TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

IÑIGO GARMENDIA AROZTEGUI

# A KNOWLEDGE-DRIVEN METHOD FOR 3D RECONSTRUCTION OF TECHNICAL INSTALLATIONS IN BUILDING REHABILITATIONS

Master of Science Thesis

Examiner: Professor Jose L. Martinez Lastra
Examiner and topic approved by the Dean of the Faculty of Engineering Sciences on 28th March 2018

# ABSTRACT

**GARMENDIA AROZTEGUI, IÑIGO**: A KNOWLEDGE-DRIVEN METHOD FOR 3D RECONSTRUCTION OF TECHNICAL INSTALLATIONS IN BUILDING REHABILITATIONS

TAMPERE UNIVERSITY OF TECHNOLOGY
Master of Science Thesis, 56 pages
June 2018

Keywords: object segmentation, visualization, point cloud segmentation, knowledge based systems

Rehabilitation of buildings often requires analysing the scene and measuring the dimension of different elements. Normally this task is done by visiting the real scene. In many cases, the planes are lost or unavailable, so the area must be measured again. Building a 3D model of the scene enables to work remotely. The scene is captured using a 3D laser scanner, which creates a point cloud. Applying a segmentation method to the point cloud, the features of the data can be extracted.

In this thesis a method for processing the point cloud data obtaining the information of each wall and pipe is developed. The point cloud is cleaned of unnecessary points and segmented applying different segmentation methods. Region growing algorithm and RANSAC algorithms are combined for an accurate detection of pipes and planes. Once the features of each element are detected, a knowledge-based model, in this case an ontological model, is created. This ontological model provides information for building the 3D model in a web application. Furthermore, it contains data about the suppliers and products, which after applying some reasoning, the purchase options for a pipe is shown.

The research work is about scanning the scene with a 3D laser scanner and building an application for creating the model out of the scanned data. The web application recreates the structure and pipes on a 3D model, making the intersections between pipes and providing information about the replacement's purchasing options.

# PREFACE

The current master thesis was done during my stay in at FAST laboratory in Tampere University of Technology as an exchange student. It is the final step of my university studies, the finishing of student life and the beginning of a new stage.

I would like to thank Jose L. Martinez Lastra and Borja Ramís for encouraging and giving me the opportunity to develop the present thesis at FAST laboratory. Thanks also to Wael Mohammed, my supervisor for helping to carry out the thesis. I would also like to thanks Luis Gonzalez for helping and advising me with some issues.

This would not be possible to my parents, thanks for giving me the opportunity to go abroad to have a new experience and supporting me everything I have done.

Finally I would like to thanks all my friends that I met in Finland. It was a pleasure sharing with you this amazing experience.

# CONTENTS

## LIST OF FIGURES

# LIST OF SYMBOLS AND ABBREVIATIONS

List of Symbols and abbreviations

| | |
|---|---|
| CAD | Computer Aided Design |
| PCL | Point Cloud Library |
| RANSAC | Random Sample Consesunsus |
| TUT | Tampere University of Technology |
| HTML | HyperText Markup Language |
| CMM | Coordinate Measurement Machines |
| CT | Computed Tomography |
| PCMM | Portable Coordinate Measuring Machine |
| TLS | Terrestrial Laser Scanners |
| LSHP | Line-Segment-Half-Planes |
| LMEDS | Least Median of Squares |
| MSAC | M-Estimator Sample Consensus |
| RRANSAC | Randomized RANSAC |
| RMSAC | Randomized MSAC |
| MLESAC | Maximum LikeLihood Estimation Sample Consensus |
| PROSAC | Progressive Sample Consensus |
| KBS | Knowledge Based System |
| LHS | Left-Hand Side |
| NN | Neural Networks |
| ANN | Artificial Neural Networks |
| CBR | Case Based Reasoning |

.

# 1. INTRODUCTION

## 1.1 Problem definition and motivation

Architecture has been evolving over the years, such as the way of representing it. Nowadays, we cannot imagine architecture without the use of computers. Computers and digitalization help the architects and engineers to project all the ideas that they have in mind much easier than using hand drawings or verbal explanations. Specially, 3D models are able to give a precise approximation of real objects, such as buildings.

