven point density and noise make it difficult to generate good features. Some researchers have made use of the geometry of topographic maps to classify mobile laser scanner data (Murali, 2018). However, such methods are solely classifying the points into a limited number of classes. This results in a point cloud in which different types of objects appear in different colours (see **Figure 6**). For example, buildings are coloured red and points belonging to vegetation emerge in green. Both are important tasks when it comes to providing more meaning to a point cloud. However, segmentation and classification do not suffice for the real identification of separate individual objects. Even more knowledge could be attached to LiDAR datasets if it would be possible to request attribute information for individually identified objects.

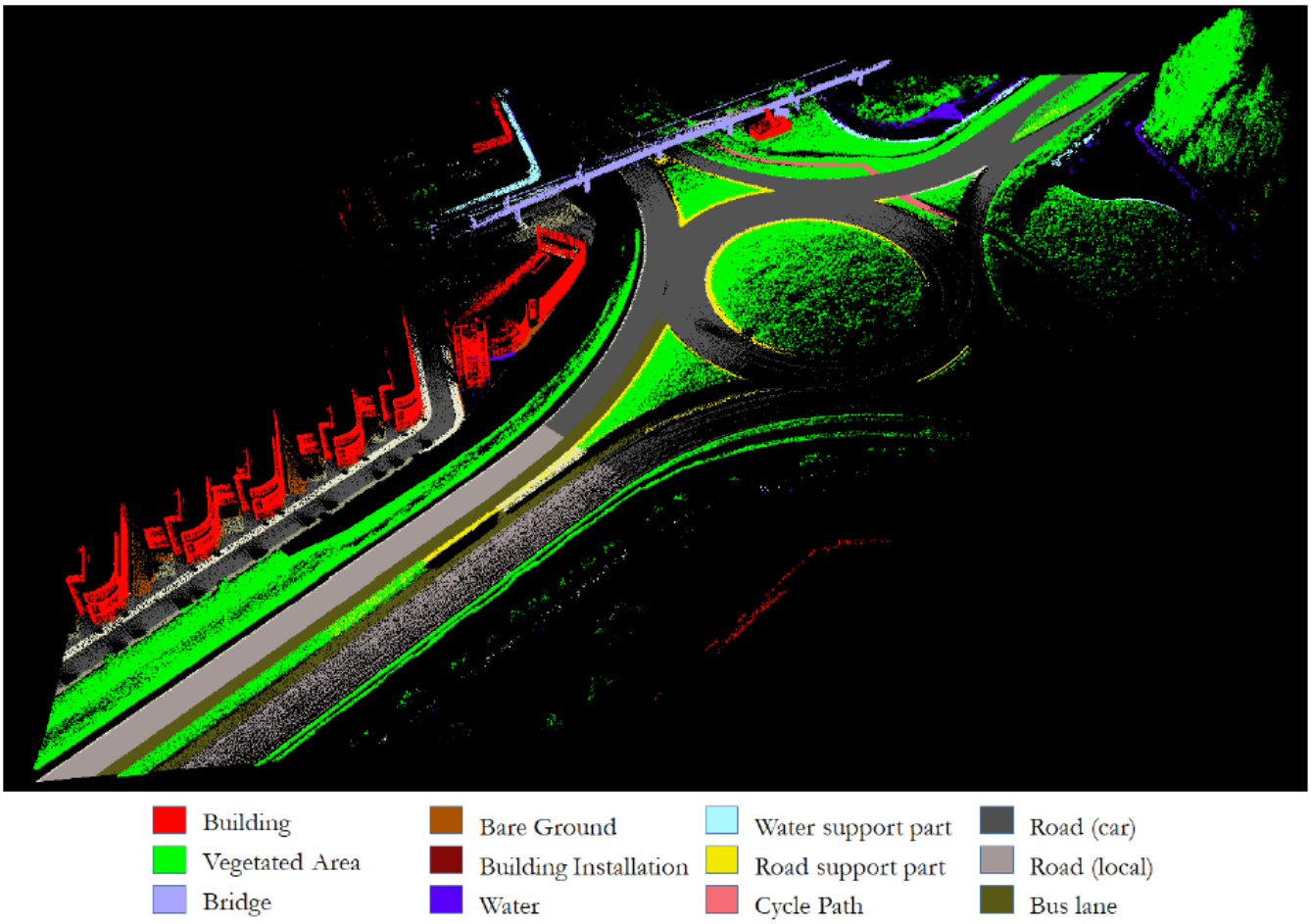


Figure 6: LiDAR classification techniques are aiming to divide the point cloud into a limited number of classes. Here, the point cloud is classified through matching the points with polygon map features (image taken from [Murali, 2018]). In a smart point cloud, objects are not solely classified into different categories (e.g. buildings, vegetated areas and roads). A smart point cloud would allow for diversity per object within a class, enabling to visualize buildings in different colours based on their attributes (e.g. type of ownership, year of construction, surface area).

There is a fine but essential line between classified and smart point clouds. Classification enables the viewer to see the distinction between classified objects. In a smart point cloud however, the user can request many different sources of information from the objects in the 3D-model, such as the year of construction or total surface area of a certain building. All such properties are available in existing datasets, they have only not been linked to LiDAR point clouds yet. In a smart point cloud, collections of points are linked to their corresponding objects with attributes. This research aims at the automated creation of a smart point cloud for a restricted research location. In the next chapter, all required prerequisites, datasets and their specifications will be discussed.

3 Prerequisites

In this section, the prerequisites of the research are further explained. First, the research location is described in **section 3.1**. Later, the associated datasets (**3.2**), software packages (**3.3**) and hardware (**3.4**) are discussed. The specifications of the Digital Cyclorama Recording System have specific implications for the research methods. Also, existing datasets were modified in a certain way to ease the registration process. Only as soon as those prerequisites are explained, an overview of the processing approach is given in **chapter 4**.

3.1. Research location

The registration will be applied to a restricted research location. Processing time and complexity are reduced which will ease the research and development. For the test area, a LAS dataset of Schiedam will be used. An extensive point cloud, captured in September 2016, is available for this area. The region has been divided in squared tiles with a width and height of 50 metres. The total extent of the region is shown below in **Figure 7**. The data covers an area of approximately 4.55 square kilometres in total. This area is significantly big enough to allow for the extracted methods to be applied on other LiDAR datasets in or outside the Netherlands later as well.

For the template matching, all point cloud files are accessed through the tiling scheme shown in **Figure 7**. The spatial division allows for multiple versions of the same point cloud in one tile. For example, this occurs at intersections, where the mobile laser scanner drives through the same tile multiple times. Point clouds of the same area with different timestamps must be treated individually, since their positionings errors might differ. Therefore, the extent of the current tiling scheme will not be used as input size for the actual matching process. For this purpose, the LAS files are first converted, arranged and iterated through time rather than geographic location. Every point cloud tile will have its own unique timestamp. As a result, point clouds with different timestamps and possibly different registration errors will be treated separately by the algorithm.

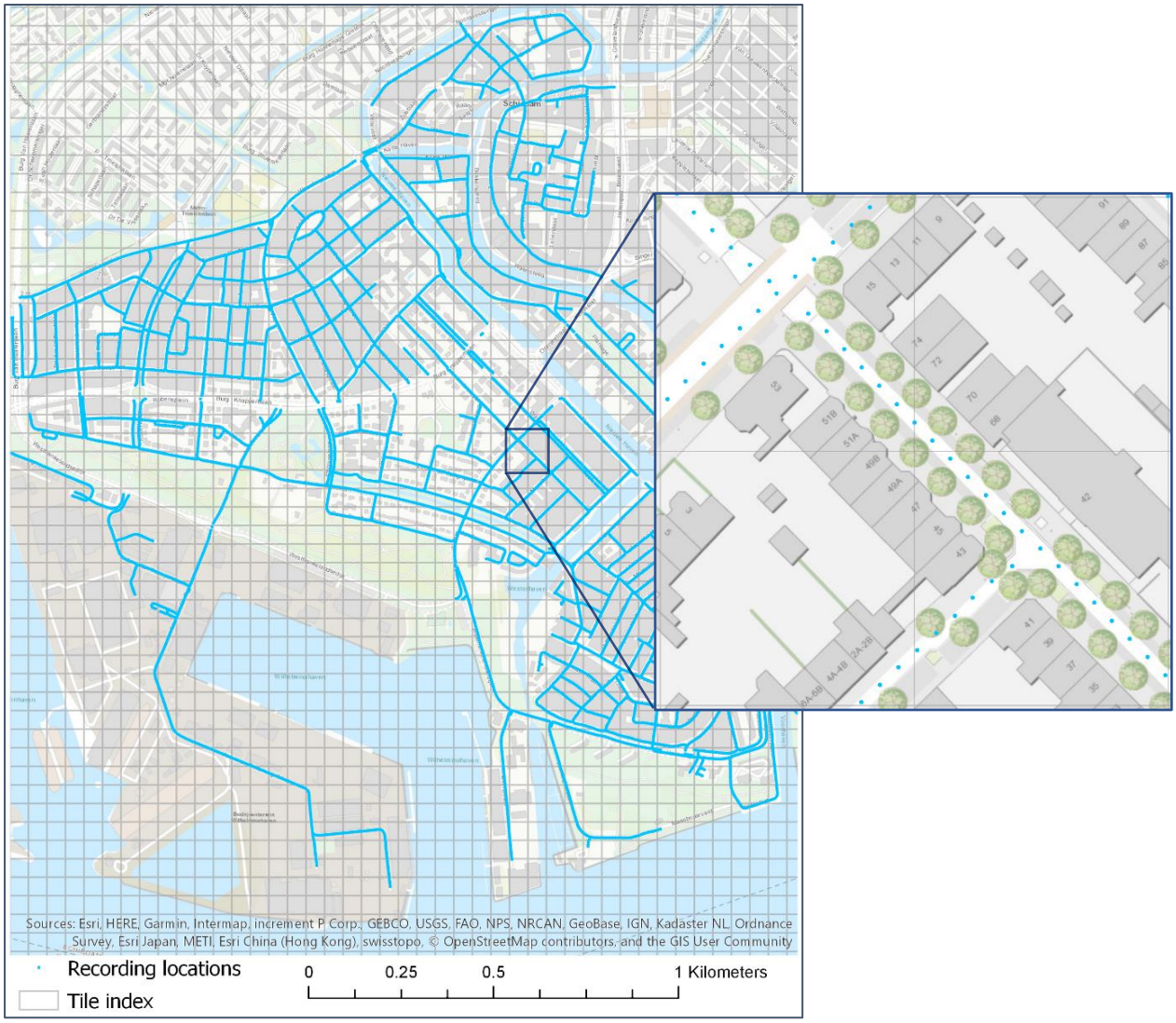


Figure 7: The initial research location consists of 12.230 recording locations (blue) in total. Those positions are positioned at an interval of 5 meters. Whereas the current tiling scheme (grid in grey) consists of static rows and columns, the new scheme for storage of the LAS files will be made based on the dynamic recording locations through time.

While the LiDAR scanner is continuously recording, extensive information from the IMU and GNSS log is available. Information on the positioning is stored at an interval of 200 Hz. This means that 200 positioning records are available for every recorded second. Since it is inconvenient to use all this information, data from the panorama recording locations are used only. The panoramic pictures are being recorded at an interval of 5 meters. Thus, the data gives an indication of the recording trajectory at an interval of 5 meters. Since the recording locations follow the trajectory of the mobile laser scanner, all LiDAR points are acquired near those locations. By using the recording locations as a centre for new tiles with the same size, every tile will contain a sufficient amount of 3D points.

For performance reasons, it is more convenient to exclude some recording locations from the registration process. Therefore, only one out of five recording locations are selected, whereby the resolution is decreased to an interval of 25 meters. This results in a division of the research region into 2.447 recording locations, as shown in **Figure 8**. Tiles with a width and heights of 50 meters are created around the recording locations. All points within the boundaries of those tiles are merged in a new LAZ file. In this way, the mobile laser scanner always has a central position in the new tile. Areas that are near the track of the mobile laser scanner will provide with more dense point clouds. Selecting a buffer of 25 meters around the recording locations ensures that the point cloud is dense enough to match it with the maps. Parts of the point cloud which are recorded from a distance beyond this range will be less accurate and dense. Therefore, it is better to exclude them from the data.

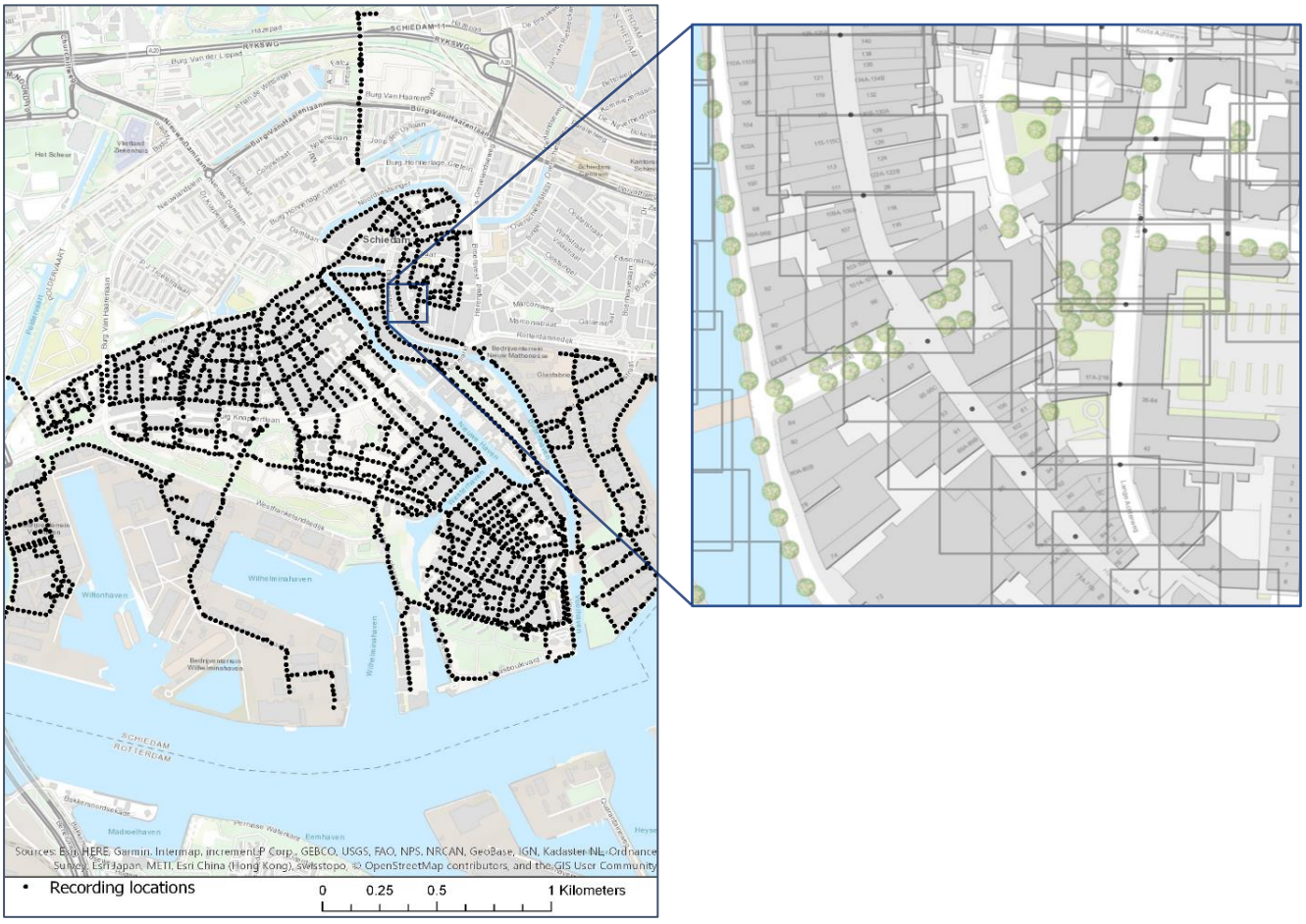


Figure 8: The adjusted research location consists of 2.447 recording locations at an interval of 25 meters (on the left). The point cloud of the research location is divided into new tiles based on those recording locations (on the right). This new tiling scheme ensures dense point clouds since the mobile laser scanner has a central position in every tile.

3.2. Data

In this thesis, different datasets are being combined. The three main sources of information that are being used, are described below. First, the LiDAR dataset from Cyclomedia is discussed in **section 3.2.1**. Afterwards, the different key registries of buildings and addresses (BAG) and buildings in the key registry of large-scale topography (BGT) are described in **section 3.2.2**.

