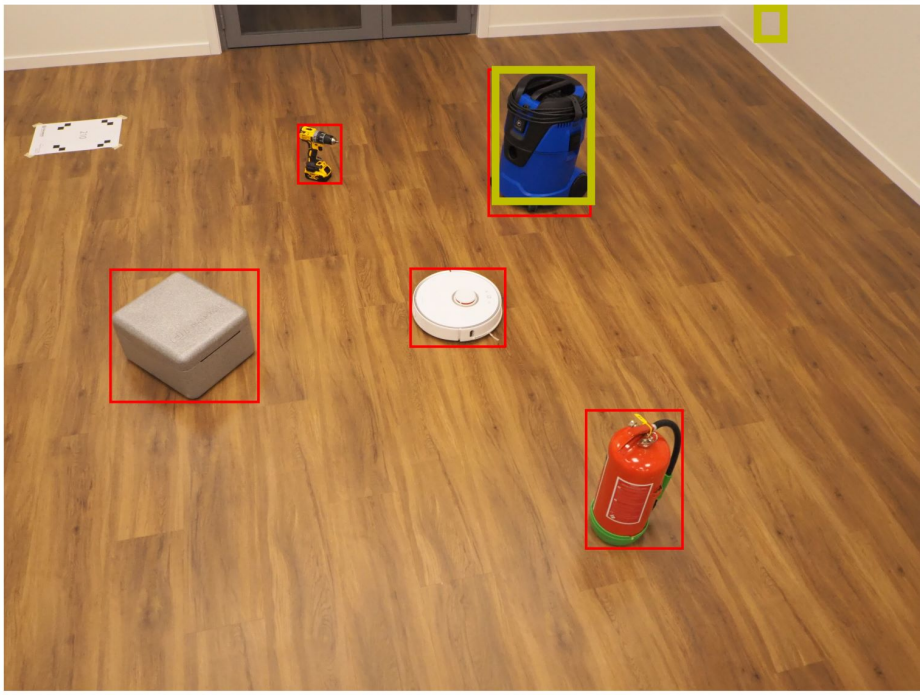
CHALMERS
UNIVERSITY OF TECHNOLOGY



Automatic update of geometrical changes for a digital twin

A method to automatically identify and estimate the pose of objects through object recognition and photogrammetry

Master's thesis in Systems, Control and Mechatronics

RIKARD KARLSSON

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Automatic update of geometrical changes for a digital twin

A method to automatically identify and estimate the pose of objects through object recognition and photogrammetry

RIKARD KARLSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems & Control
Automation
Chalmers University of Technology
Gothenburg, Sweden 2020

Automatic update of geometrical changes for a digital twin
A method to automatically identify and estimate the pose of objects through object recognition and photogrammetry
RIKARD KARLSSON

Supervisor: KNUT ÅKESSON, PETTER FALKMAN, Dept. of Electrical Engineering, Chalmers
Examiner: KNUT ÅKESSON, Dept. of Electrical Engineering, Chalmers

Department of Electrical Engineering
Division of Systems & Control
Automation
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover:Illustration of detected, added object in one of the generated test cases.

Typeset in LaTeX
Gothenburg, Sweden 2020

Automatic update of geometrical changes for a digital twin
A workflow to automatically identify and estimate the pose of objects through object recognition and photogrammetry

Rikard Karlsson
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Virtual commissioning has become a promising approach to more efficiently develop production systems. There are however drawbacks with virtual commissioning. As the systems get increasingly complex the amount of work needed for development grows, in some cases to a degree that virtual commissioning is deemed too expensive. By increasing the number of applications of virtual commissioning, it is possible to outweigh the cost of increasing development complexity. One such feature would be to automatically update the digital twin, which would extend its lifespan and expand the possible usages even after the physical commissioning is performed. This thesis presents a way to automatically identify added or removed objects within an environment, as well as estimate the pose of objects.
Using images of the environment, a 3D point cloud representation can be constructed through photogrammetry. By comparing two different point clouds of the same environment, it is possible to detect geometrical changes representing added or removed objects. These geometrical changes are then re-projected back into the images with the principles of photogrammetry, highlighting where the environmental changes occur within the images.
The same images from which the point cloud is constructed are also used for object recognition through the use of a YOLO neural network, creating bounding boxes in the images of where objects of interest are located. The overlap between the re-projection and these bounding boxes are then calculated, and used to identify the geometrical changes. The identified geometrical changes are then matched with a point cloud of the identified object, where the latter is aligned with the former to estimate the pose of the added or removed object.
The developed identification metric performs well, but is reliant on good alignment between the two point cloud instances of the environment.
The pose, estimated with global registration and iterative closest point, is calculated close to the ground truth as long as the objects have distinct geometrical shapes. Any uniformity will lessen the accuracy of the estimated pose, sometimes to such a degree that objects are turned upside down due to their uniform shape.

Keywords: virtual commissioning, digital twin, change detection, point cloud identification, pose estimation

Acknowledgements

I would like to thank my supervisor and examiner Knut Åkesson for all technical assistance during the writing of this thesis, as well as my co-supervisor Petter Falkman for his input on what the developed workflow would need to produce to being relevant for a real world application. I would also like to thank Johan Vallhagen (Volvo) as well as Jonatan Berglund (Chalmers), for providing and helping generate the data upon which the developed workflow was tested.
Last but not least I would like to thank Jason Li for his preceding work which laid the foundation for my thesis, as well as the invaluable brainstorming sessions that helped me get on the right track early on in the development.

Rikard Karlsson, Gothenburg, June, 2020

Contents

List of Figures

List of Tables

1

Introduction

Virtual commissioning has during the last decade risen in popularity as an integrated part in the process of developing new production systems. Studies on the subject has shown that there are multiple advantages with performing virtual commissioning. Lechler et al. [1] has shown that virtual commissioning decreases the required development time, as well as increasing the quality and efficiency in production engineering. Virtual commissioning also makes it possible to detect faults in an early stage of development [1] [2], and can also enable tests which has been deemed too expensive to perform in a real world system [3]. It is therefore a useful tool to verify and validate the functionality of the system [2] [4]. Additionally, Dahl et al. [5] has shown that virtual commissioning reduces the time needed for test and integration during development. To summarize, performing virtual commissioning is a possibility for a company to both save time and money during development [6].
Virtual commissioning can be performed on a multitude of systems with a varying degree of complexity. Due to this a recently published paper has been defining categories of virtual commissioning based on the level of integration [7]. According to this paper, virtual commissioning can be classified into five different levels based on the different implementations that are needed to fulfill the requirements. The simplest forms of virtual commissioning aim to emulate a controller for code verification, while the highest level requires full simulation of the whole system. The highest level of virtual commissioning integrates all parts of the system, including concepts like Human-in-the-loop and visualization.
According to Albo et al. [7], an integrated part of the highest level of virtual commissioning is the digital twin. The digital twin aims to bridge the gap between the physical and virtual environments [8]. By creating a digital twin which is capable of simulating the whole system, combined with virtual reality, it is possible for humans to interact with the system. This opens up for the possibility of early stage testing of human-robot collaboration [1], as well as safe, off-site training of the workforce even before the system has been constructed [1][6][9].
While there are clear advantages of high level virtual commissioning, there still exists drawbacks. As the scope of virtual commissioning increases, so does the complexity and implementation costs of the digital twin [1][10]. Creating a detailed simulation model is a tedious, manual task that is deemed to not outweigh the costs [2]. The usefulness of the digital twin is further diminished over time, since as time progresses changes to the physical system occurs which increases the gap between the model and the physical system [11].

1.1 State of the art

Recent research in the field of virtual commissioning has partly been focused on two main areas which aims to improve the development and use of virtual commissioning.
Firstly there has been proposed concepts which would aid in the construction of virtual commissioning, reducing the amount of work that needs to be performed. This includes concepts such as clever component repositories [2], a standardized way of structuring simulation-crucial information of each component. These clever components is proposed to contain information such as CAD and kinematic models, description of dynamic behavior, and control and safety logic.
Secondly, some research has been focused on how to extend the usability of the virtual model beyond the development phase. Especially the possibility of real time simulation alongside production has been discussed to extend the possible use cases of virtual commissioning after physical commissioning has taken place [12]. Updating the virtual model with data from the real system, for example the wear of components in the physical system, allows for better simulations and predictions [13] [14].
To realize this continuous update of the virtual environment, the use of so called anchor points have been proposed [11] [15]. By defining references within the physical system it is possible to update the virtual model to match the current state of the physical system. These anchor points can be divided into the different domains of software, electrical and mechanical depending on their nature, allowing multiple ways of synchronizing the digital model with the physical system.
There has also been methods developed which aims to listen in on the actions performed by the physical system [16]. By sequencing these actions into operations, it is then possible to determine if the system ever deviates from its defined model.

1.2 Problem formulation

The state of the art research mentioned above aims to aid the development of virtual commissioning by either:

A: Reducing the amount of work needed to perform virtual commissioning

B: Broaden the usage of virtual commissioning and digital twins

The digital twin is a centerpiece within each of these aspects, and developing and updating the digital twin is a task which requires a lot of work.
If a method to automatically update a digital twin could be developed it could reduce the amount of work that has to be performed during early stages of virtual commissioning, as well as increasing its usefulness once the physical system is operational.

1.3 Purpose and aim

This thesis aims to propose and verify a workflow which is capable of identifying geometrical changes to an environment; namely if an object has been added to or removed from the environment and where the object is or was positioned. The objective is to produce results that has the possibility of being used to automatically update a digital twin.

1.4 Research questions

To be able to determine if the proposed workflow fulfills its purpose, the results should aim to answer the following research questions:

1. Is it possible to create an automated process for the intended purpose? If so, what are the prerequisites?
2. Is it possible to identify a geometrical change in an environment as an object, and if so to what certainty?
3. How accurate can the pose of an object be estimated?

1.5 Delimitations

This thesis does not intend to produce a fully automated system, but prove that by combining the selected principles a satisfactory result is achievable. Some of the included principles have their own varying degree of performance, but for the sake of evaluating the proposed workflow these principles have been assumed to have a 100% accuracy.

2
Problem deconstruction

The main problem presented in Chapter 1 can be deconstructed into a few underlying problems, each with the possibility to be solved separately. These subtasks are presented in this Chapter, along with a short presentation of existing methods to solve each subtask.

2.1 Information acquisition

To be able to analyze anything related to an existing environment, the first thing that needs to be done is to detect its current state.
There exists many different methods of acquiring geometrical information of an environment, all with their different advantages and limitations. A property they all share is the possibility to produce point clouds, which is a common way storing data of 3D objects and surfaces.
A few of the most common methods of acquiring geometrical information of an environment are presented below.

2.1.1 Laser triangulation

The laser triangulation technique consists of a laser and a camera, with a fixed known pose relative each other. The laser projects a line onto an object and the camera captures the distortion of the laser, which is illustrated in Figure 2.1.

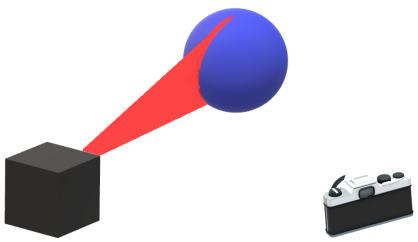


Figure 2.1: Illustration of laser triangulation and how a laser beam distorts around a surface

By knowing the relative pose of the laser with respect to the camera, it is possible to triangulate the surface of the scanned object from the distortion. The scanning can either be done through turntables and a fixed laser and camera, or by manually moving the laser and camera over the object. Note that in both cases the relative pose between the laser and the camera is fixed. The main advantage of the laser triangulation technology is the accuracy. With an accuracy on the scale of micrometers, it can produce a good representation of the object.
The main drawback is the small scanning area. Only the area onto which the laser is projected can be detected at each time instance, making the time needed for scanning scaling with the size of the object. Another drawback is how the scanning process is performed. The scanning takes place in close proximity to the object itself, making scanning of a large environment a tedious and time consuming task. During this time any part or object related to the scan has to be fixed, since any movement will render the scan useless.
Since the laser has to pan over all surfaces, there is no simple way of applying this scanning method in an automated fashion for objects of varying shapes and size.