There are many reasons for making changes in a room, building or any facility. They can be deteriorating over time or we just want to make some changes. When reforming a room, the measures of the elements like pipes or walls are needed. These measures can be taken from previous planes that have been used to build it, but sometimes, the planes defining the reconstruction area are not available, forcing to take measures manually in the real environment.

Nowadays, there is need to take advantage of technology that can help us to solve problems in an easier and more efficient way than using conventional methods. In the problem we are facing, 3D scanners can be very useful, providing point cloud data of the scanned room. Point clouds are a set of points in a three-dimensional space, which can be used for many purposes as creating 3D models, taking measures, animation, rendering, metrology and mass customization applications.

## 1.2 Objectives

The objective of this thesis is to create an application that creates a 3D model of technical installations for building rehabilitation. This means creating a method that processes and converts the scanned data. The application should be able to detect the walls and pipes from the point cloud data and make a visualization of them, reducing the file size comparing to the scanned one. The application has to take the measures of the walls and the pipes and convert them to a model, increasing the amount of information given through the 3D model.

In the final application, the user is able to visualize the 3D model of the scanned scene and explore the available replacements for the pipes in the market.

## 1.3 Hypothesis

The problem that is presented here, needs actual technology like 3D scanners [1]. This type of scanners collect 3D coordinates from the surface of the scanned object, in this case a room. 3D scanners can be used for many purposes like reverse engineering, industrial design and manufacturing, healthcare, art and design. Here we have used the 3D scanner for digitalizing interiors.

The collected data can be processed in many different ways, depending on the finality of the problem that want to be solved. Point Clouds can be useful for measuring distances, detect objects in the space or for many other utilities. In this case, the point cloud is going to be used for detecting pipes and walls by using the coordinates of each point. The object detection can be also possible by observing different colours of the points of each object. But assuming that not all the scanners detect the texture of the scene, the processing of the scene is going to be without using the textures.

For object detection and segmentation there are many useful algorithms, like region growing, model fitting, machine learning… Model fitting segmentation is valid for this problem as the objects like pipes and walls can be approximated with geometrical shapes like cylinders and planes. By using RANSAC algorithm it is expected to identify the planes and cylinders of the scene.

Finally, the 3D model is going to be built with the information that have been recollected from the segmentation. Using Three js library a lightweight model with smooth navigation is expected.

## 1.4 Challenges and limitations

The principal challenge of this work is creating an exact 3D model of the real scene that includes all the features like pipes or columns. There are many factors that affect in the process of getting a good result.

It is important to take into account the limitation of the 3D scanner. Although the results of the actual scanners are quite good, they are not perfect. Aspects like light conditions, type of material or laser positioning can influence, creating a point cloud with noise or holes, affecting the processing of data. The zones which are in shadow for example, are not well recognized.

Another significant challenge would be to manage point clouds with a lot of points. The executing time of the application would increase exponentially and maybe getting bad results. In this work, the trials have been done with a part of the scanned room, which has been cut manually using a point cloud specialized software. To reduce the execution time, the density of points is reduced, as all the points are not needed to identify the geometrical

shapes.

Ideally, all the pipes should be recognized, without mattering the direction or the radius of them. Practically, vertical and horizontal pipes, which have not a small radius, are recognized. Unfortunately, some false cylinders are recognized. The cause of these recognitions are the edges of the walls that have a little curve. The intersections of the pipes with other pipes and the pipes with the walls are not recognized either.

For the reconstruction of interiors, it is very important to know where the columns are, which cannot be modified or pierced. This application is able to recognize most of the planes and those that meet some conditions are considered as part of a column.

Finally, the 3D model is just created to open it in a web browser. This model can be developed for more platforms like virtual reality or CAD (Computer Aided Design) format.

# 2. THEORETICAL BACKGROUND

## 2.1 3D Digitizing

Digitization nowadays is in our everyday life, most of the things are in a digital version. It has a great impact in the economy and lots of companies are emerging in this area. It is considered as public interest and a modern management method [2].

Digitization is the process of introducing digital technology in our society, the production networks and the applications based on it [3]. We can divide this process of digitization into two phases. The first phase consists of digitalizing the sectors which the production, consumption and communication are based on transactions, information and the use of data. This includes the press, financial services or music production [3].