3.2.1. Cyclomedia's point cloud

The entire LiDAR dataset of Schiedam comprises over 16 billion points. The total size of all those points together is approximately 84 gigabytes (GB). This size applies to the compressed LAZ files. The total size of the actual uncompressed LAS files is much higher. After decompression, the file size becomes four up to fourteen times larger. In other words, compression reduces the size of the original files by 75 up to 93 percent (Isenburg, 2013). Even after strong compression, the point clouds remain rather voluminous.

The exact trajectory of the LiDAR scanner is stored in the points of origin. Since one scene contains many points of origins, they occur as a solid line in point cloud visualizations. Due to inaccurately retrieved GNSS positions, a slight drift can occur in this line (Le Marchand, Bonnifait, Ibañez-Guzmán, Bétaille, & Peyret, 2009). Such a drift is hardly visible, but it results in a misplacement of the point cloud. To align the point cloud with existing registries, the points of origin will have to be replaced in the correct relative position compared to existing reference points in the map. In this way, the trajectory of the mobile laser scanner is adapted to fit the outlines of the map.

The positioning quality of the point clouds is the result of many different geometrical corrections. The data from positioning sensors (GNSS, IMU) is used to determine accurate positions. The IMU closely tracks the relative movement of the mobile mapping system, whereas the GNSS receiver uses satellite signals to determine its absolute position. The reference data from external high quality GNSS receivers is also used to limit the impact of GNSS data errors collected in the vehicle. Through combining all those positioning methods, the geometrical accuracy is very good, but not perfect. There will always be a discrepancy between the true (but unknown) and calculated position. No matter how many calibrations and corrections are conducted, the system remains subject to satellite errors, atmospheric errors and multipath errors. All together, these errors can have a serious impact on the calculated locations, leading to a deviation of a few meters or worse (Joosten, 2016).

3.2.2. Key registries

Besides this point cloud of Schiedam, several maps of the Dutch Kadaster will be used for georeferencing. The registries of addresses and buildings (BAG) and the registry of topographic maps (BGT) are accurate and up to date maps which contain objects that are well suitable for alignment purposes.

Buildings are assumed to be the most convenient objects to match with maps. Their walls appear in point clouds as dense vertical agglomerations and can therefore easily be detected in density histograms. The suitability of other map objects was tested as well. Trees, street lights and road surfaces turned out to be less suitable for map supported registration of point clouds. Maps of these objects are often more inaccurate or outdated and their shapes are harder to detect in the point cloud. For example, the more complex shapes of tree crowns are causing problems for successful object identification. As soon as their branches show overlap with other objects such as street lights, it becomes very hard to distinguish them and provide a way to separate the objects. Structures of building faces and roofs on the other hand, are rather smooth. Therefore, buildings are better suited for automated registration. It might be possible to include other map features later as well, but buildings are the most obvious for now.

Different versions of maps with buildings exist. Selection of the right version is important since they obey different definitions. When comparing the shape of a building in both maps, they can turn out to be different. This is possible since different rules of registration are applied. The registries of addresses and buildings (BAG) mostly use the outer circumference of a building as a starting point for the map. The registry of large-scale topography (BGT) on the other hand, uses the footprint of a building at ground level as the outline. Differences between the BAG and BGT occur especially for buildings with balconies, galleries, pillars and porches (Ministry of I & M, 2018).

Both BAG and BGT show high precision rates. Exact numbers on the data quality are not available for the specific research location of Schiedam. However, the law mandates source holders to persist a positional accuracy of at least 30 centimetres (Rossem et al., 2012). In real life, precision rates are much higher. Most buildings are measured with high precision land survey equipment. Those measurement devices measure distances with a general accuracy of a few centimetres. In Schiedam, the vast majority of buildings is mapped with survey equipment. According to the source holder of the land registry of large-scale topography in Schiedam, only a few remote buildings are acquired from aerial imagery (Dick Wijma, personal communication, June 28, 2018). Such locations

have a higher risk of being inaccurately mapped. However, since remote places are hard to reach for the mobile laser scanner as well, chances are low that these parts are included in the new tiling scheme.

3.3. Software

Different combinations of software packages are being used in this thesis. An overview of the most used software packages is shown below in **Figure 9**. For modification of the maps, ArcGIS Pro is used. This application is convenient since it allows the user to combine and edit maps and point clouds altogether in one view. Most of the point cloud inspections in 3D, are conducted with the help of Quick Terrain Reader. It is an adequate tool since it accepts many different file formats, while ArcGIS Pro requires some data conversions. For the actual processing of the data, Python is being used in combination with many different available libraries for geographic data, LAS files and image processing. The Open Source Computer Vision Library (OpenCV), for example, is one of those powerful libraries for image processing. In this thesis, it is being used to execute the template matching. While LAStools is a suitable set of tools to process LAS and compressed LAZ files, it will be used for creation of the smart point cloud. It offers a range of impressively fast tools for simple queries. For larger datasets, other solutions might be more suitable (Van Oosterom et al., 2015). But since the research region is relatively small, the program will be sufficient.

Figure 9: ArcGIS Pro, QT Reader, Python, OpenCV and LAStools are used for viewing, analyzing, preprocessing, programming and editing the data.

3.4. Hardware

The Digital Cyclorama Recorder system (DCR) of Cyclomedia consists of many different components. Below, the major parts with reference to the positioning accuracy are described. To create a 3D representation from the surroundings, the Velodyne HDL-32E is used (see **Figure 10**). It delivers about 700,000 points per second with a typical accuracy of +/- 2 centimetres. The usable range is up to 70 meters (Velodyne LiDAR Inc., 2013).

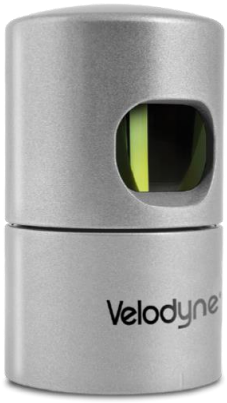


Figure 10: Cyclomedia's newest recording systems are equipped with the Velodyne HDL-32E High Definition LiDAR Sensor. It fires 32 laser beams with a frequency of thousands of times per second, providing a rich 3D point cloud.

Different types of GNSS receivers are installed on the cars, all of which are made by NovAtel. Pictures of the NovAtel GNSS antenna and iMAR IMU are shown in **Figure 11**. The IMU is triggered by the NovAtel receiver to ensure that all IMU measurements are synchronized with the GNSS time. Realtime positioning information is used for triggering the cameras every five meters. The raw data is also used in combination with reference data for post processing in the office.

Figure 11: To ensure high positioning accuracies, every DCR is equipped with an iMAR FSAS IMU (left) and a NovAtel GPS-702GG active antenna (right). Together, the IMU and GNSS are responsible for the absolute positioning.

4 Processing approach

This chapter describes the required processing techniques that are being used to registrate and develop the smart point cloud. **Section 4.1** provides a schematic overview of the processing approach. Thereafter, the pre-processing of the datasets is elaborated in **section 4.2**. **Section 4.3** explains how the datasets are being matched with each other through template matching techniques. In **section 4.4**, the process of rejecting unreliable matches is described. Only reliable map translations are implemented in the pose graph optimization, as explained in **section 4.5**. The chapter concludes with a description of the transfer of attributes in **section 4.6**.

4.1. Workflow diagram

As it is depicted in the workflow diagram of the algorithm in **Figure 12**, the processing approach consists of different phases. These research steps are related to different corresponding chapters within the thesis. In **chapter 2**, existing literature concerning registration and smart point clouds was discussed. Afterwards, the first step will be the creation of pairs of images from the datasets in a structured way. This is important for debugging purposes. One pair consists of a point cloud scene with a corresponding part of the map.

Both versions are converted into images so that they can be matched in an efficient way. As soon as the pairs of images are created, template matching will be applied on the created datasets. The correspondence matrix results in values for the calculated shifts with associated reliability indicators for each recording location. Only the reliable results are included in the pose graph optimization. Those values will be used to edit the point cloud eventually in such a way that it aligns with the map. In the end, attributes from the map can be transferred to the corresponding buildings in the point cloud.

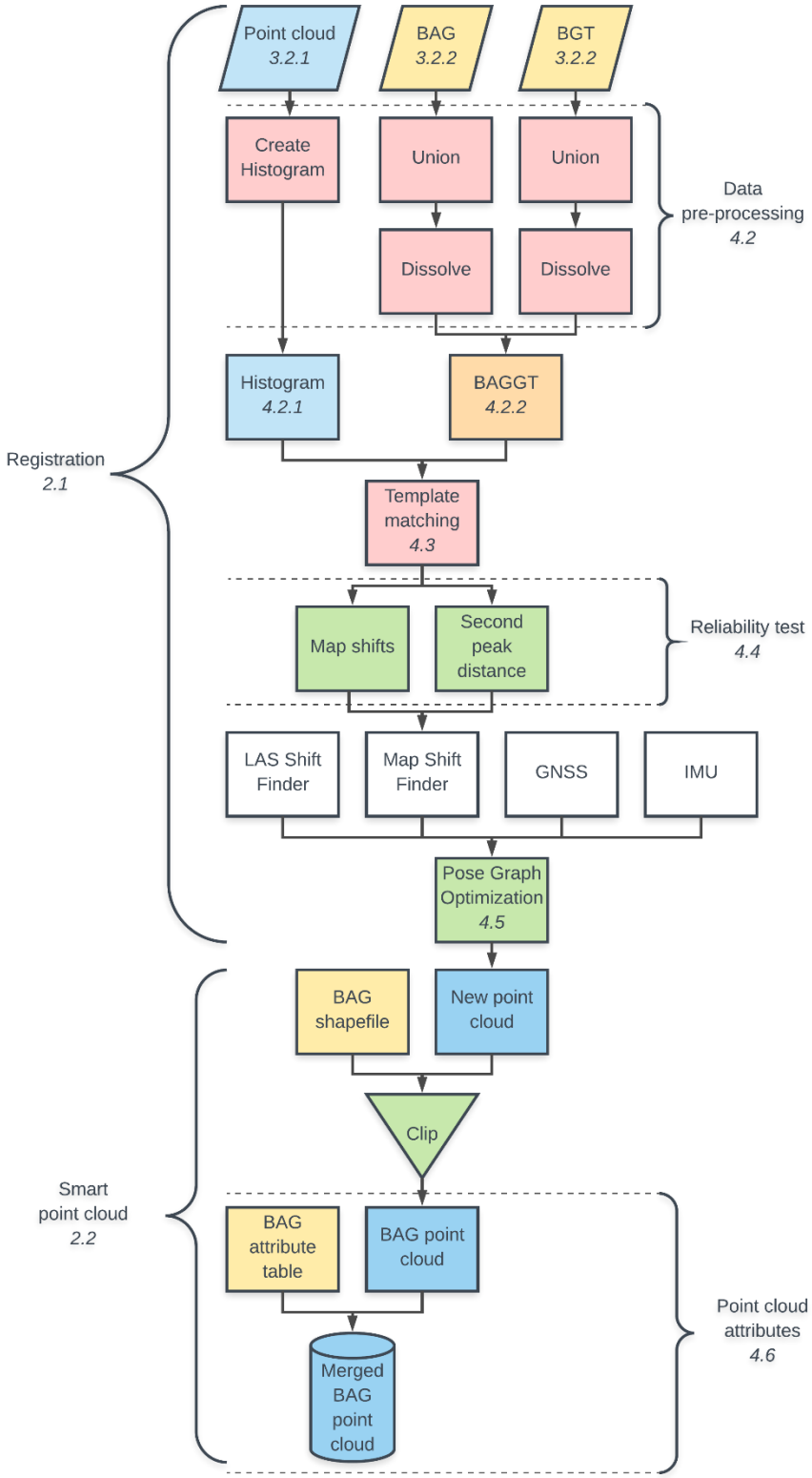


Figure 12: The methodology is depicted in one summarizing work flow diagram with corresponding chapter numbers. It describes all required research steps to achieve the registration and creation of a smart point cloud. The registration process consists of two main parts. First, shifts are calculated through template matching of point clouds with maps. Second, calculated shifts are used in the pose graph optimization. The resulting point cloud with improved positioning is than combined with the attribute table of the BAG. This smart point cloud contains separate 3D models of all buildings with corresponding attributes.

4.2. Data pre-processing

Before the template matching can be conducted, the data files must be prepared first. The map as well as the point cloud are edited with the goal of performing the eventual matching. The representations of the point cloud and map must be as much comparable to each other as possible to ensure high matching accuracies.

4.2.1. Point cloud histogram

The LiDAR dataset is divided into tiles of 2500 square meters. This tile size provides enough information to conduct the template matching, while at the same time preventing generalization of the found shift for a very large area. The point cloud tiles are created at an interval of 25 meters, meaning that the centres of the scenes are spaced 25 meters apart. While LiDAR points are continuously being obtained by the mobile scanning device, the recording positions of the panoramic images are more static than the actual path of the car. However, to simulate the actual path of the car as accurately as possible, the points are interpolated at a later stage. The three-dimensional point clouds are first converted into two-dimensional density histograms. These histograms are raster images that show the amount of points within each cell from a top view (**Figure 13**). For readability purposes, the point cloud density histogram images will from now on be referred to as **density histogram.**

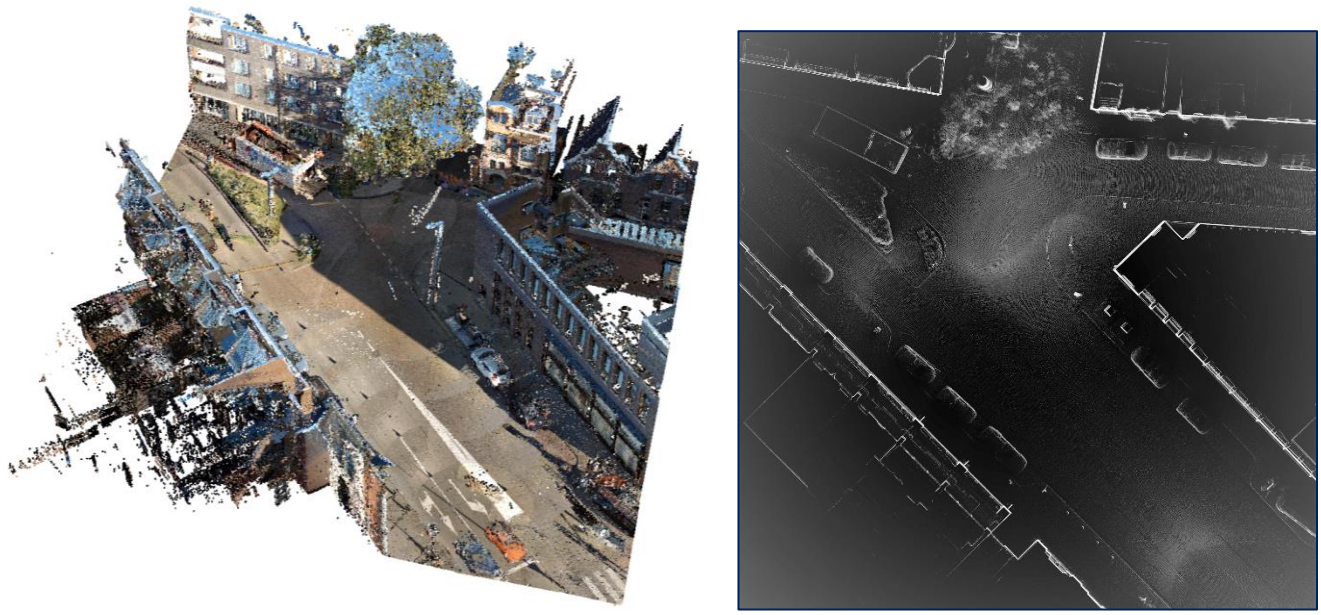


Figure 13: Each point cloud tile (left) is converted into a two-dimensional histogram (right). In the histogram, vertical agglomerations of points appear in the image as a white pixel. This conversion is necessary since it allows to compare the three-dimensional point clouds with two-dimensional maps. The newly created point cloud density histograms have a width and height of 50 meters and a pixel resolution of 2.5 centimeters. The image consists of white pixels on a black background since this color-scheme is easier to process for computer algorithms.