2.1.2 Structured light

Scanning methods based on structured light are of a similar nature as the laser triangulation in the sense that it measures the deformation of a light-based projection onto an object. The difference lies in the type of pattern projected; where laser triangulation only projects a laser line or dot, the structured light instead projects a two dimensional structured image. Like the laser triangulation method, the shape of the object is then calculated through triangulation based on the distortion of the projected structural image.

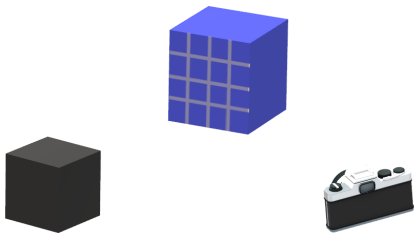


Figure 2.2: Method of structured light scanners

The main advantage of structured light is the larger field of vision compared to laser triangulation. This reduces the time needed to capture any relevant geometry. A drawback of the structured light method is, just like the laser triangulation, the need of the object in question to be in a fixed position during scanning. It also requires the scanner to pan over all objects and surfaces which are of interest, and the time

required to perform a scan scales with the area that needs to be covered.

2.1.3 Time-of-flight laser scanners

Just like the laser triangulation, time-of-flight laser scanners utilizes a laser and a receiver to detect objects in the surroundings. The difference lies within how the laser is utilized; while a constant laser is emitted in laser triangulation, the time of flight scanners instead send out a laser pulse. The time it takes for the laser to be reflected on a surface, bounce back and be registered by a receiver is then timed, which can be used to calculate the distance traveled by the laser in that direction. This is repeated multiple times, where each successful measurement represents a single point in the point cloud.
Compared to laser triangulation and structured light, which moves the scanning tool around the object, the time-of-flight scanners have a fixed point in space from which the scanning is performed. The data gathered from each scan will therefore be the environment from that point of view, no matter the distance.
Due to the technology, once the scanner has been placed into the environment it will need to send out multiple impulses to capture the point of view from this exact position. This is performed automatically, compared to previously mentioned methods which needs to pan over the object of interest.
Since this method scans the environment from one specific point in space, objects might obscure anything which is positioned behind them from the point of view of the scanner. Therefore the scanning equipment has to be moved to multiple locations to capture the whole environment. In addition to multiple scans, a few reference points also needs to be placed into the environment. These are used to locate the relative position between the scans, so that a complete representation of the environment can be constructed.
The number of scans needed to create a good environment representation differs based on the complexity of what is to be scanned, as well as the area that is intended to be covered. Due to this each scanning session needs to be planned out ahead based on the current scenery. Like the previous methods the area scanned needs to be immobile for the scanning to work. Since each scan usually takes a few minutes to complete, this scanning process can become time consuming.

2.1.4 Photogrammetry

Another way of creating a spacial representation of an environment is through photogrammetry. Photogrammetry is the technique to create a 3D model from 2D images by utilizing the principles of the pinhole camera model (equation 2.1, illustration in figure 2.3) and extracted features from the images. The foundation of the pinhole camera model states that there exists global coordinates U which can be transformed into the local, image pixel coordinates u through multiplication with the camera matrix P.

$$u = PU \Longrightarrow \begin{bmatrix} x_{im} \\ y_{im} \\ \lambda \end{bmatrix} = P \begin{bmatrix} X_{gl} \\ Y_{gl} \\ Z_{gl} \\ 1 \end{bmatrix} \tag{2.1}$$

The camera matrix P further consists of the following components: a 3x3 calibration matrix K, which contains information needed for converting between pixels and world coordinates a 3x3 rotation matrix R, representing the orientation of the camera relative the world coordinate frame a 3x3 identity matrix I a translation vector C, representing the translational movement of the focal point of the camera in global coordinates.

$$P = KR[I|-C] = KR \left[\begin{array}{ccc|c} 1 & 0 & 0 & -C_x \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 1 & -C_z \end{array}\right] \tag{2.2}$$

By analyzing and detecting features in each image, these features are used to match images which are taken from positions in close proximity to each other. Then, by utilizing these features alongside RANSAC and triangulation, the relative position from which each image is taken can be calculated.

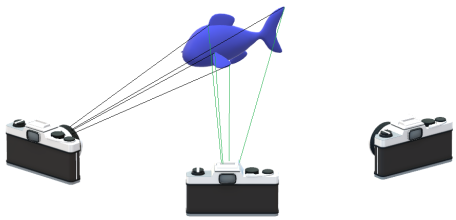


Figure 2.3: Illustration of feature matching in photogrammetry

The main drawback with photogrammetry is that it needs images to overlap to be able to find matching reference points, and the images needs to be processed to know if they create a good enough result.
Like the previously mentioned methods, photogrammetry needs the environment to be still during the data gathering. Any deviations or conflicting poses will not be recognized as part of the environment, and could in worst case be completely left out. Compared to laser scans, photogrammetry is more flexible since a camera isn't fixed in space the same way a laser is. Since it is easier to move around it can be easier to capture some intricate details which might only be visible from one point of view.
There exists a few different software which can create a point cloud through photogrammetry, for example Reality Capture and Alice Vision.

Another drawback of photogrammetry is that it is lighting sensitive. Varying light within images will produce different features, and too dark environments might completely omit feature detection. The accuracy of the resulting point cloud also depends on the resolution of the images, since a low resolution increases the uncertainty of the location of the point in the real world.
The time it takes to complete a photogrammetry session depends mainly on two things. First, the level of detail detection increases time consumption. Since at least two images is needed of the same area to be able to calculate the depth, an object with a lot of intricate details will need a lot of images taken from many different angles. This will slow down the process considerably.
Secondly photogrammetry has an advantage over the previously mentioned methods, namely the possibility of using multiple cameras at the same time. As long as no parts are moving in between the shots, there is no limitation on how many that can be used at the same time. Therefore the time needed for data gathering with photogrammetry mainly comes down to the cost of the cameras that should be used. If the relative positioning of the cameras is known, it could also help speed up the processing. Even clustering cameras together, and thereby knowing their position relative each other, could be of great help reducing the processing time.

2.2 Change detection

Since the goal is to be able to detect geometrical changes that has occurred in an environment, a comparison between two instances created at different times needs to be made.
As mentioned in the beginning of 2.1, a common way of representing 3D objects is through point clouds. Due to this, special tools has been developed to aid in analyzing point clouds. These are mainly divided into either programming libraries, like PCL, or specialized software such as Cloud Compare.
By aligning the two instances of point clouds and selecting one of them as a reference, the distance between the closest points within the point clouds can be calculated.
Comparison of point clouds has previously been researched at Chalmers [17]. This thesis compared the accuracy of photogrammetry with a time-of-flight laser scan by calculating the distance to the closest point between the point clouds. The deviating points occur mainly to the different detection ratio between the two methods, which in turn shows that both capturing methods is capable of creating an accurate representation of the environment.

2.3 Identification

To be able to tell a computer what object has been added to or removed from the real environment, there needs to be some sort of identification process. This is commonly done through a trained neural network, which is a proven method for computers to identify objects from digital media.
A known method of accomplishing this is through computer vision. They have time and time again showed promising results in digitally being able to detect and classify

objects appearing within images.
The main drawback with a neural network is that it needs a lot of images for training. These images needs to be annotated based on the level of detection, ranging from general content word tags to defined bounding boxes or segmentation of images. This is a time consuming and labor heavy task, in some cases being borderline futile.
To reduce the work needed for annotating images, research have tested if synthetic images can be generated and used to train a neural network. In [18] it is shown that this is indeed possible, but not without its drawbacks. The main difference is the reduction in prediction accuracy, which can be attributed to the rendering quality of the images.
Even though there is a reduction in prediction accuracy, one can argue that the reduction of work associated with automatically generated images and annotations outweighs the drop in accuracy.

In [19], a method of identifying objects within a point cloud is presented. This is done by projecting the point cloud into 2D depth images. These images are constructed from planes either cutting through the point cloud, or evenly positioned and spread out on an encapsuling sphere.
The same procedure is performed on objects of interest. Using the projected 2D depth images from the object, features are extracted which is used for identification. By matching features between the environment and object point clouds projections it is possible to detect within each image where an object of interest is located. This area is highlighted and projected back into a 3D environment. The objects location is then determined from the overlap between the projected areas.
Since the pose of the planes which the point clouds are projected into is known, once a match between object and environment is found an orientation of said object can be estimated. This is achievable since the pose of each image is known, and through a chain of transformations the orientation of the object within the environment can be estimated.
Even though this method is capable of detecting objects within an environment, there are a few drawbacks with using this method to detect changes in an environment. Primarily there exists no coordinate system, which makes the calculated position only relative the whole point cloud. Secondly the accuracy of the estimated pose is dependent on the spread of the generated 2D depth images. Thirdly it has to be known what object is sought or all objects of interest will be marked. This is typically not the purpose of detecting changes to an environment, where an unknown object should be identified. As a fourth remark, when searching for all objects of interest the computational time scales with the size of the analyzed environment.

2.4 Pose Estimation

To be able to automatically update a digital twin it is also necessary to be able to determine the pose of the objects that are the source of the geometrical changes. there exists mainly two different methods which allows a computer to do this manually, which is either through the use of a trained neural network or by manipulating and

matching the geometrical data of objects that are stored in a point cloud format.

2.4.1 Neural network

Neural networks can be used to estimate the pose of an object in a similar way to object classifiers. By training a neural network to identify certain feature points on an object, a computer is capable of estimating the pose of an object from a 2D image.
The main drawback with this method is that it requires a lot of data to train such a network, and to the authors knowledge there has not been any attempts at training such a network on synthetic data. Therefore, for the purpose of integrating such a network in the workflow, a lot of training data would have to be manually generated. Coupled with the accuracy and reliability of the pose estimation from such a network, this method has been removed as a viable option at the current time.

2.4.2 Point cloud data manipulation

Point clouds generally contain a lot of data and information themselves. Since each point in the point cloud represents a distinguishable instance within the environment, it can be assumed that points with similar features should be located in close proximity to each other. This can be used as a reference when comparing and aligning point clouds with each other.
Global registration is a method which aims to roughly align two point clouds with each other. This can be done in many ways, but the common denominator is to identify features related to the points in the point clouds and then use RANSAC on these points. Global registration is a method which is best suited to roughly estimate the pose of an object fast, but is inefficient for fine tuning.
Another way of comparing point clouds is the iterative closest point (ICP). As it name entails, it iterates through a comparison between two point clouds based on the distance between the closest point in the other point cloud. By making small adjustments to the aligned pose in between each iteration instance, the point cloud alignment can be fine tuned.
A known drawback with ICP is the possibility of reaching a local minimum. This mainly occurs when trying to align point clouds without similar starting positions. By using global registration to estimate a rough alignemnt, and ICP to fine tune the pose, it is possible to avoid the local minimums which might occur when only running a ICP algorithm.

3

Proposed workflow

Taking all methods and techniques presented in Chapter 2 into consideration, the identification and change detection process stands out with its limited options. For identification, all methods uses some form of computer vision applied on images. This leads to images being a core component of the developed method. In the case of detecting geometrical changes, it is possible as long as the environment is represented in a point cloud format.
All methods which is capable of detecting the geometry of an environment can produce point clouds as the output format. Even though laser triangulation and structured light is capable of producing accurate representations of the environment, both of their applicability is limited based on the need of an operator. A general way to automate these methods does not simply exist.
As can be shown in [17], both time-of-flight scanning and photogrammetry are capable of producing similar results. Due to this photogrammetry is to be preferred since images is needed for identification and can be reused. The time needed to scan an environment with a time-of-flight scanner scales with the complexity of the environment, while the time needed to capture photographs of the environment can be reduced by using multiple cameras at the same time. Using cameras can also be considered less invasive, since they are easy to remove from the environment should it be needed.

Based on these arguments, and the sub-problems presented in Chapter 2, the method to detect changes in an environment can be divided into three sub-processes. Following is a more in depth presentation of these processes.