In the second phase, the objective is to digitize physical objects of all types. Currently, this is known as "Internet of things", smart and intelligent devices that are connected to embedded software as well as interactive applications [3]. Many fields are being digitized like housing, medical stuff, transportation systems or industrial production.

In object or ambient digitizing, which is this thesis interested in, is about creating 2D drawings or 3D models. 3D modelling is a useful tool that reduces the physical manipulation of objects or spaces. It is a backup of the real object that can be used in case of destruction or for making upgrades [4]. It is easy to make modifications virtually before making changes in the real environment and also, it is possible to compare different object or spaces which are located in different situations [4].

When an installation must be modified, the planes of the area may be necessary, but sometimes these are not available. A 3D model of the actual environment can be very useful, representing all the objects or installations. There are many different tools and techniques for generating three-dimensional models of small objects, buildings and cities. Digitizing this space make this reconstruction, preservation or rehabilitation works easier facilitating data [5].

### 2.1.1 Digitizing methods

**Time of flight laser scanning.** For calculating the distance between the object and the scanner, a laser pulse is sent and the time between the signal transmission and reception is measured, after reflecting on the object. The accuracy of this technique is nearly similar for the whole object and is around millimetres. The measurements accuracy may differ depending on the beam angle [6].

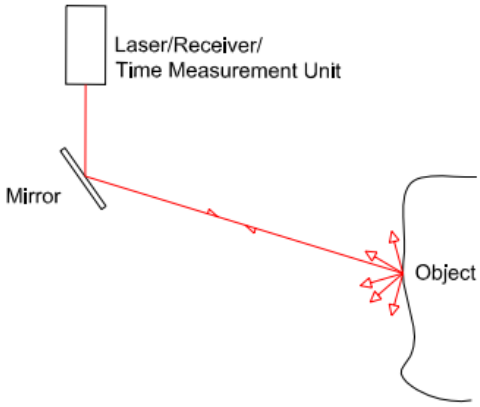


***Figure 1. Time of flight principle [6]***

**Phase comparison laser scanning.** This is a similar method to the previous one, but in this case, the distance is measured by sending a harmonic wave and calculating the phase difference of the received wave. The processing of this signal is more complicated and needs and a complex algorithm, but the accuracy of the obtained result is better. As the receiving signal has to be clear, these scanners have a reduced range and they return more wrong points than the other [6].

**Triangulation laser scanning.** The laser beam impacts the object and a camera is used for detecting the location of the laser point. Depending on the distance that the laser hits a surface, the laser points appear in different places of the camera sensor.

This technique is called triangulation because the laser, the object and the camera form a triangle. The length defined by the laser and the camera is known, as well as the angle of the vertex of the transmitter. The angle of the camera can be determined by observing the laser point. With these three values, the measures of triangle can be defined.

These instruments are very accurate (thousands of millimetres), more than ranging scanners. Although these scanners are intended for scanning small objects [6].

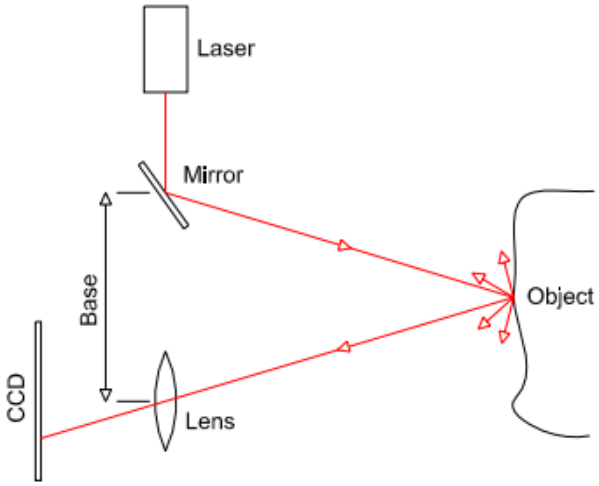


*Figure 2. Triangulation principle [6]*

**Shape from structured light.** The device projects specific light patterns sequentially on the surface of the objects while a camera takes photos. Then the images are processed to obtain the geometry giving very accurate results, with textures on it. They are easy to configure, portable and can be used in real applications, but they are still working on developing the technology to get more resolution [7]–[9].