The source LAS files are organized in folders according to their geographical locations. To be more precise, they are divided in a grid with tiles of 50 by 50 meters. Within these folders, different versions of the same environment may exist. Separate LAS files are created with different timestamps if the mobile laser scanner drives through the same tile multiple times. Since those point clouds may have differing positioning errors, they must be treated separately in the density histograms as well. Therefore, the point cloud images are created from a certain selection of points rather than the entire file(s) in the folder. The selection of relevant points is automated in a Python script. In short, it iteratively selects all points recorded within a time range of 6 seconds before and after the exact timestamp of the recording location. Hereby, the central assumption is made that the internal deformation is limited within this short time interval.

4.2.2. BAGGT

Two different registries of buildings will be combined in one new map image, renamed the BAGGT. This newly created map is a combination of the Dutch registries of buildings and addresses (BAG) and the Dutch registry of large scale topography (BGT). In most cases, the building outlines in the BAG and BGT look similar. In some cases, however, the building outlines can differ from each other since the method of acquisition differs as well. The BAG is extracted from aerial imagery. In most cases, this means that buildings are defined by their outlines as they can be seen from above. The BGT on the other hand, defines a building by looking at the ground level footprint. An image of the differences between the two maps in show below in **Figure 14**.

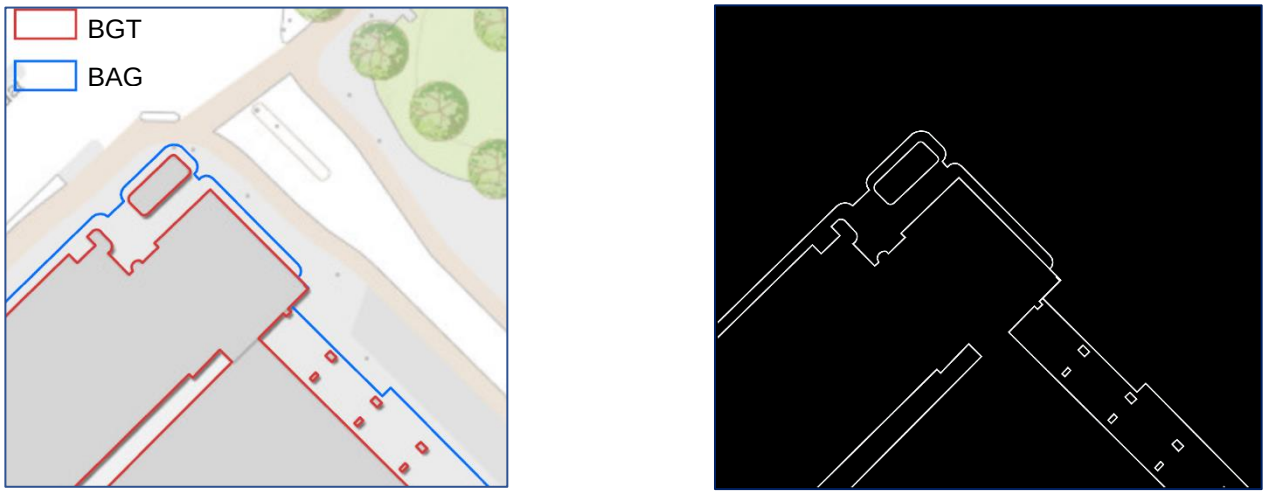


Figure 14: The BGT defines buildings by their footprints whereas the BAG uses the outlines of a building based on the top-view of aerial imagery (left). Since both footprints and rooftop prints are visible in the point clouds, the two different registries will be combined in one new view, the self-appointed BAGGT (right). Again, this image is plotted in white pixels on a black background to ease the template matching process.

Differences between the BAG and BGT occur especially for buildings with balconies, pillars and porches. Since all such features of a building will be visible in the point cloud, it is better to match the point cloud to both maps at the same time. This is likely to improve the accuracy of the template matching algorithm. Besides combining the two different registries in one new map, the building outlines for each of the registries are edited as well. This adaption of the outlines is automated in ArcGIS Pro. The steps taken to edit the input data are described in further detail in the appendix (see **APPENDIX I: Pre-processing workflow**). The mobile laser scanner is unable to scan the walls between contiguous neighbouring houses. Such unreachable walls are being removed from the map since they would increase the chances for false matches.

After all data preparations have been conducted, two main inputs are created for the template matching. The point cloud is transferred into a two-dimensional histogram of the point density from a top down view. The maps are combined in one new view and their buildings are reduced to their outer walls only through the removal of inner walls. For the readability of this thesis, the combined image of the registries of buildings and addresses (BAG) and the registry of large scale topography (BGT) will from now on be referred to as **BAGGT image**. In the BAGGT, only the buildings from the BGT are included. Note that the original BGT also includes many other mapped assets and objects such as roads, waterways, railways and greenery. Those objects are excluded since they are harder to detect, as was described previously in **section 3.2.2**.

4.3. Template matching

The newly created density histograms and BAGGT images are input for the template matching. This method uses a bigger template image to locate a smaller image that is similar to a part of the template. In other words, a source image is shifted and compared to the template image to detect the area with the highest matching score. In this case, a patch of 50 by 50 meters of the point cloud histogram is shifted over all possible positions of the larger BAGGT template image of 70 by 70 meters.

The BAGGT image, as a reference point, will remain in a fixed position. The density histogram is shifted in such a way that it matches with the map, as shown in **Figure 15**. On purpose, a slightly bigger extent is chosen for the BAGGT. In this way, the template can change the projected coordinates without crossing the borders of the map. Computer algorithms can solve template matching issues in a structural way with high accuracies (Yoo, Hwang, Kim, Ki, & Cha, 2014). In this thesis, the resolution of the template matching will be 2.5 centimetres. This has specific consequences for the accuracy of the entire process, which will logically not exceed 2.5 centimetres as a result. However, higher resolutions are unnecessary since they would transcend the accuracy of topographic maps as well as mobile laser scanners.

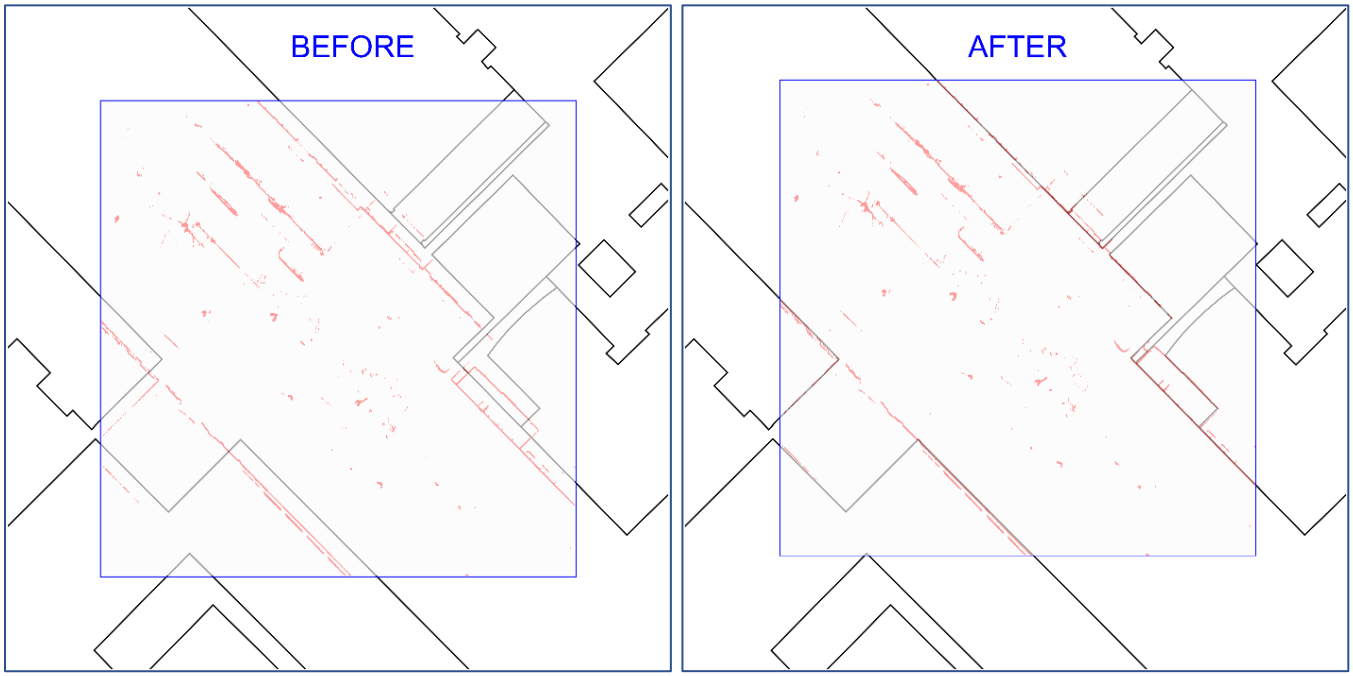


Figure 15: The point cloud tile (in red) is aligned to its corresponding BAGGT image (in black). Before the template matching is conducted (left), the spatial difference can count more than 2 meters. After the template matching (right) the density hisogram should seamlessly fit the BAGGT image. The outer map has a width and height of 70 meters, while the density histogram of the point cloud is 50 meters wide and high. In this way, the algorithm is given enough clearance to effectively shift the point cloud.

Different methods for template matching are available in OpenCV, the Python library which is used for image processing. After comparing the available methods, cross correlation (see **Equation 1**) was selected as the most suitable one. The formula for cross correlation calculates where the source image should be placed in the template image to ensure the highest possible amount of overlap. The output consists of pixel coordinates in the template image that indicate the required position of the source image's upper left corner.

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x', y+y'))$$

Equation 1: For the template matching, cross correlation is used to calculate the amount of overlap for each pixel position in the template image. All correspondence values are saved in the result matrix (**R**). **T** is the template image and **I** is the source image. Each location (**x, y**) in the result matrix contains a value representing the correspondence between the template and source image. This value is calculated for each location (**x', y'**) in the template image.

This equation is used as a function to automatically detect the shifts between the BAGGT and density histograms. The computer uses the architecture of Fast Fourier Transform (FFT) to convert correlation processes into arithmetic operations. This reduces the computational complexity and plays a critical role in the efficiency (Chang & Ma, 2017). In brief, the computer decomposes an image into its sine and cosine components. The efficient algorithm transforms data from the spatial domain into the frequency domain (Drongelen, 2007). As a result, the equivalent process of dragging the source image to every possible location in the template image becomes much faster. The method quickly returns a correspondence value for every possible position in the entire picture.

The entire template matching is automated for all tiles in the research location. A Python script is written, iterating over all recording locations in the city of Schiedam. The pseudocode is provided in **Figure 16**. The entire developed script consists of almost 400 lines of code and is provided in the appendix (see **APPENDIX II: Python script**). The main idea behind the Python script is that the required shifts to align the data are calculated first. After those shifts have been applied, the buildings will be extracted from the point cloud. Folders are created for each recording locations. In these folders, all relevant output data is stored to permit for monitoring and conduction of visual checks. Besides that, a file with Comma Separated Values (CSV) is created in which the data is saved and analysed later. For example, a reliability indicator is calculated and stored in this CSV. This reliability indicator is used later in

combination with a unique identifier of the position to exclude unreliable shifts from the pose graph optimization, as will be explained in the following chapter.

```
for each recording location:
        create folder
        create map:
                for 70m x 70m extent:
                        draw BAG shapes
                        draw BGT shapes
        create histogram:
                for 50m x 50m extent:
                        select adjacent LAS tiles
                        select all points within a margin of 6 seconds
                        make point density histogram
        match map template with point cloud image:
                calculate X coordinate shift
                calculate Y coordinate shift
                calculate reliability
                save all information in CSV
        apply shifts from CSV to original LiDAR data:
                use reliable map observations as input
                shift tiles using pose graph optimization
        clip shifted point cloud to map polygons:
                create LAS file for each separate building
                set LAS attributes
                        save year of construction in intensity field
                        save building ID in classification field
        merge all separate buildings into one file
```

Figure 16: The Python script makes use of template matching to align the point cloud with the BAGGT. Calculated local map translations cannot be applied on the dataset without taking the neighboring tiles into account. Therefore, the translations are implemented in Cyclomedia's existing software for pose graph optimization. This script aims to simulate the trajectory of the mobile laser scanner as accurately as possible while preventing inconsistencies within the point cloud at the same time.

4.4. Reliability test

As soon as the translations between the BAGGT and corresponding density histogram have been calculated for every tile, the point cloud can be shifted to match the outlines of the buildings in the map. But before the point cloud is shifted, correct matches must be separated from the false matches. To distinguish them, a certain threshold is applied to the outcomes of the template matching. This threshold is currently set at a normalized difference of 0.14 between the first and second peak. The establishment of this peak distance indicator is a decisive factor for the eventual results and will therefore be explained in the next paragraph.

As shown in **Figure 17**, the reliability of a match is derived from the response image. The pixels in the response image contain a correspondence value for every possible overlay between the BAGGT and density histogram. The distance between the real centre of the image and the pixel with the highest value gives an indication of the required shift that is needed to match the density histogram with the BAGGT. However, the suggested new position of the point cloud can sometimes be doubtful. The images can contain many potential positions with high overlap, such as is the case in the lower images of **Figure 17**. If this is the case, there is a high chance that the template matching method will select the wrong match. Therefore, the suggested shifts for these locations are rejected.

The highest and second highest peak are subtracted from each other to give an indication of the peak clarity. If the difference between those two values is small, it can be assumed that multiple sharp peaks exist. Another possibility is that the first peak is fuzzy and the match untrustworthy. In any of those instances, the match should be rejected. The peak difference value is a normalized number ranging from zero to one. Unfortunately, a threshold which accepts all true matches and rejects all false positives does not exist. Since it is important to know for certain that the found matches are reliable, the threshold is relatively high and thereby excludes some correct matches. For the determination of the threshold, many different matching results and response images were visually checked. Conducted samples proved that the threshold of 0.14 results in rejection of almost all false positives without removing too many true positives.

Unfortunately, it is impossible to empirically justify the correctness of a certain value for the threshold. The only method for selecting the most appropriate value is through trial and error. Different thresholds have been applied on the data. Lower thresholds turned out to approve too many false matches and higher thresholds turned out to

decline too many correct matches. The eventual threshold of 0.14 allows for an appropriate number of reliable shifts while at the same time only accepting one obvious false match. This single outlier is discussed in **chapter 5**, where the results are shown.

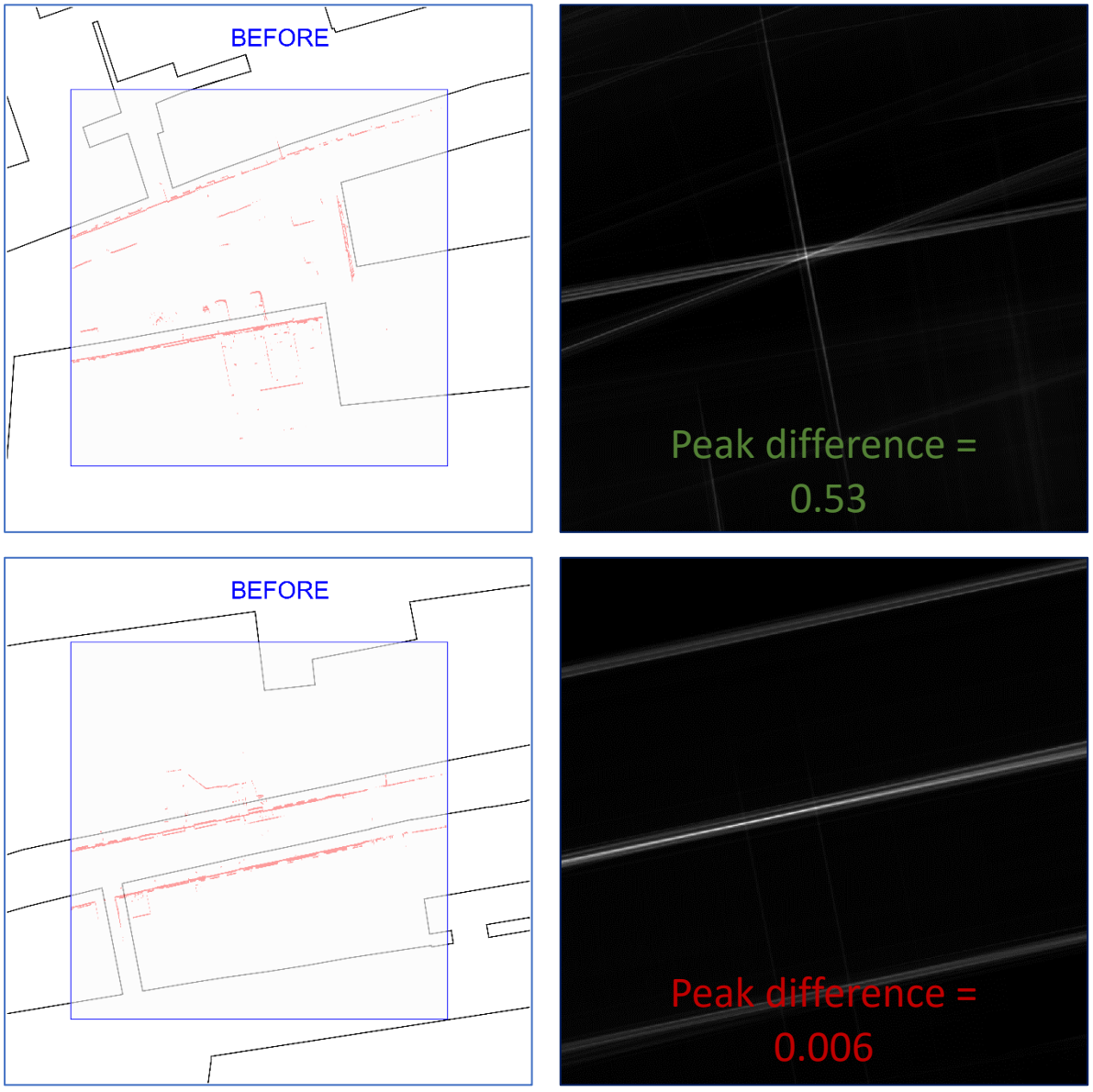


Figure 17: Overlap between the map (black) and point cloud (red) is shown twice on the left. The response images from the template matching are shown on the right. The upper right image shows one clear peak. Therefore, the normalized difference in value between the first and second peak is rather high (0.53). The lower right picture shows many points with high resemblance values along a central horizontal axis. This results in a small normalized difference between the first and second peak value (0.006). For this location, the peak is ambiguous and the found shift will therefore be rejected.