3.1 Prerequisites

For the intended method to function, a few assumptions based on the performance of previously presented state of the art research have been made.
Firstly it is assumed that there exists some sort of clever component repository, containing a CAD file of each object of interest. Secondly it is assumed to exist a neural network which is capable of detecting these objects within images. For convenience, it is also assumed that this network is capable of identifying each object with 100% success rate, as long as a reasonable part of the object is visible within the image.

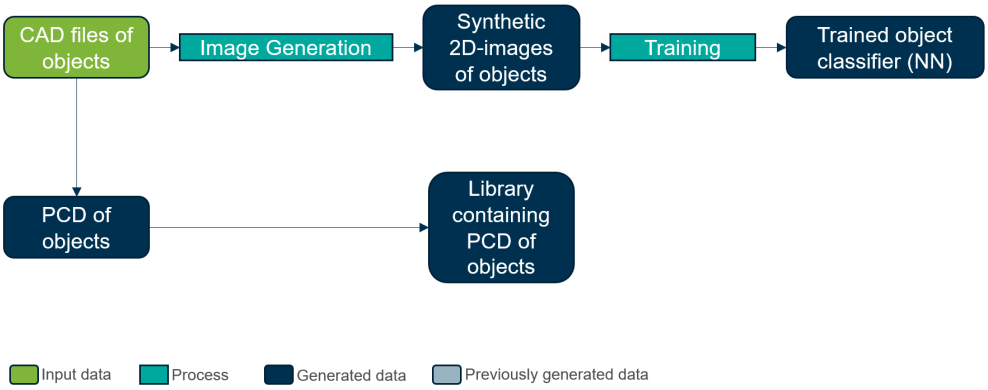


Figure 3.1: Diagram of input data processing workflow

3.2 Processing of input data

Compared to the information acquisition mentioned in 2.1, the processing of input data also includes all the work that needs to be prepared for the proposed workflow to function. This work is mainly related to the object classifier used for identifying the objects of interest.
For full automation of the proposed method, an external trigger is needed to initiate the process. This trigger can favorably be connected to an anchor point, and is hereon assumed to exist.

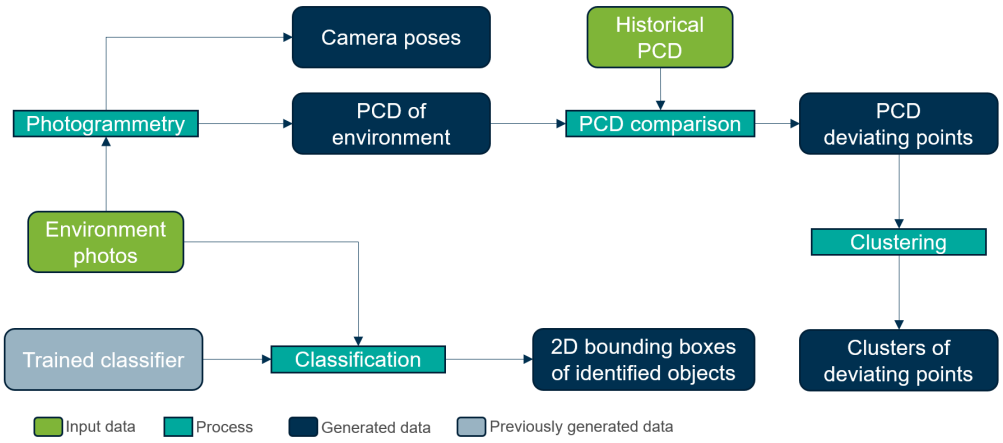


Figure 3.2: Diagram of input data processing workflow

3.2.1 Photogrammetry

Once the trigger is activated, new photographs of the environment is captured. By using these images and photogrammetry, a point cloud representation of the environment can be generated. Due to familiarity, previous use and the accuracy of

the software, Reality Capture, created by Capturing Reality, have been used when developing this workflow.
For reference and scaling of the environment, a reference sheet has been used. This reference sheet will be placed within the environment, creating a plane of four points with known distance to each other. This ensures that the point clouds will have the same size, which is crucial for further analysis.

3.2.2 Object classification

Alongside the photogrammetry, the photos will be used to identify any objects of interest. The intention is that this should be done by passing each image through the pre-trained neural network, which generates bounding boxes of all identified objects according to the YOLO notation. The neural network is intended to have been trained on synthetic images created from the CAD-files of the objects that are relevant to the environment, as stated in 3.1. By doing so the developed workflow will have a focus on automating the process.

3.2.3 Point cloud comparison and clustering

The newly generated point cloud of the environment is now compared with a historical point cloud of the environment. Even though the coordinates of the points generated in each photogrammetry session are different, the distance between the points representing the same object in the same position is short [17].
By comparing the two point clouds it is possible to create a scalar field associated with the compared point cloud, representing the distance to its closest neighbor in the reference. By filtering the points based on this distance it is possible to omit all but the deviating ones. These points then represents some sort of geometric change that has occurred within the environment.
The detection of deviating points is closely related to which point cloud is used as the reference. By comparing the new point cloud with the historical one, any added object will be detected. By changing the comparison order, and use the newly generated point cloud as reference instead, an object that has been removed can be detected in the same fashion.
It is safe to assume that filtered points that are positioned closed to each other belong to the same object. Therefore the deviating points are grouped together based on the distance between them, generating clusters of points which is assumed to represent an object.

3.3 Identification

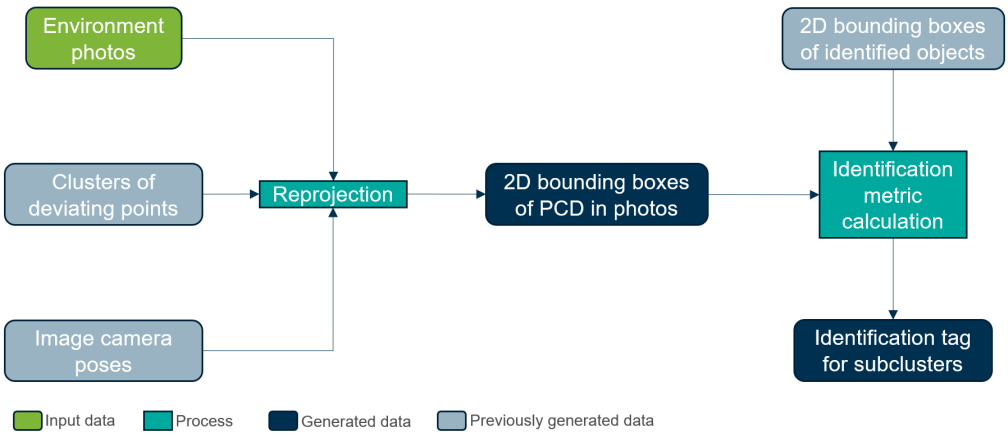


Figure 3.3: Diagram of identification workflow

When the deviating points have been separated into clusters of geometrical change, each cluster has to be analyzed to determine if it is an object of interest. This is done by projecting the deviating points back into the photographs of the environment through reversed photogrammetry. Then a 2D bounding box is generated for each cluster by extracting the maximum and minimum pixels in x- and y-direction, encapsuling all 2D points of that specific cluster.
Once all the clusters have been re-projected, the 2D bounding boxes from the YOLO neural network is imported. By calculating the intersection over union (IoU) between the neural network bounding boxes and the cluster bounding boxes, each cluster of deviating points can be matched with an identified object of interest. The IoU is calculated for the re-projection of clusters in all images, from which a mean overlap is calculated.

To properly identify the relevant clusters, while also avoiding faulty identifications, an identification metric is created. This metric is the product of the IoU and the ratio between the number of times bounding boxes overlap and the number of times an object occurs within all the images.

$$metric = IoU * \frac{nr\_overlaps}{nr\_occurrences} \tag{3.1}$$

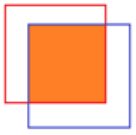


Figure 3.4: Illustration of IoU, intersection over union

This metric weighs in favor of a high level of overlap, while limiting faulty identifications that can occur when an object is positioned in such a way that it perfectly obscures the real object that is responsible for the geometrical change.

3.4 Pose estimation

After the clusters representing the objects of interest have been identified, the next step is to estimate the pose of the object. Each cluster is paired with a point cloud of the identified object, where the cluster is used as a reference. As described in section 2.4.2, a combination of two algorithms is used to estimate the pose. First a global registration is performed to estimate a rough transformation, which is then improved upon with ICP.

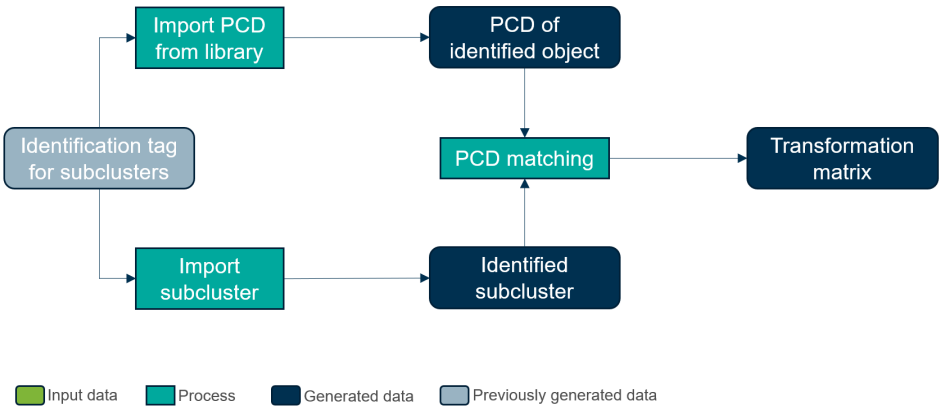


Figure 3.5: Diagram of pose estimation workflow

Once an estimation of the pose have been made, the resulting transformation matrix can be forwarded to a software which updates the digital twin.

4

Test case generation

To test the performance of the proposed workflow a simple test environment was created, which contains up to five pre-selected objects. These objects are a power drill (PD), a robot vacuum (RV), a manual vacuum (MV), a fire extinguisher (FE), and a drone case (DC). To generate a few different test cases from the environment, some of the objects were either added to, removed from or re-positioned within the environment.
For each test case a number of photos were taken using an Olympus PEN E-PL9[20]. These images will be used for creating a point cloud representation of the environment, as well as input for the identification process. Due to limited time for this thesis all images were manually annotated. Since the output from the neural network is already assumed to have a 100% accuracy this won't affect the results, and the intention of this thesis is to evaluate whether or not an automated process for the intended purpose can be developed.
For creation of the point clouds the software Reality Capture was used. It was chosen due to its performance and familiarity with the interface as well as the possibility to export the camera poses, which are needed to re-project the deviating points back into the image.
Since a point cloud database doesn't exist, the objects used in these test cases were scanned using a handheld 3D scanner. The three objects used for testing were either scanned using an Artec Eva Lite[21] or an Artec Space Spider[22], creating a point cloud representation of these objects.
After generating these test cases, before testing the identification and pose estimation process, a comparison was made between each captured setup of the environment to make sure that the process would be able to filter out the object of interest. In this chapter the data from these comparisons will be presented.

4.1 Benchmarks and measurements

To validate the accuracy of the point clouds, a reference setup was created. This setup was first scanned with a FARO time-of-flight laser scanner, before photos were taken which were used to create a similar point cloud through photogrammetry. By comparing these two point clouds the deviations in distance could be measured, which can be seen as a benchmark of the accuracy of the photogrammetry-generated point clouds.

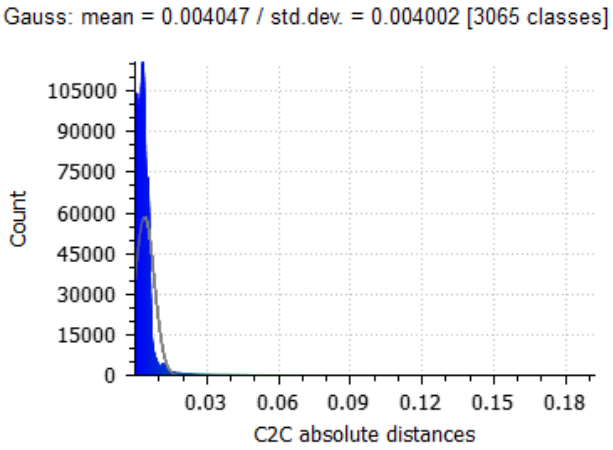


Figure 4.1: Histogram of deviation between photogrammetry and laser scan

As can be seen in Figure 4.1, the mean deviation between the point clouds generated from photogrammetry and time of flight laser scan is just above 4mm. This is in line with the findings from [17], which shows that it is possible to generate point clouds with the expected accuracy with the test setup.

