Master of Science Geomatics

Deducing the Location of Glass Windows in 3D Indoor Environments

Mels Smit

January 2022

TU Delft

CGI

Deducing the Location of Glass Windows in 3D Indoor Environments

Mels Smit
Student 4601866

January 2022

Delft University of Technology
Master of Science Geomatics for the Built Environment

Abstract

Over the years, the pace at which data is generated keeps on increasing. As a consequence, the data itself no longer holds the highest value, but rather the information and context the data captures are. This principle also holds in the 3D environment modelling scene, as accurately depicting an environment holds more value than the number of models there are of it.

One of the major problems in 3D environments, especially when the environment represents a building, is the presence of glass. A lot of the data captured to model these 3D environments is captured using LiDAR laser scanning. This is where glass becomes a problem as glass is almost completely transparent to laser beams at the typical wavelengths used when using LiDAR laser scanning. As a consequence, glass can lead to problems with navigational routes as it is invisible in the environment but still blocks the path. It can also create false spaces in the captured environment as it can also partially act as a mirror reflecting the laser beam and showing these reflections in space as if they were captured in a straight line.

Alternative manners for capturing and identifying glass in environments captured with laser have been created over the years, but they often need a dedicated set-up, expensive equipment or a lot of data. These solutions are not always feasible for users of point cloud data.

Therefore, in this thesis a focus is put on how can a low entry solution be created for this problem, which leads to the main research question: ***How can the location of glass be deduced using only information acquired from 3D point clouds and a reference position?***

To answer this question, this thesis focuses on the deduction of the locations of glass windows in the provided input. To find these a projection from 3D data to 2D is performed. In 2D image space, contours are then detected that match the criteria of window frames. These contours are then used to segregate parts of the 3D point cloud that should contain the window detected in the projection. After clustering these parts and the best matching cluster is deduced to be a window.

In this thesis, it is shown that using the proposed methodology it is possible to deduce the location of glass in a LiDAR point cloud using only an additional reference position, but there are some flaws with the simplified input of the method.

Preface

This Master Thesis could not have been made possible without the help of a lot of people. It has been quite a long journey but I am grateful for each and everyone who supported me during this past year and a half. I especially want to thank my mentors Edward Verbree and Martijn Meijers from the TU Delft and Robert Voûte from CGI for the patience they have had with me. They have helped me from the start, provided feedback on my work, provided opportunities to get hands-on with data collecting, clarified issues I had with the material and reached out to me when I was having a rough time during the thesis. So from the bottom of my heart thank you all.

Next, I would like to thank Sean Vink, one of the student counsellors at the faculty of Architecture and the Built environment, for helping me find my way again last September and to help me finalize this work. I would also like to thank Roderik Lindenbergh for being my co-reader and providing feedback at the P4.

The data collecting in this research was only feasible because of the help from Leica Geosystems, who provided the Leica RTC360 to capture data at the Orange Hall in the Faculty of Architecture, so I am very grateful to them for getting this opportunity.

I would also like to thank my fellow students from the Geomatics year for the fun first year of our studies and the support in the second. I really loved working with you all at the faculty and hope to see some of you very quickly!

During my thesis I was also working at CGI The Netherlands, where a lot of people have helped me with brainstorming, getting insight into data and figuring out problems while coding, so thank you all for your help! From these people, I would especially like to thank Arjan Vonk and Arend Pool who were co-students of mine. Your feedback, positive insights and in general the fun we've had have really helped me a lot during the roughest time of the past year.

Finally, I would like to express my gratitude to my friends and family for the amount of trust, patience and support I have gotten from them in my years at the TU Delft. After a rough start in the first year of my Bachelor and some more rough patches along the way, you all kept me going and made me find a path in life that I now walk with joy and excitement, so thank you all for being in my life and making it better now than it has ever been.

Contents

List of Figures

Chapter 1

Introduction

Data keeps being generated at a faster and faster pace, due to all kinds of technological advances. With it, however, the constantly growing body of data itself is not the property with the highest value anymore. Rather, the information and context the data captures are what people are most interested in (Wang et al., 2015).

This principle also holds within the 3D environment modelling scene, as having an accurate representation of what the data represents is often more valuable than having a large number of points with features describing them. The environments modelled today can vary a lot from large buildings or complexes like hospitals, shopping centres and airports to outdoor environments that are harder to model with satellite data like tunnels and caves. These models can then be used in navigation tools in larger areas where other navigation tools such as GNSS would fail (Staats et al., 2017), planning evacuation routes or managing assets. However, such models are not always kept up-to-date diminishing the potential of their uses the more time progresses (Nikoohemat et al., 2019).

One of the most promising ways to generate these 3D environment models is from LiDAR point clouds (Zou and Sester, 2021) generated by laser scanning. In recent years there have been numerous improvements in the field of Mobile Laser Scanning (MLS), which make it possible to dynamically scan large complexes in just a few hours by traversing the environments themselves with a carryable scanner (Lehtola et al., 2017). With some more time Terrestrial Laser Scanning (TLS) can also be used by making several static scans at different locations and joining these together using, for instance, the methodology presented in (Zou and Sester, 2021). These solutions produce point clouds and images as output in a time-efficient manner, which capture the current status of the building (Nikoohemat et al., 2019), making it easier to keep the vital 3D environments up to date.

These 3D environment models are, however, more than likely incomplete. LiDAR point clouds are a great medium to convey 3D spatial information, which can be time-efficient using the Mobile Laser Scanning approach or accurate using a more traditional although slower Terrestrial Laser Scanning approach. They are, however, still held back by the physical properties of their capturing medium, namely laser, and its interaction with glass which is the focus of this thesis.

1.1 Problem statement

The capturing medium of laser brings along some potential problems when creating point clouds that accurately model 3D environments. Glass is one of the root causes of these problems as it is a very unique material in nature, having the property that light can go through it while it can also reflect it and absorb it ((Yang and Wang, 2011) and (Koch et al., 2016)). This interaction with light and thus laser has as a consequence that it generates inaccurate data by creating artefacts due to reflecting the laser beam. It also creates noise as it reduces the laser beam's intensity and creates varying point densities with faulty points intermingled with true points. Furthermore, it inaccurately introduces areas in the space that might not exist or would not be found in the scan should glass not have been present, which can also provide the illusion of navigable space where there was actually a blockade of glass. Such faulty perception of the scene can lead to problems when using the scene when planning navigational routes through the scene and could potentially lead to dangers if these routes are used for evacuating people.

The above-mentioned downsides are only a few of the possible ways that glass can interfere with LiDAR scans, so limited perception of glass can be a real hazard in accurately representing a 3D environment (Yang and Wang, 2011), but at the very least knowing that glass is present in certain locations will already help with such a representation.

Some of the other famous causes of problems with representation using 3D environments are inaccurate data, missing areas in the point cloud, low density of points and sensor noise (Zou and Sester, 2021). Some of these causes are well-known downsides of TLS and MLS like varying point densities and missing areas due to occlusion (Weinmann et al., 2013), where inaccurate data and sensor noise are part of measuring with sensors themselves. Although these problems are present and will be covered partially when relevant in the thesis, they are not the focus of the thesis and as such are mostly out-of-scope.

1.2 Research question

To solve the problem of deducing the location of glass in LiDAR point clouds this thesis aims to answer the following research question:

How can the location of glass be deduced using only information acquired from 3D point clouds and a reference position?

To answer this question and achieve the proper results the following research questions will also have to be answered:

- *What properties are required from the scanning medium to be able to deduce glass?*
- *How reliable is the deduction of the location of glass from 3D point cloud data within the chosen research scope?*
- *How can the deduction of glass be improved using the characteristics of regular windows?*

  (A regular window in this thesis is defined as a rectangular window that is placed perpendicular to the floor.)

- *What are the advantages and disadvantages of trying to deduce glass in a 1D, 2D and 3D view of the scene?*
- *To what extent can points behind glass be used to improve the deduction of glass candidates?*

1.3 Research goal and scope

A lot of the research performed on glass detection and deduction when using point clouds is done in the field of robotics (see Chapter 2). To tackle these problems often extra information or steps are required as is explained in the work of (Ye et al., 2015). Here solutions ranging from taping windows to ignore the glass in the scene to using different sensors, like sonar, to enhance the data are presented. These methods are also not always suited for the 3D scene as when using SLAM algorithms, glass windows can be detected by robots but only in 2D as the SLAM is used to navigate the robot and not to model a 3D environment ((Yang and Wang, 2011) and (Koch et al., 2016)). Next to this, there is also a dependency on creating a model on the fly whereas there are many existing datasets that would increase in value should the presence of glass be shown in them.

What this thesis strives to deliver is a solution that applies to a minimalist approach of point cloud analysis. The proposed methodology will work on the input of a single 3D point cloud that by definition stores the XYZ coordinates of points (Waldhauser et al., 2014) and a reference position of the scanner at the moment of capturing the data. This means that an initial deduction of glass can be performed on any 3D point cloud, enhancing its semantic value by adding the presence of a previously unknown material.

The deduction of glass in itself is quite a broad topic, as it comes in all kinds of shapes and forms, and there is a lot of assumptions that can be made around it, e.g. where to look for it in space and under which angle the glass is captured with relation to the ground (for more information on the challenges with glass see §3.1). As the goal of this thesis is to perform glass deduction using a minimalist approach where only a basic point cloud and a location in it are used, the scope of glass deduction of this thesis is limited to the deduction of the location of regular windows as defined in the subquestions.

1.4 Reading Guide

This thesis discusses the following topics in separate chapters. Chapter 2 shows related work that has been performed by others about different ways in which glass can be deduced. Chapter 3 then covers the theoretical background needed to fully understand the thesis. Chapter 4 shows the proposed methodology by first giving an overview of the total method and afterwards explaining each step more in-depth. Chapter 5 then shows the results acquired in this thesis where the methodology is applied to a use case. Chapter 6 then provides a conclusion with the answers to the research questions and a discussion. Finally, Chapter 7 discusses the future potential of the work in this thesis.

Chapter 2

Related work

As previously indicated, the presence of glass has been a prevalent problem when capturing any kind of environment using LiDAR data. The fact is, however, that in indoor environments glass is very prevalent. It comes in the form of windows, doors, separation walls and so on, and fulfils tasks like sound isolation and letting more light into the environment. Therefore a lot of work has already been performed to accurately capture the indoor environment in such a way to minimalize the influence of glass or explicitly work around it, with the end goal of getting the best overview of the scene. In this thesis the focus lies on generic LiDAR point clouds having only X, Y and Z coordinates to make it applicable to as much data as possible. In this section, a broader overview of related work into glass deduction is shown from previous research where all kinds of techniques have been performed to better deduce the presence of glass, work around glass or enhance LiDAR point clouds as a whole.

First, an overview is given for techniques that can deduce the presence of glass but do not necessarily use the more conventional LiDAR approaches, to show that different data can be used to deal with the presence of glass. Then work on mirror/specular reflection detectors is shown, which uses LiDAR and logic to deduce the location and type of the material. This is followed by work on enhancing point clouds themselves to get an even better representation of the scene. Finally, a conclusion on the related work is presented.

2.1 Different techniques for deducing the location of glass

A lot of different techniques for deducing the location of glass have been presented over the years. In the related work section of Ye et al. (2015), a great initial overview is given of such works. They split the topic of reconstructing transparent and refractive objects, like glass, into 4 major categories being *Physical manipulation, Active illumination, Passive methods and Sensor fusion.*

1. **Physical manipulation:** The phenomenon of physical manipulation is possibly one of the more straightforward approaches that can be taken when trying to identify glass. It involves changing the properties of the target surface in the real world to improve their detectability, which in the case of glass could involve putting paint on the surface, applying window foil, placing a thin cardboard layer or using a spray to coat the surface which would be detectable with the scanner. While this approach may be valid it does have its downsides as physical manipulation is not always possible or desired, costs time and resources, and also cuts off data that was originally in view by completely blocking the signal at the glass surface. This last downside can be circumvented by making two scans, which can then be overlapped and points that are found in the scan with physical manipulation but are not in the scan without can then be labelled as glass, but this once again costs a lot of time and work.

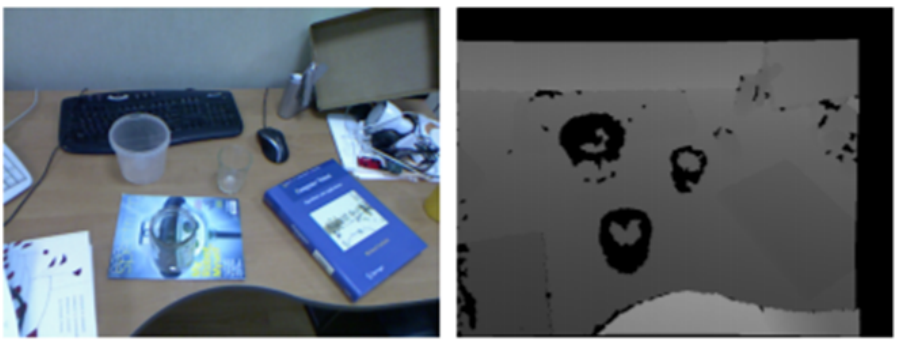
Figure 2.1: Example of physical manipulation. Black window foil is used to cover the glass which makes it able to be detected by the laser beams. [1]

2. **Active illumination:** These methods involve illuminating the glass surface with special illumination methods, such as structured light or coded illumination. By analyzing the distortions in the patterns these methods output, the location of the distortion source (i.e. glass) can be retrieved. The main downside active illumination approaches have is that they are often very dedicated in setup. This means that a dedicated setup is needed to get the results for specific properties, which hinder the scalability of these approaches and requires prior knowledge of the scene that would be captured. Even if these conditions can all be met, it can still be the case that scanning larger buildings will prove to be rather difficult as active illumination methods tend to have a high processing load.

[1] Acquired from `https://www.amazon.nl/Rhodesy-Raamfolie-zelfklevend-anti-uv-hitte-slaapkamer/dp/B07RHYW1SN/ref=asc_df_B07RHYW1SN/` (11-01-2021)

Figure 2.2: Example of Active Illumination. Structured light with a striped pattern is projected onto the glass. By registering the distortions of the pattern the location of the distortion source, in this case the drinking glass, can be deduced. [2]

3. **Passive methods:** Passive methods can directly generate 3D reconstructions from captured imagery or other data, without needing an active interaction with the scene. For example, mirror shapes can be reconstructed by observing the distortions of known patterns inside the scene, provided that the mirrors can refract light significantly or transparent materials can be detected by comparing different types of images from a depth camera. An upside of such methods is that no extra actions have to be taken to get a result, making them cost-efficient. A downside, however, is that prior knowledge of the scene is needed to calculate the results and that the scene needs to be suited for this kind of approach. This is because these methods are dependent on the condition of the scene and available scanners, sometimes making them less flexible for different environments.



(a) Kinect RGB image (b) Kinect depth image

Figure 2.3: Example of a Passive Method. In the figure, an RGB image created with a Kinect is shown on the left and a depth image is shown on the right. As can be seen, the more transparent materials like glass are quite hard to distinguish in the RGB image, whereas they are a lot more noticeable in the depth image. Using these differences in the data the presence of glass can be deduced. Acquired from Lysenkov et al. (2013).

[2] Acquired from `https://www.stemmer-imaging.com/media/uploads/sis/ST/STEMMER-IMAGING-EN-Inspecting-transparent-objects.pdf` (11-01-2021)

4. **Sensor fusion:** The last of the categories is sensor fusion which quite literally means combining multiple sensors, to generate a better result. Lots of different sensors have been combined with LiDAR data over the last few years to enhance it. For example in their work Ye et al. (2015) look into the addition of an ultrasonic sensor to a Microsoft Kinect camera to give the depth camera an additional reference to compare with (e.g. sonar) which is capable of seeing glass. Other researchers have performed similar approaches of combining LiDAR data with Sonar data to navigate robots indoors in an office building with glass walls (Wei et al., 2018). By adding sonar information to enhance the original LiDAR data, transparent objects can be identified in the scene to counteract one of the weaknesses of just using LiDAR data. A downside of this approach is, however, the cost of an additional sensor and the need to make software that is capable of joining the datasets and deciding which is the right one.

Figure 2.4: Example of Sensor Fusion. Here a fusion of a Microsoft Kinect camera and an ultrasonic sensor is shown. The fusion is performed to enhance the color and depth data of the Kinect with the Sonar information of the ultrasonic sensor. Acquired from Ye et al. (2015).

Aside from trying to reconstruct transparent and refractive objects one can also try to deduce glass by looking at the data that is missing from the scene instead. Over the years methodological approaches for deducing the presence of glass windows have been exploiting the property of glass appearing as a hole in 3D point clouds to deduce the location of glass. In the work of Kaniouras et al. (2019) a method is presented for detecting holes in point cloud surfaces. Once these holes were found, pattern and shape analysis is applied. Should a repeating pattern for similar shapes be found, then this would indicate that these are windows, turning a lack of information into a deduction.

A somewhat similar approach was performed by Tuttas and Stilla (2011) but instead of inspecting holes in facades, they assumed the repeating structure of windows in these facades and looked at the points behind the facade instead. These points captured indoor surfaces behind the transparent windows creating a pattern of data behind holes, which allowed for deducing the location of other windows by filling in the missing steps in the pattern. This made their approach more robust to windows with closed curtains and sparse input data, with the trade-off of being more dependent on the structure of the buildings itself, as a pattern is needed for the approach to work.

The final technique discussed in this section uses the property of glass to transmit energy or heat. As glass transmits heat into or out of buildings, a fluctuation in temperature in the building facades can often be seen around windows. This property of windows and other openings in facades was used by Jarząbek-Rychard et al. (2020) to deduce the presence of windows as the spots where the most heat left the building. Using a supervised machine learning approach these spots in the thermal images were found and afterwards classified accordingly, showing that thermal information can also be used for the deduction of glass windows.

2.2 Mirror / Specular Reflection Detectors

In the previous section work on deducing glass has been presented that does not use conventional LiDAR as its way of collecting points, but research has been performed that shows the possibility of deducing specular reflective materials like mirrors and glass before using LiDAR.

In the work of Yang and Wang (2011), a practical application of the previously mentioned sensor fusion has been performed, between LiDAR and Sonar. The goal of their work was to find mirrors and glass in the scene to allow for proper robot navigation in an environment, as these can lead to false assumptions made by the robot. They found that simple sensor fusion is not enough to avoid these false assumptions as artefacts of reflections were left in the scene and labelled as proper navigable space. To circumvent this they assumed all gaps found in the data to be reflective surfaces, like glass panes or mirrors, thereby removing the need for sensor fusion with Sonar. They then tested points captured behind these gaps by checking if a reflection is present in the mirrored space by applying mirror symmetry to the gaps in the scene. If the probability of a reflection is high enough the gap is identified as a mirror and if not the gap is identified as glass, showing the possibility of glass deduction using only LiDAR and logic.

In the work of Koch et al. (2016) this feat was achieved using a slightly different approach. In their work, they used a Hokuyo 30LX-EW multi-echo laser scanner, which is a special type of LiDAR scanner that produces multiple echoes per point. This makes it possible to compare results for the captured point to see if any differences are found in-between echoes, as differences indicate surface reflection properties. This means that for a non-reflective surface the differences are negligible but for reflective surfaces like glass, mirrors and metallic surfaces, there is a notable difference in returns when the reflective surface is captured at a perpendicular enough angle with the laser beam. Using this property and following their methodology, it is then possible to deduce mirrors, glass and other specular objects in the scene.

2.3 Enhancing LiDAR point clouds

Now that it has been shown that glass deduction is possible there is still another limiting factor for this field of work, namely the dependence on input data. Point clouds are known for having imperfections in them especially after only a single scan. These imperfections can be caused by all sorts of influences like noise in data, moving artefacts during scanning, occlusion and other poor conditions. To circumvent these downsides Zou and Sester (2021) have created a method to refine point clouds by adding new captures of the scene and taking the strong suits of each separate scan. This is done by creating a hierarchical model of the building to save detected features, like windows, doors, balconies and occlusions, so the point cloud is represented in semantic groups. In doing so the semantics of the scene can be incrementally improved by merging in new perspectives, scans in different conditions and scans with fewer or different occlusions to fill in each other's gaps. It can also be used to stitch together datasets, provided there is enough overlap, enlarging the total represented work. Using these techniques, processing data of varying quality and captured from different positions can be used to better complete the overview of the whole scene. This could also make Terrestrial Laser Scanning capable of capturing a complete scene by making scans in multiple positions in the room, so occlusions can be filled in with other data.

2.4 Conclusion of related work

This chapter started by giving broader insight into what kind of techniques are available for deducing the location of glass while showing some of their upsides and downsides. The most notable reoccurring downside of these methods is that they often cost a lot more time and money to perform, which is not accessible for everyone, hence this leaves more to be desired. Then methods that use only LiDAR have been presented to show that there is potential in deducing the location of glass with only a LiDAR point cloud as data. These experiments have been done using the SLAM methodology, to create a 2D map a robot could use to navigate the terrain. For 3D models however this leaves more to be desired as 3D spaces are often more complex. What can be seen from the related work is that a lot of different approaches have been tested, often bringing more materials or sensors that need to be managed with it. This is why this thesis introduces a method of scaling up the deduction of windows to 3D using similar concepts to deduce the location of glass as shown in the work before. For this thesis TLS was used to capture the scene, hence the introduction of work on how to stitch point clouds together while enhancing them in the process. The method can also be classified as a passive method as no active additional actions need to be performed in the scanning process as a point cloud and a reference position for the scanner are all the input required.

Chapter 3

Theoretical background

In this chapter concepts used in the Methodology (see Chapter 4) are explained. These concepts are things used in the Methodology that need to be understood to fully grasp the Methodology itself. Some of the concepts might be considered general knowledge for someone in the Geo-ICT scene but for a full overview of the work performed in this thesis they are included here. The concepts covered in this chapter are *The properties of glass, 3D to 2D projection, Edge detection, CLAHE, Morphological operations, Contour detection, the Douglas-Peucker algorithm, Clustering, SVD.*

3.1 Properties of glass

This thesis is focused on deducing the location of glass in LiDAR point clouds. To do this the fundamentals of the interaction between glass and laser need to be known. Laser scanning is done with light at particular wavelengths to get the best results, taking into account all kinds of factors and physical limitations, such as reflectivity, absorption by surroundings and, most importantly, preventing the possibility of harm to human eyes. The human eye is capable of seeing all so-called "Visible light" which means light of wavelengths ranging from 380 to 780 nanometers (nm) (Schott AG, 2020).

To ensure that the Scanning process is not annoying or even harmful for humans, most LiDAR scanners tend to scan with a wavelength that is in the Near-Infrared (NIR) region of the colour spectrum with wavelengths between 800 nm and 2500 nm. Often the smaller wavelengths in this range are taken as the larger the wavelength the more energy the light has, which can still cause damage to human eyes over time. Specifically, light with a wavelength above 1400 nm can be absorbed by the cornea and lens of the eye after which it is converted to heat which can cause damage to the eyes (las), so even though some wavelengths higher than 1400 nm are still allowed to be used via the FDA eye-safety standard IEC 60825 (for instance some scanners from Velodyne are capable of measuring using 1550 nm wavelengths (VELODYNE LIDAR, 2018)) it is safer and thus more common to use lower wavelengths.

This choice, although very understandable, is the source of not being able to detect certain materials when using LiDAR, as these materials are difficult to identify using light at these wavelengths. Glass is one of these materials and

has a fairly unique property in that it is transparent for all visible light and a large part of the infrared spectrum. How the type of glass exactly behaves does depend on the type of glass and its condition (for example damaged glass has a rougher surface making it more likely to reflect a bit of the laser beam).

To illustrate how glass and LiDAR interact with each other in general, figure 3.1 presents the internal transmittance of the glass type SCHOTT N-BK7, where a transmittance of 100% can be seen for both visible light and most of the NIR spectrum. Higher and lower wavelengths have lower transmittance making them better suited to capture the glass. They do however violate previously mentioned safety constraints, so are not preferred in practice as LiDAR scans are not guaranteed to be performed in areas without people.

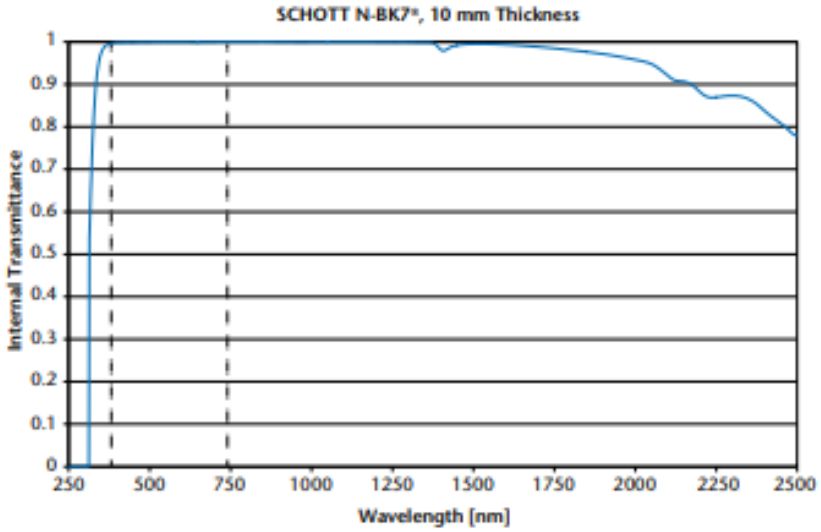


Figure 3.1: The internal transmittence of light in a 10mm thick SCHOTT N-BK7 glass plate. The blue line indicates the transmittance to waves with the wavelength on the x-axis and the dotted lines show the extent of the visible light spectrum. Acquired from (Schott AG, 2020)

Somewhat contrarily to what was stated above an internal transmittance of 100% does not necessarily mean that the material is incapable of being registered by the laser beam. Depending on the condition of the glass, angle with which the laser beam hits the glass, illumination of other kinds and so on, parts of the laser beam can be reflected directly, be reflected under an angle or even be absorbed by the glass ((Yang and Wang, 2011) and (Koch et al., 2016)). For most commonly used wavelengths in LiDAR scanners multiple options occur, where most of the light is transmitted through the glass, a small portion is reflected and an even smaller portion is absorbed.