**Shape from stereo.** It takes images of the object from two different views at least and finds the common points in the different images to calculates the 3D geometry by triangulation. The advantages are the low cost the devices, portability and that is able to capture textures. The disadvantage of this method is the low resolution [7], [8], [10].

**Shape from motion.** This is similar to the previous one, but instead of using two cameras, a video camera is used that records the object from different perspectives. In this method the object can't have movable parts, it must be still [7]. Image-based methods are popular due to the low cost, but they are mainly used for visualization. They capture geometry and texture, they are portable, and they have low accuracy and resolution [9].

**Shape from silhouette.** It consists of taking many photos from different views and deducting the geometry from the silhouette. Many developments have been made in this technology, now it is able to capture the texture of the objects. This procedure is cheap and portable. On the contrary, the resolution is low [7].

**Shape from shading.** Many images are taking with different light conditions to capture the geometry of the object. The position of the light source must vary, causing variation in the shading [7]. It can be combined with other techniques to obtain better results. It is a low-cost technique that can capture texture and geometry, although in shaded areas the texture may not be well detected, and the accuracy is low [8].

**Shape from photometry.** This is a variation of the shading method. There are taking many photos with different light conditions but with a certain light angle. It also needs

objects as a reference in the scene for a good calibration. This technique is considered as portable but with a constraint, it must be in a laboratory environment. The accuracy is better compared to the shading one [7], [8].

**Shape from focus.** The system takes many images with different focus adjustment from the same perspective. Knowing the position of the focus plane, it is possible to map the pixels in the three-dimensional space [7]. Many lenses are needed for focus adjustment, usually used with professional microscopes and very small objects [8]. The method is expensive but easy to apply, getting reliable results.

**Contact digitizing methods.** The machine traverses the object touching it and recording the coordinates of the points, defining the geometry. These machines are called Coordinate Measurement Machines (CMM) and they are mainly used for metrology. They are very accurate but very slow also. Complex geometries may be not possible to capture as they have to be in contact and some parts may not be reachable [7], [10].

**Topographic methods.** It uses a 3D orthogonal coordinate system using complex devices, mainly Geodesic Stations. It is able to capture the distance and angles in big areas. The points are first obtained in the orthogonal coordinate system and they are transformed to reference coordinate system. They are very accurate, even in difficult conditions like complex shape or complicated access[7].

### 2.1.2 Point Cloud Data

Most of the scanner create point cloud data. A point cloud is a set of vertices in a three dimensional coordinate system. These vertices normally are identified as X, Y and Z coordinates and they represent the external shape of an object.

In this thesis, a laser scanner is used to digitize an interior of a building. There are many types of laser scanners, each of them for a certain situation. The products we can find in the market are CT (Computed Tomography) scanners, turntable scanners, PCMM (Portable Coordinate Measuring Machine) scanners, handheld scanners, terrestrial scanners and mobile scanners [11].

Terrestrial Laser Scanners (TLS) are tools that can record the coordinates of an indoor 3D scene efficiently. It has many advantages like a high rate of data acquisition, high accuracy and very good density of points. TLS can take 500.000 points per second with ±6 mm precision. They use non-contact laser pulses for capturing the shape of the objects. They acquire the distance of different points and store them in a database which is called point cloud [12]. In addition to geometry, some scanners are able to capture texture integrating a camera to the laser, like FARO scanners [13].

A point cloud is a data structure to represent multi-dimensional points and normally used in 3D spaces. A three-dimensional cloud contains a list of coordinates in X, Y and Z of

the points from a certain surface. It converts a 3D cloud to 4D cloud when colour is added.

***Figure 3. Point cloud with colour***

Point clouds have many applications like quality control in metrology [14], 3D modelling and many other in visualization, animation and texturing. Although point cloud data can be analysed and rendered directly, they are quite difficult to manage in three-dimensional software [14]. For that reason, they are converted to polygonal or triangular mesh models, NURBS surface models or CAD models [15].

The 3D modeling based in the point clouds can be done manually using several programs like Mehslab for creating meshes or Geomagic Design X for creating 3D CAD models. This work is about getting the 3D model automatically, creating an application in which the scanned data is the input and the output is a 3D model, which can be visualized using a web browser.