4.2 Test case description

Each point cloud generated for the test cases were compared with the point cloud of the reference setup that was created through photogrammetry. This was done to try and emulate a real world application, since the comparison is intended to be done with point clouds generated from photogrammetry.
The distance between the points in the point clouds was then calculated. The distribution of distance deviations was then used to create four filtering thresholds. Any points in the compared point cloud with a distance to its closest match in the reference point cloud less than this threshold is then filtered out, leaving only the deviating points. These thresholds were set at *mean*, *mean + 0.5 std dev*, *mean + 1 std dev* and *mean + 1.5 std dev*, which later will be used to determine if a particular threshold while filtering the point clouds is preferred.
For each test case a reference transformation matrix was manually extracted by aligning the object point cloud with the object in the environment generated by the time of flight laser scan. This transformation matrix was then converted from a 4 by 4-matrix into a pose of rotation (roll, pitch, yaw) and translation (x,y,z), which is used as ground truth when comparing the automatically calculated poses. This conversion is done for clarity purposes, since it is easier to interpret rotational and translational changes.

4.2.1 Added object

The first test case consists of detecting an object that has been added to the environment. In this case a blue manual vacuum has been added, which can be seen in Figure 4.2b.

(a) Reference environment.

(b) Environment with added object.

Figure 4.2: Images of added object

All objects were kept in the same position, with the same orientation, except for the manual vacuum. After comparing the two point clouds, the following histogram and thresholds were generated.

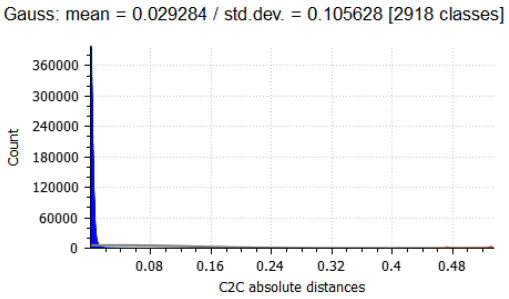


Figure 4.3: Distance distribution between point clouds, added object

Table 4.1: Filtering thresholds, added object

Threshold	Distance (m)
mean	0.029284
mean + 0.5 std dev	0.082098
mean + 1 std dev	0.134912
mean + 1.5 std dev	0.187726

Compared to the benchmark, the mean and standard deviation is relatively large. This could be due to the size of the added object, the vacuum has a height of approximately 60cm. When looking at the different filtering thresholds, it can be determined that parts of the vacuum will be present among the filtered clusters. Lastly, the manually acquired reference pose is presented in Table 4.2, where the origin is placed on the calibration sheet that is visible in the upper left corner of the images.

Table 4.2: Reference pose, added object

X-dir (m)	Y-dir (m)	Z-dir (m)	Roll	Pitch	Yaw
+1.009	+2.120	+0.333	64.11	-5.41	-39.29

4.2.2 Removed object

The second test case is to detect an object that has been removed from the environment. In this case a powerdrill has been removed, which can be seen in Figure 4.4.

(a) Reference environment.

(b) Environment with removed object.

Figure 4.4: Removed object

Taking a closer look at the distance distribution, all values are smaller compared to when the manual vacuum was added to the environment. This further strengthens the theory that the threshold is dependent on the size of the object of interest, since the powerdrill is smaller than the manual vacuum.

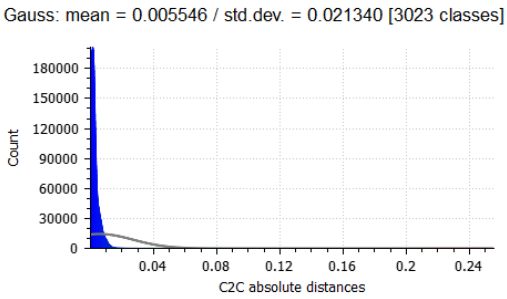


Figure 4.5: Distance distribution between point clouds, removed object

Table 4.3: Filtering thresholds, removed object

Threshold	Distance (m)
mean	0.005546
mean + 0.5 std dev	0.016216
mean + 1 std dev	0.026886
mean + 1.5 std dev	0.037556

Table 4.4: Reference pose, removed object

X-dir (m)	Y-dir (m)	Z-dir (m)	Roll	Pitch	Yaw
+0.700	+1.230	+0.090	62.05	-0.79	-128.82

4.2.3 Re-positioned object

If an object is present in both the before and after point clouds, but not located in the same position, it can be assumed that the object in question has been moved. This will result in the same object being detected when searching for both added and removed objects, and the same principles can be extended to include this type of change to the environment. For this test case, the powerdrill was moved to another position in the environment.

(a) Reference environment.

(b) Environment with re-positioned powerdrill.

Figure 4.6: Re-positioned powerdrill

First the new point cloud was set as reference, to detect if any object of interest has been removed. The calculated distance deviations are of a similar magnitude as the case when the powerdrill was removed which is what to be expected, since in both cases the same object has disappeared compared to the reference.

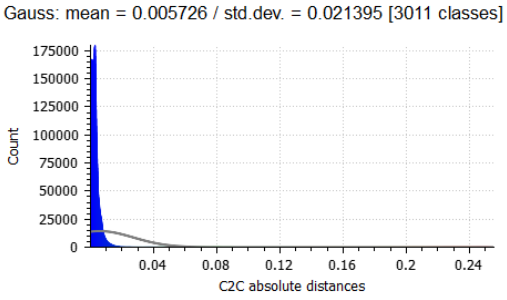


Figure 4.7: Distance distribution between point clouds, before object re-position

Table 4.5: Filtering thresholds, before object re-position

Threshold	Distance (m)
mean	0.005726
mean + 0.5 std dev	0.0164235
mean + 1 std dev	0.027121
mean + 1.5 std dev	0.0378185

Since the position of the powerdrill is identical to before it was removed, it will in this case have the same reference pose as before removal.

Table 4.6: Reference pose, before object re-position

X-dir (m)	Y-dir (m)	Z-dir (m)	Roll	Pitch	Yaw
+0.700	+1.230	+0.090	62.05	-0.79	-128.82

Now swap the comparison reference to the old point cloud to detect any added objects. As can be seen in the histogram in Figure **??**, some deviation is detected. Compared to the thresholds used to filter the before point cloud, these are of a larger magnitude even though the same object is used. This could be explained by looking at the upper end of the distance distributions of the two comparisons. The largest distance deviation in Figure 4.7 is about 24cm, while the scale in the after comparison reaches 70cm. This could be due to a higher level of noise within the after-point cloud, which also explains the higher filtering threshold. in this case.

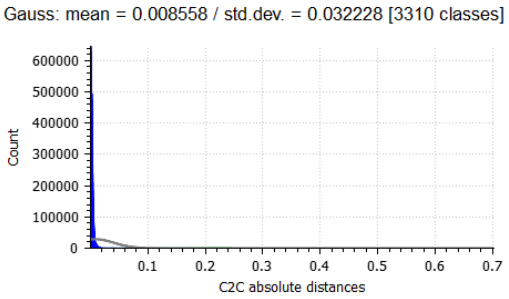


Figure 4.8: Distance distribution between point clouds, after object re-position

Table 4.7: Filtering thresholds, after object re-position

Threshold	Distance (m)
mean	0.008558
mean + 0.5 std dev	0.024672
mean + 1 std dev	0.040786
mean + 1.5 std dev	0.0569

Table 4.8: Reference pose, after object re-position

X-dir (m)	Y-dir (m)	Z-dir (m)	Roll	Pitch	Yaw
+0.881	+1.773	+0.086	63.19	-1.02	87.45

4.2.4 Swapped object

Similar to the case of an object being re-positioned, the same principles of combining the comparisons can be applied to detect if an object of relevance has been swapped out. For example, if an industrial robot has been swapped out for a newer model. The limitation is that there has to be a big enough geometrical difference between the objects. Objects with similar shape will be impossible to differentiate from each other, since they will appear identical in the point cloud.

(a) Reference environment.

(b) Environment with swapped vacuum and removed powerdrill.

Figure 4.9: Swapped vacuum

In similar fashion as the previous test cases, a change that has occurred to a smaller object leads to a smaller distance deviation. In this case the powerdrill has been removed and the robot vacuum has been replaced, which results in both objects contributing to the distance distribution between the two point clouds.

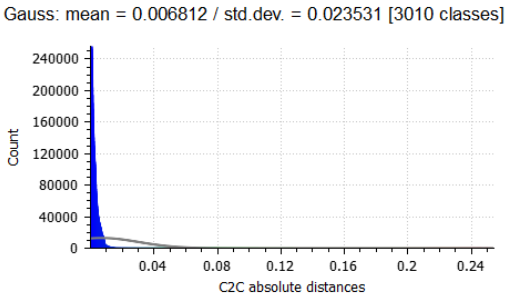


Figure 4.10: Distance distribution between point clouds, before object swap

Table 4.9: Filtering thresholds, before object swap

Threshold	Distance (m)
mean	0.006812
mean + 0.5 std dev	0.0185775
mean + 1 std dev	0.030343
mean + 1.5 std dev	0.0421085

Table 4.10: Reference pose, before object swap (robot vacuum)

X-dir (m)	Y-dir (m)	Z-dir (m)	Roll	Pitch	Yaw
+1.726	+1.744	-0.002	57.89	-1.53	74.01

When comparing the after-swap environment to the before, the distribution once again approaches the levels of when the manual vacuum was added.

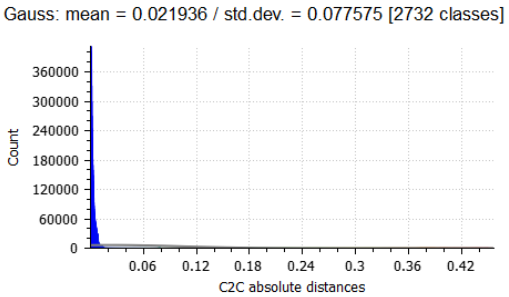


Figure 4.11: Distance distribution between point clouds, after object swap

Table 4.11: Filtering thresholds, after object swap

Threshold	Distance (m)
mean	0.021936
mean + 0.5 std dev	0.0607235
mean + 1 std dev	0.099511
mean + 1.5 std dev	0.1382985

And the reference pose of the manual vacuum is similar to the reference pose of the robot vacuum. This is an indication of an object swap, and could possibly be used to distinguish a replacement from a re-position.

Table 4.12: Reference pose, after object swap (manual vacuum)

X-dir (m)	Y-dir (m)	Z-dir (m)	Roll	Pitch	Yaw
+1.720	+1.665	+0.299	14.01	-22.40	-33.22

From looking at the cloud to cloud distribution of points, it can be determined that the distance between the points in the different point clouds is small. Therefore it can be assumed that the point clouds have been well aligned with each other. This is crucial to be able to detect any geometrical changes to the environment, and through these comparisons the alignment has been verified to be satisfactory to continue to test the next step of the workflow.

5

Results - identification and pose estimation

The previous chapter indicated that the filtering works as intended. As mentioned in section 1.4, there are two main measurements regarding the performance of the proposed workflow. Below are presented the results of the identification and the pose estimation.

5.1 Identification

For each generated test case, the point clouds are filtered with each of the four thresholds presented in Chapter 4. All points with a distance above the set threshold is kept and clustered together, since they can be assumed to be part of the same geometric shape and thus part of the same object. Each cluster are then re-projected back into each image that the environment is constructed of and bounding boxes are created, from which an identification metric is calculated according to equation 3.1.