In figure 3.2 drawn representation is given of a return that is directly reflected from the glass surface. This return has a significantly lower intensity than the original laser due to the absorbance and the transmittance of the laser beam. This return can only happen when the laser beam is fired at an angle close to being perpendicular to the glass surface as otherwise, the return would not find its way back to the sensor in the scanner.

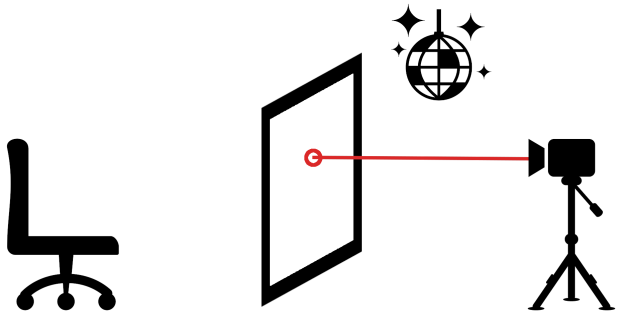


Figure 3.2: The laser beam partially gets reflected directly back to the scanner from the glass pane resulting in a point on the glass pane itself.

Returns can also come from a reflection that comes from another object than the glass which is shown in figure 3.3. The laser beam in this case bounces from the glass pane with a specular reflection to the disco ball, which would reflect it directly back to the scanner or reflect it via the mirror to the scanner, losing intensity with each reflection. When the laser beam has returned to the scanner it however only measures how long the laser was on its way, assuming it moved in a straight line. This, in the end, results in a false second disco ball forming behind the glass. The point registered in the result will also not be positioned at the red circle on the right in figure 3.3 but the blue circle on the left.

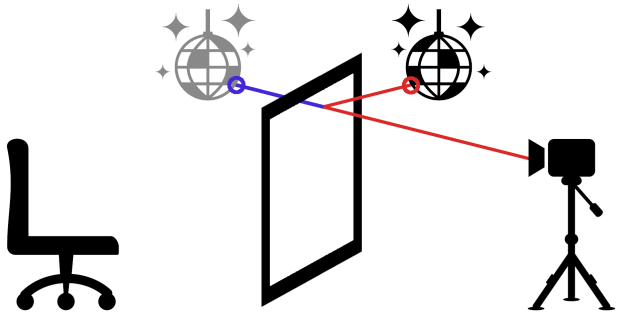


Figure 3.3: The laser beam (in red) partially gets reflected by the glass and hits a disco ball. This reflects the laser beam to the scanner via the window or directly back. The scanner, however, measures only the time the laser spent before returning, assuming that the blue path is taken when reflecting on the glass falsely creating a second disco ball behind the glass.

The next option for acquiring returns is when the laser beam is simply transmitted by the glass and this is also the one with the least amount of intensity is lost as most of the laser goes through glass without any issues. The result as shown in figure 3.4 is a point as expected on the object behind the glass pane almost as if the glass was not even there.

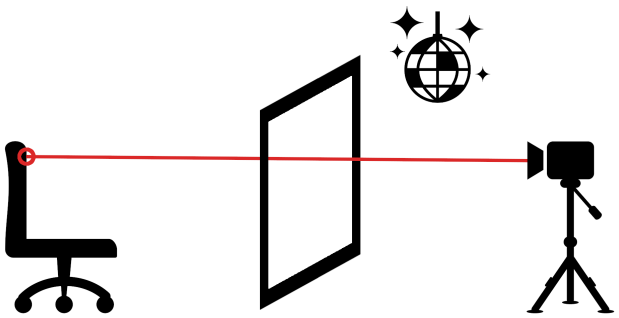


Figure 3.4: The laser beam goes through the glass with almost no interference and registers a point location on the chair behind it without problems.

Finally, the phenomenon of absorbance of the laser beam by glass is less trivially displayed as there is no return to indicate this behaviour making it much harder to inspect in the data.

Most of the information presented in this section has been theoretical, so to show that these phenomena happen in actual captured data, figure 3.5 and figure 3.6 show some parts of the scene used in Chapter 5 where points have been found directly on glass and reflection artefacts are created.

Figure 3.5: Example from the test scene used in Chapter 5 where points are captured directly on glass, here being the blue and grey points in the right window.

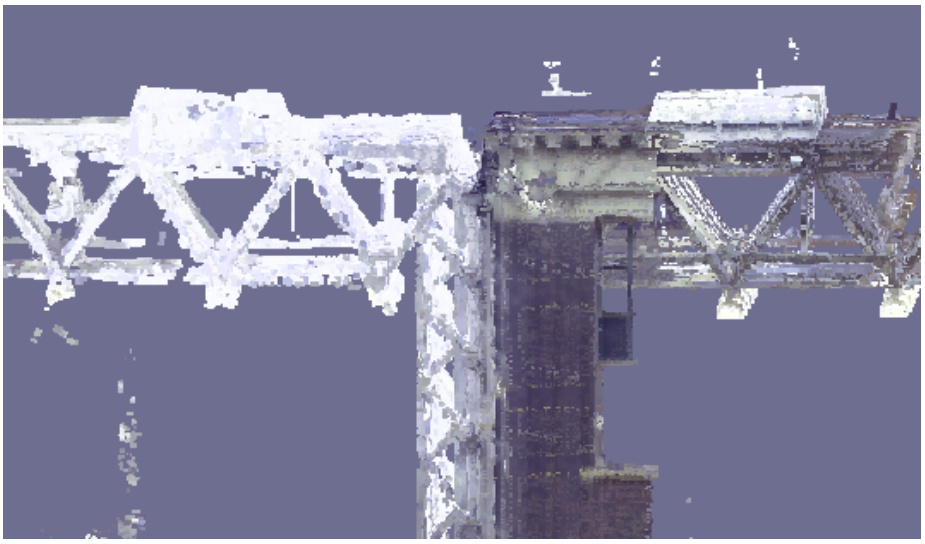


Figure 3.6: Example from the test scene used in Chapter 5 where points are shown that are direct reflections of the actual scene, which in this case are easily identifiable due to the reduced color information creating mostly white points.

3.2 3D to 2D projection

One of the steps taken in the Methodology performed in the thesis involves converting 3D Point Cloud space into 2D image space. To do this a projection is used similar to the Mercator projection. The Mercator projection is one of the most well-known projections in the world, often used in for instance world maps and Atlases (Vis, 2018). It is a Cylindrical projection, meaning that the Earth (or in the thesis' case the point cloud) is projected onto a cylinder, which is then cut open and rolled out to get a rectangular overview of the object of interest (Vis, 2018). The benefit of this projection is that angles between projected objects stay similar. This is because the lines denoting the projection of the objects are scaled similarly both horizontally and vertically. The downside, however, is that the areas of the projected objects further from the middle of the cylinder get inflated, creating a large difference in the size of objects in the projection, when compared to their real-world size.

A visual example for the Mercator projection is shown in figure 3.13, where the size distortion effects can be seen at the top and bottom of the projection, where Greenland became larger than the continent of Africa.

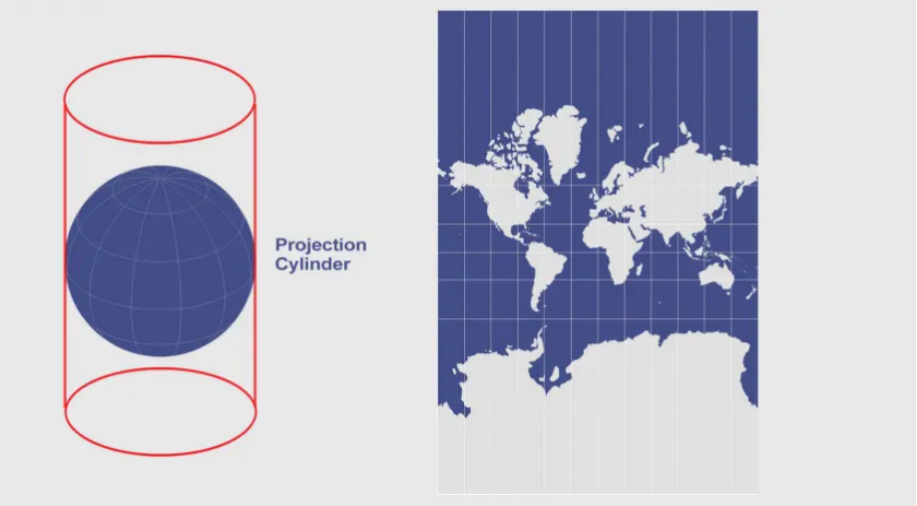


Figure 3.7: Visual example of how a Mercator projection works. The Earth on the left is projected upon the projection cylinder, after which the cylinder is cut open and flattened to produce the map on the right. The angular information is preserved, while the size information gets inflated more the further the object in Earth is from the Equator [1].

[1] Acquired from `https://gisgeography.com/cylindrical-projection/`

3.3 Edge detection

Edge detection is in short a mathematical method that aims at identifying edges or curves in a digital image at which the image values have discontinuities. Classical methods of edge detection convolved images with operators, which are sensitive to large gradients in images. Some of the most common operators used are given in the list below (Maini and Aggarwal, 2009) and a visual aid for these operators is provided in figure 3.8:

- *Sobel's operators* which are designed to give the best response with edges running vertically or horizontally
- *Prewitt's operators* which are similar to Sobel's operators but give less weight to values touching the reference pixel in a plus pattern
- *Robert's cross operators* which are designed to give the best response with edges running diagonally at a 45°angle
- *Laplacian of Gaussian operators* which highlights regions of rapid intensity change and looks all around the current pixel instead of focussing on a specific edge direction

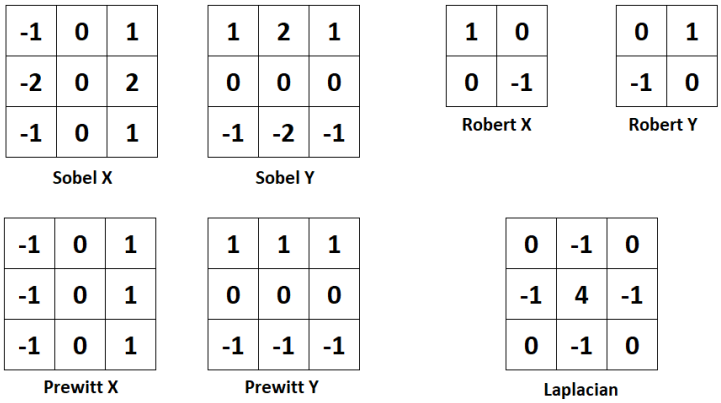


Figure 3.8: This figure gives an overview of the most common operators according to Maini and Aggarwal (2009). An X versus a Y after the operator name indicates that changes in a different direction are captured using filters. For the Sobel and the Prewitt operators, X equals vertical changes and Y equals horizontal changes. For the Robert operators, X and Y focus on a different diagonal.

All of these operators have their strengths and can work great on their own should they match the image criteria, but in 1986 Canny managed to enhance the many operators at the time with his own Edge Detection Algorithm (Canny, 1986). The Canny edge detection algorithm involves first smoothing the image using Gaussian smoothing to reduce the influence of noise. This is followed by applying the Sobel X operator and Y operator to create a combined edge strength that checks all around the current pixel. Afterwards, the primary direction of the edge is traced. Finally, non-maximum suppression and hysteresis are used to improve the results. A comparison of some of these edge detectors

can be seen in figure 3.9, where it can be seen that Canny edge detection gives the result with sharp edges without succumbing to the influence of noise.

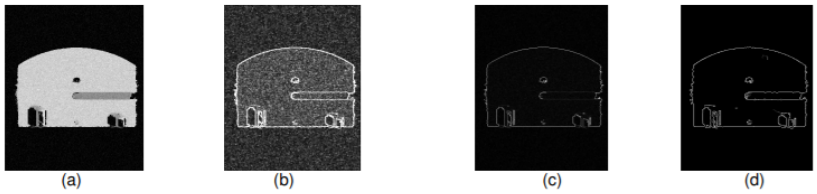


Figure 3.9: Comparison of Edge Detection techniques on a noisy image. (a) shows the original image, (b) the joint result of a vertical and horizontal Sobel operator, (c) the joint results of a positive and negative diagonal Robert's cross operator, (d) the result of the Canny Edge Detector. Acquired from Maini and Aggarwal (2009).

3.4 CLAHE

CLAHE is an acronym for Contrast Limited Adaptive Histogram Equalization. It is a technique well-known for its local contrast enhancement in images (El Abbadi and Al Saadi, 2013), which is a good step to do before trying to detect shapes in images, especially when the image has a large difference between the maximal and minimal values. This can be seen in the projects of Yadav et al. (2014) and El Abbadi and Al Saadi (2013), where it has been used to enhance the contrast of foggy video imagery (see figure 3.11) and to aid in the automatic detection of shapes in an image. CLAHE itself is an enhanced version of Adaptive Histogram Equalization, which itself is an enhanced version of Histogram Equalization. The method, therefore, takes the steps its predecessors take and adds one on top of it. The concept of Histogram equalization is shown in figure 3.10, meaning that it takes the range that a histogram is using with its values and stretches this to use the maximum range that is available, giving more values that can be used to distinguish between values in the image. This stretching can, however, result in images that become too bright or too dark as the used values are forced to stretch to fit the whole available range of values.

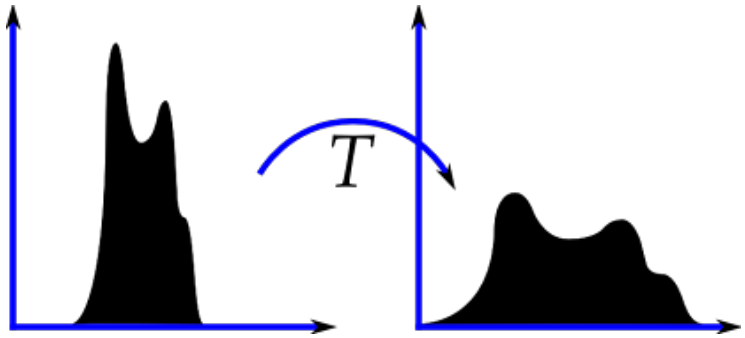


Figure 3.10: Visual example of Histogram Equalization[2]. It shows how a more narrowly used range of the Histogram on the left is stretched to use the whole available data range.

[2]Acquired from `https://sites.google.com/site/5kk70gpu/assignment-s/color-conversion`.

Adaptive Histogram Equalization takes this problem and solves it by dividing the image into smaller tiles giving a more local context and applying Histogram Equalization on all tiles separately. The downside of this approach is that it tends to overamplify the contrast in near-constant regions, which means that it tries to create significant contrast where there was none.

CLAHE fixes this issue by limiting this contrast amplification to reduce the noise that it creates. Both CLAHE and Adaptive Histogram Equalization when they would be implemented in their purest forms would be prone to introduce clear borders between the different borders, but this problem is taken care of by most implementations through the use of bilinear interpolation at the edges to make these transitions smoother. The full effect of CLAHE is shown in figure 3.11, where on the left for instance the vegetation is very dark with little detail present, whereas the enhanced version on the right shows bright green vegetation, presenting a lot more detail using the same data as input.

Figure 3.11: This figure shows the effect of CLAHE. The left image is the raw input, whereas the right image is the image acquired after using the methodology proposed in (Yadav et al., 2014).

3.5 Morphological Operators

Morphological operators are operators that can take a binary image and a structuring element (like a 3x3 square kernel as shown in figure 3.12) as input and combine them using a union set operator (Bhatia and Chahar, 2011). These operators can be used to reconstruct damaged regions in images (Raid et al., 2014) or contrarily to destroy weakly connected regions in images. In image analysis, these processes are valuable tools to connect and enhance results acquired from processes like edge detection (see §3.3).

1	1	1
1	1	1
1	1	1

Figure 3.12: A standard 3x3 square kernel which is used in figure 3.13 as the kernel for acquiring the results of the morphological operators.

The two most fundamental morphological operators are Dilation and Erosion. With dilation, every 0 that touches a 1 via the structuring element intersection is turned into a 1. The opposite happens with erosion, where every 1 that touches a 0 via the structuring element intersection is turned into a 0. This means that dilation grows regions in the image and can fill holes in them, whereas erosion shrinks regions and can break a single region into multiple regions (Bhatia and Chahar, 2011).

Opening and closing are two additional operators that combine erosion and dilation. Opening first erodes the regions splitting potential objects open and afterwards dilates the result to fill in possible holes created during the process. Closing first dilates the regions joining potential objects and afterwards erodes the result to get rid of the excess gained at the outer parts of the region. A visual representation of the operators is shown in figure 3.13.

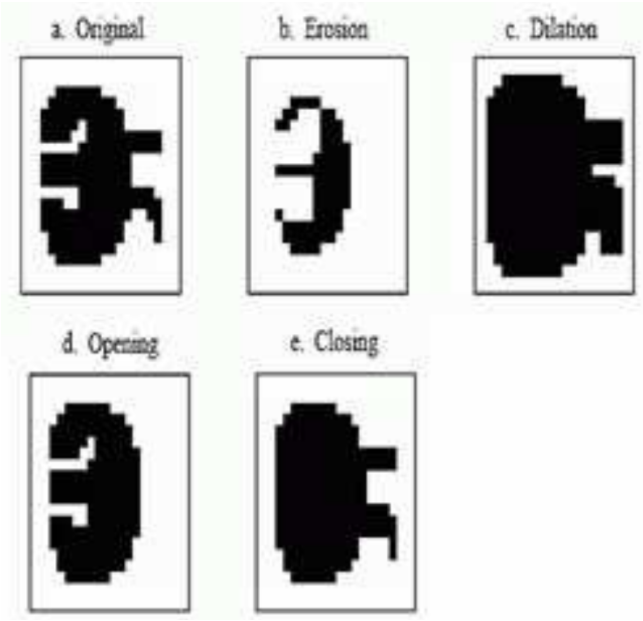


Figure 3.13: b to e show the results of the four basic Morphological operators applied to the example binary image (a). Acquired from (Bhatia and Chahar, 2011)

3.6 Contour detection

Contour detection is the process of detecting the outline of a shape. It can roughly be divided into 4 categories being: pixel-based, edge-based, region-based and deep-network-based (Gong et al., 2018). For this research, an edge-based approach has been used and as such, this is the approach that will be explained in this subsection.

Edge-based contour detection is mostly focused on taking the edges provided by an edge detector, to determine whether they surround a specific object in the image. Finding these contours is however not a trivial task as often the edges do not capture a closed object. A closed contour can be created by eliminating irrelevant data and sorting the other edges into groups using probabilistic models

as guidelines to join the separate parts into one coherent shape. In figure 3.14, this process is visualized. It starts by applying edge detection on the input image as is explained in §3.3. It then finds the outlines of the melon and the pear in the image by first removing the clutter in the data and then grouping the stronger outlines to accurately capture the shapes of the objects in the image.

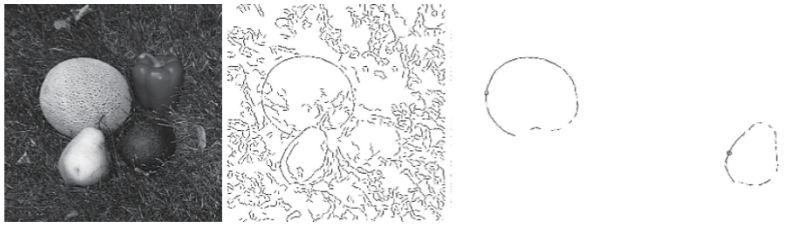


Figure 3.14: Visual example for the contour detection pipeline. From left to right the input is shown, then the result of edge detection and finally the resulting contour groups for the melon and pear are found after removing irrelevant data and grouping the edges. Acquired from (Gong et al., 2018)

3.7 Douglas-Peucker Algorithm

In 1973, David Douglas and Thomas Peucker published their algorithm for simplifying line shapes as they found that as a general rule all digitizing methods used far more data than necessary when recording lines (Douglas and Peucker, 1973). This was the origin for the Douglas-Peucker Algorithm that is still being used today although with some efficiency improvements that have been made over the years, like dynamic thresholding to better suit the specific applications level of detail (Al-Asadi and Baiee, 2014).

What the Douglas-Peucker Algorithm tries to achieve is to simplify the original line by selecting some points along the line called critical (or anchor) points which represent a generalized line that is close enough to the original. To select these critical points a threshold value, $T > 0$ is defined beforehand. Then a segment between the first point A and last point Z is traced to start the algorithm with (see figure 3.15). The distances d_{index} to the line segment AZ are then calculated for all other points. The point with the largest distance to AZ, B in this example, is added as a critical point replacing AZ with two line segments AB and BZ. However, this only happens should d_B be larger than T, if this was not the case the program would have finished instead of splitting up. After the segments AB and BZ have been created the previous steps are repeated using them as the reference and this continues until all critical points have been added and redundant points are left out of the generalized line. In figure 3.15 this resulted in leaving two points out of the generalized line.

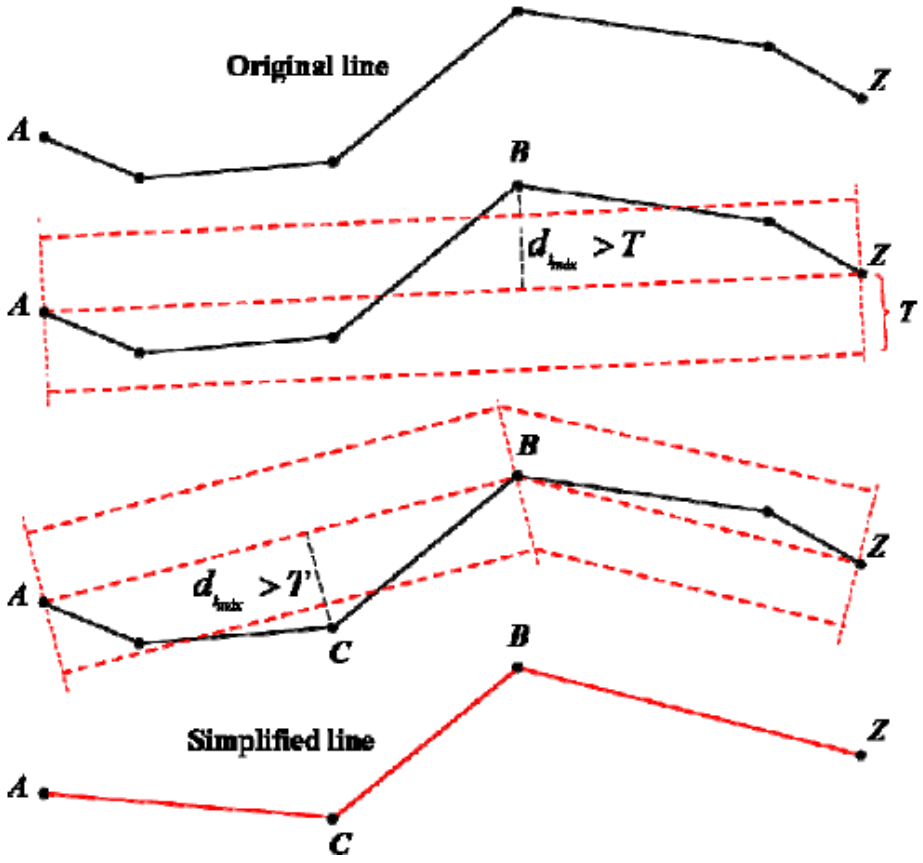


Figure 3.15: A visual explanation of how the Douglas-Peucker algorithm works to generalize the original line by keeping only the most critical points A, C, B and Z to represent it. Acquired from (Crespo et al., 2014)

3.8 Clustering

Clustering is the method of grouping data into separate groups, used in many applications such as pattern recognition and image processing (Wang et al., 2015). There are a lot of different techniques for clustering data that can be subcategorized into categories that work on the same general principle. In this subsection, the main focus will be on the DBSCAN (Density-Based Spatial Clustering of Application with Noise) clustering technique as this is the technique used in the Methodology. The K-Means clustering technique will also be explained to show the contrast and uses of DBSCAN vs K-Means.

DBSCAN is a density-based clustering method based on the concept of density reachability (Wang et al., 2015). This entails that the method loops over the points and sees if any points are within a given radius of it. If such points exist, then these are added to the cluster and referenced. This allows the cluster to continuously grow into any shape as long as there are points that are close enough. Clusters that can not grow to contain more points than a minimum threshold are considered outliers. This means that the DBSCAN algorithm has

two user parameters that need to be tweaked. One is the radius around points which implies which points are included in the cluster. The other is the minimum number of points a cluster needs to have to not be considered an outlier.

K-Means clustering on the other hand is a partition-based clustering method. It works by having the user initialize k starting points in the cluster, meaning that there will always be k clusters that are found at the end of the clustering. Each of these k starting points will form a cluster with all points in the area that are closer to it than to another starting point. This means that all data, including outliers, will be assigned to a cluster. Once all points have been assigned the means of each cluster will be computed and these will be the new starting points for a re-clustering of the set. This continues until none of the means makes a significant shift anymore.

K-Means is a form of clustering that always labels the whole dataset and can guarantee a split into k clusters even if the data is tightly knit together, but as can be seen in figure 3.16 it does not always split the dataset into logically separated objects. Furthermore, K-Means clustering is very dependent on the first placement of the starting points making it possible to have different results when applying the same process to the same set multiple times. DBSCAN is a lot more suited for logically separating objects. It can ensure that even if the number of objects in the scene is not known beforehand that the result can give a logical split between most objects. In a full scene with lots of objects close together, it can be the case that merges between some objects will occur that are not intended. To minimalize this the radius for points to be included in a cluster should not be too large but this comes with a downside of creating more outliers, so this has to be properly chosen per scene.