We can divide into two categories the conversion from point clouds to meshes, which are volumetric approaches and surface-based methods [16]. In the first category, the point cloud is converted to a voxel volume to reconstruct the implicit shape using marching cubes algorithm. In the second category, they create a triangle network from the vertexes of the point cloud using Delaunay triangulation, alpha shapes and ball pivoting [16].

In metrology, measured point clouds are compared to the original design that it is in a CAD file or even with another point cloud. Representing them in a visual mode, the user can see the differences in an easy way [14].

Point clouds are also used in medical tasks, normally for making a visualization of a volume. Also, maps with volumes and textures are created by scanning the real surface [17].

Point cloud data is an expanding and progressing technology that will allow many new applications (to) emerge. Therefore, it is probably that in the future all the robots will be able to see in three dimensions.

## 2.2 Segmentation methods

Nowadays, 3D models and point cloud data are commonly used in many applications. The procedure of processing the point cloud depends on the field of application and the desired results. Some of these processes can be filtering, down sampling, registration or segmentation. The objective of segmentation is to group points that have same features, in a homogeneous region [18], [19]. In this chapter, different segmentation methods are analysed, presenting the advantages and disadvantages of each of them.

A segmentation method should have three properties. First, the algorithm must be able to distinguish the different features of certain objects or geometries, when the number of features increase, automatically has to reject them. Second, it has to be able of deducing

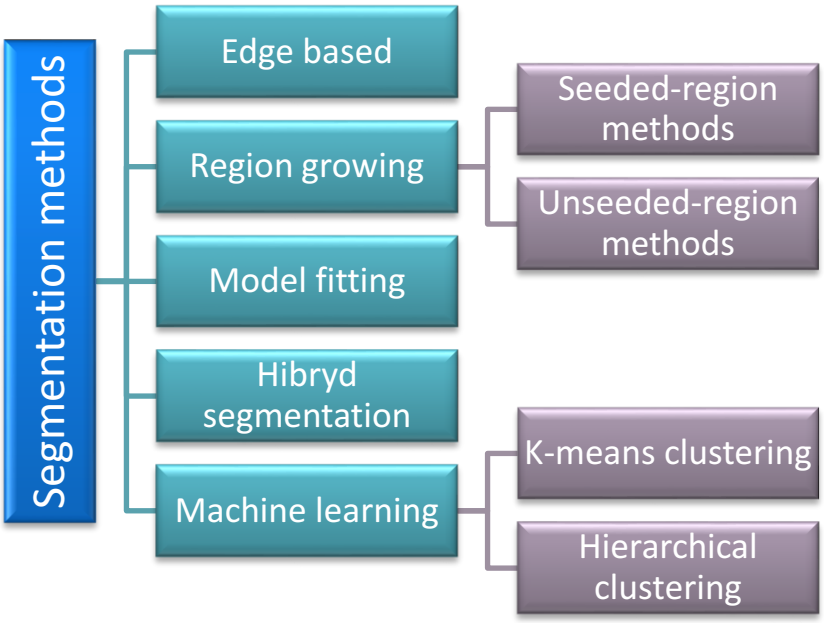


***Figure 4. Representation of the segmentation methods***

the label of the points which are in a very few populated segments by observing their neighbours. Third, the algorithm has to adapt to the 3D scanner used, because there can be many differences in the point cloud data, even in the same environment [18].

### 2.2.1 Edge based segmentation

Edge based segmentation defines regions of the point clouds that are limited by the edges. It consists on finding points that have big change of a certain feature [18]. The algorithm has two steps: first, the edges are detected to outline the borders of the different regions. Then, the points inside the boundaries of the region are grouped [19].

There are many procedures in the edge segmentation. One method is computing the gradient, fitting lines to a group of points and identifying the variations in the normal vectors on the surface [18]. Another technique is to use statistical methods for detecting the sharp

features, segmenting the neighbourhood of points in regions that belong to the same surface [20].

Lin et al. [21] suggested a technique to identify accurately the edges of two planes that are intersecting from a large raw scan points. The 3D line-support region which is, in other words, a point located closed to a straight linear structure, is obtained concurrently. A geometric limitation for a segment of a line is provided by the 3D line support region that is suited by our Line-Segment-Half-Planes (LSHP), allowing a more defined and accurate line segment.