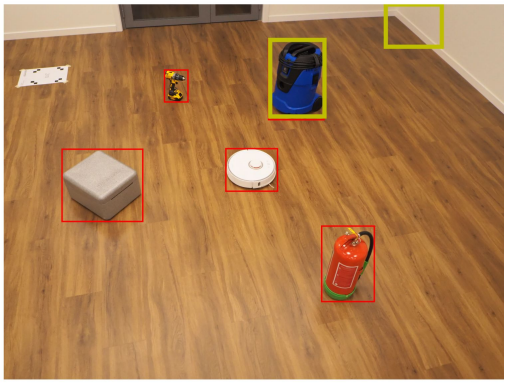
5.1.1 Added object

Taking a closer look at Table 5.1, 7 clusters remain after filtering with the lowest threshold. Among them is one cluster that produces an identification metric of 0.905, which on a scale from 0 to 1 is a good indication of a good overlap appearing in many images. In Appendix Table A.1 it can be seen that only a single cluster produces an identification metric above 0.02, which is a good indication for the feasibility of the developed cluster identification method.
As the filtering threshold increases the number of clusters and the identification metric decreases, which is logical since more and more deviating points are removed before the re-projection. This is illustrated when comparing the images 5.1a and 5.1b in Figure 5.1, where the yellow bounding box indicating a cluster decreases in size.
The decrease in the identification metric correlates with the filtering distance, since a higher threshold will remove a larger part of the object. At the highest threshold the filtering distance is set to 18.8cm which is almost a third of the height of the manual vacuum, and it results in approximately a 30% decrease in the identification metric.

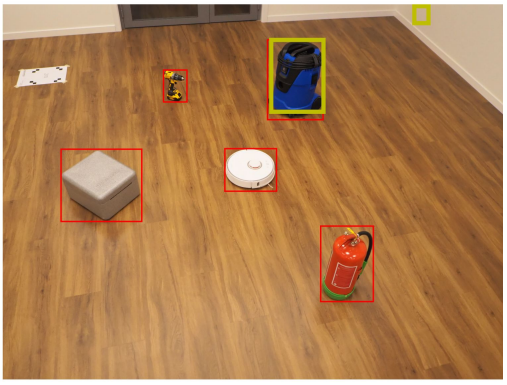
Table 5.1: Identification metric for added vacuum

Threshold	Nr clusters	PD	RV	MV	FE	DC
Mean	7	0.002	0.001	0.905	0.002	0.001
Mean+0.5	7	0.001	0.001	0.826	0.002	0.000
Mean+1	3	0.001	0.001	0.702	0.002	0.000
Mean+1.5	1	0.001	0.001	0.613	0.001	0.000

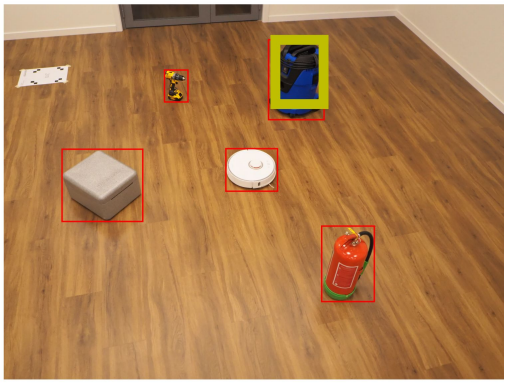
Even though only a single cluster remains after the highest level of filtering, it still produces a small result in the identification metric for one of the other objects of interest. This occurs when one of the other objects obscures the manual vacuum in any of the images, but the low identification metric indicates that this is not a feasible identification.



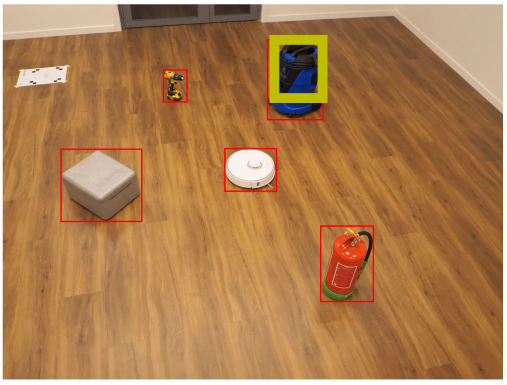
(a) Threshold: Mean deviation



(b) Threshold: Mean + 0.5 std deviation



(c) Threshold: Mean + 1 std deviation



(d) Threshold: Mean + 1.5 std deviation

Figure 5.1: Overlap illustration with added object

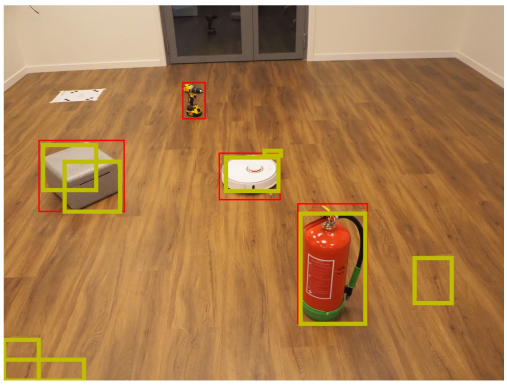
5.1.2 Removed object

For the second test, the number of separated clusters increased at the lowest filtering threshold. This is to be expected since the filtering distances are lower, and more noise could remain after filtering. What is interesting here is that neither of the

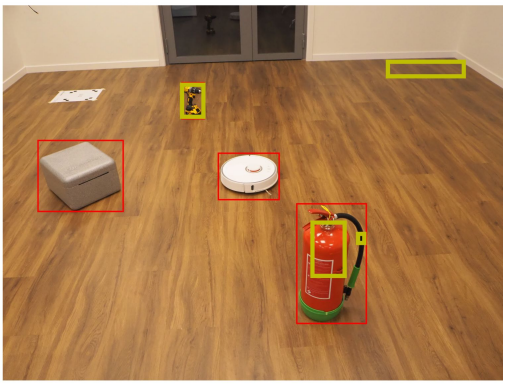
clusters are identified as the removed powerdrill, but instead they occur around the other objects in the environment (see Figure 5.2a). The set threshold is incapable of separating the points representing the powerdrill from the immediate environment, and thus the identification fails. This is probably due to a poor alignment between the two point clouds, since a deviation was detected in the objects that was not moved.
However, as the filtering threshold increases the proposed method is capable of separating the points representing the powerdrill, and becomes capable of identifying it correctly.

Table 5.2: Identification metric for removed powerdrill

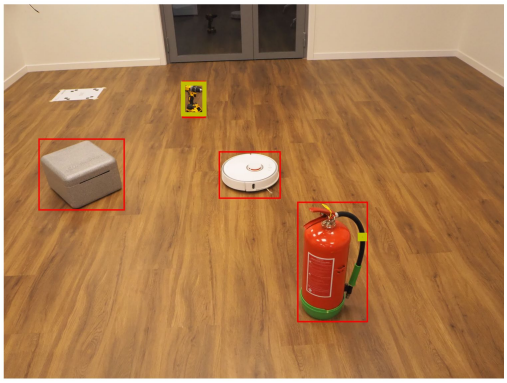
Threshold	Nr clusters	PD	RV	MV	FE	DC
Mean	10	0.001	0.785	0.000	0.829	0.381
Mean+0.5	5	0.802	0.001	0.000	0.211	0.001
Mean+1	2	0.777	0.001	0.000	0.002	0.001
Mean+1.5	1	0.753	0.001	0.000	0.000	0.001



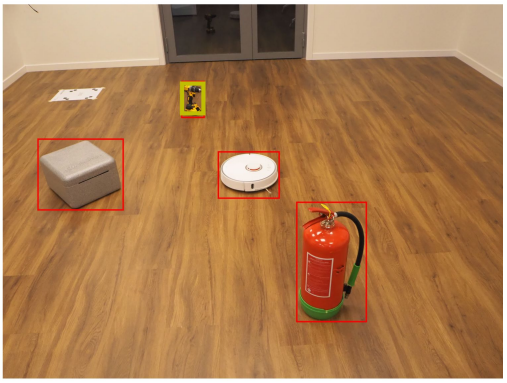
(a) Threshold: Mean deviation



(b) Threshold: Mean + 0.5 std deviation



(c) Threshold: Mean + 1 std deviation



(d) Threshold: Mean + 1.5 std deviation

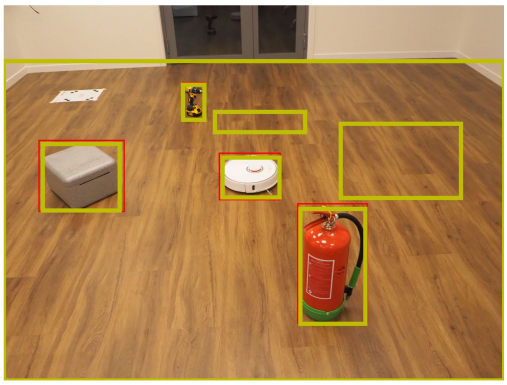
Figure 5.2: Overlap illustration with removed object

5.1.3 Re-positioned object

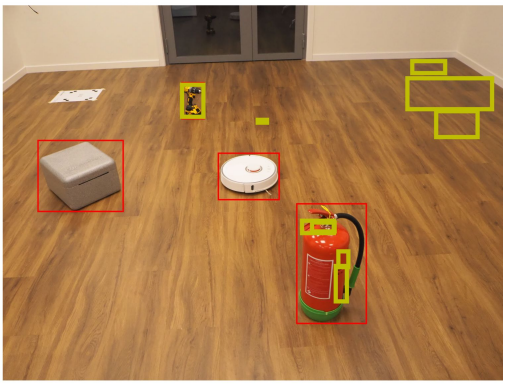
Compared to the test case of removing the powerdrill, when it is re-positioned the points representing said object can be separated from its immediate environment. This strengthens the argument that the results produced when removing the powerdrill occurs due to a poor alignment between the point clouds. However, multiple small clusters are still present and they increase in number as the filtering threshold is increased to the next step. This is an indication of noise being present throughout the point cloud, but once the filtering threshold reaches a high enough level the overlap between the bounding boxes becomes so small that a good identification of a cluster can still be made.

Table 5.3: Identification metric before re-positioning of powerdrill

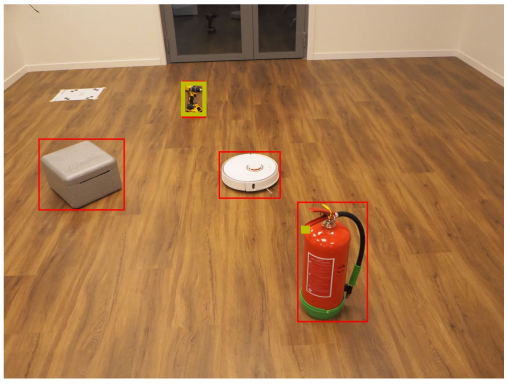
Threshold	Nr clusters	PD	RV	MV	FE	DC
Mean	7	0.838	0.852	0.000	0.882	0.865
Mean+0.5	13	0.801	0.002	0.000	0.047	0.002
Mean+1	2	0.776	0.001	0.000	0.001	0.001
Mean+1.5	1	0.751	0.001	0.000	0.000	0.001



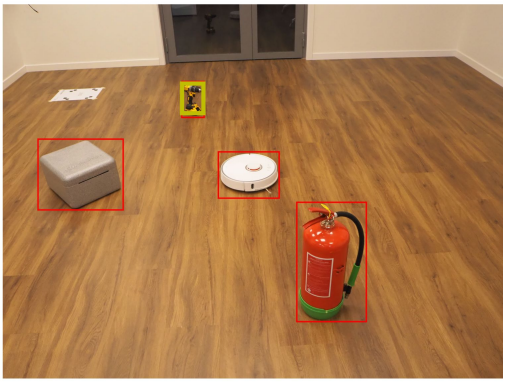
(a) Threshold: Mean deviation



(b) Threshold: Mean + 0.5 std deviation



(c) Threshold: Mean + 1 std deviation



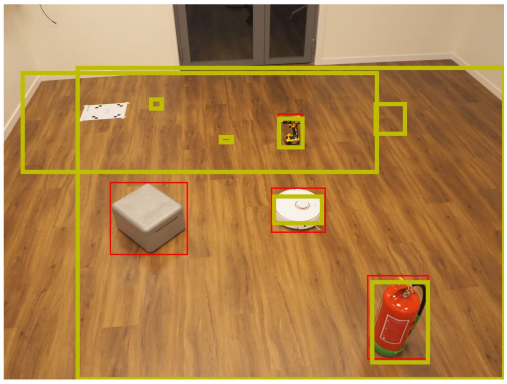
(d) Threshold: Mean + 1.5 std deviation

Figure 5.3: Overlap illustration with re-positioned object, before

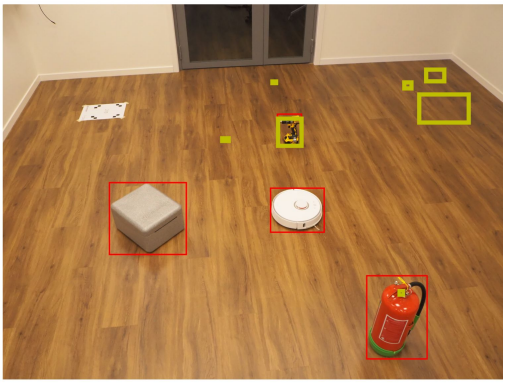
The same problem appears when analyzing the environment after the re-positioning of the powerdrill. A reasonable explanation for this, in all three cases, is that the first filtering thresholds are too low. In similar fashion as previous cases, once the filtering threshold increases it is possible to correctly identify the object that produces the geometrical changes.