4.5. Pose graph optimization

After the removal of unreliable matches, the remaining translations are implemented during the pose graph optimization. This method takes several parameters as input to find the optimal configuration of the different tiles. Hereby it shifts the point cloud tiles while considering the GNSS/IMU log, trajectory intersections and map translations. It tries to adhere to the suggested shifts as much as possible without provoking irregularities or gaps in the data.

The template matching is valid for the registration on a local scale. For each recording location and its corresponding tile, it will calculate the shift that is required to align that specific tile of the density histogram with the BAGGT image. However, the entire point cloud must match with the entire map as well, without creating irregularities between the different tiles. Therefore, the global registration aims for improvements in the path of origin rather than matching separate point cloud tiles with local parts of the map. This is where the Simultaneous Localization and Mapping (SLAM) algorithm enters the scene. The separate tiles can be considered as nodes in a graph. The SLAM algorithm will try to find a configuration of the nodes that shows the least possible amount of errors in the data. In other words, it will reposition the virtual path of the car in such a way, that the point cloud shows as much similarities with the map as possible.

The process of finding the most optimal configuration for the car positions is called **Pose Graph Optimization** (PGO). The different positions of the car are considered as nodes in a graph. Different measurements have an influence on the estimated positions of the car, such as the recorded GPS locations, calculated IMU movements and the required shifts to match the density histogram with the BAGGT. Those variables are all implemented in the pose graph optimization as constraints with different weights. The weights can be determined based on reliability indicators such as standard deviations and error estimations. In the end, two different configurations of the pose graph will be created. The first graph determines the point cloud's positioning as it is currently calculated in Cyclomedia's solutions. The second output will include the map shift finder besides the existing positioning constraints.

4.6. Point cloud attributes

As soon as accurate registration to the BAGGT has successfully been conducted, the resulting point cloud will fit better with the map. The final step is to create a new smart point cloud with attached knowledge from the corresponding topographic map objects. Each separate building will be linked to a collection of points that belong to the building. In this way, a 3D model for each building becomes available together with as well map attributes (e.g. building year and height) as point cloud attributes (e.g. acquisition date and RGB value). This linked dataset will be created in a new database.

Since the BAGGT and point cloud are now showing an almost perfect overlay, it should be possible to transfer the attributes form the former to the latter. This process consists of several steps. In short, the first step is creation of separate point clouds for each building. Thereafter those point clouds are assigned identifiers from their corresponding object in the register of buildings and addresses (BAG). A database is built with a linked table in which attributes from the BAG are connected to their associative point cloud. In this database, it will be possible to query the point clouds and select or view buildings with a certain specified attribute.

A smart point cloud does not only imply a distinguish between buildings and other classes, it also enables identification of separate buildings or houses. To make this possible, the point cloud must be separated into separate collections of points for each building first. This is done through splitting the registries into separate shapefiles for each individual building. The model with a sequence of different required actions to achieve this is included in the appendix (see **APPENDIX III: Process to split the BAG in Modelbuilder**). With a buffer of 20 centimetres around those shapes containing the outlines of the buildings, the point cloud can be classified. This classification consists of assigning the corresponding building identifier and year of construction to the las attributes.

This transfer of attributes is conducted in an indirect file-based manner. A more direct, and perhaps more beautiful way to do this would be through the addition of actual new custom attributes to the LAS file. This is possible in the version of the LAS 1.4 file format (ASPRS, 2013). However, visualisation of those extra attributes in the LAS 1.4 files is problematic. Thus far, tools for interactive visualisation of custom attributes are lacking functionality. For example, existing widely used tools such as ArcGIS Pro, CloudCompare and Quick Terrain Reader are only supporting usage of well-known fields like the intensity, return number and RGB value. The current file-based

method is an intermediate step which has been decided upon for compatibility reasons. Soon, after essential developments in 3D processing and visualisation, custom made viewers should be able to visualize true new point cloud attributes as well.

The last step is to store the new smart point cloud in a structured manner. The extracted matches will be given unique object identifiers which can be used to link the buildings to their corresponding point cloud. Many separate collections of point clouds are being created in one database. Attributes from the map are attached to the separate files. In this way, a smart point cloud is created which contains attributes as well as the three-dimensional shape of the building in colour.

5 Results

In this chapter, the outcomes of the afore mentioned methods and techniques will be discussed. The results are based on a prototype implementation of the suggested workflow, which is tested in the research location of Schiedam. First, results from the template matching are analysed together with their influence on the registration in **section 5.1**. Then, the reliability of those results is discussed in **section 5.2**. Eventually, **section 5.3** shows the transfer of attributes from map to point cloud.

5.1. Map shift finder

The template matching was conducted for 2.447 recording locations (and their corresponding tiles) in the city of Schiedam. Since the template matching is calculating shifts between the BAGGT image and density histogram, it can be called the 'Map Shift Finder'. The image below (see **Figure 18**) shows a map with the calculated discrepancies between the point cloud and map. As expected, densely built areas with more high-rise show bigger discrepancies between the BAGGT and density histograms due to the poorer GNSS reception conditions. Many recording locations (in grey) have been rejected. This occurs especially in locations where the corners of a building are not included in the BAGGT image. Those unique outlines in the map are needed to result in one clear peak in the result matrix. Locations with big buildings and long straight lines show multiple peaks with similar values in the response image. Such tiles are rejected due to a lack of corners of walls or other distinctive shapes in the point cloud and BAGGT image.

Corners are important since they allow for one single clear peak in the response image. The presence of any building in the map is a requirement to initiate the template matching. However, as soon as the corners of the building are not included in the tile, there will be no clear match. Instead, the correspondence values will be high along the entire axis of the wall. Situations as such occur quite often, buildings or uninterrupted building blocks of more than 50 meters are not rare. But since all buildings do have edges, the mobile laser scanner will pass by the

landmark at some point. For this very reason, the map clearly shows that matches are less frequently rejected near intersections, where building corners are often located.

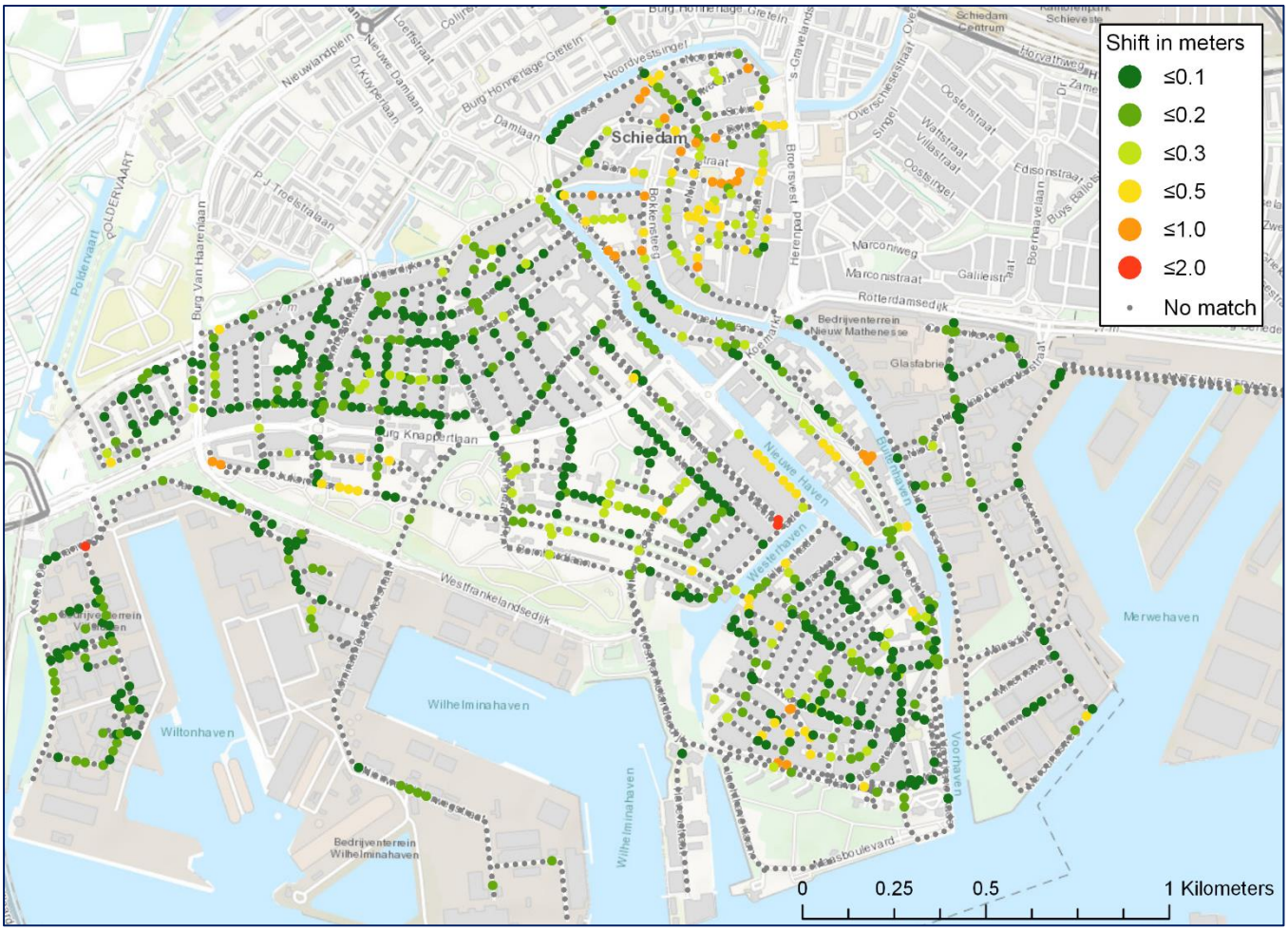


Figure 18: In most parts of the city, the point cloud seems to be quite coherent with the geometry of the map. Some other areas are showing a euclidean difference between the map and point cloud of almost two meters. The map also shows that locations with few and large buildings, such as the industrial area in the South-West, are often rejected. Those locations are considered unreliable due to a lack of building corner walls in the map. Corners of buildings are the most important landmarks for the template matching to base the registration on.

The map can be interpreted as an overview of the locations where the map supported registration is required for creation of a smart point cloud. Suppose that buffers are created around the existing building blocks and all LiDAR points within the buffer are assigned with the corresponding BAG-id. In that case, wrong values would be assigned to all attributes from points near red and orange locations on the map.

The biggest shift is located quite centrally in the map. At the red dot, by the water, a true shift of approximately 2 meters was discovered. This proposed solution was correctly detected, meaning that the point cloud in that tile was indeed positioned in the coordinate system with an accuracy of less than two meters. The second biggest discrepancy between the point cloud and map was falsely detected in the far western part of the map. Through visually checking the location, it turned out that the building consists largely of glass. Since the LiDAR system is unable to correctly depict glass shields, the registration was conducted based on the wrong wall. All other accepted matches shown above were visually checked as well. Luckily, no other matches false matches were found.

Figure 19 shows the differences between the former Pose Graph Optimization (without map supported registration) and the newly conducted PGO (with map supported registration). Again, the false match in the western part of the map is clearly visible in the figure. Here, the map shift finder experienced some issues with a building that largely consists of glass, as is explained in **Figure 20**.

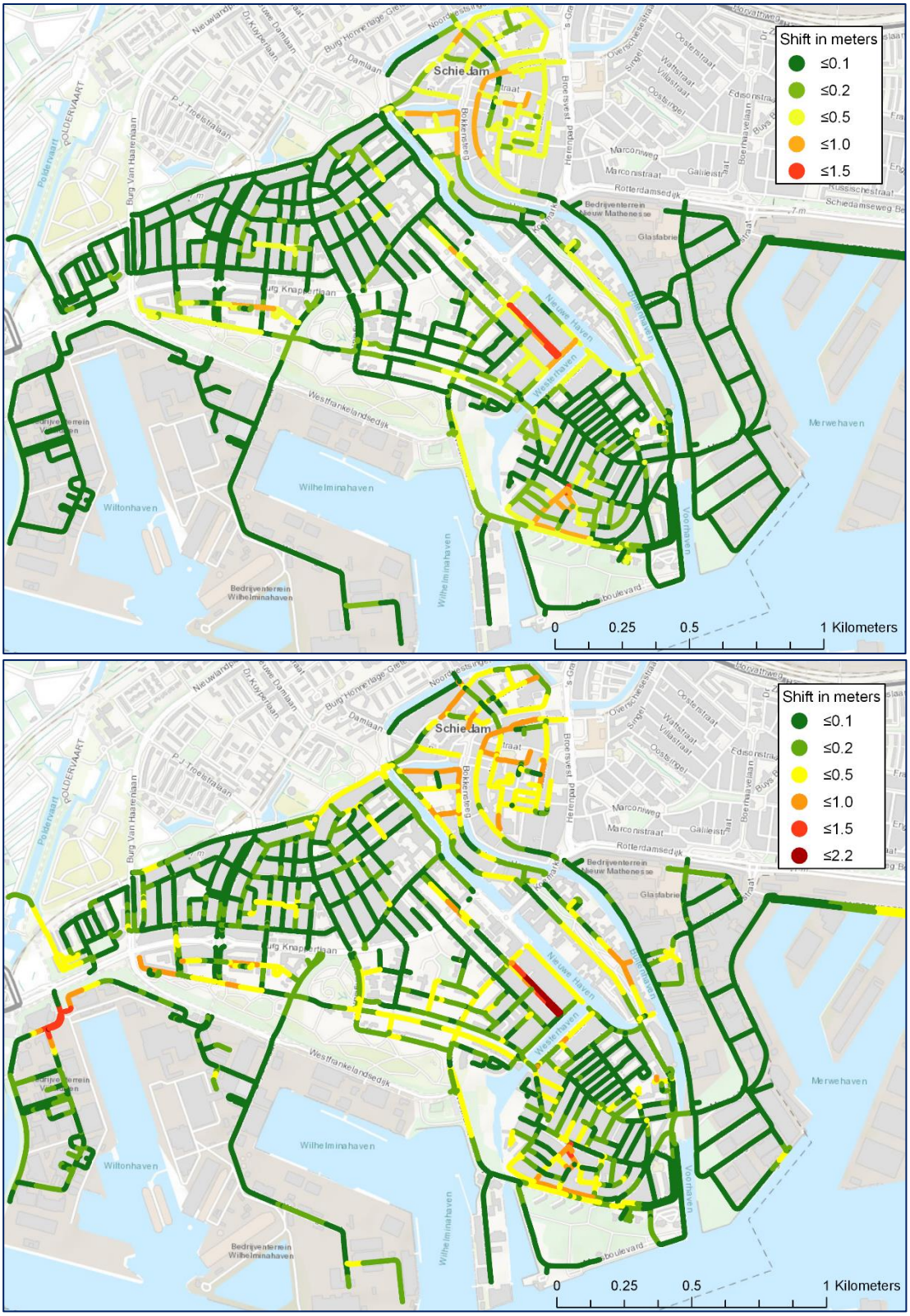


Figure 19: The upper image shows the shifts calculated with Cyclomedia's current method for positioning improvements. The lower image also takes the newly created map shift finder into account. In general, the two methods seem to agree on the spatial distribution of shifts. Only, the map supported pose graph optimization indicates that the positioning errors are generally underestimated by the former method.