There are many other techniques in edge segmentation, most of them use surface properties like normals, gradients, principal curvatures or higher order derivatives [19].

Edge based segmentations are fast, but they have lack of accuracy because they are sensitive with noisy and uneven point clouds, which is common in point cloud data [18].

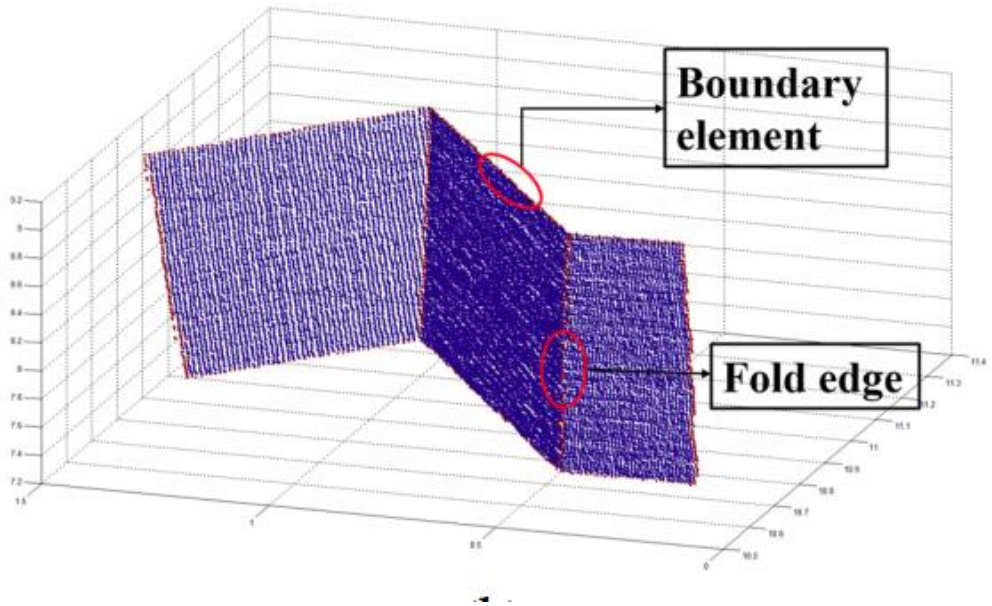


*Figure 5. Edge detection sample [22]*

### 2.2.2 Region growing segmentation

Region growing segmentation use local features extracted from a neighbourhood around each point to add nearby points with similar properties and extract the region from a point cloud. Region based methods are more accurate than edge-based methods and have better behaviour with noisy data, but they have problems under or over segmenting and detecting edges accurately. There are two main categories in region-based segmentation:

seeded-region methods (bottom up) and unseeded-region methods (bottom down) [18], [19].

**Seeded region method.** It starts choosing a number of seed points, then from each seed point, the region will grow if the nearby points satisfy the regions features. The algorithm introduced by Besl [23] identifies the seeds points based on the curvature of each point and grow them depending the defined feature as proximity of points or planarity of surfaces. This method is very sensitive with noise and takes too much time.

There are many variations in seeded region segmentation algorithms. Rabbani [24], [25] proposed a method using normals and their residuals, according to the parameters defined by the user to group points associated to smooth surfaces. This algorithm is intended to avoid over-segmentation resulting an under segmented point cloud.

Seeded points depend considerably on selected seed points. Selecting the seed points imprecisely can affect the result and produce a bad segmentation. Choosing seed points along with controlling the growing process is time consuming. The segmented results can change easily according to the characteristics of the chosen threshold. An added complication is to decide when to add point to a region as the decision is made locally, being affected by noise [18].

Unseeded region method. On the contrary to the seeded methods, all the points are assigned to the same region. Then this region is divided in smaller regions. If the chosen figure for fitting is bigger than a threshold, then the subdivision carries on [18]. Chen used this method for grouping planar regions to reconstruct an architectural building [23]. It is based on confidence rate of the local area for planar recognition. The problem of this technique is over-segmentation.

The main problem of unseeded-region methods is to decide how and where to subdivide. It is necessary to know some features like number of regions or model objects that most of the cases are unknown scenes are not available.