Table 5.4: Identification metric after re-positioning of powerdrill

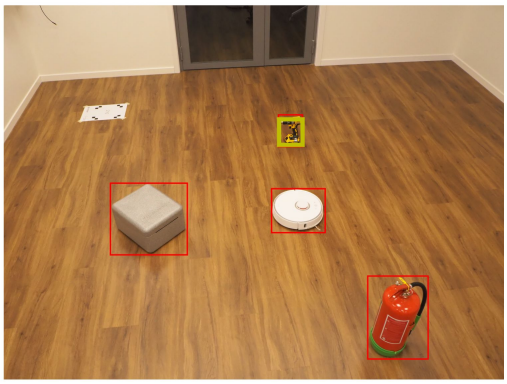
Threshold	Nr clusters	PD	RV	MV	FE	DC
Mean	8	0.844	0.615	0.000	0.799	0.071
Mean+0.5	9	0.803	0.003	0.000	0.002	0.002
Mean+1	1	0.738	0.003	0.000	0.001	0.002
Mean+1.5	1	0.679	0.003	0.000	0.001	0.002



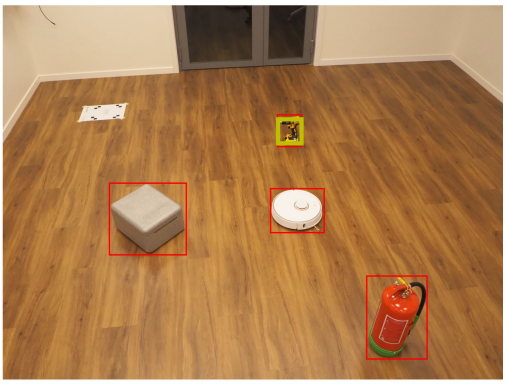
(a) Threshold: Mean deviation



(b) Threshold: Mean + 0.5 std deviation



(c) Threshold: Mean + 1 std deviation



(d) Threshold: Mean + 1.5 std deviation

Figure 5.4: Overlap illustration with re-positioned object, after

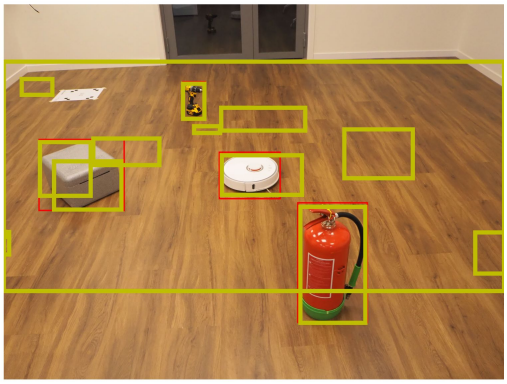
5.1.4 Swapped object

In the case of before a swap of objects occurs, a filtering threshold similar to the removal and re-position of the powerdrill is applied. The same problems at the

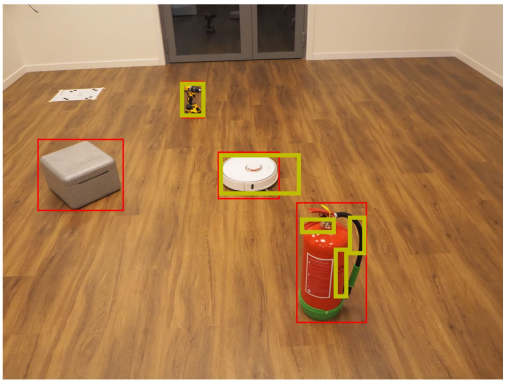
lowest filtering level occurs in this scenario as well, which further strengthens the theory of a too low threshold. As the filtering threshold increases the number of clusters decreases, just like previous test cases, but here something interesting occurs at the higher levels. The robot vacuum is separated into two different clusters that both generate similar identification metrics, more accurately 0.47 and 0.35 as can be seen in Appendix Table A.24.

Table 5.5: Identification metric before replacing of robot vacuum

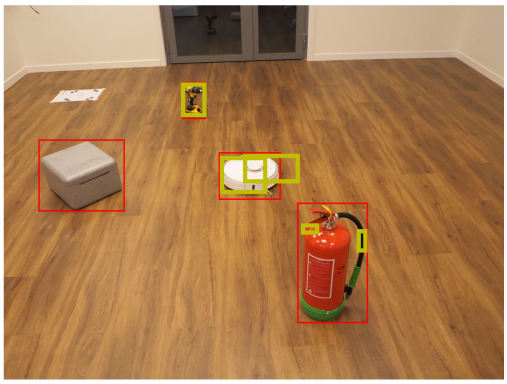
Threshold	Nr clusters	PD	RV	MV	FE	DC
Mean	14	0.837	0.619	0.000	0.877	0.581
Mean+0.5	5	0.794	0.655	0.000	0.061	0.001
Mean+1	5	0.769	0.594	0.000	0.019	0.001
Mean+1.5	5	0.731	0.470	0.000	0.012	0.001



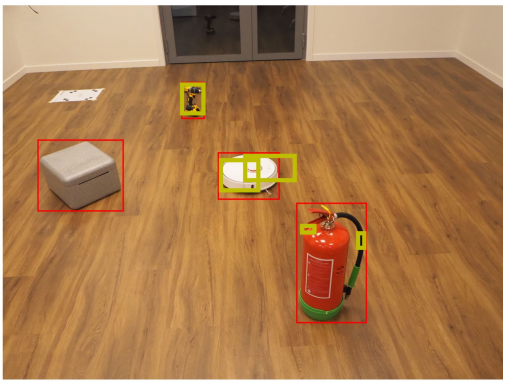
(a) Threshold: Mean deviation



(b) Threshold: Mean + 0.5 std deviation



(c) Threshold: Mean + 1 std deviation



(d) Threshold: Mean + 1.5 std deviation

Figure 5.5: Overlap illustration with swapped object, before

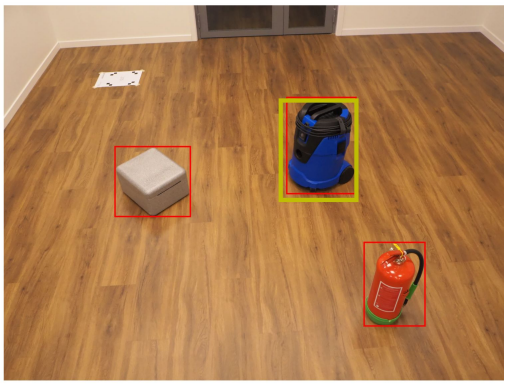
Further analysis into why this occurred resulted in the probable cause of being inflicted by the positioning of the manual vacuum in the after-environment. Since the filtering is based in the distance calculated between the point cloud of interest

and the reference, the manual vacuum becomes part of that reference. Points in close proximity of the manual vacuum is therefore filtered out, which results in a separation if the points representing the robot vacuum.

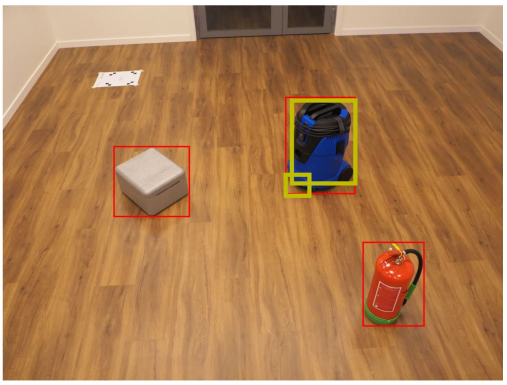
As can be seen in Table 5.6, once the filtering threshold increases all noise can easily be removed even at the lower threshold, resulting in a correct identification even at the lower levels.

Table 5.6: Identification metric after replacing of robot vacuum

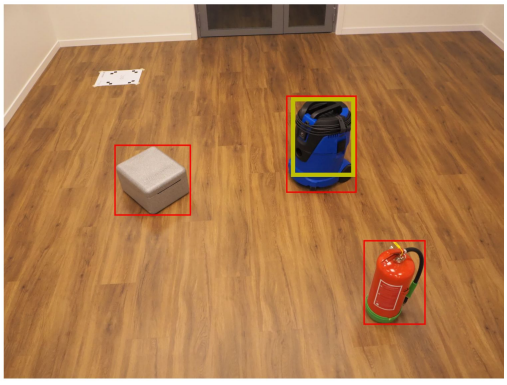
Threshold	Nr clusters	PD	RV	MV	FE	DC
Mean	1	0.000	0.000	0.834	0.015	0.009
Mean+0.5	2	0.000	0.000	0.775	0.010	0.006
Mean+1	1	0.000	0.000	0.698	0.009	0.005
Mean+1.5	1	0.000	0.000	0.545	0.007	0.004



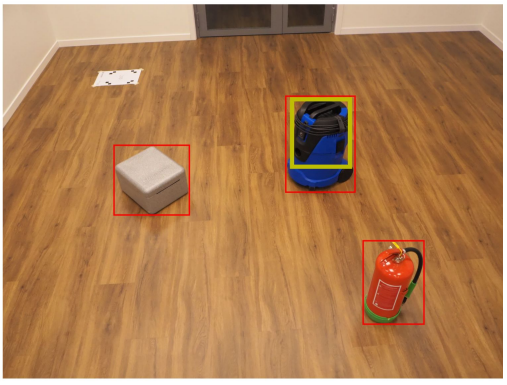
(a) Threshold: Mean deviation



(b) Threshold: Mean + 0.5 std deviation



(c) Threshold: Mean + 1 std deviation



(d) Threshold: Mean + 1.5 std deviation

Figure 5.6: Overlap illustration with swapped object, after

5.2 Pose estimation

For each of the test cases generated, once the clusters has been identified, a global registration followed by a fine registration was performed. The resulting pose was calculated by taking the average of ten calculations, which is then compared to the manually created reference. The difference between the estimated pose and the reference is calculated, and then separated into translation and rotational error.

5.2.1 Added object

As can be seen in Table 5.7, the estimated pose is within a 2cm distance of the reference and the estimated orientation is also quite good. The yaw angle was the hardest to estimate, which is reasonable since the vacuum has a cylindrical shape.

Threshold	X-dir (m)	Y-dir (m)	Z-dir (m)	Roll	Pitch	Yaw
mean	0.011	-0.001	0.000	0.13	0.06	5.36
mean + 0.5 dev	0.011	0.003	-0.005	0.43	-0.07	5.16
mean + 1 dev	0.015	0.001	0.001	0.35	0.11	7.85
mean + 1.5 dev	0.007	0.001	0.001	-0.37	1.08	4.96

Table 5.7: Error compared to reference, added object

Figure 5.7: Illustration of estimated pose, added object

5.2.2 Removed object

For the first filtering threshold in this test case the identification process was unable to correctly identify the powerdrill as the removed object, and therefore no pose estimation was performed after the first filtering. For all other filtering thresholds it was capable of estimating the pose of the powerdrill to a satisfying degree.