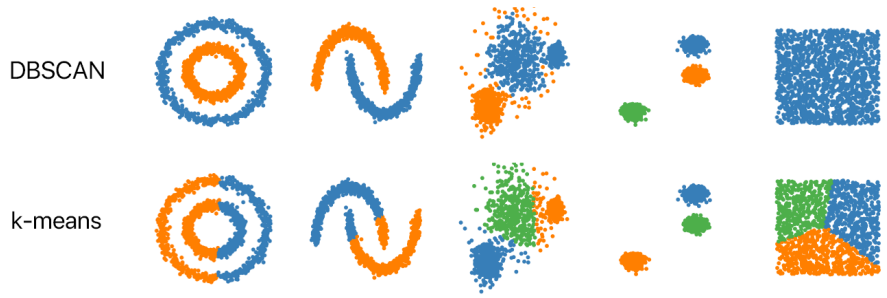


Figure 3.16: A visual comparison between the functionality of the DBSCAN and the K-Means clustering algorithms on multiple point neighbourhoods [3].

[3] Acquired from `https://github.com/NSHipster/DBSCAN`.

3.9 Principal Component Analysis

Principal Component Analysis (PCA) is one of the most important concepts in linear algebra for revealing simplified structures in data (Shlens, 2014). PCA achieves this by fitting vectors on an orthonormal basis to the regions with the most variance in data, assuming that the data follows a linearity constraint. What this essentially means is that it searches a line in the data along which the variance is the greatest, which is the first principal component (see figure 3.17 for visual feedback). Then a vector orthogonal to it is fitted to the most variance in that restricted direction for the next principal component and this is repeated until a principal component is selected for each dimension of the data. After all principal components have been found, they can be turned into unit vectors, which denote the eigenvectors of the data and eigenvalues can be extracted from them as well.

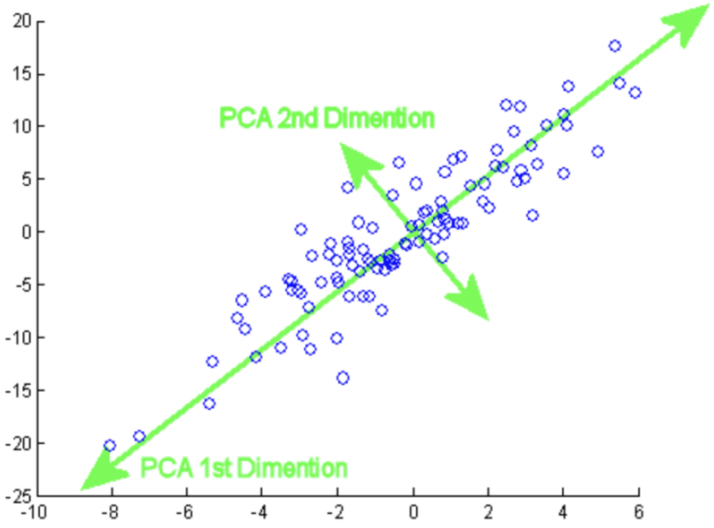


Figure 3.17: Visual example of PCA on 2D data. PCA 1 is located along the spread of data with the largest variance, with PCA 2 being placed orthogonally to it [4].

From this data, a lot of features can be calculated to further describe the properties of the dataset. In this thesis the data used is 3D so 3 principal components are expected in the data which are essentially the eigenvalues of the data cluster used as input. For the rest of this section these values are referred to as λ_1, λ_2 and λ_3 where it holds that $\lambda_1 \geq \lambda_2 \geq \lambda_3 \in \mathbb{R}$ and they have corresponding eigenvectors e_1, e_2 and $e_3 \in \mathbb{R}^3$. Using these components the following properties can be calculated which will be used in the methodology:

[4] Acquired from
`https://programmathically.com/principal-components-analysis-explained-for-dummies/`.

- **Linearity:** The linearity feature shows how much the cluster can be modeled by a 3D line ((Waldhauser et al., 2014), (Thomas et al., 2018), (Weinmann et al., 2013)). The feature is defined by equation (3.1).

$$\frac{(\lambda_1 - \lambda_2)}{\lambda_1} \tag{3.1}$$

- **Planarity:** The planarity feature describes the smoothness of the cluster surface ((Waldhauser et al., 2014), (Thomas et al., 2018), (Weinmann et al., 2013)) and is defined by equation (3.2). It gives some insight in how rectangular the cluster is, as a full score of 1 can only be achieved if $\lambda_2 = \lambda_1$ and $\lambda_3 = 0$.

$$\frac{(\lambda_2 - \lambda_3)}{\lambda_1} \tag{3.2}$$

- **Sphericity:** The sphericity feature is a simplified form for describing how spherical or scattered the cluster is as it compares λ_3 with λ_1 as is shown in equation (3.3) ((Thomas et al., 2018), (Weinmann et al., 2013)). The higher the outcome the closer these eigenvalues are and the more spherical or scattered the cluster is.

$$\frac{\lambda_3}{\lambda_1} \tag{3.3}$$

- **Verticality:** The verticality feature describes how vertical the cluster is compared to the z-axis (e_z) of the scene, where the z-axis is assumed to be the axis depicting height in the scene (Thomas et al., 2018). Following equation (3.4), verticality is calculated by taking the angle between the normal of the cluster (e_3) and the z-axis (e_z) in radians which lies in the range $[0, \pi]$. Then this is subtracted from $\frac{\pi}{2}$ to shift the range of values, made absolute as positive vs negative is obsolete in this comparison and normalized by dividing with $\frac{\pi}{2}$ to generate results between 0 and 1. A verticality of 1 would mean here that the cluster would be flat on the ground, whereas a verticality of 0 would mean that the cluster would be perpendicular to the ground.

$$norm(abs(\frac{\pi}{2} - angle(e_3, e_z))) \tag{3.4}$$

- **Change of curvature:** The change of curvature feature describes how much the smallest eigenvector influences the rest of the cluster. This can be seen in its definition shown in equation (3.5) ((Weinmann et al., 2013), (Thomas et al., 2018)). Change of curvature is very suited to distinguishing between planar structures such as facades, the floor and windows and non-planar structures like furniture and plants (Weinmann et al., 2013).

$$\frac{\lambda_3}{(\lambda_1 + \lambda_2 + \lambda_3)} \tag{3.5}$$

Chapter 4

Methodology

In this chapter, the methodology used in the thesis will be introduced. First, a short overview of the proposed method is given. Then, the different steps which are introduced in the overview are elaborated upon. Finally, an overview will be given of possible ways in which the challenge of detecting glass in 3D point clouds could have been solved and why the used methodology was chosen.

4.1 Method overview

Following the main research question, the proposed methodology (see figure 4.1 for a visual overview) aims to deduce the location of glass in 3D point clouds, using as little information as possible. This methodology is structured in four layers signifying a split in the general tasks that are covered in the method.

Layer 1 is the input that is required for the method to work. This input is a 3D point cloud and a reference position in the 3D point cloud, like the scanner position.

Layer 2 is about the conversion of the 3D data to 2D data. The first step in this process is to calculate the distance of all points in the point cloud to the reference position. These distances are used in the next step, which converts the point cloud from 3D space to 2D space, using an approach similar to the Mercator projection technique explained in §3.2. The intermediate result of the conversion is the creation of a 3D histogram that contains all points that have approximately the same direction from the reference position in the same bin. These bins are then converted into a single value based on a criterion to reduce the dimension from 3D to 2D. In this thesis, the variance of the distance between all points in the bin and the reference position is taken as this criterion. Due to this conversion, a 2D image with values indicating the spread of points in a similar direction from the reference position is created, where the shapes of the 3D objects are preserved as best as possible in the 2D projection. The benefit of the conversion from 3D space to 2D space is that a vast array of image processing techniques can be applied to the data afterwards as shown in the following steps.

Layer 3 is about finding areas in 2D where glass windows are expected to be in the 3D data. These areas are named window candidates in this thesis. To find these candidates, first CLAHE (§3.4) is applied to the image to make local

changes more prevalent, as the values in the spread of points are limited by their local context due to having a static reference position. Then Canny edge detection (§3.3) is applied to the image to highlight the areas that have a significant change in variance when compared to their local neighbourhood. These changes in variance indicate the edges of windows, which means that the result of the edge detection gives the first indication for window candidates. These candidates are then extracted using contour detection (§3.6), which connects the edges into a contour, after which the contours are simplified and validated to see if it matches the general window shape.

Layer 4 is about using the candidates found to select areas in the 3D point cloud where window deduction is performed. For each of the found window candidates, it is now known in which area of the 2D image the window is present. This area is then reverted to 3D space to allow for the deduction of the window in part of the point cloud. The 3D points are then clustered using density-based clustering (§3.8) after which for each point cluster properties will be calculated and scored to see which of the clusters represents the glass window in 3D space.

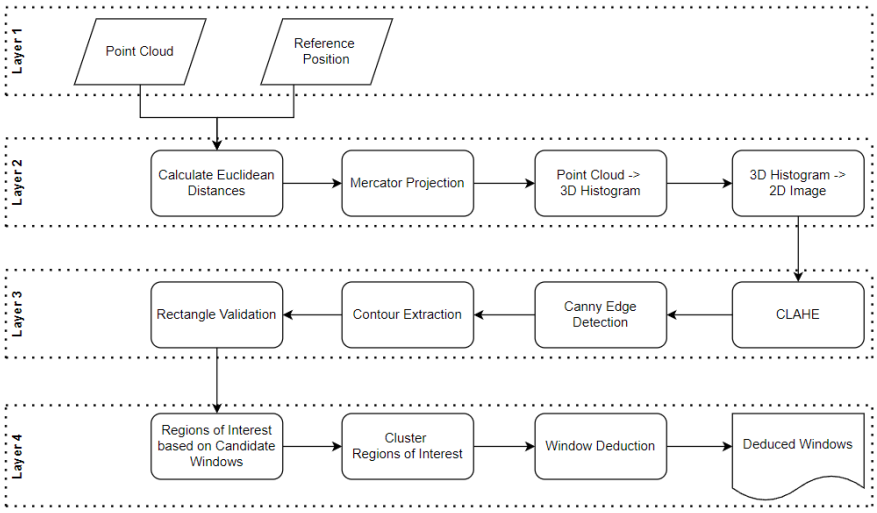


Figure 4.1: A visual overview of the steps performed in this thesis

4.2 Elaboration of steps

In this section, all of the steps of the Methodology are elaborated upon. Reasoning as to why certain choices have been made will be provided. Furthermore, visual aides and examples will be given to give more insight into the process itself.

4.2.1 Providing input data

As stated before, the methodology needs two parts of input data to fulfil its goal.

The first part of this input is LiDAR point cloud data. LiDAR point cloud data in this thesis follows the definition given in Waldhauser et al. (2014), meaning any set of 3D points that contains at the very least the x, y and z positional components.

To accommodate this data and to be able to verify it in space there is also in need for a reference position. The reference position in this methodology will act as a central point for the spatial conversion from 3D space to 2D space. This position can be either the position of the scanner itself for instance when using TLS or any position that is desired by the one performing research. In the case, an arbitrary position is picked it should have an unobstructed line to all points in the point cloud for the intended result.

The reason why these inputs are chosen to be this generic is to make sure that this methodology can be used in almost any situation where there is a single 3D point cloud that has been delivered. More value can, in turn, be added to these pre-existing point clouds without the need of having to redo the whole scanning process or generating other data than was already present.

4.2.2 Calculating the distance for all points towards the reference position

After the point cloud and the reference position have been loaded, the first step is to calculate the distance of all points in the point cloud towards the provided reference position. The calculation of this distance can be done using the euclidean distance formula as the distance between scanner and point is always under the assumption to have nothing in between them, so a direct distance calculation like the euclidean distance is suitable for this application. The formula used for the calculation is shown in equation (4.1) with n being equal to 3 for the three dimensions x, y and z.

$$d\left(p,q\right) = \sqrt{\sum_{i=1}^{n=3} \left(q_i - p_i\right)^2} \tag{4.1}$$

This is done for two specific reasons, the first being that the distances and respective positions from the reference position to the points are needed for the projection of 3D space onto 2D space. The second reason is that this data will be stored in the projected 2D space and will become the main criteria for identification in the resulting 2D image. This projection to 2D space aims to use the variance property of the distances in a similar line of sight to find the edges

of holes in facades, which will be used as candidates for the window estimation later on. This phenomenon is shown in figure 4.2 where on the left a line of sight only hits a facade which therefore creates a single group of points as the LiDAR signal is fully blocked by the facade. This single group of points is comprised of points that are all close to each other and will therefore have a very similar distance to the reference position resulting in a small variance. The line of sight on the right on the other hand captures multiple groups of points as due to the hole the line of sight can see both the edges of the hole as some of the points behind it. These multiple groups of points are further away from each other, therefore, increasing the variance towards the reference position, validating that a greater variance is an indicator for the detection of holes.

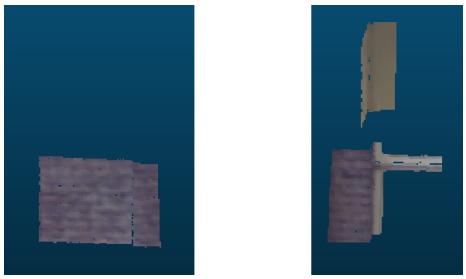


Figure 4.2: On the left, the result of a line of sight can be seen from the reference position, which is the position of the scanner in this case, towards a facade, which has only a single group of points. On the right, the result of a line of sight can be seen of the edge of a hole in the facade and as can be seen here, there are multiple groups of points indicating a higher variance in the distances of points towards the reference position for this line of sight.

4.2.3 Convert from 3D point cloud to 3D Histogram

Once the euclidean distances for all 3D points have been calculated, the conversion from 3D to 2D can begin. The conversion is done using the Mercator projection technique explained in §3.2. This projection takes a point on the Earth's surface and using its distance and angle from the Earth's centre projects it onto a 2D map preserving angular information in the process (Vis, 2018).

As the goal of our Methodology is to find the shape of window objects in the data to deduce their position, the distortion of size is mostly irrelevant. This is because the rectangular shapes of the objects are mostly preserved due to the preserved angular information. The largest distortions, where this distortion would not be trivially solved, occur on the roof and floor of the indoor space where the data is captured, where it is assumed that no windows are present.

This projection is recreated in the methodology by using the 3D points as points on the Earth's surface and using the scanner location as the Earth's centre. The resulting formulas for this conversion are shown in equation (4.2) and equation (4.3) and are visually supported by figure 4.3, which calculate a latitude angle ϕ and longitude angle λ for each point p that act as an index reference for the 3D histogram.

$$Latitude(\phi) = degrees(sin^{-1}(\frac{p_z}{d(p, scanner_{origin})})) \tag{4.2}$$

$$Longitude(\lambda) = degrees(tan^{-1}(\frac{p_y}{p_x})) \tag{4.3}$$

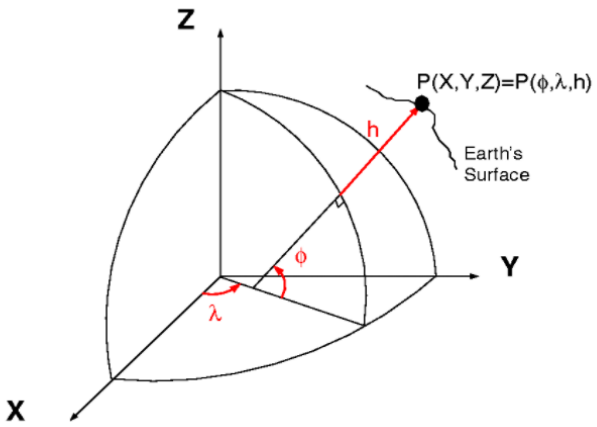


Figure 4.3: Visual example to support equation (4.2) and equation (4.3). In the figure the relation between the XYZ position and the $\phi\lambda h$ position of point p is shown. For the purpose of the methodology the height is neglected as only the general direction of p to the origin is needed [1]

This 3D histogram has a latitude and longitude which act as identifiers so it is known to which bin, which is acting as the third dimension, the current point needs to be added. This process ensures that all points with the same rounded latitude and longitude will be stored in the same bin, meaning all points with the same directional angles, ergo a similar line from the reference point are stored together. In figure 4.4, this step is visualized.

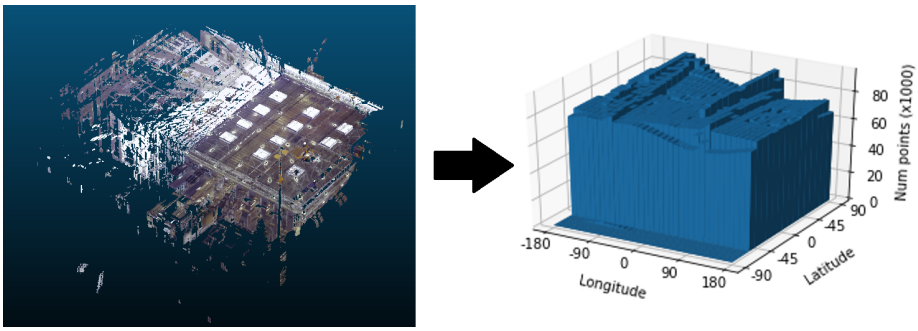


Figure 4.4: Conversion from LiDAR Point cloud on the left to the 3D histogram on the right. As can be seen there has been a fairly stable collection of points with some holes and peaks along the way in the histogram. Notable is that lower latitude readings (-90 to about -50) are substantially lower than the rest of the cloud, which is because of the used scanner which could capture less to no points below a certain angle due to the scanner itself physically blocking the signal.

[1] Acquired from `https://geodesy.noaa.gov/TOOLS/XYZ/xyz.shtml`

4.2.4 Convert from 3D Histogram to 2D Image

The 3D histogram now still needs to be converted to a 2D image before image analysis techniques can be applied to it. To do this for each bin the variance σ^2 of the distance from the points inside of the bin to the scanner location is computed, using equation (4.4) where n = the total number of points in the bin and μ = the mean distance of the points in the bin to the scanner origin.

$$\sigma^2 = \frac{\sum_{i=1}^{n} (d(p_i, scanner_{origin}) - \mu)^2}{n} \tag{4.4}$$

This value is then stored as the pixel value of the resulting image of size max latitude by max longitude. This conversion is shown in figure 4.5.

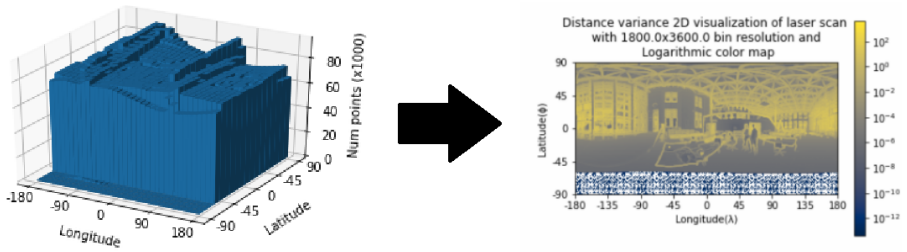


Figure 4.5: Conversion from the 3D histogram on the left to the 2D image on the right. On the left, a logarithmic colormap is used to display the differences in the Variance more gradually. Also note that the decreased data in the 3D Histogram (explained in figure 4.4) is visible as white pixels at the bottom in the 2D image.

4.2.5 CLAHE

Now that a 2D image has been achieved it is possible to start deducing the window candidates from the data. There is only one problem with the image as of now which is that the context of variance difference is taken for the whole image instead of just the local area of the pixels. This results in that smaller but still clearly defined differences between pixels will not be detected as well as the extremes of the image take precedent in the analysis. To limit the amount of influence these extremes have on the result CLAHE (see §3.4) is applied to the image before further analysis is done. This process makes it so that differences in variance are not viewed in the context of the whole image but in a limited local context highlighting local extremes.

The difference that the application of this algorithm makes to the whole scene visualization can be seen in figure 4.6. CLAHE enhances the differences from the initial image on the left and gives a result with much more detail into the notable variance peaks shown in yellow.

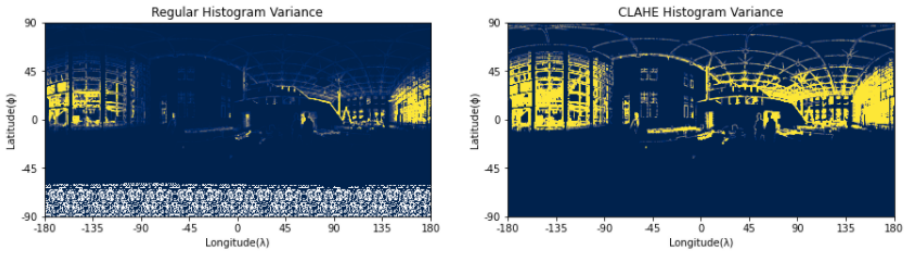


Figure 4.6: On the left the initial image is shown, with some notable reactions mostly on both sides, but a lot of the edges that are expected to be seen are left out. After applying CLAHE the image on the right is found, showing a lot more edges which could be window candidates.

4.2.6 Canny edge detection

To use the image of variances to deduce the location of windows Canny edge detection (see §3.3) is then applied to the image. Canny edge detection is chosen in this methodology as it looks from both a top-down and a left-right perspective at the changes in the pixels making all sides of the windows appear in the result. After the discontinuities have been found in the image they are linked together to form an edge or curve that because of the data it is based upon indicates a jump in space. These edges are however often close to each other but not fully connected yet, therefore closing (see §3.5) is also applied to the result to connect closely detected edges and make them more robust by filling in the small gaps in them.

In figure 4.7 these edges are visualized and the first indications of where the window candidates can be found already become more visible. It is however also clear that not only the window candidates have been detected in this process, but also a lot of other jumps in variance, so extra steps need to be taken to eliminate these from the result set.

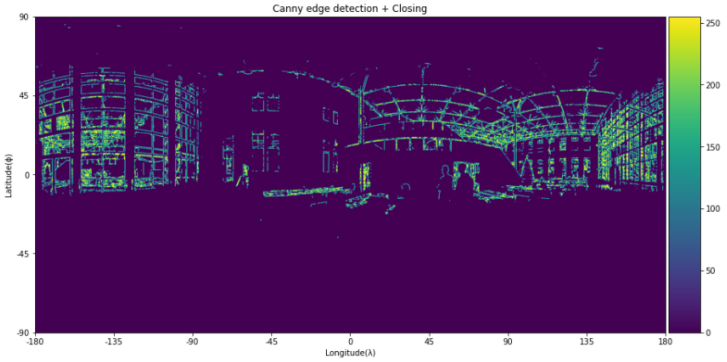


Figure 4.7: Intermediate result of Canny edge detection with closing applied afterwards. The colour in the image shows the strength of the edge.

4.2.7 Contour extraction

The first of these extra steps is contour extraction, to join the different loose edges into connected objects. First, the contours need to be detected from the separate edges, so these are joined together based on a combination of their intensity, location, the direction they follow and possibly a probabilistic model. To enhance this process, images can be simplified beforehand by putting their colour scheme to grayscale to have a clearer distinction in intensity and parts that are not important can be filtered out to make the contours be better focused on the features that are wanted. For this reason, edge detection has been applied, as this results in a grayscale image, where higher values indicate stronger edges and the unimportant readings have been removed mostly from the space of interest. The result of these detected contours for the scene is shown in figure 4.8, but it is clear that a lot is selected and the difference with the edge detection is not yet apparent. What these contours are most useful for is that they are grouped and are the building blocks for all kinds of inspections, which will be done in the next step.

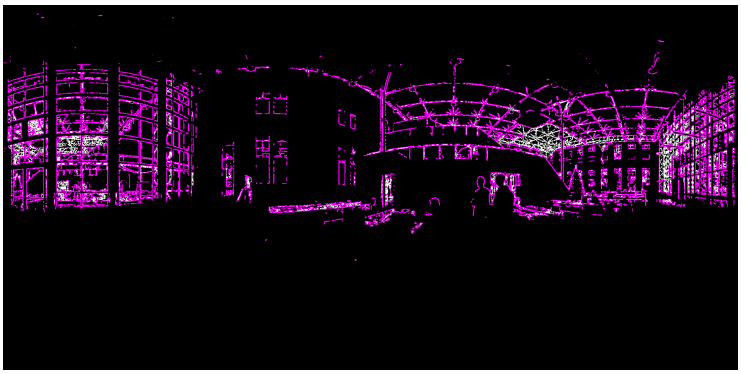


Figure 4.8: Visual example of contour extraction. The contours found based on the binary result of Canny edge detection are quite close to the edges itself as expected, but the contours deemed to small to be useful are filtered out.

4.2.8 Rectangle validation

A window candidate is defined as a close to rectangular object in the slightly warped 2D space that was the result of projection in §4.2.3. Once the contours have been extracted the window candidates can be deduced from them. To do this, the shape of the contour is generalized by using the Douglas-Peucker algorithm (see §3.7) to get rid of points in the contour that do not contribute as much to the general shape of the contour. Lines of points with a slight curve can therefore be captured as a straight line with only two endpoints. After this process the more complex contours are simplified into being made up of just a fraction of the points they had before this, which makes window candidates simpler to detect as per definition the window candidates are rectangular, and when simplified have only four strong points. Once the simplified contours have been filtered out there is another criteria for the rectangles, which is that all of their corners have an angle of 90 degrees. This is however not the case for the generated scene as it has been slightly warped due to the view the scanner had

of the scene even if the projection used is one where the angles are preserved as best as can be. By using an error margin around the 90 degrees angle, the general shape of a close to perfect rectangle can still be approximated quite well as all corners need to uphold the criteria. In figure 4.9 this is shown more clearly, with the shape on the left fulfilling the requirements for a window candidate, but the shape on the right does not stay within the error margin around 90 degrees so is therefore rejected.