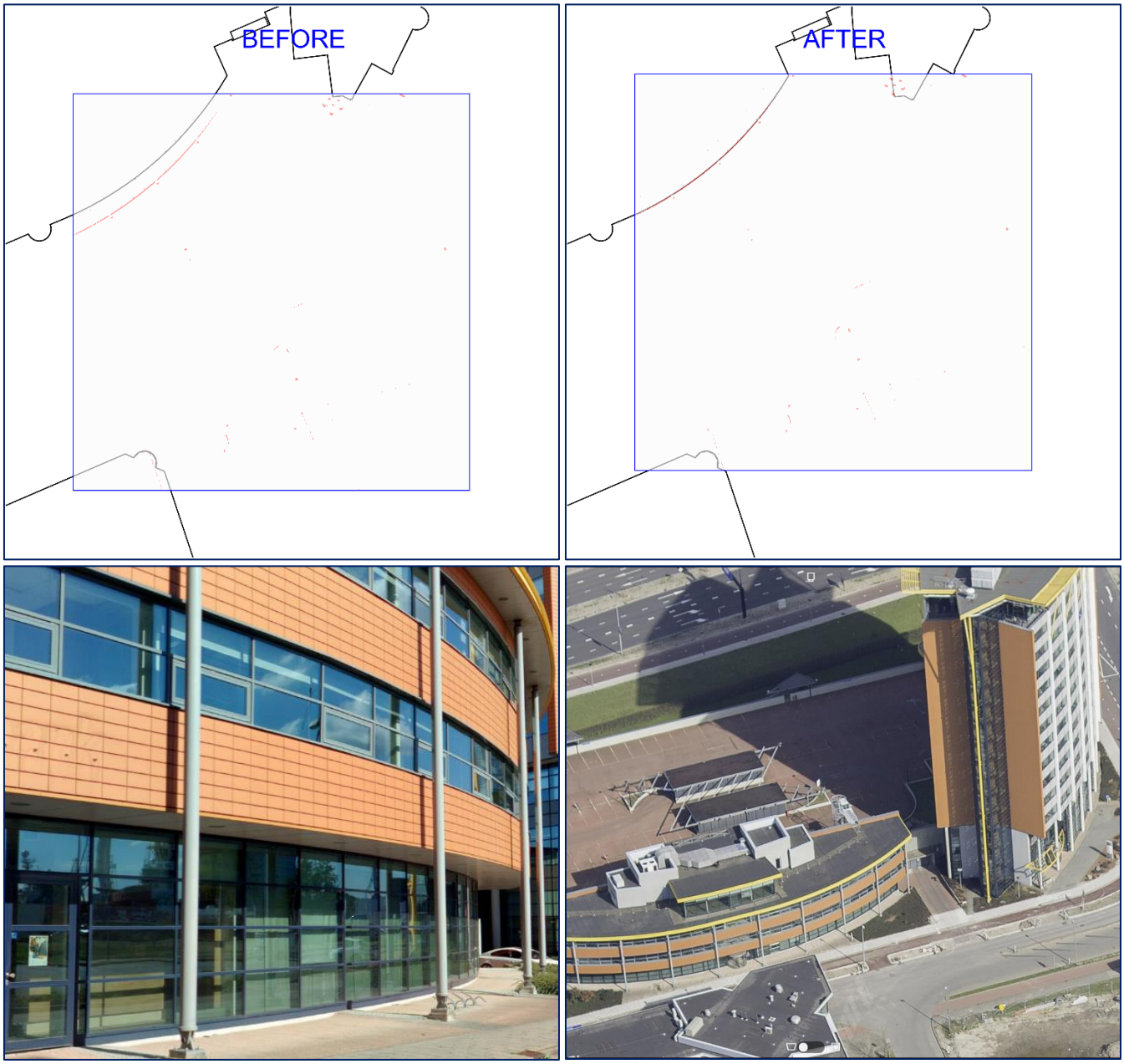


Figure 20: Buildings with lots of glassware create difficult source images the template matching. Glass is poorly recorded by LiDAR systems. In the specific case shown above, the footprint of the building is significantly different from the rooftop print. Unfortunately, both the BAG and BGT based their registration on the footprint, which consists largely of glass in this case. Therefore, the map supported registration fails by matching the impermeable bricks with the footprint.

In **Figure 21**, the differences between the registration with and without the map shift finder are mapped through extracting them from one another. Besides the falsely matched westernmost red part of the map, few big differences are visible. In most of the locations, the map supported registration seems to agree with the initial registration method. While conducting several checks on the orange and yellow parts of the map, no registrations errors were detected. Therefore, it can be assumed that the positioning of the point cloud is improved at those locations through shifting the points up to 90 centimetres. In other words, the former method fails to correct some positioning errors accurately enough with error underestimations of up to 90 centimetres.

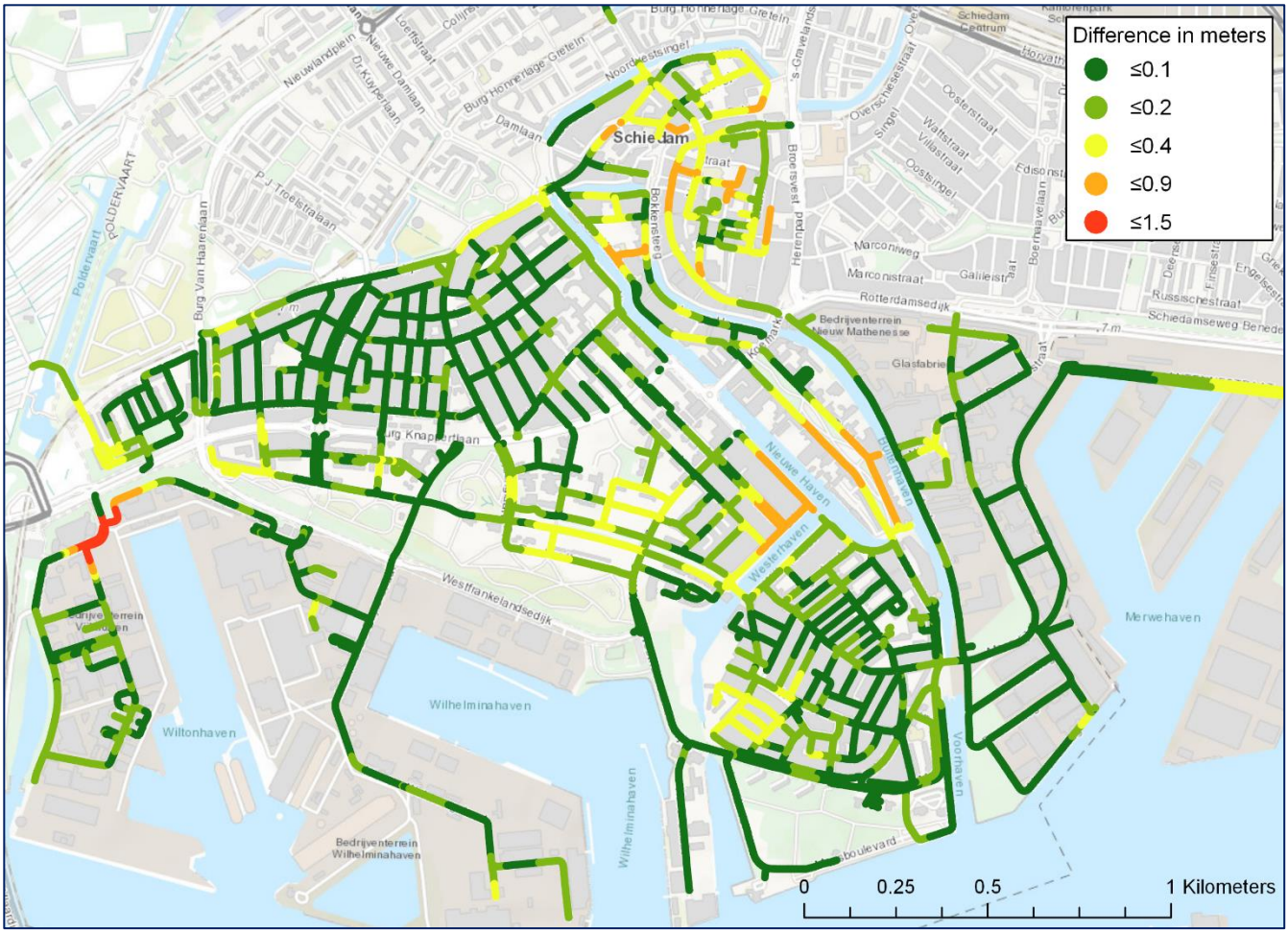


Figure 21: Whereas the old method for positioning included GNSS / IMU and point cloud shifts at intersections, the new method also includes map supported registration. In 82 percent of the cases, the difference between the new and old configuration is less than 20 centimeters. This can be considered as a confirmation of the reliability of both methods.

Besides that, the maps show many similarities. In this way, the methods are confirming each other's reliability. In general, the map shift finder shows bigger shifts than the initial pose graph optimization. It seems that big shifts are automatically tempered or moderated during the original optimization. Since the map shift finder clearly detects those areas where the discrepancies are underestimated, it can function as a valuable replenishment to the method. The map shift finder was implemented with a Dell Precision M6800® computer, 2.8 GHz quad-core, Intel® Core™ i7 processor, 16 GB RAM memory.

With the reported performance metrics, the algorithm needs approximately 26 hours to compute all calculations and implement the shifts for the research location. The total run time could be significantly decreased through the implementation of multi-threading. Processing time could also be decrease through making use of the computing power of a Graphics Processing Unit (GPU). For the current research location, a Central Processing Unit (CPU) was sufficient. However, the computing time could easily be improved if this is aimed at.

5.2. Reliability

Many unreliable matches are rejected, as shown in the previous chapter. This chapter will explain the reliability of the results in further detail. The distance between the first and second peak gives an indication on the reliability of the template matching.

As shown in **Figure 22**, the relative number of accepted shifts is rather low. Approximately one out of four suggested translations are accepted after application of the threshold. On purpose, a relatively high threshold was applied. This was decided upon since it is important to be sure that the surviving matches are reliable. It is more valuable to have a low number of matches with high reliability than the other way around. As explained before, the interval of calculating the differences between the BAGGT and density histogram along the path of origin does not have to be very high since drifts are occurring and fading out gradually. Including more recording locations would result in a higher resolution, but the overall advantage in respect of interpolation would not be very high.

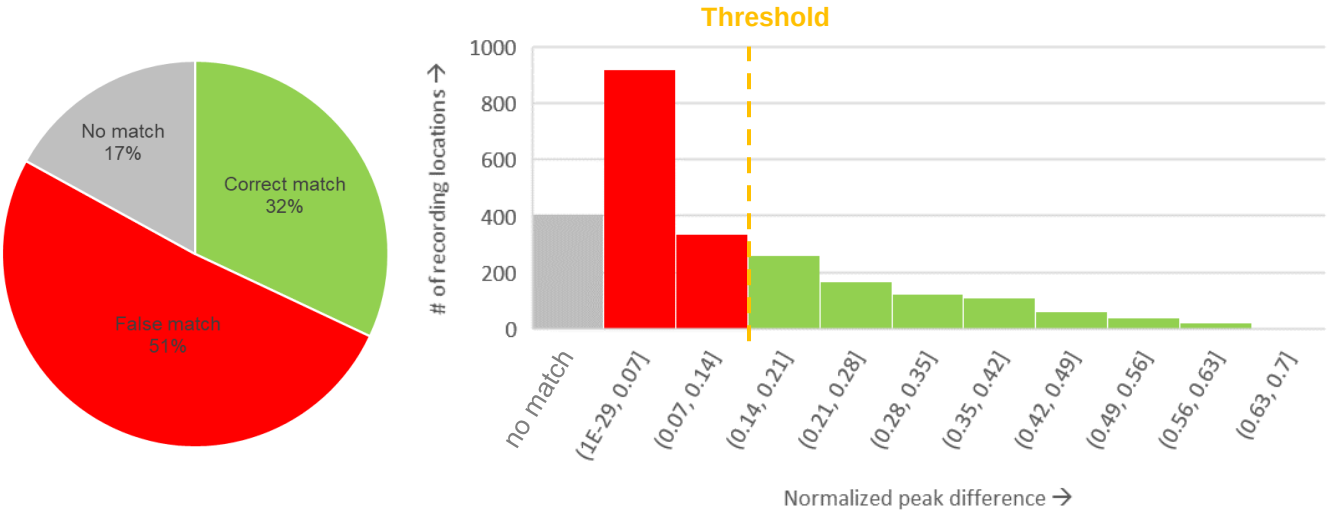


Figure 22: A match is only accepted if the difference between the two most probable matches is big enough (more than 0.14). From the 2.447 recording locations in total, 407 locations were showing no match at all. This occurs when a pair of images does not contain any buildings in the BAGGT at all. In 1.254 cases, the match was considered untrustworthy after comparing it to the threshold. Eventually, only 786 matches were accepted and used to align the point cloud with the map. Percentagewise, this seems to be a low matching accuracy. However, since the research location is rather small, this absolute number of reliable matches is sufficient for the registration to be conducted succesfully.

The example below (see **Figure 23**) clearly illustrates the importance of rejecting cases with multiple high peaks. In this example, the map shift finder is unable to make the correct distinction between walls. In occasions as such, the overlap remains almost similar when the wall in the point cloud is aligned with an incorrect wall in the BAGGT. Luckily, the reliability test indicated a normalized difference between the first and second highest peak of 0.06 for this specific location. Since 0.06 is below the threshold of 0.14, the found match for this specific tile is rejected. This is not a big problem since the template matching for surrounding tiles was finished more successfully.

Figure 23: The point cloud (in red) was perfectly aligned with the map (in black) before the template matching was conducted, as visible in the upper left image. However, after the template matching, the point was aligned with the wrong part of the map, as visible in the upper right image. In this specific case, different walls with different heights cannot be distinguished from each other. Luckily, in dubious cases as such, where two different shifted positions of the point cloud show similar overlap, the found shift is rejected due to the small peak difference.

Still, not all false matches are being declined after the threshold is applied. In some rare occasions, false matches are mistaken for a reliable shift. An example as such is given below in **Figure 24**. In this rare case, very few buildings are included in the density histogram, whereas they do appear in the BAGGT image. The buildings are located quite far from the road, while high hedges are separating them from the mobile laser scanner at the same time. Through this unfortunate concurrence of events, the dense hedge in the point cloud is mistaken for an actual building. As a result, the point cloud is shifted entirely to a wrong position.

Figure 24: Due to the absence of dense buildings in the point cloud in combination with the presence of a rather high hedge, the natural fence is mistaken for walls of a building. In this rare case, the false match is accepted since the difference between the first and second peak in the result matrix is high enough.

Since few other dense objects are present in the point cloud, the distance between the first and second peak is not high enough to exclude the recording location from the eventual results. To be more precise, the difference between the highest and second highest correspondence pixel in the response image is 0.23. Whereas only values below the threshold of 0.14 are being rejected. However, the pose graph optimization manages to reject most extreme shifts as the one shown above. Especially if it is very inconsistent with neighbouring measurements. It is also important that the location is well connected with other parts of the map. This prevents the algorithm to allow for one major shift that does not provoke further inconsistencies.

5.3. Smart point cloud

Now that the entire point cloud is aligned with the map, the next step is to transfer attributes from the map to the point cloud. This is done through the creation of buffers around the map while clipping all points within this bounding box. After the points have been grouped according to their corresponding building identifier from the map, attributes such as the year of construction are attached to the collections of points. In the end, a new database of LAS files is created with separate collections of points for each building. In this database, point clouds can be queried based on their year of construction, ownership type, address or surface area.

First, all points belonging to buildings are separated from all other points (see **Figure 25**). For this step to be successfully conducted, it is crucial that the registration was done precisely enough. Buffers of 20 centimetres are created around the original BAG footprints to make sure that all relevant points are included in the 3D model of the separate buildings. The buffers are ensuring that all recorded walls are included in the clipped point cloud. Unfortunately, (parts of) balconies can be excluded from the new point cloud. The BAG represents the outer circumference of a building in most cases, but unfortunately not in all cases.