There are several region-growing algorithms available in Point Cloud Library (http://pointclouds.org). In the Fig. 6, one part of the Fastory lab is segmented by region growing algorithm implemented in the the pcl::RegionGrowing class. The objective of this algorithm is to group points of the same smooth surfaces in different regions.

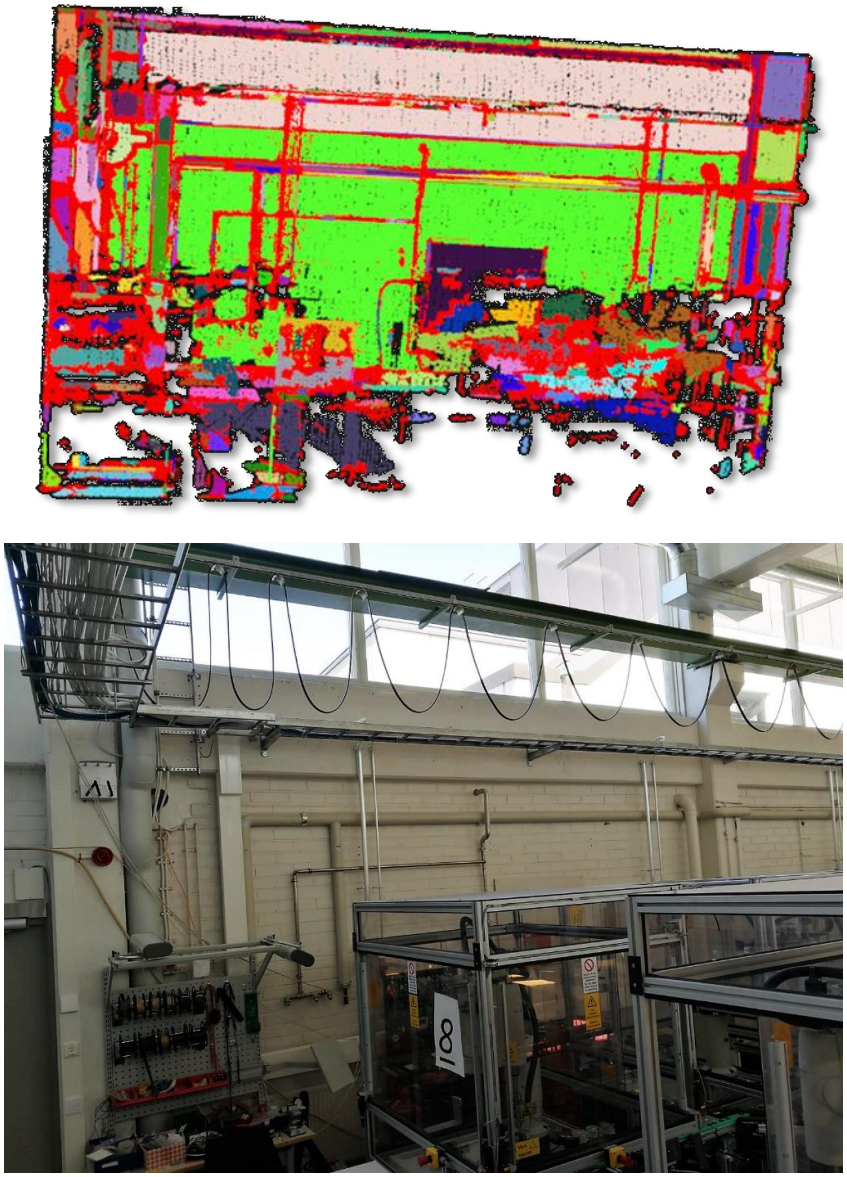

***Figure 6. Point cloud segmented with region growing algorithm from the Point Cloud Library***

This is the process of the Region Growing class algorithm of the PCL library:

- The picked point is added to the set called seeds.
- For every seed point algorithm finds neighbouring points.
  - Every neighbour is tested for the angle between its normal and normal of the current seed point. If the angle is less than threshold value then current point is added to the current region.
  - After that, every neighbour is tested for the curvature value. If the curvature is less than threshold value then this point is added to the seeds.
  - Current seed is removed from the seeds.