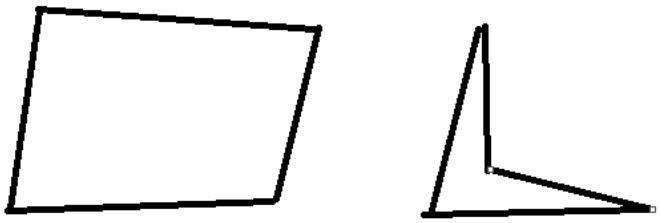


Figure 4.9: The shape on the left shows a close to perfect rectangle which will be detected by the algorithm as its errors in the corners stay within the error margin for each corner. The shape on the right fulfils the 4 points requirement but does not stay within the error margin for its corners and is therefore not labelled as a window candidate

Using these two criteria, contours that have 4 strong points are tested by first calculating their corner angles using the Cosine law (equation (4.5)), where a and b are adjacent the lengths of the adjacent sides of the triangle to $\angle\alpha$ and c is the opposite side. After the corners are calculated, they will all be tested to see if they stay within the error margin.

$$\angle\alpha = degrees(cos^{-1}(\frac{a^2 + b^2 - c^2}{2 * a * b})) \tag{4.5}$$

The result of this selection can be seen in figure 4.10 where the window candidates are shown in green and the rejected contours with 4 strong points are shown in yellow.

Figure 4.10: Visual feedback for detected window candidates in the scene. The green regions are those that are accepted as candidate windows for the next steps whereas the yellow regions fulfil one but not all of the listed criteria.

4.2.9 Acquire regions of interest based on candidate windows

Once the window candidates have been found, the analysis in 2D space has concluded. For each of the window candidates, a bounding rectangle is created which denotes the pixels in the image that the window candidate is inside of. These pixels are then used to cut out a part of the 3D data where the window is inside of, which essentially is picking the corresponding bins from the 3D histogram which correspond to the pixels in the 2D image. The result in 3D is a pyramid-shaped region of interest from the total scene containing all points in that direction as well as the points that make up the window. In figure 4.11 this concept is depicted, with the tip of the pyramid being the location of the scanner and the lines in red demonstrating the area that has been cut out of the total scene.

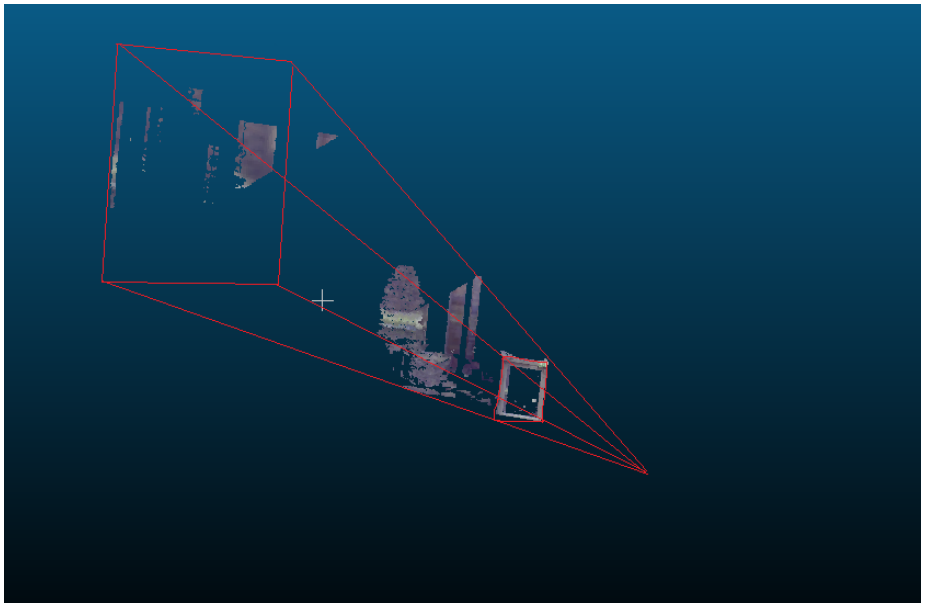


Figure 4.11: Part of the 3D scene that has been cut out using the window candidates as a base. The red lines show the area that has been cut out in total to indicate the magnitude of this operation. The tip of the pyramid-like shape the lines make is where the scanner was located and the width of the pyramid is dependent on the size of the window candidate.

4.2.10 Cluster regions of interest

The region of interest that has been cut out contains the points that make up the window in 3D context but also a lot of other points. Therefore the correct points that depict the windows need to be found in this space. As it is known as a consequence of the cut in 3D space performed in the previous step, the area of points is cut off at the sides. This means that the total space is made up of groups of points in space surrounded by empty space, making it easy to split these groups from each other. This is done using Density-Based spatial clustering (see §3.8) to cluster the selected points into groups that are separate in space from each other. The result of this operation can be seen in figure 4.12, where each colour is a separate cluster to be examined.

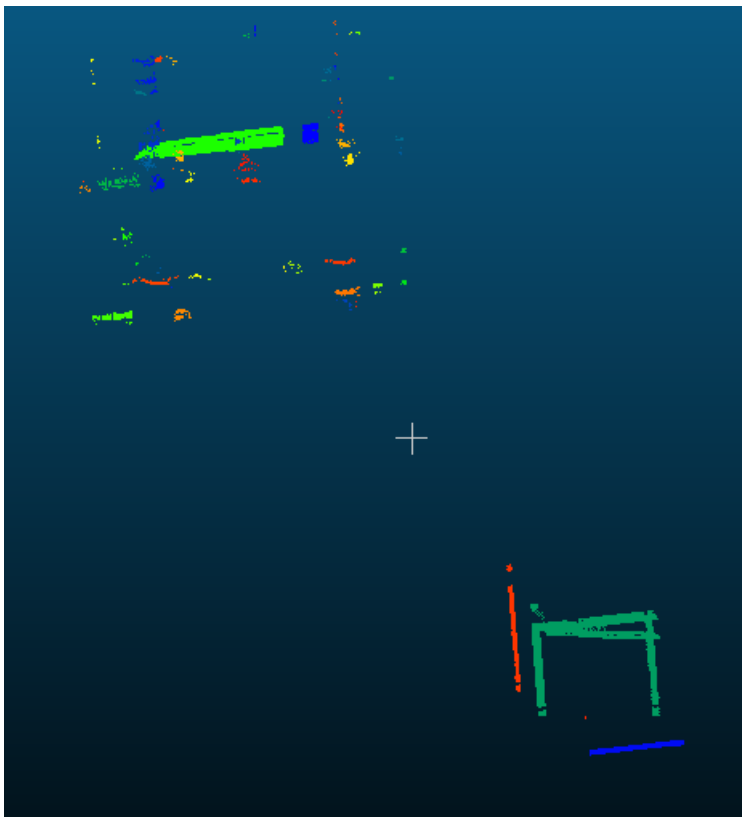


Figure 4.12: Visual example of the result of density based clustering.

4.2.11 Deduce window in regions of interest based on geometric properties

Now that the clusters have been created, they can be tested to see which of them is the window. These tests are done by calculating a set of 3D features of the clusters using the clusters as the selected neighbourhood as described in Weinmann et al. (2013) and validating these with the expected criteria for a window. But to generate these features first the 3 eigenvalues of the clusters need to be derived using PCA (see §3.9). These eigenvalues are hereafter referred to as λ_1, λ_2 and λ_3 where it holds that $\lambda_1 \geq \lambda_2 \geq \lambda_3 \in \mathbb{R}$ and they have corresponding eigenvectors e_1, e_2 and $e_3 \in \mathbb{R}^3$. This concept is shown in figure 4.13.

For this methodology, the features discussed in §3.9 were calculated on a scale from 0 to 1, so that each could easily be compared to another.

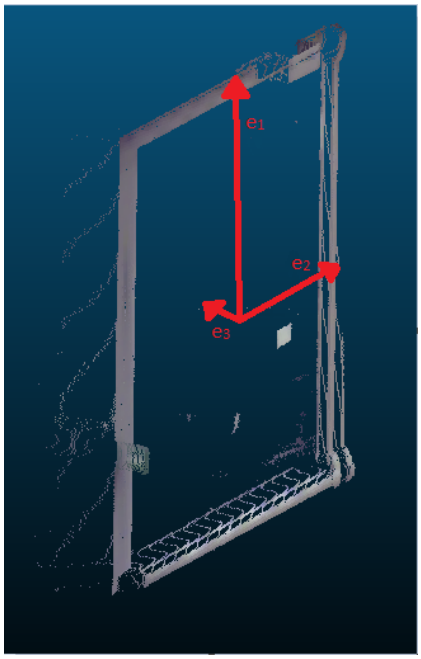


Figure 4.13: The cluster of a candidate window, that is expected looks somewhat like this. A centre point in the middle of the cluster, which should be approximately the middle of the window, from which a dominant e_1 vector to either the top or the side of the window originates which is the longest spread of the window. A less dominant but certainly significant e_2 vector perpendicular to e_1. Finally, there is a less significant normal vector to the window plane depicted by e_3.

- **Linearity:** As (the outline of) windows themselves are rectangular instead of linear it is expected that λ_1 is quite large when compared to λ_2 as shown in figure 4.13. For Linearity a somewhat lower value is desired as can be seen in figure 4.13 where e_1 is about twice as large as e_2 indicating that for these kinds of rectangular windows 0.5 is a good estimate but optimization of the parameter may be required when used in a different scene.

- **Planarity:** In the case of Planarity, a higher value is desired as e_3 is preferably as little as possible. A perfect value of 1 can only be achieved by having a square window which might not be the desired shape so some optimization for the scene may once again be required but in general, a higher score in planarity indicates a higher likeliness to be a window.

- **Sphericity:** A high Sphericity indicates very close eigenvalues, which as can be seen in figure 4.13 it is desired that e_1 is large and e_3 is small. This means that in general higher Sphericity indicates lower likeliness to be a window.

- **Verticality:** In most cases, one can expect windows to be perpendicular to the ground. This means that a lower verticality indicates a higher likeliness for being a window, but in the architectural scene this is not a certainty as windows can be placed in all kinds of angles, so modify or remove this criterion should this not apply to the scene.

- **Change of Curvature:** Finally, there is the change of curvature, which is very suited to distinguishing between planar structures and non-planar structures. As a high change of curvature implies a non-planar structure, this indicates a lower likeliness that the cluster is a window.

For each cluster, a weighted average is then calculated for these features. These weights need to be optimized to the scene itself, as in some scenes for instance the verticality of the cluster is a great indicator for windows, but in areas with tilted windows, this will be less reliable. Using the assumption that in each selected area only one window is present, the cluster that scores best on this weighted average of this data is selected to be the window.

4.3 Considered alternatives to the Methodology

One of the subquestions in this research has been:

What are the advantages and disadvantages of trying to deduce glass in a 1D, 2D and 3D view of the scene?

This question has been considered while coming up with the methodology presented in this chapter but has yet to be covered in the thesis. This section will cover the ideas around using other dimensions for the methodology by covering them one by one and then giving the reason as to why the current methodology was chosen. Afterwards, another alternative will be mentioned as a workaround to one of the main problems with the methodology being its dependency on the contours that need to be found.

4.3.1 Advantages and disadvantages of using different dimensions to deduce the location of glass.

- **1D data:** An initial idea for the methodology was to look at the points in 1D to deduce glass. The points in the captured data used in this thesis were captured in their order over time. This made it possible to see the lines the scanner captured and deduce jumps in distance and intensity. This approach could therefore be used to detect holes in the point cloud, but it was hard to distinguish any meaningful geometric information out of them. This is because such information is dependent on assumptions on how the data is captured, how the scanner moves during the capture and noise fluctuations without (much) spatial reference. The points being in the order of their capture is also not a guarantee, as the creation of point clouds does not enforce this principle. Because of this, using

such an assumption as a dependency in the methodology would lead to a significantly smaller scope. Next to this, a 1D approach will also need to be able to distinguish between specularly reflective objects, diffusely reflective objects and objects captured that are reflections themselves or captured behind glass, to be able to properly classify points in the result.

The benefit of a 1D approach would be the increase in speed such an implementation could provide. Data stored in such a format would make the analysis dependent on only the previous and next points, which makes the time complexity of the application scale with only the sample size. So should a parameter be present in the data that can properly distinguish between glass, diffusely reflective objects and specularly reflective objects, then such a 1D approach can bear fruit.

- **2D data:** As such a parameter was not present in the data used in this thesis, this thesis focused most of its attention on 2D and 3D analysis of the point cloud. A 2D view of the scene has the benefit of being able to use a vast amount of libraries available for processing image analysis. These libraries can be used to identify rectangular shapes in the data using edge and contour detection.

  Its downsides are however also apparent as point cloud data is typically 3D. A 1D overview can be created by linearly going over the points in this 3D data but for a 2D overview, a projection needs to be performed which is an expensive process and loses data like the depth of points in the conversion.

- **3D data:** An analysis of 3D data does not have these problems as the point cloud data is already present in this form. The 3D environment is however a more complex one making it harder to identify basic shapes in space than when using a 2D approach. Clusters of points can, however, be identified in this space and geometric features can be computed for them to identify the desired shapes, like for instance window frames.

  A downside of this however is that clustering a large scene and calculating geometric features for them can be a very intensive process as all points that are free in space are hard to project to just a single layer. In environments with a lot of noise, this can create a lot of artefacts or wrongly clustered objects and given that the goal of this thesis is to deduce the location of glass a lot of noise will be present.

Every view of the data has its benefits. The methodology, therefore, tried to exploit these by converting between both 2D and 3D. 2D was used to provide indications on where window frames would most likely be found in 3D space. In doing so the 3D space was split up into multiple regions of interest which are subsets of the data. The made the clustering process and the calculation of geometric features on them a lot more manageable in the end.

4.3.2 A workaround for the dependency detecting contours

After discussing the choice on how the data is used in the methodology one flaw of the method should still be discussed. This flaw is the dependency of the process on finding proper contours in 2D for the initial selection. When converting from 3D to 2D a lot of depth information is projected as if being on the same plane. This can lead to a lot of details being present in the 2D image making contour detection significantly harder. In this thesis, a lot of noise with low intensity in the point cloud was kept as low intensity is a good indicator of the presence of glass as interaction with glass reduces the intensity of the laser beam. If the presence of such noise makes detecting contours in the point cloud insufficient to produce a proper result, a workaround can be used to still acquire sufficient results.

This workaround is increasing the size of the closing kernel used after the edge detection process. When increasing this size the kernel removes a lot of accuracy and detail from the data so this should be done with caution, but it simplifies the data significantly making it easier to recognize window candidate shapes and as such creates more results although less accurate ones.

Chapter 5

Implementation and results

In this chapter, the steps taken in the implementation of the methodology for this thesis are presented. First, the software and hardware tools used during the thesis are named and their purpose of inclusion is explained in §5.1. Then in §5.2, the dataset which was collected for this thesis is shown and the steps for collection are elaborated upon. This is followed by how each step is implemented in §5.3. Finally, the results of the application of the implementation to the indoor environment are shown for multiple scanner locations in §5.4.

5.1 Tools used

In this section first, the software tools and libraries used in the implementation are presented and briefly explained in §5.1.1, whereas their full practical use is elaborated upon in §5.3. Then, the hardware used to collect and analyze data is presented in §5.1.2.

5.1.1 Software used

Python[1]

The implementation for this thesis has been done in the programming language Python. The version used for this implementation is 3.6 as the PyE57 library is dependent on this version, whereas all other used libraries also supported this version. Python has been chosen for its ease in prototyping, its available and easy to use image analysis libraries and the option to work in Jupyter Notebooks, which have been used in this thesis as it makes it easy to switch the order of code execution by running cells in different orders. Because of this, it was possible to test with multiple variables and solutions and to see how each step affects the next one without having to start from scratch once again. Especially when changing the dimensions of millions of points, having a way to ensure that such conversions are needed as little as possible has been a real timesaver while testing the implementation.

In the implementation of this thesis, the following Python packages have been used:

[1] `https://www.python.org/`

- **NumPy:**[2] NumPy is an open source project that enables numerical computing with Python. As this thesis uses a lot of conversion of large arrays of data as well as linear algebra concepts, NumPy is used extensively to speed up this process.

- **pye57:**[3] pye57 is a package created, so that .e57 point cloud files can properly be loaded intp the Python language. As the .las format is better supported with libraries such as Laspy and due to the .las format being a more common format for now although some suppliers are changing to .e57, the implementation of this thesis is made to work with .las files as input. This means that the pye57 package is used as a pre-processing step to convert data generated by the Leica RTC360 (see §5.1.2) into a .las file.

- **Laspy:**[4] Laspy is a Python library that enables reading, modifying and creating .las and .laz files. This functionality has been used a lot in this thesis to load and write (intermediate) results to files for inspection and validation of them, as well as to deliver the final result.

- **matplotlib:**[5] Matplotlib is a library that can be used to create static, animated and interactive visualizations in Python. It is used in this thesis to plot the 3D histogram data as well as to plot the 2D images generated after multiple steps. Matplotlib is chosen for this purpose due to the flexibility available when creating the plots, like modifying axes to display the proper ranges and choosing colormaps that scale with the data to enhance differences found.

- **OpenCV-Python:**[6] OpenCV is an open-source computer vision and machine learning software library. OpenCV-Python specifically is the version that is used here which is made out of pre-compiled bindings that can be used in Python to use the functions in the OpenCV library. In this thesis a lot of computer vision standards are used in the image processing part of the methodology which are mostly from the OpenCV library. The most notable functions used are the implementations of CLAHE, Canny edge detection, Morphology operators, contour detection and approximation of polygons. To make OpenCV a bit more efficient, the imutils support package is also included in the implementation of this thesis. Imutils is a series of convenience functions to make basic image processing functions like rotations and translations easier to perform with OpenCV and Python. In this thesis it is mostly used for the grab_contours function, which allows for the extraction of points along the lines found in the OpenCV contour detection implementation for better modification and inspection.

[2] `https://numpy.org/`
[3] `https://github.com/davidcaron/pye57`
[4] `https://github.com/laspy/laspy`
[5] `https://matplotlib.org/`
[6] https://opencv.org/

- **SciPy:**[7] SciPy is a collection of mathematical algorithms and convenience functions built on the previously mentioned NumPy package. It contains all kinds of useful subpackages for clustering, interpolation, spatial functions and linear algebra functions for instance. In this thesis the main benefit came from the SVD decomposition function provided in the scipy.linalg package. This provides the use of a singular value decomposition algorithm for the thesis and from this SVD it is possible to extract the PCA values, which are used to calculate the features of clusters in the methodology.

- **scikit-learn:**[8] scikit-learn is a open source machine learning Python library built upon the previously mentioned NumPy, SciPy and matplotlib libraries. It contains multiple clustering, classification and regression algorithms that can enhance NumPy and SciPy data with this functionality. For this thesis the main reason for using this library was the Density-Based Clustering algorithm it provides called DBSCAN. Using this spatial clusters can be created from the already present NumPy data and can afterwards be analyzed using SciPy algorithms making this an ideal library to use between these steps.

CloudCompare[9]

CloudCompare is a 3D point cloud processing software. It is capable of visualizing up to more than 100 million points and provides all kinds of functions to modify point cloud data inside of the software. In this thesis CloudCompare has been used for inspecting the scene, visualizing results and generating validation data which has been created by cutting desired areas out of the point cloud, namely the areas where glass is present and saving them as a validation set to compare results against.

5.1.2 Hardware used

Leica RTC360

The Leica RTC360 3D laser scanner is a Terrestrial Laser Scanner, which can generate scenes using a capture speed of 2 million points per second with a range from 0.5m to 130m, with an accuracy below the centimetre within the testing space. Colour information is added with 36 MP HDR images acquired with a 3 camera system after which the point cloud is coloured according to the captured images. The process of capturing a scene with colour information is done in less than 2 minutes.

The RTC360 is a lightweight solution that is easily placed in new locations and once it is placed correctly the scan can be performed with a single button press.

It is also possible to use a GNSS fix when capturing the data but as this thesis focuses on the indoor environment such a fix was not necessary to get the results as all data is tested locally within the same set. If multiple scans were to be performed using the same scanner then all data is linked together using the

[7] `https://scipy.org/`

[8] `https://scikit-learn.org/stable/` and (Pedregosa et al., 2011)

[9] `https://www.danielgm.net/cc/`

internal coordinate system, as was the case in this thesis. Should multiple sets be made in different intervals or with separate scanners then GNSS is needed to match them or manual matching needs to be performed.

Figure 5.1: The Leica RTC360 3D laser scanner is used for Terrestrial Laser Scanning during this thesis [10].

Computer

The dataprocessing and analysis performed during this thesis have been done on a Laptop with the following specifications:

- System model: Dell Latitude 5580
- Operating system: Microsoft Windows 10 Enterprise
- Processor: Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz
- Physical Memory (RAM): 32,0 GB
- System type: x64-based PC

[10] The image on the left is acquired from `https://leica-geosystems.com/nl-be/products/laser-scanners/scanners/leica-rtc360`. The image on the right is from the day the data was captured.

5.2 Dataset used

Decembers last year, Leica helped with the creation of this thesis by bringing multiple scanners to the faculty of Architecture and the Built Environment at the TU Delft to collect data in the "Orange Hall" (see figure 5.2)

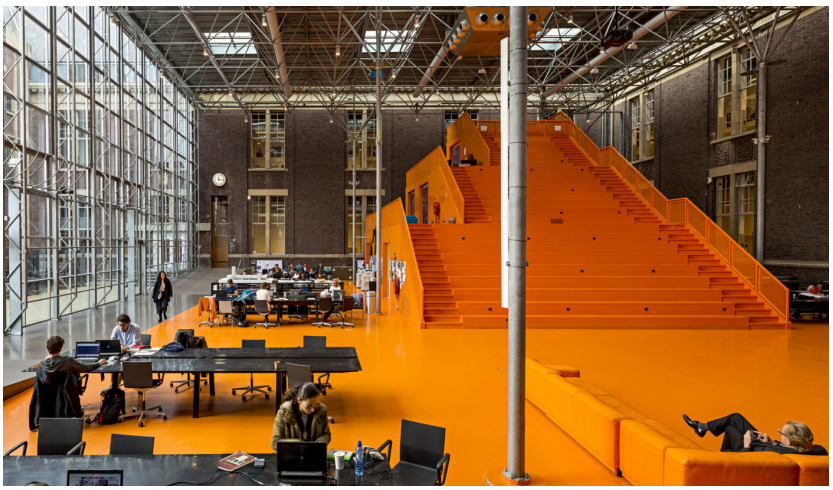


Figure 5.2: Depiction of the Orange Hall at the faculty of Architecture and the Built Environment, which is used as the test scene during this thesis[11].

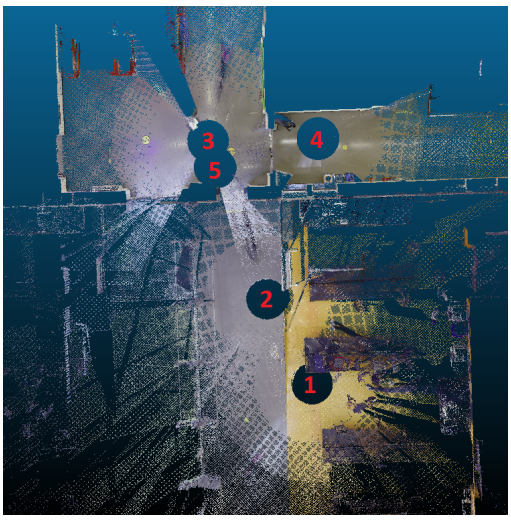
This hall is a well-known location in the TU Delft which has also been the testing location of other indoor focused theses, such as the work of Staats et al. (2017) and Flikweert et al. (2019). As stated in §5.1.2, the datasets depicting the Orange Hall shown in figure 5.3 are captured using a Leica RTC360 scanner. Each scene produced has upward of 40 million points and there has been no filtering out of points usually labelled as noise, for which the reason is that these points are actually points of interest in this research as reflections typically have a lower intensity which is one of the main criteria used to determine whether a point is considered noise or not.

The main focus for getting these scans was to capture the glass wall shown on the left in figure 5.2, as well as the windows in the back of the hall. These regions were chosen to see if it is possible to deduce the location of the glass windows that act as a barrier on the outside of the indoor environment. The windows on the inside of the building are also of interest as here distances to the scanner are shorter and lighting conditions are more similar, which changes the process of detecting reflections.

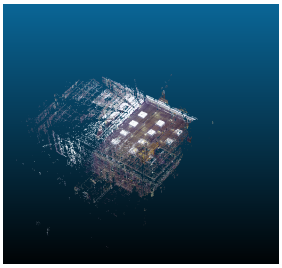
A total of 5 scans were performed as shown in figure 5.3 with the scanner positions, that are used as reference positions, for each scan being shown in figure 5.3a. The reason why there are no points in the circles around the scanner position is that the scanners in their own scan cannot capture this area due to occlusion by the tripod itself and overlap from data found in other scans has been removed in this image for clarity as to where the scanner was positioned for each scene.

A general overview of the scenes is given in table 5.1.

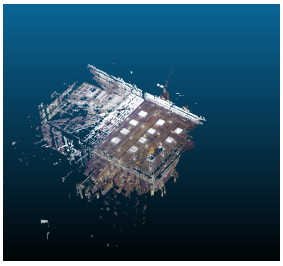
[11] Acquired from `https://www.braaksma-roos.nl/project/bk-city/`.



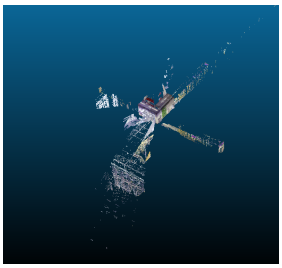
(a) Zoomed-in view of the positions of where the scanner was placed during the capturing of the scenes shown in b through f



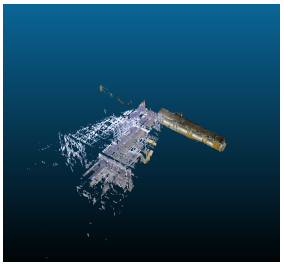
(b) Scene 1



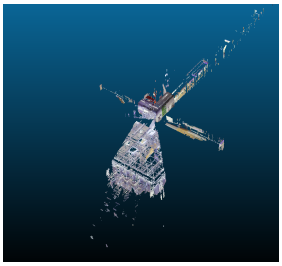
(c) Scene 2



(d) Scene 3



(e) Scene 4



(f) Scene 5

Figure 5.3: Overview of all scenes used in this thesis capturing the Orange Hall

Dataset	Nr of Points	Location
Scene 1	41.66 million	Inside Orange Hall
Scene 2	41.62 million	Inside Orange Hall
Scene 3	41.02 million	Next to Staircase
Scene 4	41.01 million	In Hallway next to Orange Hall
Scene 5	40.88 million	Next to Staircase

Table 5.1: Summary of the number of points and location of scanner position for the scenes captured.