Threshold	X-dir (m)	Y-dir (m)	Z-dir (m)	Roll	Pitch	Yaw
mean	—	—	—	—	—	—
mean + 0.5 dev	0.012	-0.007	0.008	3.03	-2.14	-2.89
mean + 1 dev	0.010	-0.005	0.011	3.02	-1.99	-2.36
mean + 1.5 dev	0.012	-0.005	0.010	3.19	-1.82	-2.65

Table 5.8: Error compared to reference, before removal of object

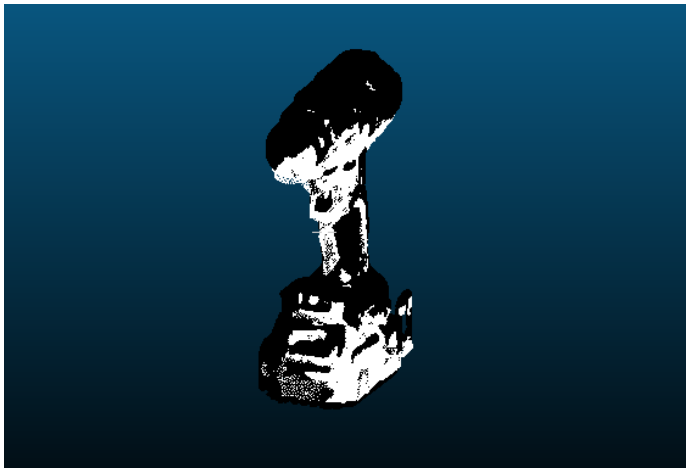


Figure 5.8: Illustration of estimated pose, removed object

5.2.3 Re-positioned object

Compared to the removal of the powerdrill, in this scenario the proposed workflow was capable of identifying the object of interest at all filtering thresholds. The estimated pose have an error of similar magnitude for both before and after the re-position, as well as the removal, of the powerdrill, indicating that there are traces of repeatability in the process.

Threshold	X-dir (m)	Y-dir (m)	Z-dir (m)	Roll	Pitch	Yaw
mean	0.015	-0.007	0.004	4.09	-1.79	0.54
mean + 0.5 dev	0.014	-0.006	0.008	3.77	-1.68	-1.53
mean + 1 dev	0.012	-0.005	0.011	2.93	-0.77	-2.62
mean + 1.5 dev	0.010	-0.003	0.012	1.68	-1.17	-1.92

Table 5.9: Error compared to reference, before re-position of object

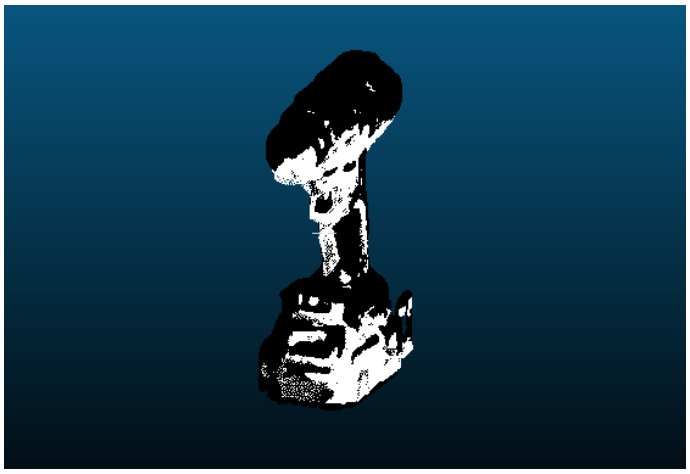


Figure 5.9: Illustration of estimated pose, before re-position of object

Threshold	X-dir (m)	Y-dir (m)	Z-dir (m)	Roll	Pitch	Yaw
mean	0.000	-0.007	0.010	-0.30	-1.20	1.15
mean + 0.5 dev	0.002	0.000	0.011	-0.33	-3.28	1.24
mean + 1 dev	-0.001	-0.002	0.010	-1.05	-1.79	1.48
mean + 1.5 dev	0.000	-0.001	0.010	0.11	-2.95	1.63

Table 5.10: Error compared to reference, after re-position of object

Figure 5.10: Illustration of estimated pose, after re-position of object

5.2.4 Swapped object

Looking at the estimated pose of the robot vacuum at the higher filtering thresholds, they appear to be fairly accurate. However, it gets the estimated orientation completely wrong. It flips the object point cloud upside down and is incapable of correctly aligning it with the cluster. This could be affected by the separation of the clusters, making them incomplete, as well as the shape of the robot vacuum. Since it is relatively uniform, except for the small, raised, circular platform on top, there are almost no distinguishable features for the algorithm to use.

Threshold	X-dir (m)	Y-dir (m)	Z-dir (m)	Roll	Pitch	Yaw
mean	0.027	-0.185	0.099	-180.66	2,97	-129.66
mean + 0.5 dev	0.013	-0.205	0.100	-179.27	2.92	-142.35
mean + 1 dev	-0.029	-0.053	0.101	-181.21	-1.43	-116.15
mean + 1.5 dev	-0.016	-0.007	0.091	183.55	2.03	-61.38

Table 5.11: Error compared to reference, before swapping of object

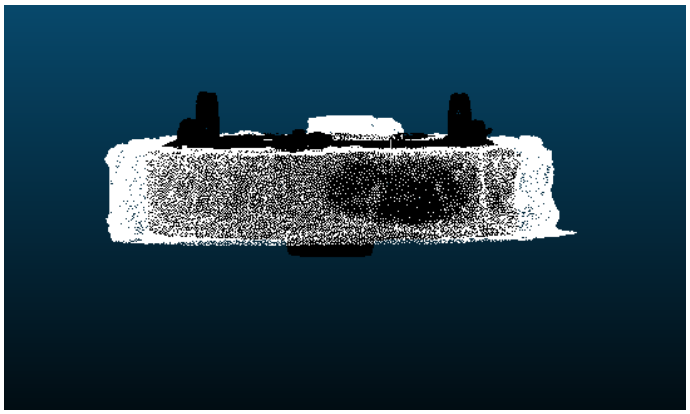


Figure 5.11: Illustration of estimated pose, swapped object before

Taking a look at the estimated pose of the manual vacuum, there is a significant difference between it the reference at all filtering levels. However, when applying both transformation matrices they all align well with the identified cluster, as can be seen in Figure 5.12. This shows that the conversion from a 4x4 transformation matrix into roll-, pitch-, yaw-angles are not without flaws of its own, and the deviation most likely occurs when converting from a transformation matrix into a more comprehensible pose.

Threshold	X-dir (m)	Y-dir (m)	Z-dir (m)	Roll	Pitch	Yaw
mean	0.001	-0.002	0.001	50.06	16.44	0.87
mean + 0.5 dev	-0.011	-0.004	0.029	50.01	16.67	-4.86
mean + 1 dev	-0.001	-0.002	0.034	49.49	15.97	0.52
mean + 1.5 dev	0.001	-0.004	0.032	49.35	16.78	0.77

Table 5.12: Error compared to reference, after swapping of object

Figure 5.12: Illustration of estimated pose, swapped object after

5.3 Result summary

Based on the results in identification, it appears that some environments tends to be somewhat noisy. This noise makes up a large portion of the total deviation, which in turn greatly affects the distribution of the deviation. From the tests conducted, it appears that a threshold of mean + 1 standard deviation produces the best result, but a fixed filtering threshold might perform better than basing the distance on the total deviation between two point clouds of the environment.
When there are distinguishable, geometrical features in the objects of interest it is possible to estimate the pose through global and fine registration, and it appears to perform relatively even between each time the algorithm runs due to the small deviations from the reference pose.

6
Discussion

Even though the results indicate that the developed method is successful, there are a few questions which arose during development that needs to be addressed.
Lets first discuss the identification process. There exists two main requirements that needs to be fulfilled to be able to accurately identify a cluster. The first, and most important, is the need for the trained neural network object detector to be able to distinguish between all objects of interest. Two objects with similar shape and color will be hard to identify correctly, which could lead to wrongly identified changes. Poor illumination would also affect the result, since all cameras utilizes a light-sensitive sensor to capture information from the environment.
It is also noteworthy to mention that no neural network was trained during the development of this workflow. This will affect the accuracy of the identification metric, but the way it is constructed should theoretically still leave room for a few incorrect classification of objects.

The second factor is that the two environment point clouds has to be aligned properly. Any alignment deviations in this step will affect all upcoming processes negatively. An example of this is in the case of low threshold filtering of the point clouds in the cases of removing and re-positioning objects. As can be seen in the Tables A.5, A.9 and A.13 in Appendix A, there exists a few irregularities when the comparisons are filtered with a low threshold. This can only be explained with either noisy and inaccurate point clouds, or a badly performed alignment. Taking a closer look at Table A.5, the test case where the powerdrill was removed, this is most likely the result of a noisy environment. Objects in the environment that was stationary have the highest identification metric while the target object cannot be identified, which is an indication of filtering difficulties. This problem is however reduced as the filtering threshold is increased, which then leads to a correct identification.
Now take a look at Table A.9 and A.13 instead. Here it can be seen that there are multiple objects which have been identified and the correct item, the powerdrill, is among them. This indicates that the filtering has been able to separate the powerdrill from its immediate surroundings such as the floor, while deviations also have been registered in close proximity of the other items. It can be assumed that the noise is lower in these point clouds since filtering is capable of separating the powerdrill from the floor. Therefore the disturbance should mainly come from a less than optimal alignment. The many small deviating clusters in in close proximity of stationary objects are therefore a result of and an indication of poor alignment.
During testing, the point clouds was aligned through the calibration sheet. Any small noise at these points would magnify the deviation further away from the ori-

gin, which is the most probable cause of the poor alignment. A possible solution to this problem could be to apply global and fine registration on the point clouds before comparing them, minimizing the total error, but it has to be investigated further.
Thirdly a discussion of accuracy is in order, more accurately what level is needed for this workflow to be considered as a useful tool. In most generated test cases it was possible to estimate the pose of the object of interest, but when converting the transformation matrix to a translation and a rotation, problems could occur. The illustrations of the overlap indicates that the produced transformation matrix can be correct, but there is currently no automated way to verify this. In addition, small positional errors could build up over time, leading to deviations between the digital twin and the real environment. This could have an effect on the simulations, especially if the digital twin is supposed to simulate work that requires high precision.

6.1 Proposed work for further development

Before the proposed workflow can be deemed successful, there are a few areas and concepts that needs to be developed and integrated into the workflow before it can be tested as a whole. Here are a few suggestions on how to further develop the workflow.

1. Construct an environment for the prerequisites and test the integration into the workflow. This includes the generation of synthetic images and training of the neural network, as well as creating the library with all the needed information.
2. Expanding the pose estimation to include colors of the objects. As of now it only considers the point coordinates and the surface normals when aligning the two instances. This however requires that all point clouds contain color information, and the result could be affected by varying lighting conditions.
3. Develop a method to merge point clouds that are divided into two or more clusters even though they are part of the same object, like what happened with the robot vacuum. If these two clusters could be merged it is possible that a correct pose estimation can be achieved.
4. Investigate the possibility of using and listen in on anchor points to improve the pose on objects in the digital twin.

7
Conclusion

The proposed method is deemed promising since the proposed identification method is capable of correctly labeling deviating clusters, and it is in most cases capable of producing good pose estimations of objects within an environment. There are areas that need further development and testing before the presented workflow should be of consideration of being used to automatically update a digital twin.

Bibliography

[1] T. Lechler, E. Fischer, M. Metzner, A. Mayr, and J. Franke, "Virtual commissioning – scientific review and exploratory use cases in advanced production systems," *Procedia CIRP*, vol. 81, pp. 1125 – 1130, 2019. 52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia, June 12-14, 2019.

[2] A. Khan, P. Falkman, and M. Fabian, "Virtual engineering framework for automatic generation of control logic including safety," in *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pp. 648–653, 2017.

[3] M. Dahl, K. Bengtsson, P. Bergagård, M. Fabian, and P. Falkman, "Sequence planner: Supporting integrated virtual preparation and commissioning," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5818 – 5823, 2017. 20th IFAC World Congress.