Figure 25: The initial point cloud (left) is being restricted to the spatial extent of the registry of buildings and addresses. Hereafter, cars, trees and road surfaces are automatically extracted from the point cloud (right). The trajectory of the mobile laser scanner is always located in the street at a height of approximately two meters above ground level. Due to the limited line of sight from this point of origin, rooftops and backs of buildings are scarcely present in the visualizations.

The new smart point cloud does not only enable users to see more information in the 3D model, it also improves the usability through decreasing the point cloud size. Since a lot of noise from cars and trees is being removed, the total size of the point cloud has decreased by approximately 65 percent for the entire research location. Of course,

reduction of the size depends greatly on the building density and total size of other objects in the map. However, through smart filtering options such as illustrated here, many obstructions with regards to the usability of point clouds can be overcome. Hierarchical filtering allows for showing only those objects in which the user is interested, without compromising on the level of detail for the visible parts in the scene.

Such filtering and dynamic viewing of LiDAR data is hardly possible in available viewing tools. Currently, only standard attributes can be visualised whereas visualization of custom attributes is lacking. At the same time, it is hard to filter points out based on their class or attribute. Simplification of the data is possible only through limiting the point density of the entire point cloud. In this way, the level of detail is reduced for the entire point cloud. Generalization of the data then makes it easier to handle large files, but the quality is reduced, and details are faded out. It is not possible yet to make certain choices regarding the desired visible objects which are of main interest for the user. For instance, it would be better to allow for multiple levels of detail within one view. This could be achieved through showing more detail for points which are nearby and gradually decrease the level of detail for points that are further away from the viewer.

Extraction of identified objects is not only important to clearly indicate differences between buildings and the rest of the environment. The same type of clipping can also be used to distinguish separate buildings from each other. If the registration is conducted accurately, and the bounding box of the BAG fits the model well, the separate buildings can easily be identified. The impact of false positioning is visible in **Figure 26**. As soon as the point cloud does not

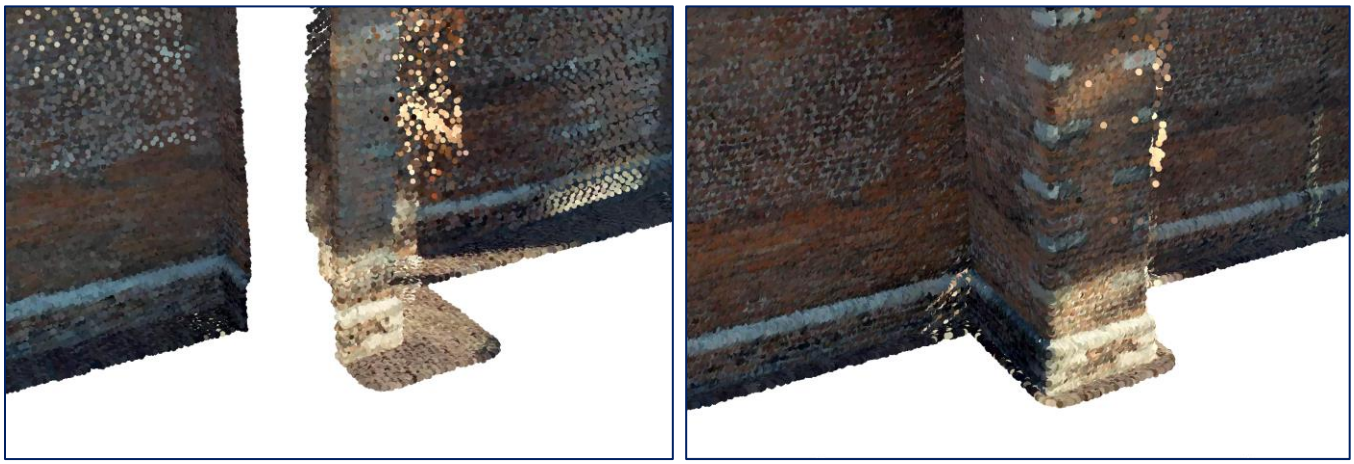


Figure 26: Bad registration (left) results in gaps in walls. However, after template matching with the map, the model fits the outlines of the buildings (right). In this way, the buildings can be extracted without missing out on any points belonging to the object.

completely align to the underlying map, parts of walls will be missing in the final point cloud. In this way, gaps will be visible in the walls of buildings.

After the extraction of buildings from the point cloud, the collections of points can be assigned their corresponding map attributes. The visualization below (see **Figure 27**) shows different identified buildings in a point cloud scene. The viewer does not allow the visualisation of new custom attributes. Therefore, the existing field for class was overwritten. Normally, this attribute is used to visualize different classes (e.g. building, vegetation and water). Here, different colours are being used to identify the different buildings. The main difference with classification is that every separate building is assigned a unique identifier. In sort of grouping all buildings into one class, the individual buildings can be distinguished based on their attributes. This is the first step towards the creation of a smart point cloud. As soon as every building has a unique object identifier, other attributes can easily be visualised as well.

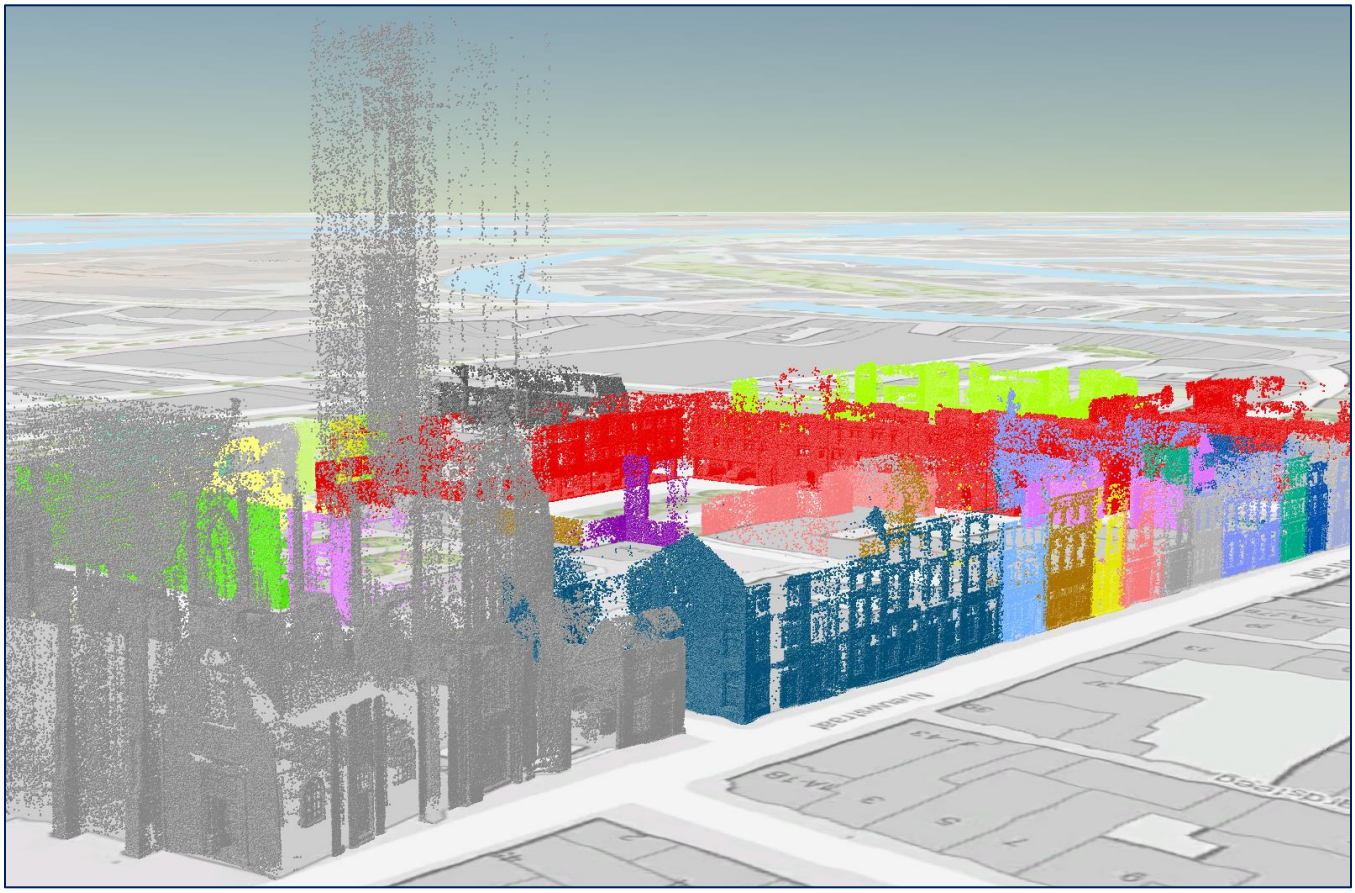


Figure 27: Overview of a small part of the research location. Here, all buildings are assigned a unique identifier. This enables the user to visualize the separate buildings in different colours. Besides that, other attributes such as the surface area or year of construction can be shown as well.

As soon as separate buildings have been identified in the point cloud, new visualisation opportunities emerge. The points are comparable based on new, formerly unknown point cloud attributes. An example of a point cloud with buildings coloured according to their year of construction is shown in **Figure 28**. Existing viewers do not support the visualization of new attributes such as the year of construction. Therefore, the intensity attribute was overwritten with the new attribute. Normally, this is not desirable since valuable information on the intensity of the returned laser beam is lost. For now, the intensity field is one of the few attributes that comprises enough bits to store a large number such as the year of construction.

However, through misusing and overwriting existing point cloud attributes, new attributes can be stored as well, allowing to visualize only old or new buildings. Besides that, the point cloud of one specific building can be separately viewed to decrease processing time for users that are mainly interested in one specific building. With this final step, the enrichment of the smart point cloud dataset is completed.

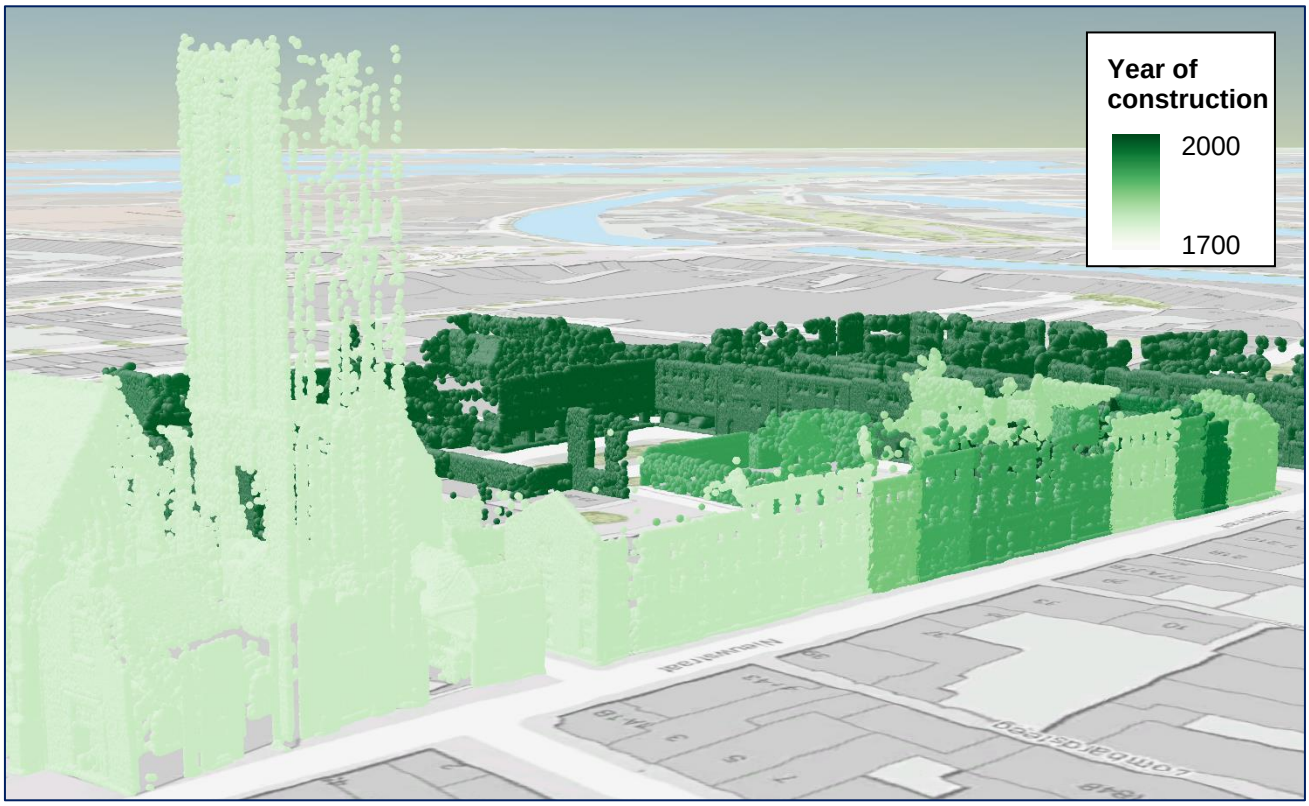


Figure 28: Different point cloud attributes can be visualized in a 3D scene. Here, the buildings are coloured according to their year of construction. Points in the scene are clickable as well, allowing to interactively request information concerning the objects shown.

6 Conclusion

This chapter aims to provide a brief overview of the answers to the stated research questions. In **section 6.1**, the most important results are summarized. **Section 6.2** discusses the findings. Suggestions for future research are given in **section 6.3**.

6.1. Summary of important results

Mobile laser scanners record their environment in three dimensions. The resulting scene is a cloud of points with relative x, y, z coordinates. Global Navigation Satellite Systems (GNSS) are used to determine the absolute coordinates of the point cloud model and project it in the appropriate coordinate system. Discrepancies emerge, especially in urban environments, where GNSS signals are occluded and become weaker. In such areas, the positioning accuracy of point clouds is restricted. Large positioning errors are constraining the possibilities to combine point clouds with other data sources.

To solve the described problem and enhance the direct usability of existing point clouds, the following question was asked: In which ways can map supported registration of mobile laser scanner data be used to create a smart point cloud of buildings? This question raised a list of other questions regarding the available registration techniques and transfer of knowledge between point clouds and maps:

1. Which registration techniques can be combined in an algorithm to improve the efficiency and robustness of the registration?
2. How can template matching be used to optimize existing point cloud registration techniques?
3. To what extent can the alignment support the transfer of semantics to the point cloud?

Many researchers have tried to attach new knowledge onto points through registration, segmentation and classification of point clouds. Most of them are focusing on point cloud to point cloud registration whereby the

relative positioning is improved. Very few however, tried to match point clouds with maps to improve the absolute positioning of point clouds. This thesis has proven that simplified histograms of point clouds can very well be automatically aligned with maps of Dutch key registries. Template matching was selected as the most suitable method for aligning the point cloud with buildings on a map. Since this method also estimates redundant solutions, the confidence level of the result increases as well.

The LiDAR dataset of Schiedam was first converted into density histogram images. In these images, every pixel represents the vertical density of points. Walls and buildings are clearly visible in the images due to their strong vertical agglomeration of points. They are compared to a customized version of the registry of large-scale topography (BGT) and registries of addresses and buildings (BAG); the self-appointed BAGGT. Buildings in those two Dutch key registries of buildings were converted into raster images as well. In this new BAGGT image, inner walls were removed whereas both footprints as rooftop prints were made visible in one combined view. The template matching of those map images and density histograms was conducted with a pixel resolution of 2.5 centimetres.

A Python script was written to automatically calculate the shifts between the tiles of density histograms and maps. The script used template matching in which the map, as a reference point, remained in a fixed position while the point cloud was shifted in such a way that it perfectly fitted the map. The proposed shift was only accepted if the result matrix showed one clear, reliable peak. Eventually, all reliable translations were optimally configured using pose graph optimization. The eventual results of this optimization which included a map shift finder where compared with Cyclomedia's existing method for positioning determination.

Different measurements have an influence on the estimated positions of the car, such as the recorded GPS locations and calculated IMU movements. The newly proposed method added the required shifts to match the density histogram with the BAGGT to the existing pose graph optimization. This new map shift finder has proven to calculate similar positioning errors as the existing methods for positioning. In 82 percent of the cases, the difference between the point cloud registration with and without map shift finder is less than 20 centimeters. This can be considered as a confirmation of the reliability of both methods. In some cases, the map supported pose graph optimization indicated that the positioning errors were underestimated by up to 90 centimeters.