5.3 Implementation

In this section, the steps of the methodology are explained one by one and results as well as details regarding the exact implementation during this thesis are given.

5.3.1 Input Data

The data acquired from the RTC360 after scanning was stored in the .e57 format. As the methodology was created to work with the ideology in mind that it could be used on as much input as possible and .las is the simpler and more common format for now, a convertor from pye57 was created. This was achieved using the pye57 library and the Laspy library.

The e57 format is capable of storing multiple scans in one file. Each of these scans is essentially a separately stored point cloud that is geo-referenced to the rest. As the .las format is not capable of storing multiple scans the converter from pye57 to las split the pye57 file into 5 separate scenes and saves the scanner position to a separate text file for later use as a reference position.

In the methodology each scene is loaded and processed separately so for the explanation of the rest of the steps, only scene 1 is covered.

5.3.2 Calculate Euclidean Distances

The first step of the methodology to perform is to calculate the euclidean distances for all points in the point cloud to the reference position. The data loaded using Laspy is stored as NumPy arrays containing a single property from all points in the point cloud. By taking the X, Y and Z attributes of these points the Euclidean distance is easily calculated using equation (4.1).

5.3.3 Convert 3D point cloud to 3D Histogram to 2D Image

The conversion from point cloud to histogram was performed as follows. First to represent the data in the histogram an empty matrix was created to represent the latitude and longitude spread. Normally latitude and longitude are in the range of -90 to 90 and -180 to 180 respectively.

However, to get sufficient accuracy and detail in the scene the matrix represented this as a range of -900 to 900 and -1800 to 1800, increasing it by a factor of 10, resulting in a 1800 by 3600 matrix. Each cell in the matrix is then filled with an empty list. Using equation (4.2) and equation (4.3) each point is given a latitude, longitude coordinate created from its X, Y and Z-coordinate, which is used to put the point in the correct list. Once all points have been processed the 3D histogram is converted to a 2D image by taking the Variance of all points in each list as pixel value for the 1800 by 3600 image.

5.3.4 CLAHE

For the application of CLAHE to the variance image, the createCLAHE function of OpenCV was used. This function takes two notable parameters being the tileGridSize and the clipLimit.

The tileGridSize turned out to be a rather forgiving setting in this function as the input image is a rather large image of 1800X3600 pixels. The tileGridSizes supported by the method were in the ranges up to 256x256. In figure 5.4 the comparison between the default of 8x8 and the maximum of 256x256 is shown and no real noticeable difference could be found. Therefore to have the local changes be higher the default of a tileGridSize of 8x8 was chosen but larger sizes would work just as well in this context. This meant that the image was split up into tiles of 8x8 and CLAHE is applied in each of these tiles after which the result is smoothed along the tile borders, by using bilinear interpolation.

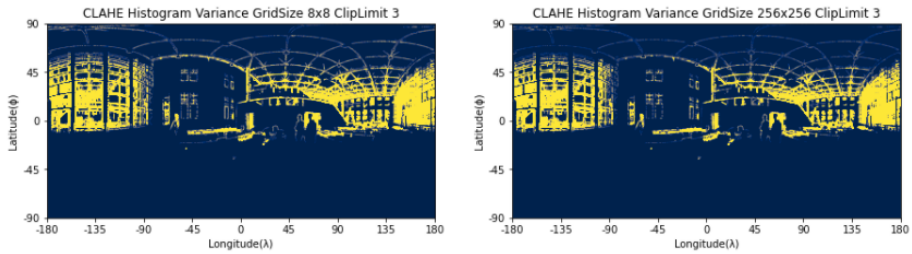


Figure 5.4: Comparison between a tileGridSize of 8x8 and 256x256.

The clipLimit on the other hand had more significant changes depending on its value. The clipLimit is the threshold for contrast limitation in the CLAHE process. This means that it clips peaks in the histogram that go over this limit and redistributes the values higher than the limit to the rest of the histogram ensuring visible differences. This limit is a factor relative to its neighbourhood meaning typical values for this clipLimit lie between 3 and 4 to ensure the cumulative distribution function of the histogram stay relevant.

In figure 5.5, figure 5.6, figure 5.7, figure 5.8 and figure 5.9 the experimented with values are shown. As can be seen, the contrast of the figures increases with a higher clipLimit. CLAHE finds it advantageous not to discard part of the histogram but to redistribute it equally among all other bins, which is visualized in figure 5.10. The data above the limitation, shown in purple is redistributed to the histogram pushing making lower contrast more apparent. This can however cascade if enough data gets pushed over the limit again by the redistribution of data, meaning it will go on until a less significant change has occurred. In figure 5.8 and figure 5.9 cases are shown when this cascading went over its intended purposes. In figure 5.8 a lot of the background data gets a larger response than desired because the cascading kept on going recursively until over 80% of the image was at the maximum. In figure 5.9 this did not stop in time making every value essentially the same resulting in every pixel getting the minimum colour value. As subtler contrast is desired as this can still identify potential window candidates the highest clipLimit that does not cascade too much is chosen, resulting in the choice of a clipLimit of 3.

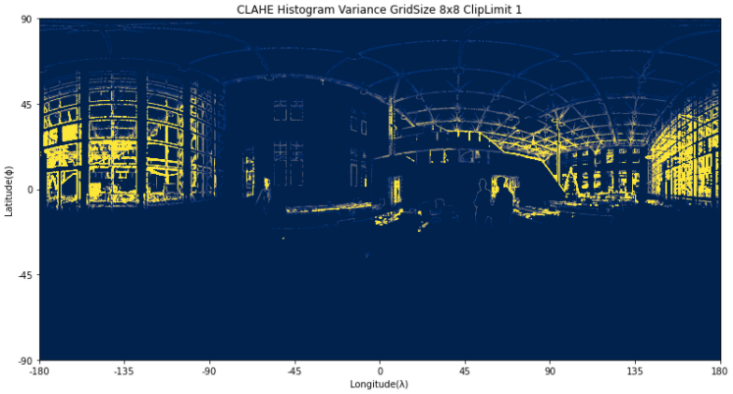


Figure 5.5: CLAHE with a ClipLim of 1.

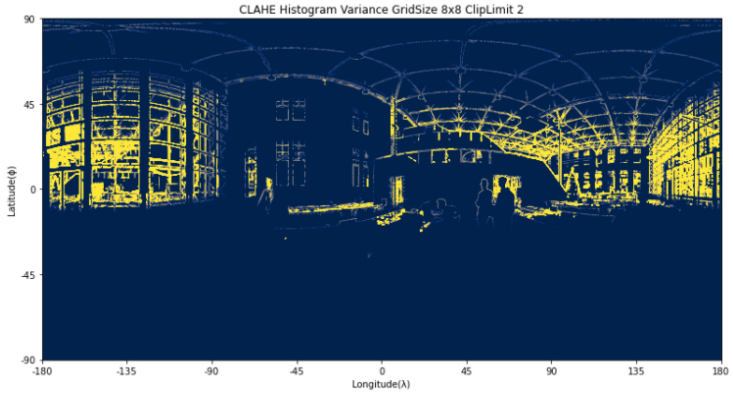


Figure 5.6: CLAHE with a ClipLim of 2.

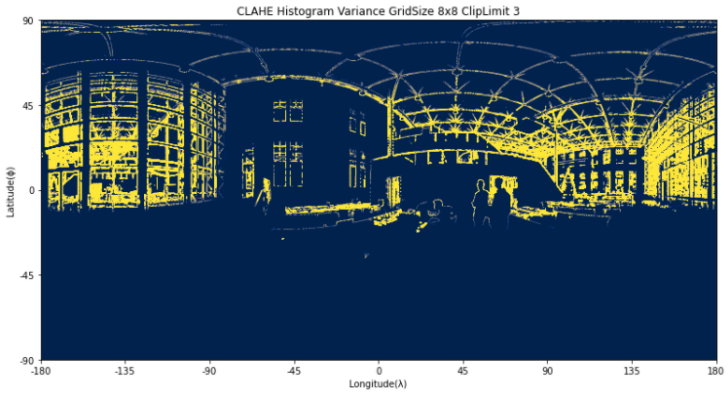


Figure 5.7: CLAHE with a ClipLim of 3. What can be seen here is that the contrast in the middle of the image jump out more than in figure 5.5 and figure 5.6 because of the redistribution of the histogram values as explained in figure 5.10.

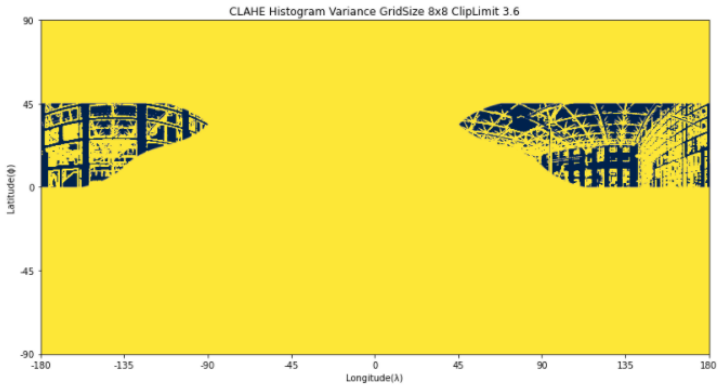


Figure 5.8: CLAHE with a ClipLim of 3.6. Here the effects of cascading redistribution of the histogram are shown making the result flooded with maximum values, limiting the insight in the local contrasts significantly.

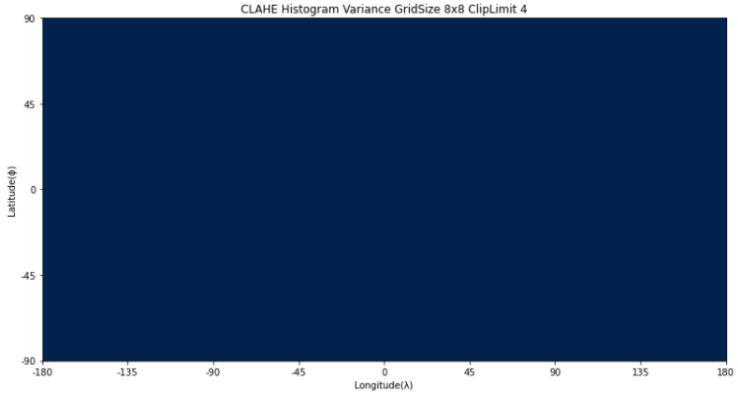


Figure 5.9: CLAHE with a ClipLim of 4. Here the cascading redistribution of the histogram eventually pushed all pixels to the clipLimit making the whole image have the same pixel values.

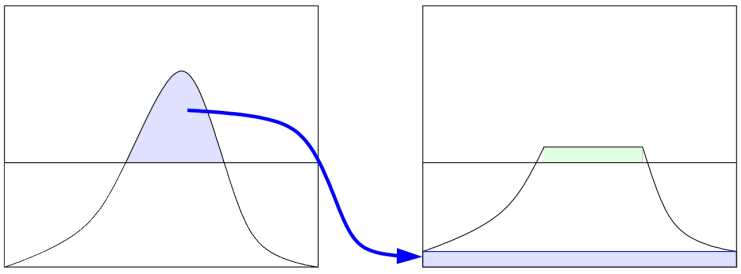


Figure 5.10: Visual example of the redistribution of the histogram. The part of the histogram above the clipLimit shown in purple on the left gets redistributed evenly over the histogram enhancing lower contrasts. This can lead to a new area being raised above the line as shown in green on the right. If this area is significant enough the process repeats until the area is not significant enough or the whole histogram is equal[12].

5.3.5 Canny edge detection

For Canny edge detection once again the OpenCV implementation is used. For this implementation, a minVal and a maxVal threshold for the hysteresis procedure of Canny edge detection needed to be chosen. In figure 5.11 this procedure is explained with a visual example. The minVal was decided to be 0 as no points should automatically be excluded in this noisy environment as that can lead to cut up edges where connections might otherwise be found. For the maxVal, some experimentation was performed as shown in figure 5.12. It is desired that as much noise disappears from the scene without losing window edges so, in the end, the choice became a maxVal of 20 as this ensured that weak edges of windows could still be retrieved whereas a maxVal of 30 led to the removal of some windows already.

Once a result has been created from Canny edge detection on the CLAHE output, morphological operators are used to connect some of the dangling edges. In the implementation, a 3x3 kernel shown in figure 5.13 is used in the closing operator. This means that this kernel is moved over all pixels in the edge detection result, first dilating everything with the plus sign and then eroding it. The plus shape is chosen to enforce straight-line connections a bit more which is expected when trying to find rectangles in data, but overall results will be similar to using a square kernel except for at the ends of the edges.

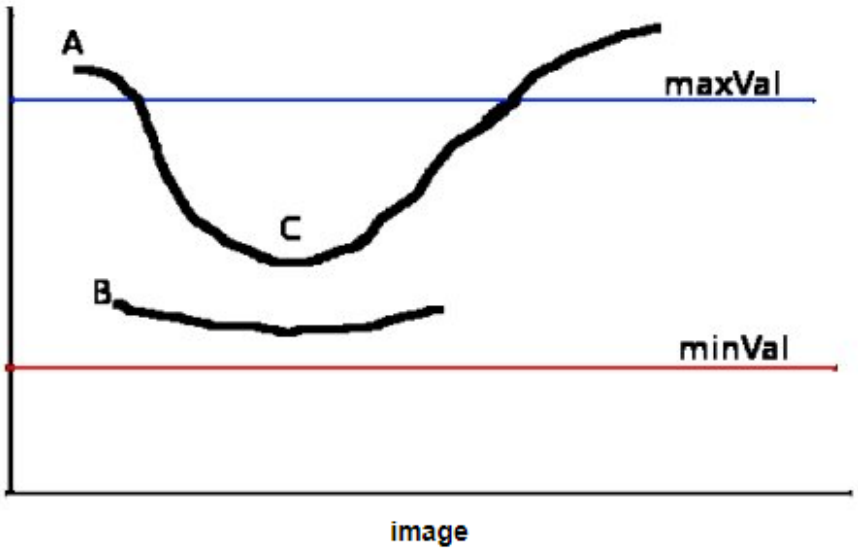


Figure 5.11: Visual example of hysteresis thresholding. Any point in the example that is above the maxVal line, like A, is automatically considered valid and kept in the output of the Canny edge detection. Any point below the minVal line is automatically considered invalid. For points B and C an additional step is needed to see if they are kept. C is connected via points to points above the maxVal line meaning that C is kept in the result as well even though it had a weaker response. For B on the other hand there are no points in its edge that are above the maxVal line meaning that the edge of B is discarded.

[12] Acquired from `https://en.wikipedia.org/wiki/Adaptive_histogram_equalization`

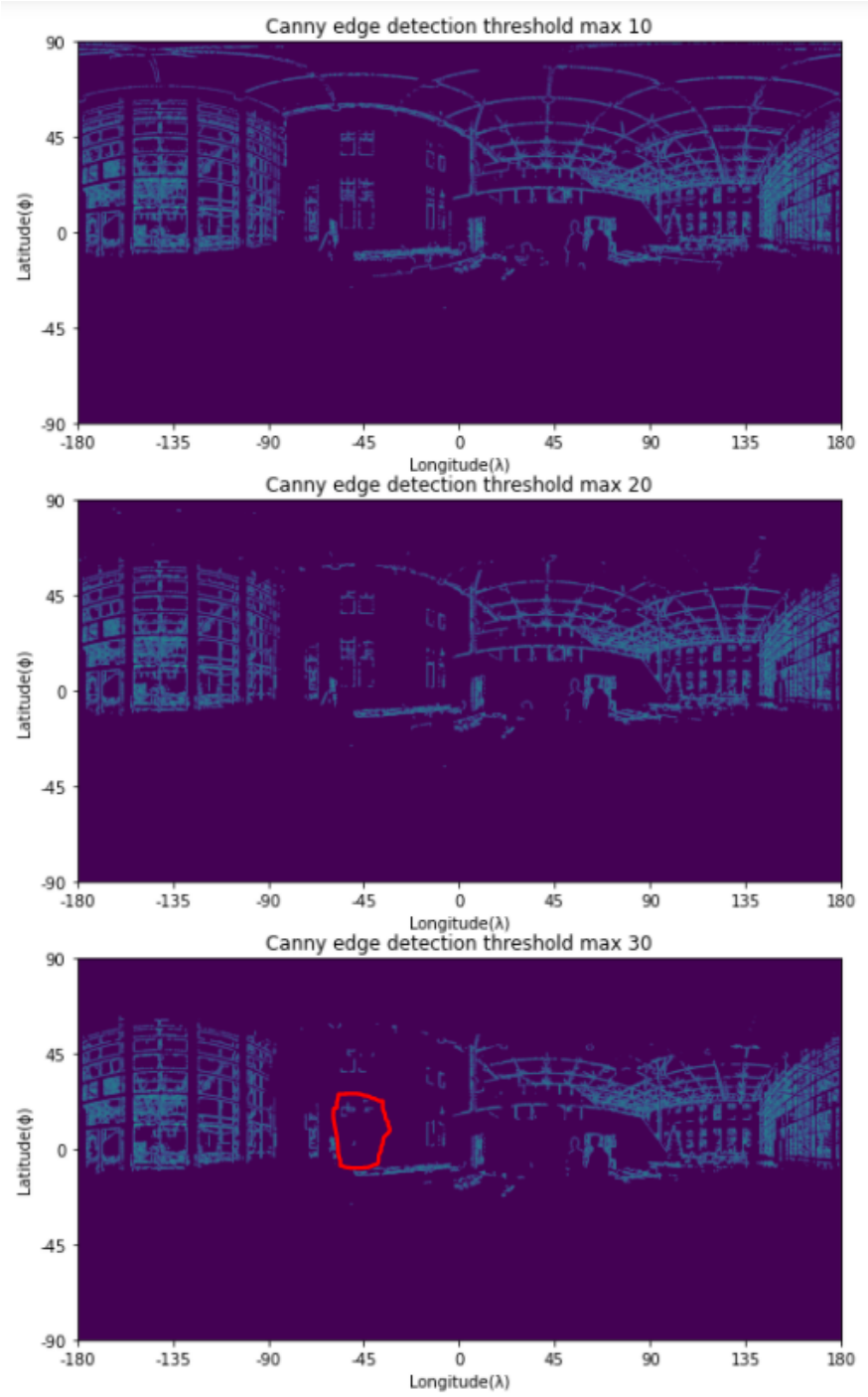


Figure 5.12: Threshold comparison for Canny edge detection. As can be seen, the maximum threshold has been gradually increased when moving down the images above. Increasing the maximum threshold of the hysteresis process removes weak edges from the result as anything above the threshold is accepted in the result. In the image with a maximum threshold of 30, it can be seen that windows start to disappear in the red circle, meaning the threshold was too high.

0	1	0
1	1	1
0	1	0

Figure 5.13: A 3x3 kernel used in the closing operator. The plus shape is chosen to remove some emphasis on diagonal connections as straight line connections are expected when looking for window candidates.

5.3.6 Contour Extraction

To find the contours in the image the OpenCV implementation findContours() is used. This function allows for multiple retrieval and contour approximation modes. For the retrieval modes, there were essentially two different results, the one generated by cv2.RETR_EXTERNAL and the one resulted by the other retrieval modes. In figure 5.14 and figure 5.15 the differences between the results are shown. As can be seen in the blue outlines the result in figure 5.15 finds more contours than the results in figure 5.14. One might argue that this is better as the retrieval modes aside from external find more contours, but for the purpose of finding the window candidates, these background contours are actually more harmful than useful. Therefore the less complete but more strict external retrieval mode cv2.RETR_EXTERNAL is chosen for the implementation.

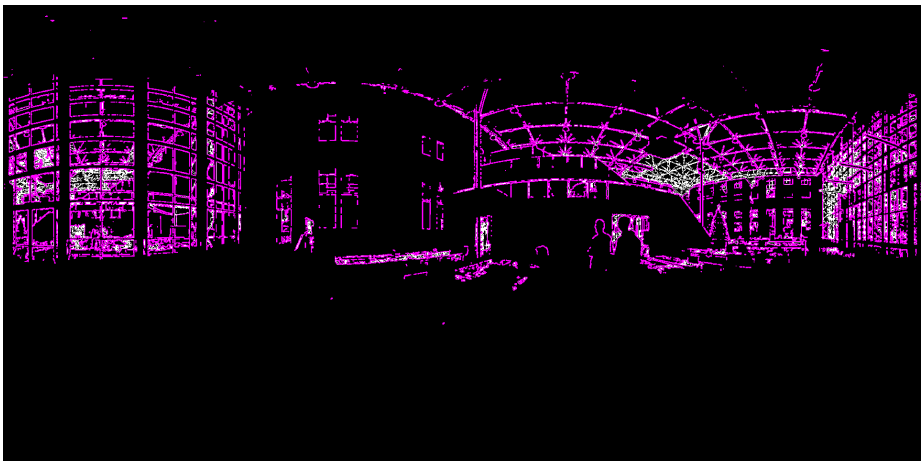


Figure 5.14: Contour detection using the external retrieval mode cv2.RETR_EXTERNAL.

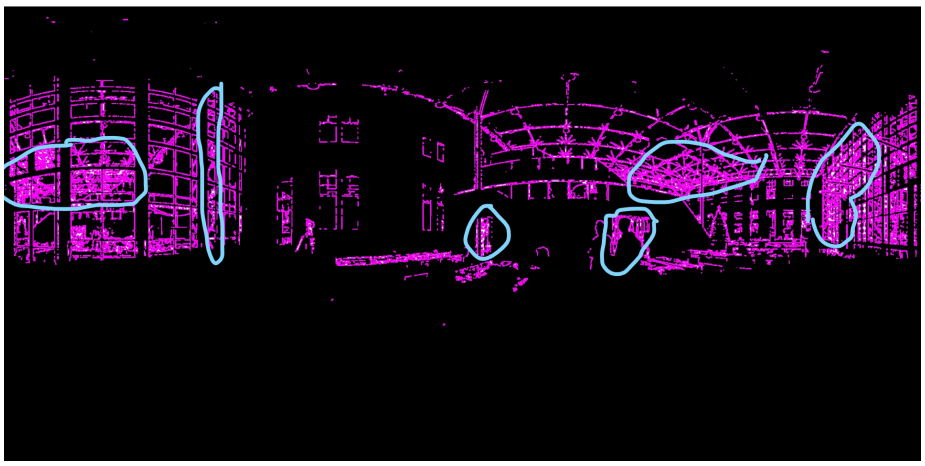


Figure 5.15: Contour detection using the external retrieval mode cv2.RETR_LIST. The results shown in this figure are similar to the results when using the other retrieval modes cv2.RETR_CCOMP, cv2.RETR_TREE and cv2.RETR_FLOODFILL. The areas in the blue circles are the most notable areas where contours are detected which are not found in figure 5.14.

For the contour approximation modes, there was hardly any difference between the found results with the exception of cv2.CHAIN_APPROX_SIMPLE being slightly more efficient as it approximates simpler shapes. At first, the idea of picking this option above the other was to better approximate simple line shapes and skip the more complex background noise but it turned out to be the case that the simple contour approximation mode also detected these so in the end any mode contour approximation mode could have been chosen here.

Once the contours have been generated they are collected using the imutils.grab_contours function. This list of contours is the first indication for candidate windows but still needs to be validated.

5.3.7 Rectangle Validation

Each of the contours found in the previous step is tested to see if it is a rectangle and thus suffices as a candidate window. Using the OpenCV approxPolyDP function the polygon around the contour is approximated and simplified using the Douglas-Peucker algorithm. If the approximated polygon is a rectangle then it should contain just 4 points to indicate its corners. These points are then checked to see if they lie within approximately 90 degrees with an error margin from each other. This error margin can be quite large as all corners need to fulfil it. In figure 5.16, figure 5.17 and figure 5.18 the valid candidate windows are shown in green and the rejected ones are yellow, similar to the situation in figure 4.10. The difference between the figures is the error margins they used which is the margin that the corners of the approximated polygon can deviate from 90 degrees. In figure 5.16 the error margin is set to 15 degrees which results in only 3 window candidates being found whereas other correct window candidates were still in the scene, so this selection was a bit too narrow.

In figure 5.17 the error margin is increased to 45 degrees and a lot more of the contours were labelled as correct. Some of these however are not the desired objects and as such are considered to be false positives

In figure 5.18 the error margin is put in the middle with 30 degrees which filters out some of the outliers while keeping the correct contours. When testing, however, the turning point of getting rid of the last outliers was the same as losing the other correct contours meaning that these were hard to separate at this stage. This specific tipping point was between 26 and 27 degrees as error margin but to make the method a bit more general an error margin of 30 was used for the rest of the scenes, meaning that as long as all corners of the contour are between 60 and 120 degrees the contour is considered a valid candidate window.

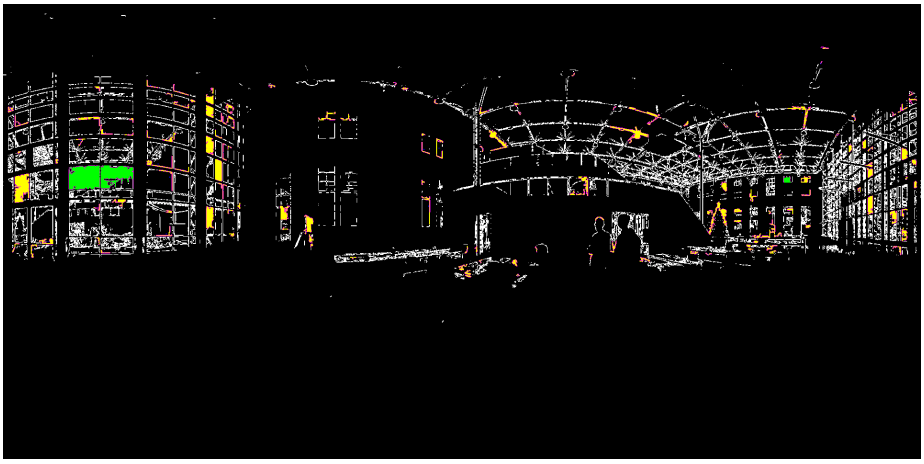


Figure 5.16: Rectangle validation with an error margin of 15 degrees. Only 3 contours are labeled as candidate windows here which is too narrow.