[4] M. Oppelt and L. Urbas, "Integrated virtual commissioning an essential activity in the automation engineering process: From virtual commissioning to simulation supported engineering," in *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*, pp. 2564–2570, 2014.

[5] M. Dahl, K. Bengtsson, M. Fabian, and P. Falkman, "Automatic modeling and simulation of robot program behavior in integrated virtual preparation and commissioning," *Procedia Manufacturing*, vol. 11, pp. 284 – 291, 2017. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy.

[6] N. Shahim and C. Møller, "Economic justification of virtual commissioning in automation industry," in *2016 Winter Simulation Conference (WSC)*, pp. 2430–2441, 2016.

[7] A. Albo and P. Falkman, "A standardization approach to virtual commissioning strategies in complex production environments," in *30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2020)*, 2020.

[8] G. Zhou, C. Zhang, Z. Li, K. Ding, and C. Wang, "Knowledge-driven digital twin manufacturing cell towards intelligent manufacturing," *International Journal of Production Research*, vol. 58, no. 4, pp. 1034–1051, 2020.

[9] M. Dahl, A. Albo, J. Eriksson, J. Pettersson, and P. Falkman, "Virtual reality commissioning in production systems preparation," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–7, 2017.

[10] R. Drath, P. Weber, and N. Mauser, "An evolutionary approach for the industrial introduction of virtual commissioning," in *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, pp. 5–8, 2008.

[11] B. A. Talkhestani, N. Jazdi, W. Schlögl, and M. Weyrich, "A concept in synchronization of virtual production system with real factory based on anchor-

point method," *Procedia CIRP*, vol. 67, pp. 13 – 17, 2018. 11th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 19-21 July 2017, Gulf of Naples, Italy.

[12] C. Scheifele, A. Verl, and O. Riedel, "Real-time co-simulation for the virtual commissioning of production systems," *Procedia CIRP*, vol. 79, pp. 397 – 402, 2019. 12th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 18-20 July 2018, Gulf of Naples, Italy.

[13] R. Rosen, G. [von Wichert], G. Lo, and K. D. Bettenhausen, "About the importance of autonomy and digital twins for the future of manufacturing," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 567 – 572, 2015. 15th IFAC Symposium onInformation Control Problems inManufacturing.

[14] T. Okita, T. Kawabata, H. Murayama, N. Nishino, and M. Aichi, "A new concept of digital twin of artifact systems: synthesizing monitoring/inspections, physical/numerical models, and social system models," *Procedia CIRP*, vol. 79, pp. 667 – 672, 2019. 12th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 18-20 July 2018, Gulf of Naples, Italy.

[15] B. A. Talkhestani, N. Jazdi, W. Schloegl, and M. Weyrich, "Consistency check to synchronize the digital twin of manufacturing automation based on anchor points," *Procedia CIRP*, vol. 72, pp. 159 – 164, 2018. 51st CIRP Conference on Manufacturing Systems.

[16] A. Farooqui, K. Bengtsson, P. Falkman, and M. Fabian, "Towards data-driven approaches in manufacturing: an architecture to collect sequences of operations," *International Journal of Production Research*, vol. 0, no. 0, pp. 1–17, 2020.

[17] J. Li, "Application of photogrammetry for virtual commissioning of production systems," Master's thesis, Technical University of Munich, Germany, 2017.

[18] J. Li, "Training convolutional neural networks with synthesized data for object recognition in the automated assembly," Master's thesis, Technical University of Munich, Germany, 2018.

[19] G. Pang and U. Neumann, "Fast and robust multi-view 3d object recognition in point clouds," in *2015 International Conference on 3D Vision*, pp. 171–179, 2015.

[20] "Tekniska data, olympus pen e-pl9." `https://www.olympus.se/site/sv/c/cameras/pen_cameras/pen/e_pl9/e_pl9_specifications.html`. Accessed 2020-01-23, webpage in Swedish.

[21] "Artec eva lite product page." `https://www.artec3d.com/portable-3d-scanners/artec-eva-lite`. Accessed 2020-01-23.

[22] "Artec space spider product page." `https://www.artec3d.com/portable-3d-scanners/artec-spider`. Accessed 2020-01-23.

A

Tables of identification metric

A.1 Added object

A.1.1 Comparing after PC with before PC

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.002	0.001	0.905	0.002	0.000
2	0.000	0.001	0.002	0.000	0.001
3	0.001	0.000	0.015	0.000	0.000
4	0.002	0.001	0.014	0.002	0.000
5	0.000	0.000	0.010	0.000	0.000
6	0.000	0.000	0.005	0.000	0.000
7	0.000	0.000	0.000	0.000	0.000

Table A.1: Metrics for all separated clusters, Mean deviation

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.001	0.001	0.826	0.002	0.000
2	0.000	0.000	0.004	0.000	0.000
3	0.000	0.000	0.000	0.000	0.000
4	0.000	0.000	0.001	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000
6	0.000	0.000	0.000	0.000	0.000
7	0.000	0.000	0.000	0.000	0.000

Table A.2: Metrics for all separated clusters, Mean + 0.5*standard deviation

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.001	0.001	0.702	0.002	0.000
2	0.000	0.000	0.000	0.000	0.000
3	0.000	0.000	0.000	0.000	0.000

Table A.3: Metrics for all separated clusters, Mean + 1*standard deviation

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.001	0.001	0.608	0.001	0.000

Table A.4: Metrics for all separated clusters, Mean + 1.5*standard deviation

A.2 Removed object

A.2.1 Comparing before PC with after PC

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.000	0.001	0.000	0.829	0.001
2	0.001	0.000	0.000	0.004	0.283
3	0.000	0.000	0.000	0.077	0.000
4	0.000	0.000	0.000	0.030	0.016
5	0.001	0.785	0.000	0.001	0.001
6	0.000	0.000	0.000	0.002	0.001
7	0.000	0.000	0.000	0.006	0.000
8	0.000	0.000	0.000	0.006	0.381
9	0.000	0.000	0.000	0.006	0.000
10	0.000	0.001	0.000	0.000	0.000

Table A.5: Metrics for all separated clusters, Mean deviation

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.802	0.001	0.000	0.000	0.001
2	0.000	0.000	0.000	0.211	0.001
3	0.000	0.001	0.000	0.007	0.000
4	0.000	0.000	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000

Table A.6: Metrics for all separated clusters, Mean + 0.5*standard deviation

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.777	0.001	0.000	0.000	0.001
2	0.000	0.000	0.000	0.002	0.000

Table A.7: Metrics for all separated clusters, Mean + 1*standard deviation

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.753	0.001	0.000	0.000	0.001

Table A.8: Metrics for all separated clusters, Mean + 1.5*standard deviation

A.3 Re-positioned object

A.3.1 Comparing after PC with before PC

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.844	0.003	0.000	0.001	0.002
2	0.052	0.049	0.000	0.059	0.071
3	0.091	0.008	0.000	0.010	0.017
4	0.002	0.001	0.000	0.799	0.001
5	0.001	0.000	0.000	0.000	0.000
6	0.006	0.615	0.000	0.001	0.000
7	0.006	0.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000

Table A.9: Metrics for all separated clusters, Mean deviation

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.803	0.003	0.000	0.001	0.002
2	0.003	0.000	0.000	0.001	0.000
3	0.000	0.000	0.000	0.000	0.000
4	0.001	0.000	0.000	0.000	0.000
5	0.000	0.001	0.000	0.001	0.000
6	0.000	0.000	0.000	0.002	0.000
7	0.000	0.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.001	0.000
9	0.000	0.000	0.000	0.000	0.000

Table A.10: Metrics for all separated clusters, Mean + 0.5*standard deviation

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.738	0.003	0.000	0.001	0.002

Table A.11: Metrics for all separated clusters, Mean + 1*standard deviation

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.679	0.003	0.000	0.001	0.002

Table A.12: Metrics for all separated clusters, Mean + 1.5*standard deviation

A.3.2 Comparing before PC with after PC

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.031	0.045	0.000	0.139	0.069
2	0.000	0.001	0.000	0.882	0.002
3	0.838	0.001	0.000	0.000	0.001
4	0.001	0.003	0.000	0.002	0.000
5	0.025	0.000	0.000	0.000	0.000
6	0.001	0.852	0.000	0.001	0.001
7	0.002	0.001	0.000	0.008	0.865

Table A.13: Metrics for all separated clusters, Mean deviation

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.801	0.001	0.000	0.000	0.001
2	0.000	0.000	0.000	0.047	0.000
3	0.000	0.000	0.000	0.000	0.000
4	0.000	0.000	0.000	0.006	0.000
5	0.000	0.000	0.000	0.020	0.002
6	0.000	0.000	0.000	0.009	0.000
7	0.000	0.000	0.000	0.000	0.000
8	0.000	0.002	0.000	0.020	0.000
9	0.000	0.000	0.000	0.021	0.000
10	0.000	0.000	0.000	0.001	0.000
11	0.000	0.000	0.000	0.000	0.000
12	0.000	0.000	0.000	0.000	0.000
13	0.000	0.000	0.000	0.000	0.000

Table A.14: Metrics for all separated clusters, Mean + 0.5*standard deviation

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.776	0.001	0.000	0.000	0.001
2	0.000	0.000	0.000	0.001	0.000

Table A.15: Metrics for all separated clusters, Mean + 1*standard deviation

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.751	0.001	0.000	0.000	0.001

Table A.16: Metrics for all separated clusters, Mean + 1.5*standard deviation

A.4 Swapped object

A.4.1 Comparing after PC with before PC

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.000	0.000	0.834	0.015	0.009

Table A.17: Metrics for all separated clusters, Mean deviation

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.000	0.000	0.775	0.010	0.006
2	0.000	0.000	0.048	0.000	0.001

Table A.18: Metrics for all separated clusters, Mean + 0.5*standard deviation

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.000	0.000	0.698	0.009	0.005

Table A.19: Metrics for all separated clusters, Mean + 1*standard deviation

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.000	0.000	0.545	0.007	0.004

Table A.20: Metrics for all separated clusters, Mean + 1.5*standard deviation

A.4.2 Comparing before PC with after PC

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.002	0.619	0.000	0.001	0.001
2	0.000	0.001	0.000	0.877	0.002
3	0.034	0.054	0.000	0.050	0.080
4	0.837	0.001	0.000	0.000	0.001
5	0.001	0.001	0.000	0.000	0.000
6	0.000	0.000	0.000	0.042	0.000
7	0.001	0.000	0.000	0.005	0.388
8	0.000	0.000	0.000	0.016	0.046
9	0.001	0.000	0.000	0.007	0.581
10	0.026	0.000	0.000	0.000	0.000
11	0.007	0.000	0.000	0.000	0.014
12	0.011	0.000	0.000	0.000	0.000
13	0.000	0.000	0.000	0.001	0.015
14	0.001	0.000	0.000	0.000	0.000

Table A.21: Metrics for all separated clusters, Mean deviation

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.002	0.655	0.000	0.001	0.001
2	0.794	0.001	0.000	0.000	0.001
3	0.000	0.004	0.000	0.061	0.001
4	0.000	0.002	0.000	0.020	0.000
5	0.000	0.000	0.000	0.054	0.000

Table A.22: Metrics for all separated clusters, Mean + 0.5*standard deviation

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.001	0.361	0.000	0.000	0.000
2	0.769	0.001	0.000	0.000	0.001
3	0.001	0.594	0.000	0.001	0.001
4	0.000	0.002	0.000	0.019	0.000
5	0.000	0.001	0.000	0.004	0.000

Table A.23: Metrics for all separated clusters, Mean + 1*standard deviation

Cluster id	Powerdrill	Robot vacuum	Manual vacuum	Fire Ext	Drone case
1	0.001	0.345	0.000	0.000	0.000
2	0.731	0.001	0.000	0.000	0.001
3	0.000	0.470	0.000	0.001	0.000
4	0.000	0.001	0.000	0.012	0.000
5	0.000	0.001	0.000	0.004	0.000

Table A.24: Metrics for all separated clusters, Mean + 1.5*standard deviation