For the effective identification of map objects in the point cloud, it is essential that the two datasets have almost no registration errors. As soon as the map supported registration was conducted, new possibilities for combination of the datasets emerged. The alignment allowed for the transfer of attributes between map and point cloud. Through the transfer of map attributes to their corresponding objects in the point cloud, a smart point cloud was created. The creation of this smart point cloud consisted of several steps. First, separate point clouds were created for each identifiable building. Then, corresponding attributes were transferred from the building registries to the point cloud. The resulting scene was a smart point cloud in which buildings could be filtered out from the environment. Besides that, it was made possible to visualize new attributes such as the year of construction and surface area.

The results are showing bigger positional discrepancies near high-rise buildings, as was expected, since densely built high constructions can occlude GNSS signals. Therefore, the biggest reliable shifts were found in the city centre. Also, the number of reliable matches was significantly higher at traffic intersections. This can easily be explained since alignment in multiple directions is required to provide one strong peak in the response image. Over 55 percent of the matches was rejected due to insufficient reliability indications. However, the registration could still be considered successful since nearly 700 matches were accepted in a total area of approximately 4.5 square kilometres. Positioning improvements are most important in densely built areas, where positioning errors are rather high. At the same time, the template matching achieves higher accuracy rates in those areas since many walls are present in the maps of these locations.

Accurate registration immediately shows possibilities for new applications. Map supported registration allows for automated identification of separate buildings based on their geographically associated map object. Thereby, the map supported registration has paved the way for point cloud enrichment through attribute transfers. As soon as point cloud viewing tools will become more acquainted with those new types of attributes, it should be possible to easily attach even more knowledge onto the point clouds. While limitations on the demanding characteristics of point clouds are being resolved, opportunities for new applications will only continue to grow. Eventually, technological innovations in hardware as well as software should relieve the burden of the rich and demanding point clouds. For now, further research must accelerate the usability of these relatively new sources of three-dimensional data.

6.2. Discussion

Template matching is a robust method for improving the positioning of datasets. However, a current weakness of the method is its inability to distinguish walls, hedges and fences. As a result, false positives occur when a building in the map is being matched with a high hedge in the point cloud. This could be solved through creating a density histogram of points above a certain height. For example, applying a threshold of two meters would result in the exclusion of lower walls and garden fences. The distinctive structure of walls or their reflection intensity could also be considered to detect them more easily. However, exceptions on defined rules will always continue to exist. In rare cases with buildings made of glass, the method might still be unable to reject a false match. Accuracy of the results could be improved through inclusion of road surfaces, street lights and trees as well. Still, buildings have proven to be one of the most convenient sources of registration due to their distinctive shape and high occurrence rates.

Although buildings are the most convenient map features for map supported point cloud registration, the reliability of the used method remained an impeding factor. More specifically, big rectangular building blocks proved to cause problems for the reliability of the template matching results. The outlines in the map need to have distinctive shapes to result in one clear peak in the result matrix. In other words, corners of buildings need to be included in the map image to allow for a reliable match. Increasing the current tile size might partly solve this problem. However, positioning errors are differing through time. Enlargement of the point cloud tiles results in a density histogram with a wider range of recording times. In this way, the positioning errors of the mobile laser scanner are further generalized. Also, the required corners are already present in the current tiles. The main problem is that some tiles are lacking a sufficient number of corners in one tile. In most cases, those buildings still show reliable matches in subsequent tiles.

Still, it would be interesting to conduct experiments with different tile sizes and pixel resolutions. The template matching results are affected by many different parameters. Conducting tests with different settings is time-consuming, but it could improve the efficiency of the method as well as the accuracy of the results. For example, multiple different map features could be included in the template image as well. Such additional features might improve the eventual results. However, they could increase the chance of false matches as well. The algorithm is

unable to distinct different map objects from each other. To include new features to the template map, objects in the point cloud must be detected first. Without this intermediary step, the risk of false matches will increase.

Threats for the data quality do not only emerge for the absolute positioning of point clouds. The relative positioning can be threatened as well. Registration implies shifting the original positions of points and this can have further implications for the mutual cohesion between neighbouring points. In this thesis however, the structure and patterns of points was kept intact. Adjustments in the LiDAR data involved translations of the x and y positions without worsening the original consistency. Only the trajectory of the mobile laser scanner was gradually adapted. Corrections were performed incrementally through pose graph optimization, making sure that the points are not shifted in big leaps. This prevented the emergence of gaps in the data and thereby guaranteed a smoothened final point cloud.

While comparing different sources of data, caution regarding the reliability is indispensable. For every piece of information, the background and method of acquisition should be considered. Maps for example, can be retrieved through many different techniques at various moments through a long range of time. This implies local differences in the accuracy of the data. In some cases, the positioning of objects in the 3D model might outperform the accuracy of maps. This can be considered a limitation or risk of map supported point cloud registration. The map is trusted while at the same time, the “real world” remains unknown. Eventually, template matching might have the power to detect unreliable facets of maps. Such map errors could be detected and solved through alignment to the more accurate LiDAR data. The tricky part however, is to make the computer understand which virtual reality – map or point cloud – it can trust.

6.3. Future work

This thesis focused on translations in x and y coordinates. Elevational shifts (z) were not considered since it was out of scope. However, registration on the elevational axis could be conducted as well. Accurate elevation maps such as the AHN (Height model of the Netherlands) are available and can be combined with the terrestrial laser scanner data. In this way, objects on top of each other could be distinguished from one another as well. For example, trees could be separated from the buildings or pavements underneath them. Further research would have to be conducted to explore the opportunities of three-dimensional map supported registration.

For the effective identification of map objects in the point cloud, it is essential that the two datasets have almost no registration errors. Through the registration and classification of point clouds, the processing time for viewing the demanding datasets can be decreased as well. Smart point clouds enable smart filtering. For instance, objects of main interest can be projected with more extensive levels of detail than others. In this way, the attachment of new knowledge does not only increase the amount of information, but it can also enhance the processing speed. Decreasing the required processing time for visualisation of point cloud datasets was out of the scope of this research, but it is an important future task in enhancing the usability of point clouds.

Accurate registration is essential to enable the extraction of more valuable information from point clouds. This thesis has proven that additional attributes can successfully be transferred to the 3D models. The LAS 1.4 file format is ready for extra attributes. Unfortunately, widely used viewing tools are still based on earlier versions of the LAS file format. As a result, custom made viewing tools are required to deal with deviant new attributes. At the time of writing, no single open source point cloud viewer can visualize such new attributes. The only workaround is to store the extra parameters in extra bytes, which can be done in the newest LAS 1.4 file format. However, this method requires more knowledge, skills and time than should be necessary. More research into smart point cloud visualisation is needed to come up with suitable techniques that deal with the growing size and possibilities LAS files.

As soon as the positions of the LiDAR dataset have been optimized, they can be linked to the topographic map. It is important to indicate where the BAGGT image and density histogram are corresponding, and where not. Some buildings in the BAGGT will have no corresponding LiDAR data in the point cloud. 3D points of a building are unavailable for locations where the car has simply not been. Other buildings with mismatches are interesting, since

they can indicate which buildings are not present in the point cloud. In other words, where has the built environment been changed and where does the map lack updates? In this way, the map supported registration could be used for change detection in the future as well.

References

ASPRS. (2013). LAS Specification Version 1.4-R13. *The American Society for Photogrammetry & Remote Sensing*, *July*(November 2011), 1–28. https://doi.org/faf

Bailey, T., & Durrant-Whyte, H. (2006). Simultaneous localization and mapping (SLAM): Part I. *IEEE Robotics and Automation Magazine*, *13*(3), 108–117. https://doi.org/10.1109/MRA.2006.1678144

Besl, P. J., & McKay, N. D. (1992). A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *14*(2), 239–256. https://doi.org/10.1109/34.121791

Billen, R., Poux, F., & Ruymbeke, M. Van. (2017). Smart Point Clouds for information modelling : application in Cultural Heritage.

Böhm, J., & Haala, N. (2005). Efficient Integration of Aerial and Terrestrial Laser Data For Virtial City Modeling using Lasermaps. *International Archives of Photogrammetry and Remote Sensing*, *36*(3), 192–197.

Catalucci, S., Marsili, R., Moretti, M., & Rossi, G. (2018). Comparison between point cloud processing techniques. *Measurement: Journal of the International Measurement Confederation*, *127*(May), 221–226. https://doi.org/10.1016/j.measurement.2018.05.111

Chang, C. Y., & Ma, C. C. (2017). Increasing the computational efficient of digital cross correlation by a vectorization method. *Mechanical Systems and Signal Processing*, *92*(1), 293–314. https://doi.org/10.1016/j.ymssp.2017.01.027

Christodoulou, A., & Van Oosterom, P. (2018). IMAGE-BASED METHOD FOR THE PAIRWISE REGISTRATION OF MOBILE LASER SCANNING POINT CLOUDS. *International Society for Photogrammetry and Remote Sensing (ISPRS)*, (in press).

Cyclomedia. (2015). *Asset inventory using high definition street level imagery*.

Drongelen, W. van. (2007). Continuous, Discrete, and Fast Fourier Transform, 91–105. https://doi.org/10.1016/B978-012370867-0/50006-1

Fischler, M. A., & Bolles, R. C. (1987). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image

Analysis and Automated Cartography. In *Readings in Computer Vision* (pp. 726–740). Elsevier. https://doi.org/10.1016/B978-0-08-051581-6.50070-2

Ge, X. (2017). Automatic markerless registration of point clouds with semantic-keypoint-based 4-points congruent sets. *ISPRS Journal of Photogrammetry and Remote Sensing*, *130*, 344–357. https://doi.org/10.1016/j.isprsjprs.2017.06.011

Goshtasby, A., Gage, S. H., & Bartholic, J. F. (1984). A two stage cross-correlation approach to template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Grilli, E., Menna, F., & Remondino, F. (2017). A review of point clouds segmentation and classification algorithms. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *42*(2W3), 339–344. https://doi.org/10.5194/isprs-archives-XLII-2-W3-339-2017

Grisetti, G., Kummerle, R., Stachniss, C., & Burgard, W. (2010). A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, *2*(4), 31–43. https://doi.org/10.1109/MITS.2010.939925

Höfle, B., Hollaus, M., & Hagenauer, J. (2012). Urban vegetation detection using radiometrically calibrated small-footprint full-waveform airborne LiDAR data. *ISPRS Journal of Photogrammetry and Remote Sensing*, *67*(1), 134–147. https://doi.org/10.1016/j.isprsjprs.2011.12.003

Houshiar, H., & Winkler, S. (2017). Pointo -a Low Cost Solution To Point Cloud Processing. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLII-2/W8,*(5th International Workshop LowCost 3D – Sensors, Algorithms, Applications, 28–29 November 2017, Hamburg, Germany), 28–29. https://doi.org/10.5194/isprs-archives-XLII-2-W8-111-2017

Isenburg, M. (2013). LASzip: lossless compression of LiDAR data. *Photogrammetric Engineering & Remote Sensing*, *79*(2), 209–217. https://doi.org/10.14358/PERS.79.2.209

Joosten, P. (2016). *On CycloMedia ' s Georeferencing Accuracy (Technical report for internal and external use)*. Zaltbommel.

Jung, S., Song, S., Chang, M., & Park, S. (2018). Range image registration based on 2D synthetic images. *CAD Computer Aided Design*, *94*, 16–27. https://doi.org/10.1016/j.cad.2017.08.001

Kain, J., & Yates, C. (1996, March 22). Airborne imaging system using global positioning system (GPS) and inertial measurement unit (IMU) data. Retrieved from https://patents.google.com/patent/US5894323A/en

Le Marchand, O., Bonnifait, P., Ibañez-Guzmán, J., Bétaille, D., & Peyret, F. (2009). Characterization of GPS multipath for passenger vehicles across urban environments. *ATTI Dell'Istituto Italiano Di Navigazione*, *189*, 2584–2592. Retrieved from https://hal.archives-ouvertes.fr/file/index/docid/445114/filename/Characterization_of_GPS_multipath_for_passenger_vehicles_across_urban_environments.pdf

Leonard, J. J., & Durrant-Whyte, H. F. (1991). Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91* (pp. 1442–1447). IEEE. https://doi.org/10.1109/IROS.1991.174711

Lewis, J. (1995). Fast Normalized Cross-Correlation, Vision Interface. *Vision Interface*, *10*(1), 120–123. Retrieved from http://scribblethink.org/Work/nvisionInterface/nip.pdf

Maffini, G. (1987). Raster versus Vector Data Encoding and Handling: A Commentary. *Photogrammetric Engineering & Remote Sensing*, *53*(10), 1397–1398. Retrieved from https://www.asprs.org/wp-content/uploads/pers/1987journal/oct/1987_oct_1397-1398.pdf

Mendes, E., Koch, P., Lacroix, S., Mendes, E., Koch, P., & Lacroix, S. (2017). ICP-based pose-graph SLAM To cite this version : HAL Id : hal-01522248 ICP-Based Pose-Graph SLAM.

Ministry of I & M. (2018). inwinningsregel BGT en BAG | Objectenhandboek BGT | IMGeo. Retrieved March 30, 2018, from http://imgeo.geostandaarden.nl/def/imgeo-object/pand/inwinningsregel-bgt-en-bag

Murali, S. (2018). *Map Supported Classification of Mobile Laser Scanner data (Master's thesis)*. University of Twente.

Peng, H., Zhi, X., Wang, R., Liu, J. Y., & Zhang, C. (2014). A new dynamic calibration method for IMU deterministic errors of the INS on the Hypersonic Cruise Vehicles. *Aerospace Science and Technology*, *32*(1), 121–130. https://doi.org/10.1016/j.ast.2013.11.005

Poux, F., Hallot, P., Neuville, R., & Billen, R. (2016). Smart Point Cloud: Definition and Remaining Challenges. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *IV-2/W1*(October), 119–127. https://doi.org/10.5194/isprs-annals-IV-2-W1-119-2016

Rossem, R. Van, Eekelen, H. Van, & Reuvers, M. (2012). Basisregistratie Grootschalige Topografie: Managementsamenvatting

BGT|I:MGeo Standaarden.

Someren, B. Van. (2017). *Neural multi-view segmentation-aggregation for joint Lidar and image object detection (Master's thesis)*. Utrecht University.

Talaya, J., Alamus, R., Bosch, E., Serra, A., Kornus, W., & Baron, A. (2004). Integration of a Terrestrial Laser Scanner with GPS / IMU Orientation Sensors. *Proceedings of the XXth ISPRS Congress*, (35), 1049–1055.

Van Oosterom, P., Martinez-Rubi, O., Ivanova, M., Horhammer, M., Geringer, D., Ravada, S., … Gonçalves, R. (2015). Massive point cloud data management: Design, implementation and execution of a point cloud benchmark. *Computers and Graphics (Pergamon)*, *49*, 92–125. https://doi.org/10.1016/j.cag.2015.01.007

Velodyne LiDAR Inc. (2013). *HDL-32E User's manual and programme guide manual*. Retrieved from http://velodynelidar.com/lidar/products/manual/63-9113 HDL-32E manual_Rev E_NOV2012.pdf

Verbree, E., & Van Oosterom, P. (2015). *Exploratieve Puntenwolken (Rapport in opdracht van: de Raamovereenkomst Rijkswaterstaat)*. Research for the Built Environment, Faculty of Architecture and the Built Environment, Delft University of Technology.

Verbree, E., Zlatanova, S., & Dijkman, S. (2005). Distance-Value-Added Panoramic Images As the Base Data Model for 3D-Gis. *ISPRS Archives*, *XXXVI-5/W8*(February), 24–25.

Yoo, J., Hwang, S. S., Kim, S. D., Ki, M. S., & Cha, J. (2014). Scale-invariant template matching using histogram of dominant gradients. *Pattern Recognition*, *47*(9), 3006–3018. https://doi.org/10.1016/J.PATCOG.2014.02.016

Zhang, C., Du, S., Liu, J., Li, Y., Xue, J., & Liu, Y. (2016). Robust iterative closest point algorithm with bounded rotation angle for 2D registration. *Neurocomputing*, *195*, 172–180. https://doi.org/10.1016/j.neucom.2015.06.107

Zhang, L., Li, Z., Li, A., & Liu, F. (2018). Large-scale urban point cloud labeling and reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, *138*, 86–100. https://doi.org/10.1016/j.isprsjprs.2018.02.008

Zolanvari, S. M. I., Laefer, D. F., & Natanzi, A. S. (2018). Three-dimensional building façade segmentation and opening area detection from point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*. https://doi.org/10.1016/j.isprsjprs.2018.04.004

APPENDIX I: Pre-processing workflow

This document describes the techniques being used to fit Cyclomedia's LiDAR point clouds to existing maps such as the BGT and BAG. Through calculating differences in positioning between the map and point cloud, accuracy can be improved, just as well as changes can be detected.