Figure 5.17: Rectangle validation with an error margin of 45 degrees. A lot more contours are labeled as correct but this did introduce false positives into the results as well.

Figure 5.18: Rectangle validation with an error margin of 30 degrees. This is the middle ground chosen for the implementation during this research.

5.3.8 Acquire regions of interest based on candidate windows and cluster them

When the contours have been found regions of interest in 3D space are created again. This is done by taking a bounding box around the found contour to properly align it with the image grid and then a small margin around the bounding box is taken to ensure the window is in the result. Once the proper cells have been defined these points can be retrieved from the original matrix and form the region of interest in which the window is supposed to be found. To find the windows first the region of interest is clustered using DBSCAN. DBSCAN is a part of the sklearn.cluster module and can perform density-based spatial clustering.

For the implementation of DBSCAN, an eps of 50000 is chosen which represent a distance of about half a meter in the dataset, which connects any points that are within that range of each other. The minimum sample, also known as the minimum number of points, required for a valid cluster is set to be 1000 as smaller clusters than this tend to be too small to be able to represent a window using TLS data. While testing with these numbers the ones above have been chosen to best match the dataset representing scene 1, where an eps of 50000 was sufficiently large to ensure that close groups of points were properly clustered together unlike what is shown in figure 5.19 and a minimum sample of 1000 was enough to ignore the clusters shown in red in figure 5.20 increasing the processing speed significantly.

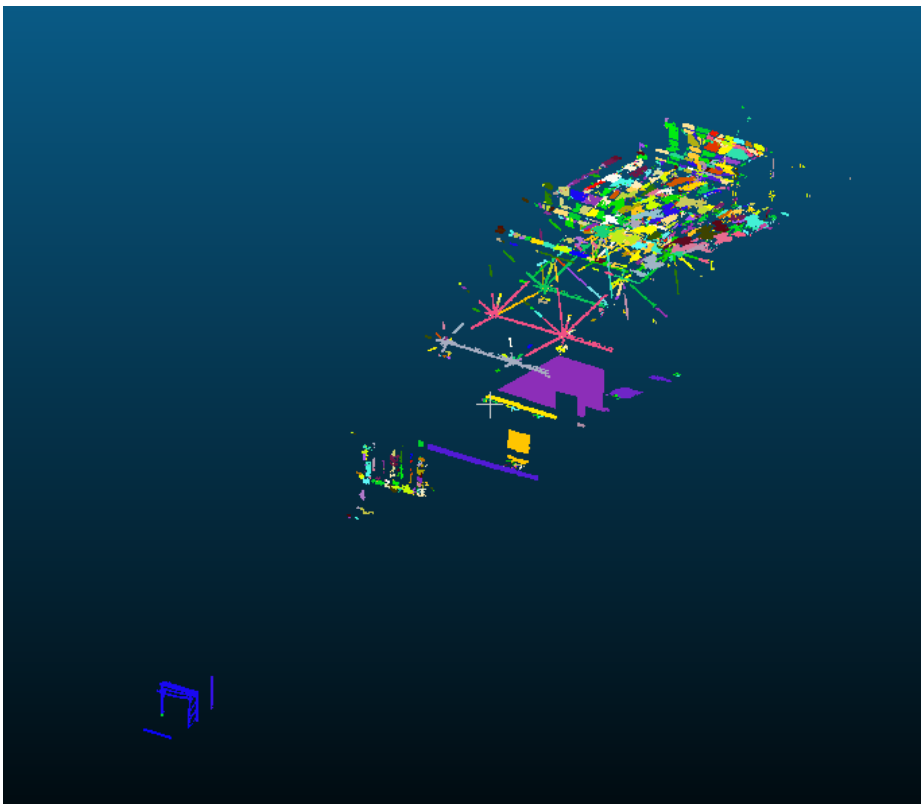


Figure 5.19: Visual example of the result of clustering using DBSCAN with an eps of 5000 and a minimum sample of 0. As can be seen, a lot of very small groups are found indicated by their different colours. Not only should these groups be considered connected in most cases, but they would also be separately processed in the next steps significantly increasing the workload of the program.

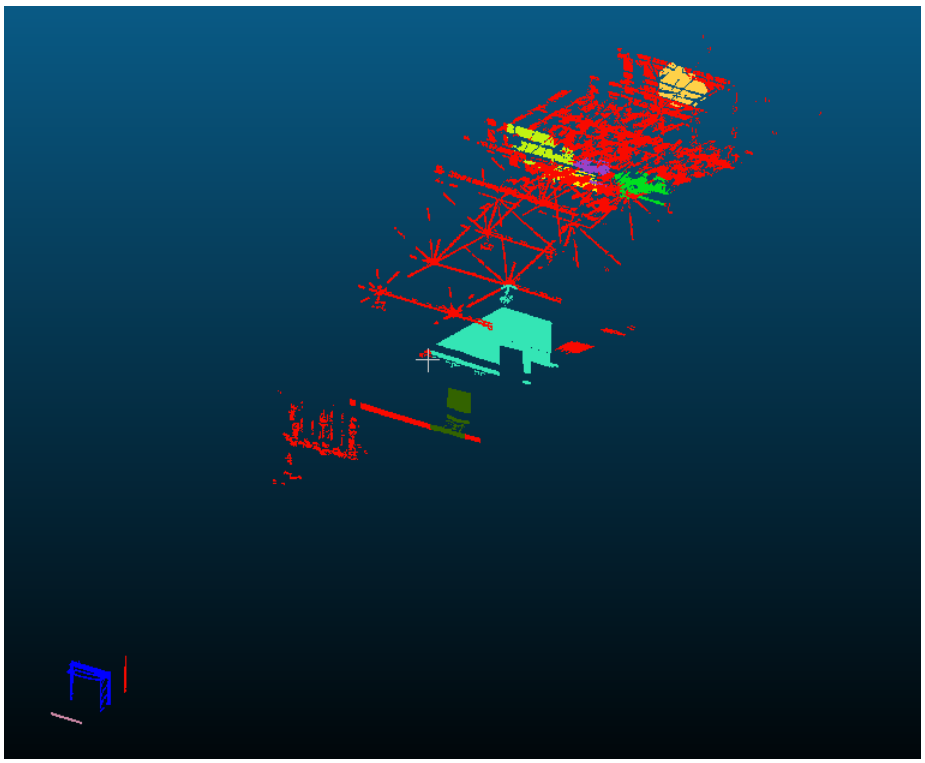


Figure 5.20: Visual example of the different clusters found using an eps of 50000 and a minimum sample of 1000. The points displayed as red are excluded from the further analysis process as they are to small and therefore rejected.

5.3.9 Deduce windows in regions of interest based on geometric properties

Finally, the last step in the methodology is deducing which cluster in the region of interest represents the window. This step is performed using the geometric properties Linearity, Planarity, Sphericity, Change of Curvature and Verticality whose formula's are found in §4.2.11. But before this can be done the eigenvalues of the clusters need to be calculated. This is done using the singular value decomposition implementation of SciPy.linalg, after which they can be found by squaring the values found along the diagonal matrix which is ordered to have the largest eigenvalue to the left.

After testing it seemed that a Linearity closer to 0.5 rather than 0 seemed to be a good indicator of a window. The Change of curvature and Sphericity criteria on the other hand turned out to be less decisive in getting a result that matches a window. This resulted in the weights for calculating the likelihood of the cluster being a representation of a window to be 0.3 for Linearity and Planarity, 0.2 for Verticality and 0.1 for Change of Curvature and Sphericity.

The points found by this methodology then depict the frames of the windows, deducing the location of glass in the point cloud.

5.4 Results

After performing the steps in the implementation part the following results were generated. These results led to insights into problems that arise with looking at 3D scenes from a 2D perspective, which became apparent applying the steps of the methodology that had been tested on scene 1 on the other scenes. This led to a great divide in the quality of results that were generated. Therefore, this section will first show and discuss the results found in scene 1, which symbolizes a case better suited to the methodology in §5.4.1. After this flaws of the methodology are shown in §5.4.2. Finally, the application of the workaround presented in §4.3.2 to the scenes with flaws is presented in §5.4.3.

5.4.1 General results

The results of scene 1 following the methodology led to the detection of 11 candidate windows. This is a lot less than the actual number of windows present in the scene as can be seen in figure 5.2 on the left side. This is due to the detection of contours at the side of the window which has been kept very strict with what is considered a valid window outline to avoid accidentally capturing other spaces.

The results of the deduced window locations are shown in figure 5.21 as a general overview with all 11 sub-results shown as a whole. The clusters shown in red in the results are indicated as being likely to be windows within their subset of data, whereas the clusters in blue are labelled to not be likely. This set on its own is a bit hard to place in the context of the scene so in figure 5.22 the same results are shown but now with scene 1 also being represented without the roof, so the results are still visible.

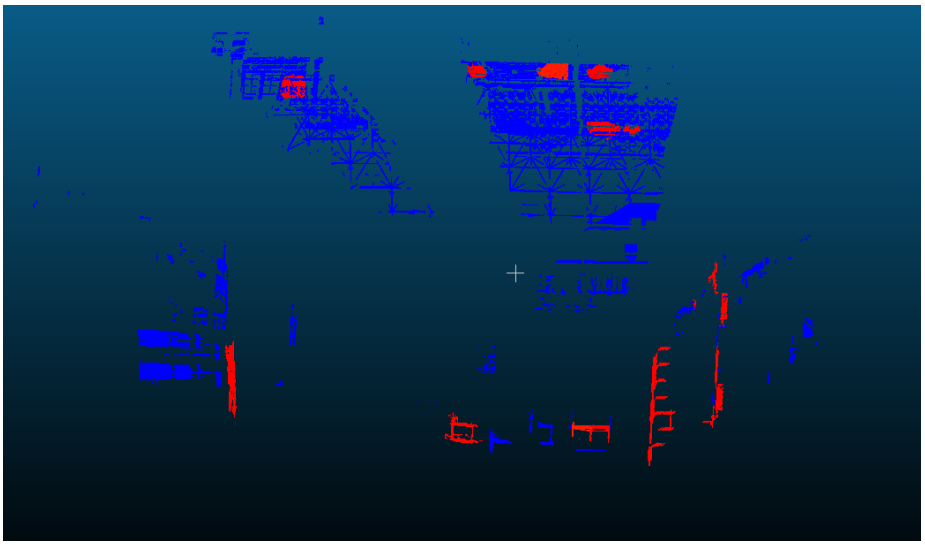


Figure 5.21: Overview of the results from scene 1. In the results the clusters of 11 candidate windows are shown along with their likelihood of being a window, where red means a high likelihood and blue a low likelihood.

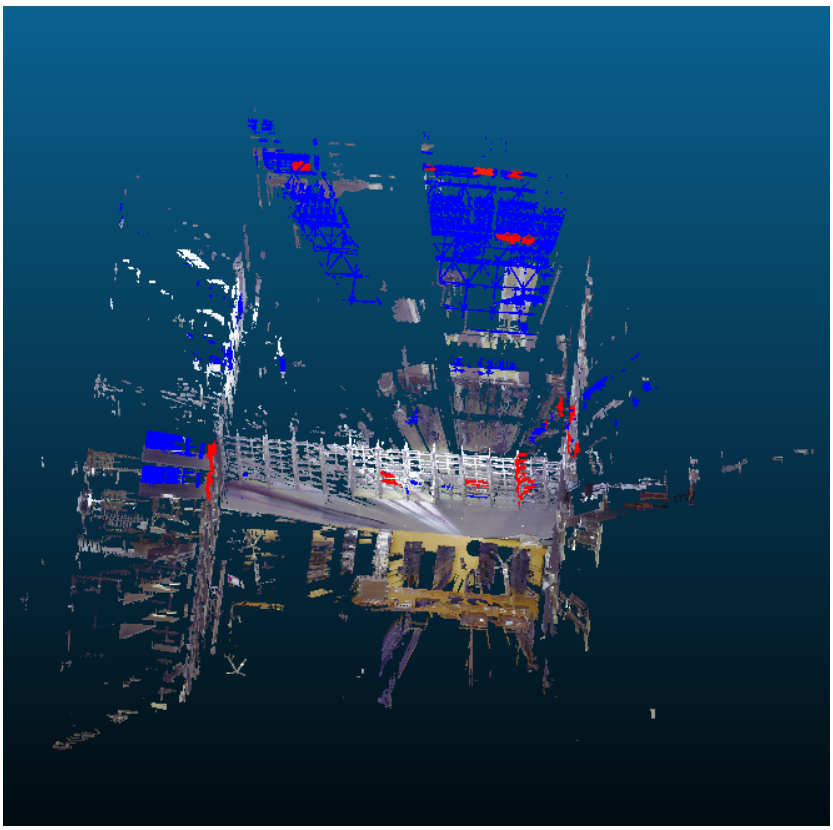


Figure 5.22: Overview of the results on scene 1 within context of the scan without the roof for additional reference.

As can be seen, the labelling of window clusters did not produce a completely accurate result. Multiple clusters are falsely labelled as windows and some parts of windows that are captured are not properly labelled at all.

Out of the 11 sub-results, only 6 had a proper window in them, from which 5 out of the 6 have correctly been labelled as windows. So when finding a cluster with a window in it the results (although little in volume) are as expected. Nonetheless, a lot of mistakes are present in the results.

First and foremost, one of the windows that was expected to be found was not. This case is shown in figure 5.23. For some reason the intended to be found cluster in the lower right scored very poor on the linearity and planarity criteria which were very high with the other similar structures in the data. This is why this cluster was missed during the deduction. The cluster in the back also managed to be a false positive, something that occurs more often in the results generated by this methodology. What seems to be the case with these false positives is that they are flat surfaces, which score similarly to windows in linearity, planarity and verticality test. This means that the normally definite criteria to identify windows are also identifying these clusters making it hard to distinguish between the windows and the flat rectangle like surfaces.

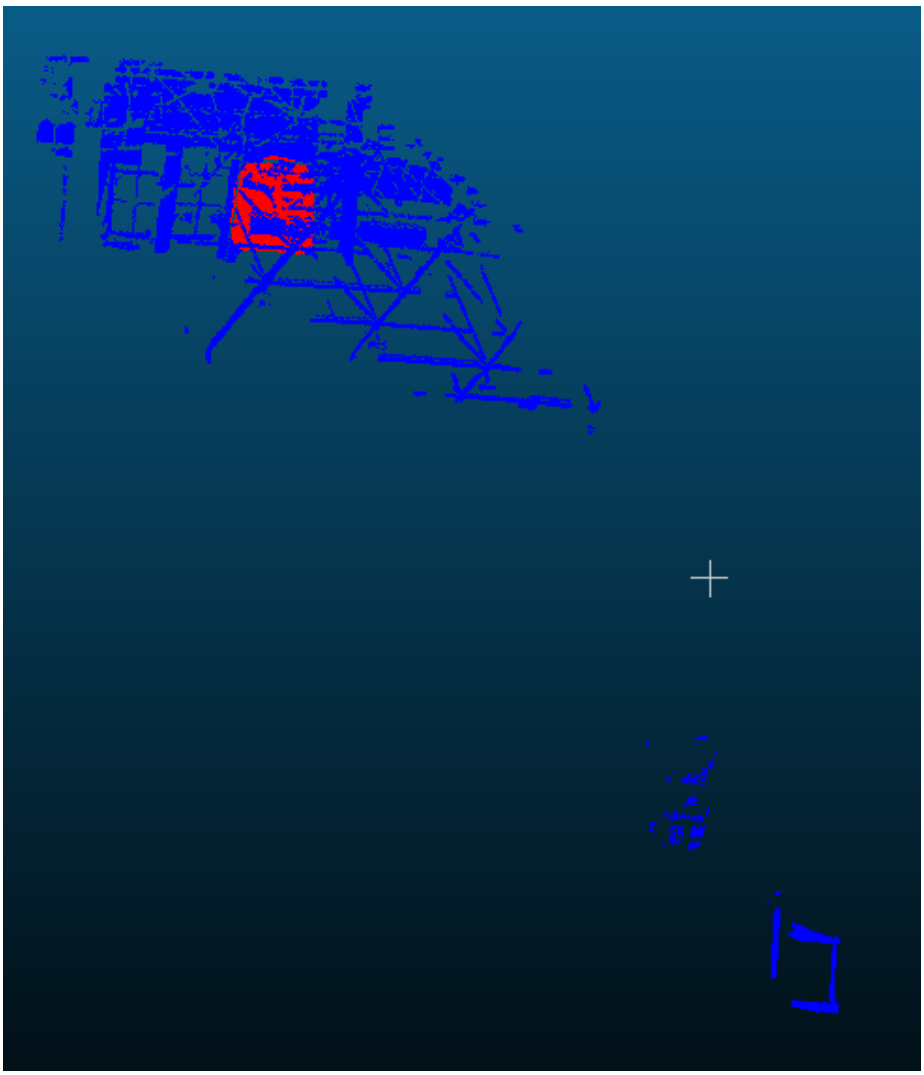
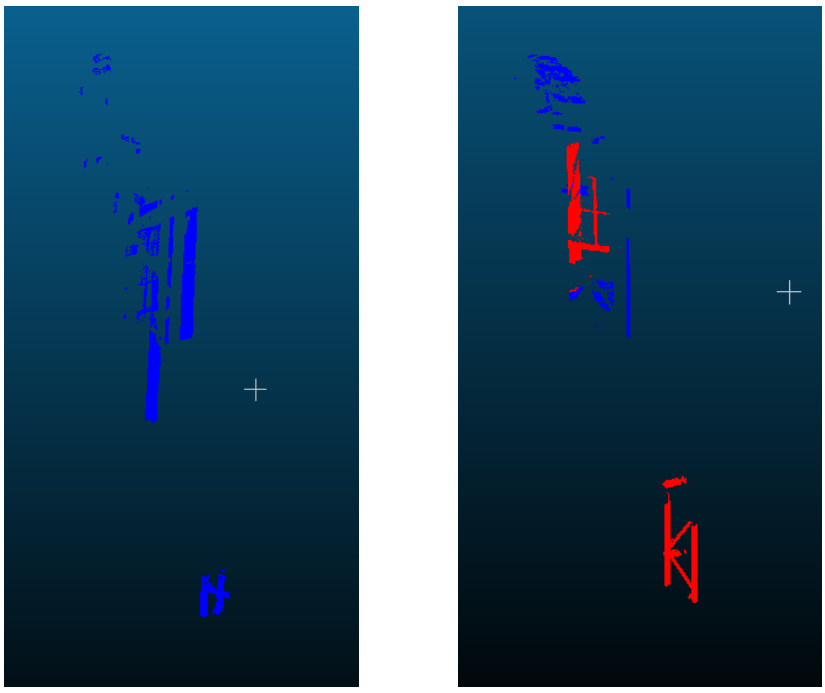


Figure 5.23: The cluster labeled as likely to be a window is in the back, whereas the intended cluster in the lower right is not labeled as a likely candidate

The next outlier is shown in figure 5.24a. In this region of interest, no matches have been found which is correct, but this does mean that improper contours were detected as candidate window. In the Orange Hall, there are a lot of metal beams very close to the windows which got detected during the contour extraction as if they were a window candidate themselves. This is most likely what happened in the case of figure 5.24a but no improper matches occurred afterwards. This is however not always the case as can be seen in figure 5.24b, where both a window in the back (which is most likely captured behind glass itself but a window nonetheless) and a piece of the beam are labelled as likely windows. In this case, the part of the beam that was captured followed the same logic as the flat surfaces, in that it scored similarly to windows on linearity, planarity and verticality because of the small part that was captured in this view.

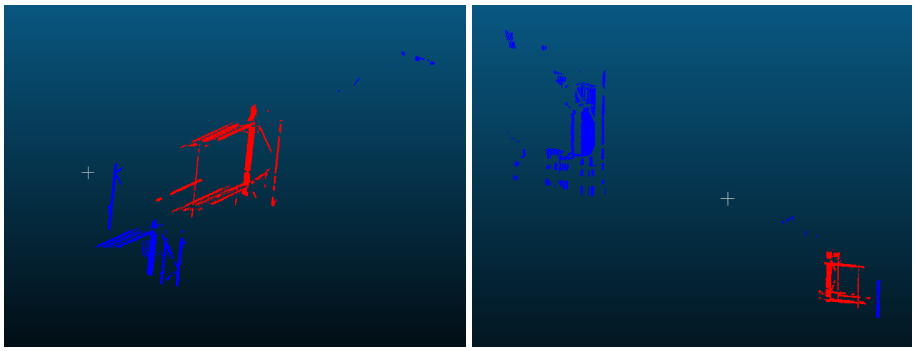


(a) In this region of interest no matches have been found.

(b) In this region of interest a beam has incorrectly been deduced to be a window.

Figure 5.24: Interference of metal beams on window deduction.

Another area where beams turned out to be a hindrance is in the initial clustering process. As the beams in the Orange Hall are very close to the windows on the side, windows and beams were often clustered together. So even though the result in figure 5.25b is expected and correct, more is found than had to be the case. In figure 5.25a the same cluster is shown but zoomed in and from the other side to properly show the merging between the window and the beam. To deduct the presence of a window, this result will more than suffice but it is not the cleanest result possible.



(a) View from the back to show the connection between the beam and the window

(b) Result is properly deduced but due to the clustering a beam is merged to the window

Figure 5.25: Results found are sometimes merged with beams that are closeby.

Another case where the cluster found turned out to be not the cleanest can be seen in figure 5.26. Here the area found by the contour was simply too large. The deduction still worked in this case as the surrounding area was flat and vertical but it is a stretch to say that the windows have been properly found in the end.

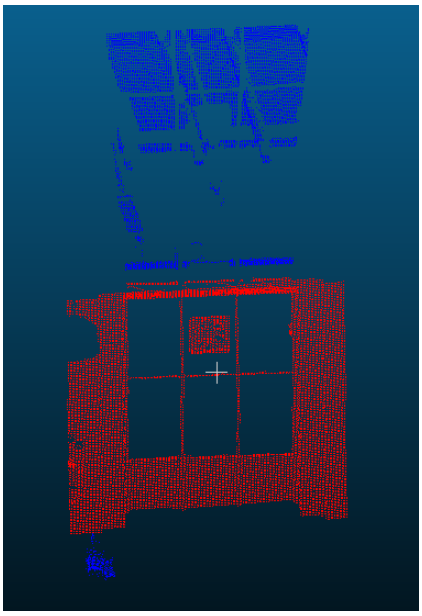


Figure 5.26: Window has been found but a lot of points around it have also been added meaning the contour was to large at the start, so a good deduction has been performed on a too broad initial selection

5.4.2 Flaws of the methodology

As shown in the general results the methodology provided is capable of generating results for deducing the location of windows in the scene even with the inherited limitations of the difficult scene itself, like points that are too close together to decently be split apart and the similarities between windows and flat isolated surfaces or steel beams. However, the flaw of dependence on finding proper contours in a noisy environment mentioned in §4.3.2 appeared when applying the methodology on scenes 2 through 5.

To show the extents of this and the separate reasons for this, the contours found in scene 2 are shown in figure 5.27. Scene 2 is captured quite close to Scene 1, so the two were expected to deliver similar results, but this turned out to not be the case entirely. In the process up to the contour detection step, most of the edges found in the analysis of scene 2 that were not connecting properly could not be fixed even with the closing morphological operator. This resulted in a lot of unconnected edges in the data, instead of connected rectangles.

Noise behind the windows, caused by objects behind them or reflections, is also one of the reasons for this as this gets connected to the edges of the windows in the scene. This makes it hard for the contour extraction algorithm to extract the proper contours along the edges and can create earlier cuts in data or join wrong clumps of data together. This issue was also present in scene 1 but to a lesser extent enabling it to still generate results.

There is also the issue that scene 2 was captured closer to the windows. This makes the window shapes closer to it appear larger, but shapes more to the side of it appear smaller and noisier as well as making occlusion from being behind steel beams even more of a problem. Resulting in less valid clusters and no practical results.

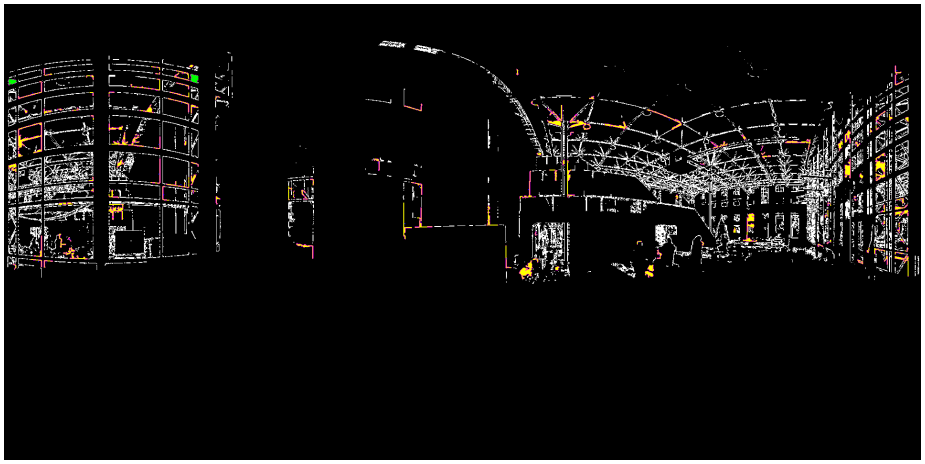


Figure 5.27: Contours found in Scene 2

When examining the contours in scenes 3 to 5 (see figure 5.28, figure 5.29 and figure 5.30 respectively) another issue became apparent, which is the problem of where the distance variance is most prevalent. These scenes were not captured in the Orange Hall, but rather in the hallway adjacent to it and the area next to the staircase. As can be seen and also as expected the contour images of

these scenes are a lot darker. This is due to there being a lot less variance in the captured setting as there is a lot less glass and other openings present in these scenes.

What is, however, very troublesome is that the outer lines of the windows and other openings are also not apparent in this overview. This is because the changes captured through the openings in these scenes have far higher variance changes than the fairly broad and gradual changes of the window frames in these scenes. As a consequence, only the small details in the background are captured properly and the data that is desired is lost. Differentiating between foreground and background is, however, not doable in 2D making this a fundamental downside of this approach.

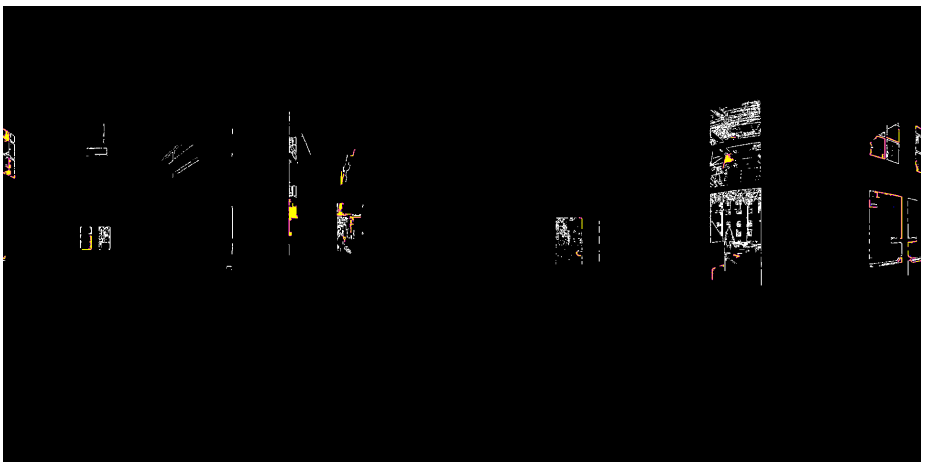


Figure 5.28: Contours found in Scene 3

Figure 5.29: Contours found in Scene 4

Figure 5.30: Contours found in Scene 5

5.4.3 Results from enlarged closing kernels

During this thesis, a simple method to circumvent these issues has been used to be able to generate output data for these scenes as proposed in §4.3.2. The implementation of this workaround was enlarging the kernel used by the closing operator after the edge detection from a 3x3 plus-shaped kernel to a 13x13 square kernel. This makes it so, a lot of detail is lost but with the amount of noise present in the currently used data, using closing to get rid of the noise and join together separate contour segments solves the problem of not being able to find the correct contours by making them simpler but also more apparent in the scene. This workaround can not be used on every scene however, as the more data is present in the scene the more gets merged even though they might be separate objects, so picking the proper size for the kernel used in the methodology should not only be chosen on a use case by use case basis but even on a scan to scan basis if the data demands it.

With the now enlarged kernel the new contours found for the scenes are shown in figure 5.31, figure 5.32, figure 5.33 and figure 5.34. As can be seen, the contours found are extremely simplified and a lot of detail around the edges has been lost, but these do provide useable and also larger contours.

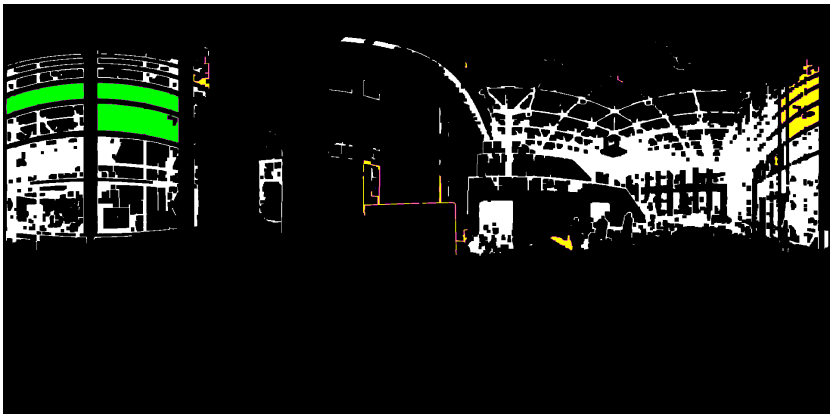


Figure 5.31: Contours found in Scene 2 after enlarging the closing kernel.

Figure 5.32: Contours found in Scene 3 after enlarging the closing kernel.

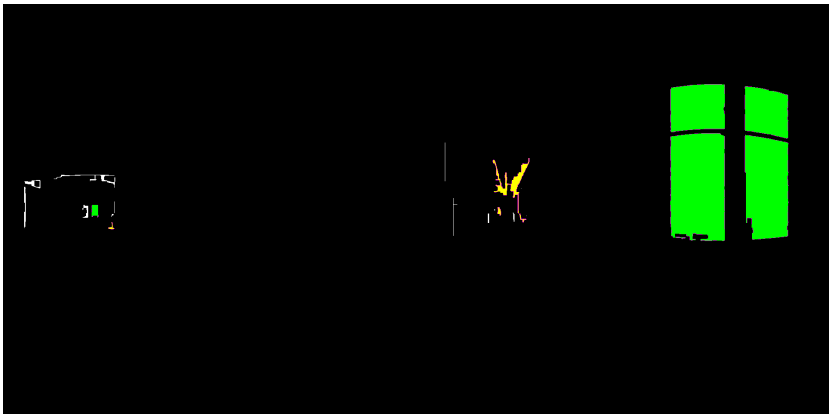


Figure 5.33: Contours found in Scene 4 after enlarging the closing kernel.

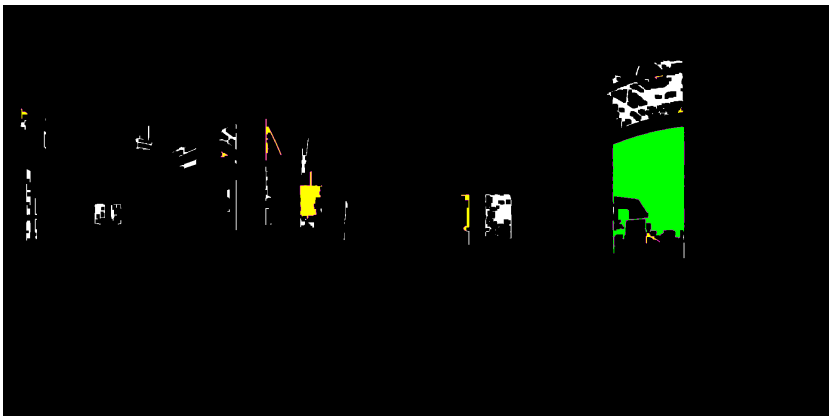


Figure 5.34: Contours found in Scene 5 after enlarging the closing kernel.

The results for scene 2 with the enlarged kernel are shown in figure 5.35 and the placement in the scene is shown in figure 5.36. The result following the detected contours has created three sub-results each finding two windows and a bit extra points around it. The reason why each contour finds two windows together instead of one lies in the enlarged contour visible in figure 5.31. The contours have grown in width making it so each of the individual contours has joined with its neighbour.

Flaws similar to scene 1 can be identified with captured flat surfaces as well as beam-like structures but the result for the windows is a lot better than not getting a result at all. The remarks made in §5.4.2 regarding the downsides of capturing closer to the windows still hold as can be seen in the results. The areas captured are the ones closest to the scanner position, used as reference position here, as these appear the largest on the scene. Enlarging the kernel in these areas meant that they could properly connect but edges further away from the scanner (see the right part of figure 5.31) have lost their details in the dilation process and have merged into something that cannot easily be identified anymore even by humans, showing once again that this method has its up and downsides.

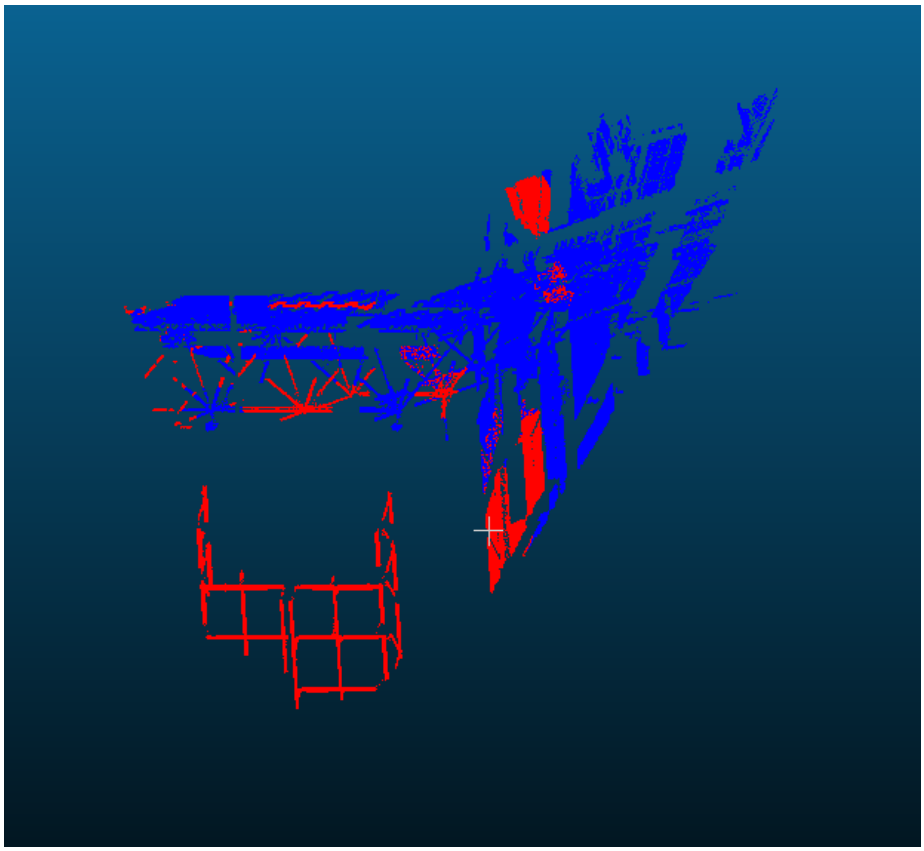


Figure 5.35: Results acquired using enlarged contours for scene 2.

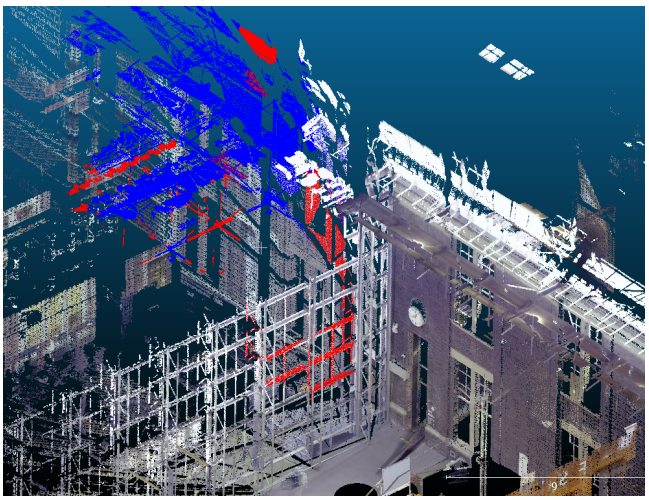
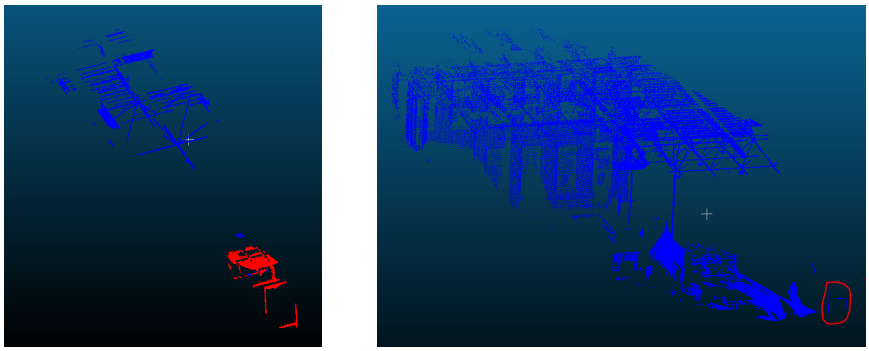


Figure 5.36: Overview of the results from scene 2 within context of the scan. Once again the roof is removed to give a look inside the point cloud for additional reference

Scenes 3 and 5 where both captured close to each other, but once again show differences in figure 5.32 and figure 5.34. Here scene 3 properly captures a window above the opening to the Orange Hall, whereas scene 5 identifies a contour for the opening itself. The results they generate can be seen in figure 5.37a and figure 5.37b, and figure 5.38 and figure 5.39 put them in their respective scene context. As this overview of the results in their scenes is quite zoomed out a zoomed-in reference to see the result match the window and opening they depict is shown in figure 5.40a and figure 5.40b.

Notable is that the regions of interest found in scenes 3 and 5 are captured on top of each other, which makes it odd as to why the contour was good enough in one scene but not in the other. The cluster found in scene 2 correctly shows the correct window but once again extra data is added onto the result due to clustering. In scene 5 no cluster is labelled as a window, which is correct in this context and happened because the contour detected was made out of separate clusters in 3D space which did not cluster together as they were too far apart.



(a) Scene 3 **(b)** Scene 5

Figure 5.37: Results from Scene 3 and 5 with enlarged kernels. The points representing the candidate window found in scene 5 are circled in red.

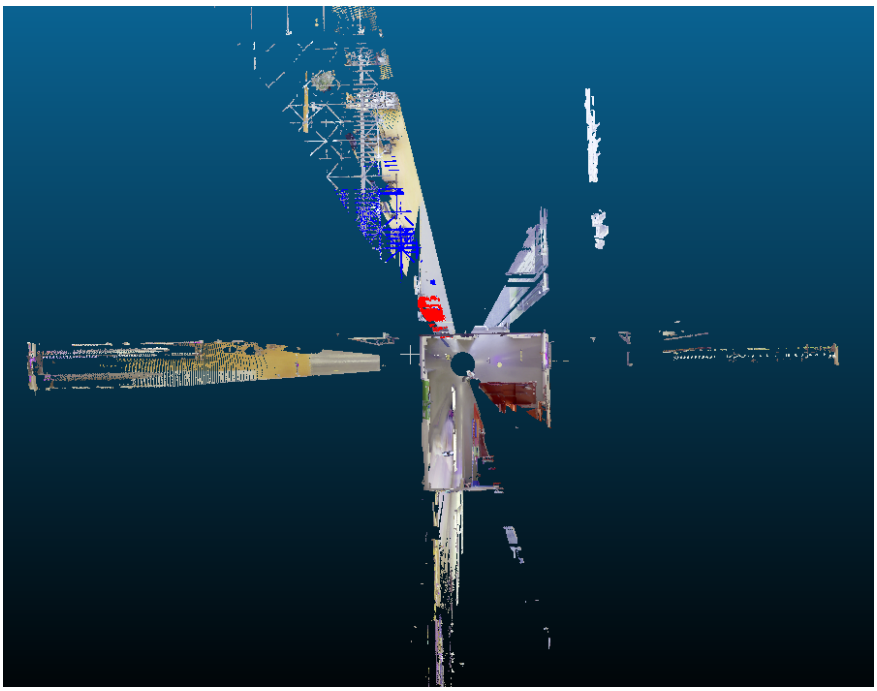


Figure 5.38: Overview of the results from scene 3 within context of the scan without the roof.

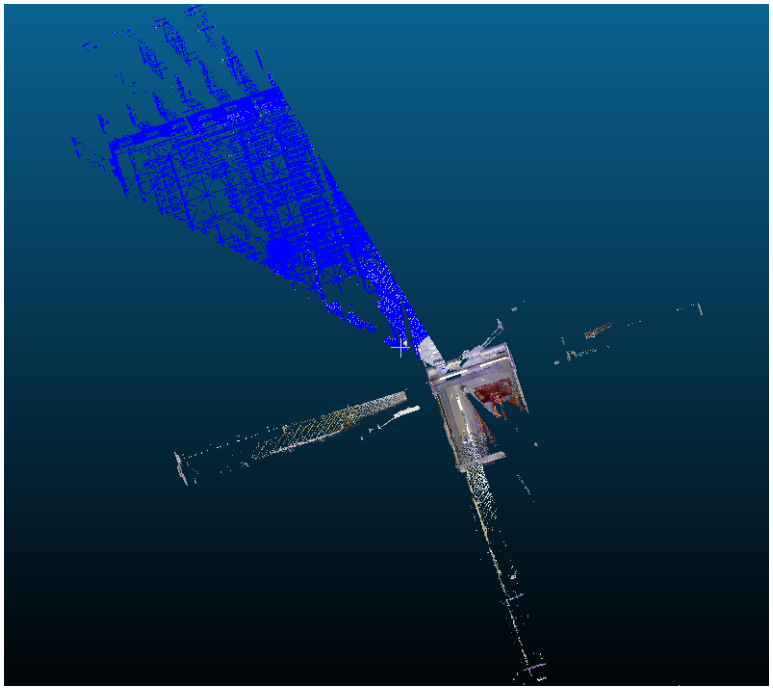
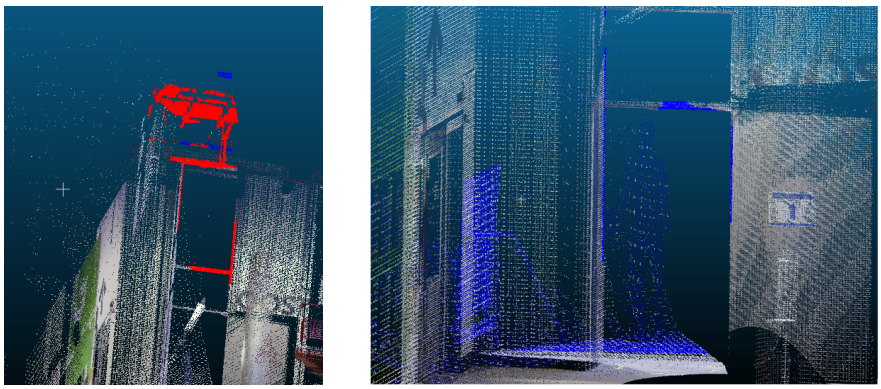


Figure 5.39: Overview of the results from scene 5 within context of the scan without the roof.



(a) Scene 3 **(b)** Scene 5

Figure 5.40: Zoomed in look at where the result in scene 3 and 5 were captured.

The methodology did, however, not capture all of the glass present in scenes 3 and 5, which can better be seen in figure 5.41. In the scenes, there were glass windows and glass doors present right next to the openings they did find. figure 5.38 and figure 5.39 show that some data behind these glass objects has been registered but when looking at the contours found the responses were lacking especially in scene 5, which is most likely due to the windows being more obstructed.

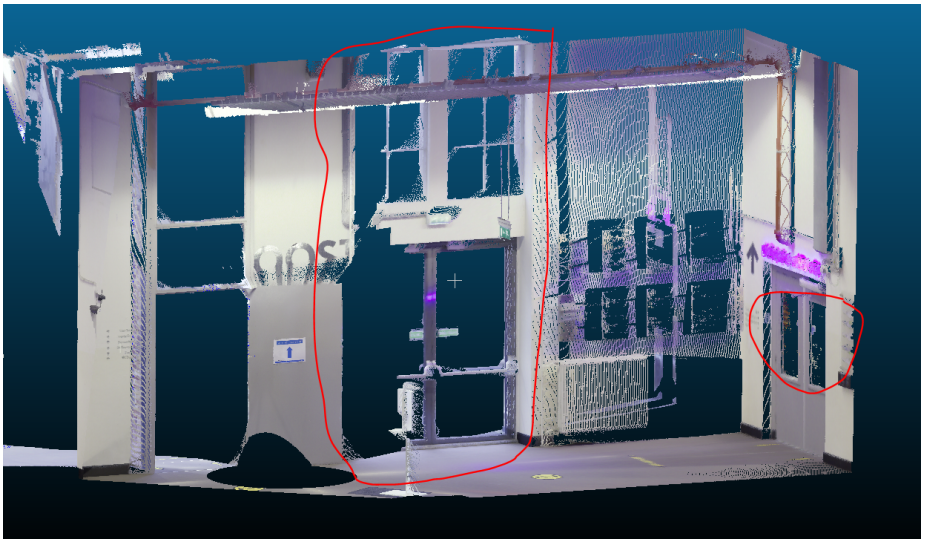


Figure 5.41: Circled in red are glass windows and doors that are not properly found in either of scene 3 and 5.

The final scene to cover is scene 4 which was captured in the hallway next to the Orange Hall. The results for this scene are shown in figure 5.42 with the context in the scene given in figure 5.43. As the regions of interest became quite large in this result a zoomed-in view of the results can be seen in figure 5.44 and figure 5.45. In this scene, there were 5 regions of interest detected and all 5 of them found the window that matches the contour. The results found in

this scene had similar problems to the other results such as flat areas also being deduced to be window locations as well as beam-like structures, but what is most interesting in these results is the number of points directly reflected on the glass, which is especially prevalent in figure 5.45 which is even more special as this is the window furthest away from the scanner.

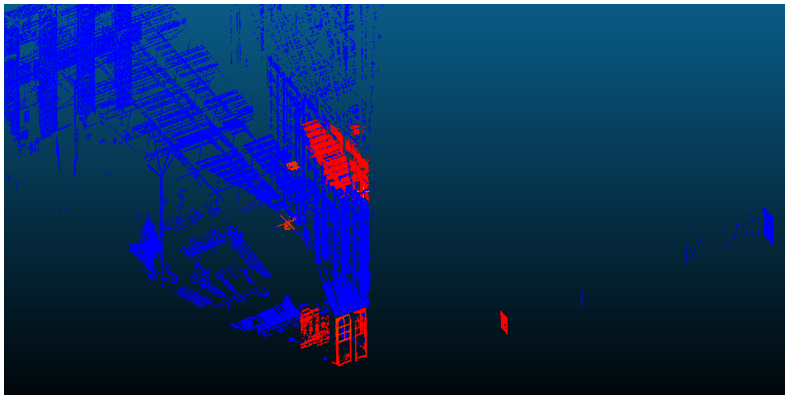


Figure 5.42: Results from Scene 4 with enlarged kernels.

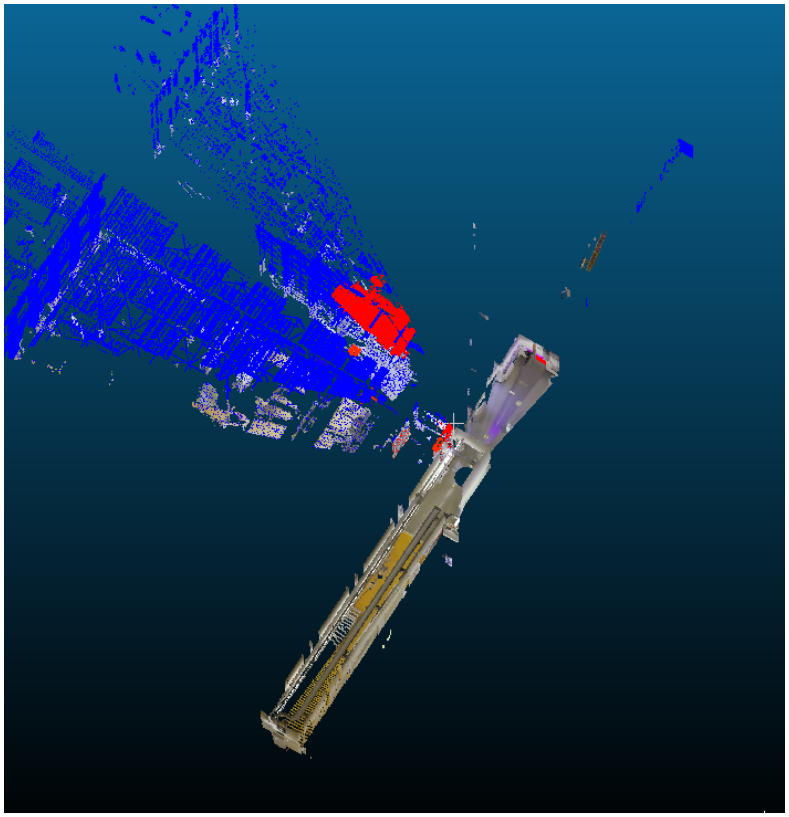


Figure 5.43: Overview of the results from scene 4 within context of the scan without the roof.