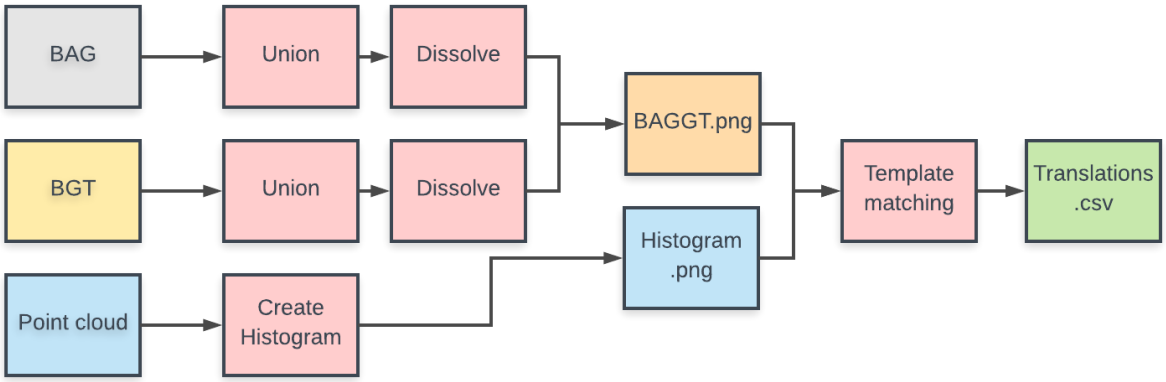
Preparation of shapefiles

The BAG and BGT will be edited in such a way that the template matching will be more successful. For instance, inner walls are removed because the LiDAR scanner mainly detects outer walls as well.

First, gaps in the polygons are removed. This is necessary since python does not handle shapefiles with gaps. Besides that, it is impossible for the lidar scanner to detect those walls. For the last processing step, the dissolve tool is being used. This step ensures that inner lines are removed (see image below).

- *Union tool in ArcGIS (option 'Gaps Allowed' should be unchecked)*
- *Dissolve tool in ArcGIS (option 'Create multipart features should be unchecked)*

The new shapefiles of the BAG and BGT are combined in a new image. This so called BAGGT is used as a template for registration of the point cloud. Through OPENCV template matching, translations for the X and Y coordinates are calculated. Those translations are stored in a new csv file, to be used in the pose graph optimization later on.



APPENDIX II: Python script

main_v2.py

```
# -*- coding: utf-8 -*-
"""
MAP SUPPORTED LIDAR REGISTRATION
Created on Tue May 1, 2018

@author: Falco Joosten
"""
# import file & libraries
import ...

start_time = time.time()

# iterate recording locations
for i in range(2900, 12235, 5):
    origin_id = i
    oid = poseInformation.calculateId(origin_id)

    #____________________CREATE RESULTS FOLDER____________________
    createFolder.createFolder(oid)

    #___________GET COORDINATES FROM THE RECORDING LOCATION_______
    x, y = poseInformation.calculateXY(origin_id)
    t = poseInformation.calculateTime(origin_id)

    #___________CREATE MAP OF THE AREA AROUND THE ORIGIN__________
    gauss = createBaggt.makeMap(x, y, oid)

    #___________CREATE POINT DENSITY HISTOGRAM FROM THE LAS_______
    histogram = createHistogram.pointDensity(x, y, oid, t)

    #___________GET X AND Y TRANSLATIONS FROM IMAGE MATCHING______
    match.getTranslation(oid, origin_id, histogram, gauss)

# print total run time
end_time = time.time()
run_time = end_time - start_time
print ("Run time", str(datetime.timedelta(seconds=run_time)))
```

poseInformation.py

```python
# -*- coding: utf-8 -*-
"""
Created on Tue May 1, 2018

@author: Falco Joosten
"""
import pandas as pd

# define location of csv
df = pd.read_csv('Recording_locations/recording_locations2_translations.csv', index_col='id')

def calculateXY(origin_id):
    x = float(df.iloc[origin_id - 1][0])
    y = float(df.iloc[origin_id - 1][1])
    return x, y

def calculateId(origin_id):
    oid = str(df.index[origin_id - 1])
    print ("Row:", origin_id)
    print ("ID:", oid)
    return oid

def calculateTime(origin_id):
    t = float(df.iloc[origin_id - 1][6])
    return t
```

createBaggt.py

```
4
5   @author: Falco Joosten
6   """
7   import ...
13
14  def makeMap(x, y, oid):
15      baggt = 'output_test/' + oid + '/' + 'BAGGT.png'
16      filter = 'output_test/' + oid + '/' + 'BAGGT_gauss.png'
17
18      # create empty figure
19      fig = plt.figure(figsize=(14, 14))
20      ax = fig.add_subplot(111, frameon=False)
21      ax.get_xaxis().set_visible(False)
22      ax.get_yaxis().set_visible(False)
23      plt.xlim([x - 35, x + 35])
24      plt.ylim([y - 35, y + 35])
25      fig.patch.set_facecolor('black')
26      fig.tight_layout(pad=0)
27
28      # read BAG shapefile
29      sf = shapefile.Reader("D:/Users/fjoosten/OneDrive - CycloMedia Technology B.V/DATA/Schiedam/"
30                            "BAG/BAG_Schiedam_sel_un_dis2.shp")
31
32      # draw BAG polygons
33      for shape in sf.shapes():
34          points = shape.points
35          ap = plt.Polygon(points, fill=False, edgecolor='white', linewidth=0.1)
36          ax.add_patch(ap)
37
38      # read BGT shapefile
39      sf = shapefile.Reader("D:/Users/fjoosten/OneDrive - CycloMedia Technology B.V/DATA/Schiedam/"
40                            "BGT/BGT_Schiedam_sel_un_dis.shp")
41
42      # draw BGT polygons
43      for shape in sf.shapes():
44          points = shape.points
45          ap = plt.Polygon(points, fill=False, edgecolor='white', linewidth=0.1)
46          ax.add_patch(ap)
47
48      # save the BAGGT (for template matching)
49      plt.savefig(baggt, dpi=200, facecolor=fig.get_facecolor())
50
51      # apply gaussian filter to BAGGT map
52      image = cv.imread(baggt)
53      gauss = gaussian_filter(image.astype(np.float32)[:,:,0], sigma=3.0)
54
55      # save the gaussian filtered map
56      cv.imwrite(filter, gauss)
```

createHistogram.py

```python
# -*- coding: utf-8 -*-
"""
Created on Mon Mar 21, 2018

@author: Falco Joosten
"""
# import file & libraries
import ...



def pointDensity(x, y, oid, t):

    path = '\\\\cm-nas03\\3DDATA\\data\\out\\experiments\\asw\\schie_no_cp\\sp-schi-161003-28992-laz\\las_processor_out'

    list_x = []
    list_y = []

    # calculate index for time and location
    xrow = int((x - (x % 50))/50)
    ypath = int((y - (y % 50))/50)
    t_in_seconds = int(t)
    t_min = t_in_seconds - (t_in_seconds % 60)

    # select only nearest neighbouring x row
    if x % 50 > 25:
        lowerx = xrow
        upperx = xrow + 2
    else:
        lowerx = xrow - 1
        upperx = xrow + 1

    # select only nearest neighbouring y row
    if y % 50 > 25:
        lowery = ypath
        uppery = ypath + 2
    else:
        lowery = ypath - 1
        uppery = ypath + 1

    # select only relevant timestamps
    if t % 60 < 6:
        lowert = t_min - 60
        uppert = t_min + 60
    elif t % 60 > 54:
        lowert = t_min
        uppert = t_min + 120
    else:
        lowert = t_min
        uppert = t_min + 60


    for row in range(lowerx, upperx):  # select adjacent tiles
        for col in range(lowery, uppery):
            folder = str(row) + '_' + str(col)
            for time in range(lowert, uppert, 60):  # select LAS files with corresponding timestamp
                if os.path.exists(path + '/' + folder + '/1915_' + str(time) + '.laz'):
                    lasfile = File(path + '/' + folder + '/1915_' + str(time) + '.laz', mode='r')
                    margin = 6.0
                    # select LAZ points within a 6 second range of the recording time
                    selected = np.logical_and(lasfile.gps_time < t + margin, lasfile.gps_time > t - margin)
                    selected_x = lasfile.x[selected]
                    selected_y = lasfile.y[selected]
                    list_x = np.append(list_x, selected_x)
                    list_y = np.append(list_y, selected_y)

    # compute the number of points that fall within each cell (densities)
    densities, edges_a, edges_b = np.histogram2d(list_x, list_y, bins=(2000, 2000),
                                                 range=[[x - 25, x + 25], [y - 25, y + 25]])

    # rotate the histogram by 90 degrees
    histogram = ndimage.rotate(densities, 90)

    # save the histogram
    cv.imwrite("output_test/" + oid + '/histogram.png', histogram)

    # calculate treshold
    retval, thresh = cv.threshold(histogram, 25, 255, cv.THRESH_BINARY)

    # create treshold figure
    fig = plt.figure(figsize=(10, 10))
    ax = plt.axes([0, 0, 1, 1], frameon=False)
    ax.imshow(thresh, cmap=plt.cm.gray)

    # save the threshold histogram
    plt.savefig('output_test/' + oid + '/histogram_thresh.png', dpi=200)

    return histogram
```

calculateTranslation.py

```python
# -*- coding: utf-8 -*-
"""
Created on Mon Apr 30, 2018

@author: Falco Joosten
"""
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
from PIL import Image, ImageFont, ImageDraw
from scipy.misc import toimage
from skimage.feature import peak_local_max
from skimage import img_as_float
import pandas as pd
import csv

def getTranslation(oid, origin_id, histogram, gauss):
    # define image (BAG)
    img = gauss.astype(np.float32)

    # define template (histogram to match)
    template = histogram.astype(np.float32)

    # save width and height
    w, h = histogram.shape

    # make list of 4 methods for comparison
    method = cv.TM_CCORR

    # apply template Matching
    response_image = cv.matchTemplate(img,template,method)
    min_val, max_val, min_loc, max_loc = cv.minMaxLoc(response_image)

    # save the match
    top_left = max_loc
    bottom_right = (top_left[0] + w, top_left[1] + h)
    cv.rectangle(img,top_left, bottom_right, 255, 2)
    cv.imwrite('output_test/' + oid + '/match.png', img)

    # save the resulting response image
    plt.imshow(response_image,cmap = 'gray')
    plt.savefig('output_test/' + oid + '/response_image.png')
    plt.close('all')

    # define font for visualization
    font = ImageFont.truetype("arial.ttf", 128)

    las = Image.open('output_test/' + oid + '/histogram_thresh.png')
    data = np.array(las)  # "data" is a height x width x 4 numpy array
    red, green, blue, alpha = data.T  # Temporarily unpack the bands for readability

    # Replace white with red... (leaves alpha values alone...)
    white_areas = (red == 255) & (blue == 255) & (green == 255)
    data[..., :-1][white_areas.T] = (0, 255, 255)  # Transpose back needed

    las2 = Image.fromarray(data)
    las3 = las2.convert("RGBA")
    las3.putalpha(128)

    map = Image.open('output_test/' + oid + '/merged_BAGGT.png')
    map_copy = map.copy()

    # save visualization of overlay map + las before match
    map.paste(las3, (400, 400), las3)
    before = ImageDraw.Draw(map)
    before.text((1200, 100), "BEFORE", fill=(255, 255, 255), font=font)
    before = ImageDraw.Draw(map)
    map.save('output_test/' + oid + '/before.png')

    # save visualization of overlay map + las after match
    map_copy.paste(las3, max_loc, las3)
    after = ImageDraw.Draw(map_copy)
    after.text((1200, 100), "AFTER", fill=(255, 255, 255), font=font)
    after = ImageDraw.Draw(map_copy)
    map_copy.save('output_test/' + oid + '/after.png')
```

```
76
77      # print the x, y translations
78      shift_x = round((max_loc[0] - 400) * 0.025, 4)  # compensate for border and multiply by resolution (1p = 0.025m)
79      shift_y = round((max_loc[1] - 400) * 0.025, 4)
80      print ("X translation:", shift_x, "m")
81      print ("Y translation:", shift_y, "m")
82      print ("Min similarity score:", min_val)
83      print ("Max Similarity score:", max_val)
84
85      # Comparison between image_max and im to find the coordinates of multiple peaks
86      coordinates = peak_local_max(response_image, min_distance=1)
87
88      # print the total amount of peaks in result image
89      peak_amount = len(coordinates)
90      print ("# of peaks:", peak_amount)
91
92      # create a list of the peaks
93      peaks = []
94      for i in coordinates:
95          peaks.append(response_image[i[0], i[1]])
96      peaks_list = list(zip(peaks, coordinates))
97      sorted_peaks = sorted(peaks_list, key=lambda x: x[0], reverse=True)
98
99      # count the amount of pixels in the map
100     map_pixels = gauss.size - np.sum(gauss == 0)
101     print ("Map pixels:", map_pixels)
102
103     # if map is sufficient, calculate distance between 1st and 2nd peak
104     if map_pixels > 150000:
105         first_peak = sorted_peaks[0][0]
106         second_peak = sorted_peaks[1][0]
107         peak_diff = 1 - (second_peak / first_peak)
108         print ("Peakvalue difference:", peak_diff, "\n")
109     else:
110         peak_diff = 0
111         print ("Peakvalue difference:", peak_diff, "\n")
112
113     # save the shift values and response image information
114     input = pd.read_csv('Recording_locations/recording_locations2_translations.csv')
115
116     # calculate and save the shifted coordinates
117     input.loc[origin_id - 1, 'shift_x'] = shift_x
118     input.loc[origin_id - 1, 'shift_y'] = shift_y * -1
119     input.loc[origin_id - 1, 'new_x'] = input.loc[origin_id - 1, 'x'] + shift_x
120     input.loc[origin_id - 1, 'new_y'] = input.loc[origin_id - 1, 'y'] + shift_y
121     input.loc[origin_id - 1, 'peak_amount'] = peak_amount
122     input.loc[origin_id - 1, 'peak_dist'] = peak_diff
123
124     input.to_csv('Recording_locations/recording_locations2_translations.csv', index=False)
```

smartPointCloud.py

```python
# -*- coding: utf-8 -*-
"""
Created on Mon Mar 19, 2018

@author: Falco Joosten
"""

# import file & libraries
import ...

# select BAG shapefile
shapefile_folder = "kerk/shapefiles_buffer_20cm"
las_folder = "kerk/BAG_LAZ_buffer_map_GOED"
csv = "D:/Users/fjoosten/OneDrive - CycloMedia Technology B.V/DATA/Python/kerk/shapefiles/Schiedam_centrum.csv"

# open csv as pandas database
df = pd.read_csv(csv)
df2 = df.set_index('identifica', drop=False)
print(df2.to_string())

# lasclip to extract separate point cloud for building
for filename in os.listdir(shapefile_folder):
    if filename.endswith(".shp"):
        bag_id = filename[0:16]
        print (bag_id)
        executable = "lasclip -i kerk/merged_map_GOED.laz -poly ", shapefile_folder, "/", bag_id, ".shp ", "-o ", \
                     "kerk/BAG_LAZ_buffer_map_GOED/", bag_id, ".laz"
        call([executable])


# las2las to put year of construction in intensity
for las in os.listdir(las_folder):
    if las.endswith(".laz"):
        bag_id = las[0:16]
        print (bag_id)
        year = str(df2.loc[int(las[1:16]), 'bouwjaar'])
        print (year)
        number = str(df2.loc[int(las[1:16]), 'OBJECTID'])
        print (number)
        area = str(df2.loc[int(las[1:16]), 'Shape_Area'])
        print (area)
        executable2 = "las2las -i kerk/BAG_LAZ_buffer_map_GOED/", bag_id, ".laz -set_intensity ", area, \
                      " -set_classification ", number, " -o kerk/BAG_LAZ_buffer_map_GOED_area/", bag_id, ".las"
        print (executable2)
        call ([executable2])


# lasmerge to merge all separate buildings in one las
call (["lasmerge -i kerk/BAG_LAZ_buffer_map_GOED_area/*.las -o output_buffer_map_GOED_area.las"])
```

APPENDIX III: Process to split the BAG in Modelbuilder