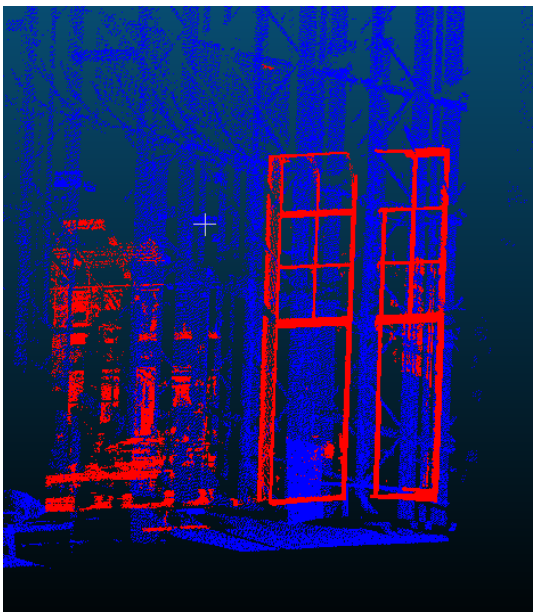


Figure 5.44: A zoomed in view on the deduced window locations in scene 4 adjacent to the Orange Hall

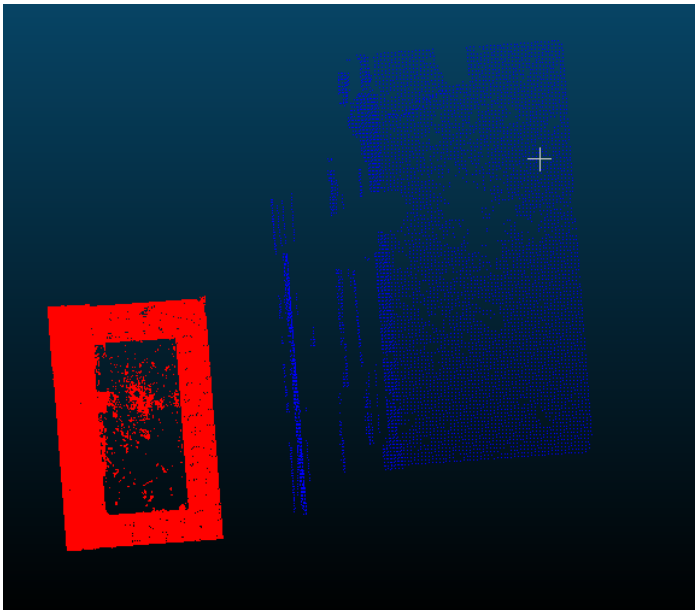


Figure 5.45: A zoomed in view on the other deduced window locations in scene 4

Chapter 6

Discussion and Conclusion

This thesis set out with the goal of deducing the location of glass windows in 3D point clouds using only some of the most basic components one could have when analyzing point clouds, to make the method accessible and applicable to multiple datasets. In this chapter, the conclusion of the thesis will be provided. First, the research questions are evaluated and answered, then a discussion of the method is given and finally, a conclusion on the thesis as a whole is provided.

6.1 Research questions

In this section first, each of the subquestions is explained after which the research question is answered within the scope of this thesis.

6.1.1 Answers to the subquestions

- *What properties are required from the scanning medium to be able to deduce glass?*

  In this thesis, it is shown that for deducing the location of glass the only properties that are necessary for an initial estimate are a standard 3D point cloud which has the spatial properties of X, Y and Z-coordinates by definition and possibly a reference point in space to make sense of it all. To do so, assumptions about the objects that are sought need to be made as using X, Y and Z-coordinates alone can not detect a material that is not captured in the scene. The assumption is made that glass is present when the structure of a regular window is found (see subquestion 3). Using this assumption it is, therefore, possible to deduce the location of glass using only 3D coordinates and a reference position, although as the results have shown it is not trivial. The reference point is needed to make extra assumptions like knowing which parts in the point cloud are indoor or outdoor, which in turn can be used to depict which points show the captured scene and which are possible artefacts in space created by for instance reflections.

- *How reliable is the deduction of the location of glass from 3D point cloud data within the chosen research scope?*

  As has been shown by the results, the reliability of the deduction of the location of glass from 3D point cloud data within the chosen research scope is questionable at best. The contour detection is something the methodology depends on a lot and in the complex environment that the Orange Hall turned out to be, detecting contours that accurately capture windows was a difficult task. Edges would not connect properly even after performing closing or the opposite would be the case meaning that noise would be too prevalent and interfere with the detection of the proper outline of the contour. Once proper contours were found the accuracy of deducing the location of window-like structures in the regions of interest is quite high, but so is the presence of false positives in them making it hard to give a final reliable conclusion that a cluster is for sure a window.

- *How can the deduction of glass be improved using the characteristics of regular windows?*

  Characteristics of regular windows in this thesis have been that they are rectangular and are placed perpendicular to the floor. Having these characteristics makes it possible to make assumptions as to what a cluster that represents a window would look like. This makes it easier to deduce which of the clusters found in the regions of interest acquired after performing contour detection is the most likely to be a window. If these assumptions can not be made it becomes a lot more difficult to deduce the presence of glass based on spatial properties alone, as glass can be found in indoor spaces in all kinds of shapes and forms. It is also a material that in itself is seldom properly captured in the data, so without a structure that indicates its presence, it might even be impossible to deduce the location of glass using only a LiDAR point cloud.

- *What are the advantages and disadvantages of trying to deduce glass in a 1D, 2D and 3D view of the scene?*

  An initial idea for the methodology was to look at the points in 1D to deduce glass. The points were in the order of their capture over time making it possible to see the lines the scanner captured and deduce jumps in distance and intensity. This approach could therefore be used to detect holes in the point cloud, but it was hard to distinguish any meaningful geometric information out of it as the capturing is dependent on assumptions on how the data is captured, how the scanner moves during the capture and noise fluctuations without (much) spatial reference to cover for them to name a few. Also, the points being in the order of their capture is not guaranteed with all point clouds making this approach not even applicable in the first place. The benefit of the approach would be the increase in speed such an implementation could provide, as the data is stored in such a format so that an analysis of it is only dependent on the previous and next points, which makes the time complexity of the application scale with only the sample size. So should a parameter that properly identifies glass be found in the data, then such a 1D approach can bear fruit.

  As such a parameter was not present in the data used in this thesis, this thesis did focus most of its attention on 2D and 3D analysis of the point

cloud. The 2D view of the scene had the benefit of the vast amount of libraries available for and research performed on 2D images. Using this, rectangular structures like windows can be identified as shown in the results using 2D spatial neighbourhood information instead of the more complex 3D environment. The amount of research into and flexibility of 2D analysis are therefore the greatest strength of this approach but reducing a dimension of the data makes it unable to accurately deduce the location of the windows in the original 3D data, which in the end is the goal as enhancing the scene with the location of glass is where the value of the methodology lies. It also removes depth in this case, which makes it possible to have overlapping points that are captured in the same pixel and can greatly influence the results of the methodology should the background data have more detail than the foreground data where glass is expected to be found, as can be seen in §5.4.2. In Terrestrial Laser Scanning as used in this thesis overlapping points are less of an issue as the scanner is fixed in place, so all data when referenced to the scanner location should have no overlap, but in Mobile Laser Scanning approaches this could lead to more problems when analyzing the scene in 2D.

Finally, when looking at the scene from a 3D view the 3D point cloud data can be analyzed in its most complete view. There is separation possible between different clusters in height, width and depth and spatial features on the spread of points can be computed in all 3 dimensions as well. But there are downsides to working purely in 3D. For one, structure analysis of the whole scene is quite an intensive process as all points that are free in space are hard to project to just a single layer especially in environments with noise this can create a lot of pseudo walls that should not even be present in the scene. Furthermore, it is possible to lose orientation when viewing 3D spaces as a lot of information is occluded in one angle while visible in the other making it harder to distinguish specific shapes if the data is not viewed from the correct angle.

So every view has its benefits which the methodology tried to exploit by converting between both 2D and 3D.

- *To what extent can points behind glass be used to improve the deduction of glass candidates?*

  The points behind glass can indicate a lot to make the deduction of glass better. They can visualize objects that are present behind glass, which is the most standard case, but this excludes that the found hole in the point cloud is caused by mirrors or other reflective materials, as they can only show reflections of what is found on the other side of the scene. They can also be used to visualize reflections which when properly detected exclude the possibility that the hole in the point cloud is simply a hole, as a reflection shows an active process that happened at the location of glass. Using these properties it is possible to classify holes found in the point cloud as glass, mirror or hole in the result, similarly to an approach that Koch et al. (2016) used in their work. After the detection of the points behind glass and using them for potential further gain the points could also be removed to better represent the actual indoor space captured. Reflected points could also be filtered out in this set to remove only the

points that should not exist in the scene saving points captured behind glass (see Chapter 7 for more information on this potential).

6.1.2 Answer to the main research question

Now that the subquestions have been answered the main research question can be answered.

How can the location of glass be deduced using only information acquired from 3D point clouds and a scanner location?

The location of glass in 3D point clouds can be deduced using only an additional reference location following the proposed methodology from Chapter 4. This entails converting the 3D point cloud into a 2D image by projecting points in similar directions onto the same cell and taking the variance of all those points to be the pixel value of the image. Then contours indicating the location of potential windows are found and extracted. These contours are then used to single out regions in the point cloud, that are clustered using a density-based clustering algorithm. Finally, the clusters of points are tested upon a set of geometrical criteria and the cluster most resembling a window is deduced to be the location of a glass window in the point cloud.

The result for this can be enhanced in all kinds of ways by adding more data or modifying parameters but the 3D spatial information provided in the point cloud combined with a reference point to make extra assumptions is enough, as long as assumptions can simplify the targeted structure that identifies glass. One of these simplifications is related to the spatial shape of glass that is desired to be found, for which rectangular windows can be an example as has been shown in the experiments performed. Properties of viewing the scene in multiple dimensions may also be used to help with the deduction of glass. The example of this used in the methodology is first using 2D for finding the direction in 3D space where windows are expected and then using these directions to simplify the analysis of point clusters in 3D. The results did, however, also show that false positives are a common occurrence so this should be taken into account when deducing the location of glass in 3D point clouds in this way.

6.2 Discussion

As shown in the results (see §5.4), finding the location of glass in the scene has partially been achieved with some of the windows being deduced properly in the different scenes. This shows that additional insight can be added to point clouds enhancing them with more semantic information even on the most standard point clouds.

The method is shown to not be very accurate due to having been applied to a complex scene with beams and threads before the windows which make it harder to extract all correct contours and even split some of them up. The method does show promise in identifying where walls with a lot of windows are present as a human after seeing the results can already determine a lot from them afterwards. The criteria used in this analysis can also be tuned to be less strict but that can introduce false positives in the results, as was seen when increasing the kernel used in the closing morphological operator. This made the method able to acquire more results but did come at the price of detail as

it registered window like structures that were larger than before because more windows were joined to be in the same set.

Another limitation is that assumptions needed to be made to get results. Glass itself is a material that comes in all kinds of forms and is therefore hard to completely find based on shape alone. To deduce its location properly, without relying on assumptions of the structure glass is contained in, other more complex data will be required, like using a sensor fusion of LiDAR and Sonar to actively detect it or potentially using deep learning to find more connections in the data that indicate its presence.

This research has been performed on TLS data, while MLS data is also a prevalent data capturing method to produce LiDAR point clouds. The reason the focus of this thesis was on TLS is that the reference location for this manner of capturing is easy to determine as it is simply the scanner location, but with MLS this becomes trickier as the scanner moves causing issues with points on two sides of walls being visible in the conversion to 2D if a substituted location has been chosen as the reference location. MLS does however have the benefit of almost guaranteeing that points are captured under multiple angles which might lead to other possibilities for glass deduction itself.

6.3 Contribution to Research

This thesis provides a way to add insight into the presence of glass, one of the most troublesome materials for LiDAR scanning, in point clouds. This method has been devised to be used on any 3D point cloud as long as a reference position for the scanner position is given, but is mostly focused and tested on data acquired from Terrestrial Laser Scanning. Having only these requirements makes the technique of glass deduction more accessible while keeping the time invested and materials used to a minimum, making sure as little extra costs are added as possible.

The method also shows a new way of using a conversion from 3D space to 2D space to apply image processing techniques to determine regions of interest in 3D space for further analysis. Although initial results may have varied, it shows the potential of inspecting 3D scenes differently, using the best of both worlds when trying to get to a result.

The results themselves give an insight into where glass is present in the scene giving a better understanding of the scene as a whole. This understanding of the presence of glass in places where initially it was thought that nothing was present allows for better use of the data in planning navigational routes or escape routes for both humans and machines, insight into maintenance costs for the building and properly splitting the indoor and outdoor environments. The methodology itself also shows a lot of potential for growth in value by using the deduced windows locations to eliminate artefacts created by them from the scene in post-processing as well as adding missed windows at a later moment based on the logic applied to already known results.

Chapter 7

Future work

This thesis proposes a methodology for deducing the location of glass windows in any 3D point cloud as long as a reference position inside of the scene, preferably the scanner position, is provided. The methodology set out what it to do and achieved this, but the result can be improved in numerous ways, such as improving the detection of the contours that depict the window candidates, making the method applicable to even more datasets like MLS data or taking into account other properties that can push the results forward by improving the classification of the results. As a consequence, the following areas for future research are presented.

7.1 Usage of different data to enhance the initial point cloud

As shown in the related work section using additional data or manipulation in the scene can get rid of a lot of the problems glass can bring to LiDAR datasets. In this thesis, the deduction of glass windows has been performed using as little extra information as possible to make the methodology as widely applicable as possible but as has been shown in related research actively knowing the location of glass can be very beneficial to improve the LiDAR results. Sensor fusion with Sonar sensing is one of these approaches that can actively detect glass, although the accuracy of the resulting Sonar data is lower than the LiDAR data. Another approach of creating a dataset with actively detected glass can be created with physical manipulation of the scene by putting something thin that can be detected by LiDAR directly in front or behind the glass or using sprays on glass that makes it detectable. Having an additional dataset that has the location of glass in it, makes it easier to investigate the neighbourhoods around the glass as will be discussed in the next section, but can also in itself bring a lot of extra information to the scene. There has already been a lot of research into how this can help in for instance robot navigation, but not to enhance indoor environments in a full 3D space where its uses can also be greatly beneficial, although its costs go up along with it.

7.2 Further investigation of point neighbourhoods

To enhance the certainty of classification of the window candidates further research could be done into the points around or behind the found window candidate. Points behind the window candidate can be checked to be reflections as has been shown in the work of Yang and Wang (2011). This could help further the classification of the window candidates by making a distinction between glass, mirrors and other specular reflective objects. Namely points in point clouds that look to be behind mirrors should always be reflections, so if the points do not properly fit as a reflection the window candidate can not be a mirror, but if everything matches as a reflection then it becomes a lot more likely that it is a mirror instead of a glass window. If the points have no reflective match whatsoever, the assumption that the window candidate is a hole in the wall becomes significantly more likely. Should the results perform some reflection matching but also some missed matches then glass is the most likely option. Combining this research with different scanner input as mentioned before, some of these possibilities can already be filtered out, but the better the material is classified the better the scene can be understood and consequently the uses of said data can also increase.

7.3 Combination of multiple scans

As touched upon by §7.1, data acquired from different sources like Sonar or data captured after manipulating the scene so glass is better detectable can be used to enhance the initial LiDAR dataset. If these sets can be combined, a better overview of the scene as a whole can be created. The work of Zou and Sester (2021) has shown that an initial step for iterative enhancement of LiDAR point clouds is possible, but this research has been performed on outdoor scenes with a single scanner. The same methodology should, however, be feasible in the indoor environment as well. Using the option of enhancing point clouds with other point clouds can enhance the results from this thesis greatly. The results from each scene are currently capable of deducing the location of some but not all windows in the point cloud. Should new scans be performed at different positions then the results of both scans joined together will be able to give a better view of the scene as a whole. A practical example of this is already provided in the results for scene 1 and scene 2, which both found different windows in the same general area, so when these are combined the total scene will be enhanced.

Combining multiple scans can also be used to make the methodology proposed in this thesis more generally available. By being able to combine multiple scans into one this methodology can also be applied to Mobile Laser Scanning data. Currently, there is the risk of improper overlap of data due to not being able to select a single centre of space in MLS data once data on multiple sides of a wall has been collected. When the option of stitching multiple sets together gets introduced, however, MLS data can be split up in time frames where the scanner was in only a single room. By doing so a position in that room can be chosen as a reference position and the methodology can be applied to this part of the data. Once this has been performed for all rooms and the results have been merged a total overview of the indoor environment can be achieved.

7.4 Deep Learning

Finally, this research did not make use of any deep learning approaches, due to the interest in trying to control all variables and logic that would be used, as well as a lack of labelled data available for training and testing. The potential that machine learning has for detecting spatial features and accurately making predictions based on sometimes unknown factors to researchers is immense. For purpose of this methodology, it could possibly be used to find more and better window candidates in 2D space, help in determining while clusters depict the windows in 3D space and possibly help better classify the results. This does, however, depend on data being available to train such a network, which would need to be able to distinguish between holes, windows and mirrors. Aside from this, the network should also be able to work with different kinds of indoor environments as they are very diverse making it harder to get robust results out of deep learning approaches. But should data become available to properly train a network to work on all kinds of indoor environments, regardless of their interior, then this could be able to become a very promising method for deducing the location of glass in indoor spaces.

Bibliography

Laser Safety – Module 4: Bioeffects. URL https://www2.lbl.gov/ehs/training/webcourses/EHS0302/story_content/external_files/Module%204%20Bioeffects.pdf.

T. A. Al-Asadi and W. R. Baiee. Improved Douglas-Peucker Algorithm with Dynamic Threshold Values. *International Journal of Digital Content Technology and its Applications*, 8(6):61, 2014.

G. Bhatia and V. Chahar. An Enhanced Approach to improve the contrast of Images having bad light by Detecting and Extracting their Background. *Int. J. Comput. Sci. Manag. Studies*, 11(2):2231–5268, 2011.

J. Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, PAMI-8(6):679–698, 1986.

R. G. Crespo, W. L. Romero, O. S. Martínez, and C. E. Montenegro-Marín. Design and Modeling to Generalized Linear Elements in a Vector Formatted Cartographic. *International Journal of Advancements in Computing Technology*, 6(3):96, 2014.

D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

N. El Abbadi and L. Al Saadi. Automatic detection and recognize different shapes in an image. *International Journal of Computer Science Issues (IJCSI)*, 10(6):162, 2013.

P. Flikweert, R. Peters, L. Díaz-Vilariño, and R. Voûte. Automatic Extraction of an IndoorGML Navigation Graph from an Indoor Point Cloud. Master's thesis, TU Delft, 2019. URL http://resolver.tudelft.nl/uuid:b11f5b57-5362-4b45-bed6-d5bc154d86aa.

X.-Y. Gong, H. Su, D. Xu, Z.-T. Zhang, F. Shen, and H.-B. Yang. An overview of contour detection approaches. *International Journal of Automation and Computing*, 15(6):656–672, 2018.

M. Jarząbek-Rychard, D. Lin, and H.-G. Maas. Supervised detection of façade openings in 3D point clouds with thermal attributes. *Remote Sensing*, 12(3): 543, 2020.

P. Kaniouras, M. Moscholaki, J. van Liempt, K. Jarocki, L. Zhang, E. Verbree, and M. Meijers. Direct Analysis on Point Clouds: Geomatics Synthesis Project 2019. 2019. URL http://resolver.tudelft.nl/uuid:4f610e66-01b5-409a-aaf9-9f03d96c7889.

R. Koch, S. May, P. Koch, M. Kühn, and A. Nüchter. Detection of Specular Reflections in Range Measurements for Faultless Robotic SLAM. In *Robot 2015: Second Iberian Robotics Conference*, pages 133–145, Cham, 2016. Springer International Publishing. ISBN 978-3-319-27146-0.

V. V. Lehtola, H. Kaartinen, A. Nüchter, R. Kaijaluoto, A. Kukko, P. Litkey, E. Honkavaara, T. Rosnell, M. T. Vaaja, J.-P. Virtanen, et al. Comparison of the Selected State-Of-The-Art 3D Indoor Scanning and Point Cloud Generation Methods. *Remote sensing*, 9(8):796, 2017.

I. Lysenkov, V. Eruhimov, and G. Bradski. Recognition and pose estimation of rigid transparent objects with a kinect sensor. *Robotics*, 273(273-280):2, 2013.

R. Maini and H. Aggarwal. Study and comparison of various image edge detection techniques. *International journal of image processing (IJIP)*, 3(1):1–11, 2009.

S. Nikoohemat, A. Diakité, S. Zlatanova, and G. Vosselman. INDOOR 3D MODELING AND FLEXIBLE SPACE SUBDIVISION FROM POINT CLOUDS. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 4, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in Python. *the Journal of Machine Learning research*, 12: 2825–2830, 2011.

A. Raid, W. Khedr, M. El-Dosuky, and M. Aoud. Image restoration based on morphological operations. *International Journal of Computer Science, Engineering and Information Technology (IJCSEIT)*, 4(3):9–21, 2014.

Schott AG. Transmittance of optical glass. 2020. URL https://www.schott.com/d/advanced_optics/5b1f5065-0587-4b3f-8fc7-e508b5348012/1.1/schott-tie-35-transmittance-of-optical-glass-february-2020-row-20022020.pdf.

J. Shlens. A Tutorial on Principal Component Analysis. *CoRR*, abs/1404.1100, 2014. URL http://arxiv.org/abs/1404.1100.

B. Staats, S. Zlatanova, A. Diakité, and R. Voûte. Identification of walkable space in a voxel model, derived from a point cloud and its corresponding trajectory. Master's thesis, TU Delft, 2017. URL http://resolver.tudelft.nl/uuid:6a827a88-ce09-43f1-adb9-da886448a1fc.

H. Thomas, F. Goulette, J.-E. Deschaud, B. Marcotegui, and Y. LeGall. Semantic classification of 3D point clouds with multiscale spherical neighborhoods. In *2018 International conference on 3D vision (3DV)*, pages 390–398. IEEE, 2018.

S. Tuttas and U. Stilla. Window detection in sparse point clouds using indoor points. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38(3), 2011.

VELODYNE LIDAR. A Guide to Lidar Wavelengths, 2018. URL `https://velodynelidar.com/blog/guide-to-lidar-wavelengths/`.

M. Vis. History of the Mercator projection. B.S. thesis, Utrecht University, 2018.

C. Waldhauser, R. Hochreiter, J. Otepka, N. Pfeifer, S. Ghuffar, K. Korzeniowska, and G. Wagner. Automated classification of airborne laser scanning point clouds. In *Solving Computationally Expensive Engineering Problems*, pages 269–292. Springer, 2014.

W.-T. Wang, Y.-L. Wu, C.-Y. Tang, and M.-K. Hor. Adaptive density-based spatial clustering of applications with noise (DBSCAN) according to data. In *2015 International Conference on Machine Learning and Cybernetics (ICMLC)*, volume 1, pages 445–451. IEEE, 2015.

H. Wei, X. Li, Y. Shi, B. You, and Y. Xu. Fusing sonars and LRF data to glass detection for robotics navigation. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 826–831. IEEE, 2018.

M. Weinmann, B. Jutzi, and C. Mallet. Feature relevance assessment for the semantic interpretation of 3D point cloud data. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 5(W2):1, 2013.

G. Yadav, S. Maheshwari, and A. Agarwal. Contrast limited adaptive histogram equalization based enhancement for real time video system. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 2392–2397. IEEE, 2014.

S.-W. Yang and C.-C. Wang. On solving mirror reflection in LIDAR sensing. *IEEE/ASME Transactions on Mechatronics*, 16(2):255–265, 2011.

M. Ye, Y. Zhang, R. Yang, and D. Manocha. 3D reconstruction in the presence of glasses by acoustic and stereo fusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4885–4893, 2015.

Q. Zou and M. Sester. Incremental Map Refinement of Building Information Using LIDAR Point Clouds. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 43:277–282, 2021.

Appendices

Appendix A

Reflection of Master Thesis

3D indoor environment models keep on evolving over the years and with their evolution so do their uses keep increasing. These models do, however, not always give an accurate representation of what the environment is actually like by either being outdated or by being incomplete. This thesis set out to contribute to solving one of these problems by proposing a methodology that is capable of deducing the location of glass windows in LiDAR point clouds. Glass is one of the easiest methods to miss while capturing an environment using LiDAR as it is almost fully transparent to the laser beam LiDAR typically uses to capture the environment. By introducing the methodology proposed in this thesis one can enhance a point cloud previously captured and by providing just a reference position in the scene, a deduction of the location of glass windows can be done. This is important because adding extra obstacles in the environment can help with purposes such as planning evacuation routes around these obstacles. Knowing the presence of glass in the environment can also help with trimming the dataset by removing points captured behind glass that indicate reflections or objects outside of the indoor environment. As this methodology can also be applied to already existing point clouds the costs for using it are also very low to enhance already existing assets.

The method presented in this thesis uses concepts of data collection, conversion, processing, analysis and visualisation, all of which fit with the Master Geomatics. The data has been collected in the point cloud format which is covered extensively during the Master and it was captured using Terrestrial Laser Scanning. The data itself was presented in 3D as it was a point cloud but using techniques learned in the courses of the master this could be converted to 2D while containing still representing the scene, after which it could be used to turn it back to 3D as well. Data processing was also a very prevalent part of the thesis, like converting data between dimensions, performing data enhancement, using edge and contour detection, clustering them in groups and calculating properties on it. These properties were then analysed and used to determine which clusters are representing windows in the end. Finally, a lot of data analysis and visualisation was performed in Cloud Compare an open-source library that was introduced to me during the Master but which has helped tremendously during this process.

The progress of the thesis itself was rough, to say the least. I personally decided to follow my three electives spread out over 3 quarters which in hindsight was a huge mistake. Doing this led to a lot of issues with my priorities as the thesis deadline seemed to be so far away, that the rest often took precedence. Because of this, the P2 was quite rushed and as such my mentors and I were not always on the same page as to which direction to go into. This became especially clear when between the P2 and P3 my initial proposal turned out to not go the way I wanted it to.

This caused me to reconsider and doubt a lot in my methodology and completely threw off my planning for the rest of the thesis. We, therefore, decided to focus on one direction and ensure that that could be finished, but at that time due to personal circumstances, my concentration was not with the thesis anymore. This caused a lot of delays until at the start of September 2021 my mentors advised me to talk to a student counsellor. With him, I started working on a plan to finish my graduation, while my mentors gave me some space to finish everything and this thesis was realised.

In conclusion, this thesis evolved a lot over its creation with different alleys being explored and a lot of confusion along the way. In the end, I am satisfied with the result, but if I could do it again most of the things would be changed. The electives would all be done in a single quarter to allow for a full focus on the thesis afterwards. In the initial planning phase more communication with the mentors would have been done to ensure everyone is on the same page regarding the intended results, before I would mindlessly start searching for solutions to unclear problems. During the data collection step, an easier scene would be picked to ensure the creation of a working methodology first which could afterwards be applied to a harder scene to see its results in that setting. I think that in doing so the process would have been a lot more pleasant and that motivation for the thesis itself would have also improved as a clearly laid out plan turned out to work significantly better for me.