If the seeds set becomes empty this means that the algorithm has grown the region and the process is repeated from the beginning. You can find the pseudocode for the said

algorithm below [26].

```
Inputs:
      Point cloud = {P}
      Point normals = {N}
      Points curvatures = {c}
      Neighbour finding function Ω(.)
      Curvature threshold c_th
      Angle threshold θ_th
Algorithm:
While {A} is not empty do
      Current region {R_c} ← Ø
      Current seeds {S_c} ← Ø
      Point with minimum curvature in {A} → P_min
      {S_c} ← {S_c} ∪ P_min
      {R_c} ← {R_c} ∪ P_min
      {A} ← {A} \ P_min
      for i = 0 to size ( {S_c} ) do
            Find nearest neighbours of current seed point {B_c} ← Ω(S_c{i})
            for j = 0 to size ( {B_c} ) do
                  Current neighbour point P_j ← B_c{j}
                  If {A} con-
                  tains P_j and cos^-1(|(N{S_c{i}}, N{S_c{j}})|) < θ_th then
                        {R_c} ← {R_c} ∪ P_j
                        {A} ← {A} \ P_j
                        If c{P_j} < c_th then
                              {S_c} ← {S_c} ∪ P_j
                        end if
                  end if
            end for
      end for
      Add current region to global segment list {R} ← {R} ∪ {R_c}
end while
Return {R}
```

***Code 1. Region growing algorithm [26]***

### 2.2.3 Segmentation by model fitting

This segmentation consists on decomposing a surface or object in geometrical shapes that can be represented mathematically like planes, cylinders, spheres, cones or tori. Primitives shapes are fitted on the point cloud and the points that adjust to the selected shape are considered as a segment [19]. There are two main algorithms based on model fitting: Hough Transform (HT) [27] and Random Sample Consensus (RANSAC) [28]. It has been proved that both methods succeed in detecting shapes in 2D and 3D, they are reliable

with a high number of outliers, but they have big memory consumption and low efficiency [29].

Hough Transform maps every point of the data to a collector, for a particular geometric model. For each point, all the possible figures that the point can belong to, are computed. The vectors that have the highest number of votes are selected, extracting the shapes. This technique requires a huge amount of memory, so it is principally used in 2D applications and not in 3D. Although there is an exception, for searching planes in 3D, because they have a small number of parameters [29].

RANSAC algorithm is an iterative method that estimates the parameters of a mathematical model from a set of data that have many outliers. The valid points that fit in the analysed model are called inliers and the non-valid one's outliers. It tries to approximate the parameters of a certain mathematical model starting from a minimal set of points and the adding points iteratively if they are inside of a threshold defined by the user. This process stops after a number of iterations are completed [30].

Moreover, RANSAC originates a reasonable result only with a determined probability [30]. It is very used in reverse engineering, segmenting the analysed object in cylinders, planes or other geometrical primitives [31]. These primitive shapes are very useful to define an object's geometry in a CAD file [30].

The work by Li et al. [32] introduced an algorithm for globally consolidating the results obtained by the RANSAC method. In this methodology, RANSAC is used for local fitting of geometric primitive shapes. It makes the correction of the geometrical shapes obtained in the local RANSAC detection and put them in an exact global alignment. This method can be used for optimizing the parameters of model fitting segmentation.

Papazov and Burschka [33] proposed a modified RANSAC algorithm for 3D object recognition for noisy, messy and unsegmented data. The method uses a robust geometric descriptor and an efficiency similar to RANSAC strategy. It is assumed that an object consists of points with their corresponding surface normal.

There are many variations to RANSAC that are available in PCL library:

- LMEDS (Least Median of Squares)
- MSAC (M-Estimator Sample Consensus) [34] [35]
- RRANSAC (Randomized RANSAC)
- RMSAC (Randomized MSAC
- MLESAC (Maximum LikeLihood Estimation Sample Consensus) [34]
- PROSAC (Progressive Sample Consensus) [35]

Model-based methods are robust and fats with outliers. It has been tested several times obtaining good results for the detection of 3D geometric primitive shapes like cylinders,

cones, spheres, planes, torus or cubes. The main problem of this technique is the inaccuracy with different point clouds. With the same configuration of the algorithm parameters, the result quality may be different. In architecture, it is difficult to fit details with primitive shapes. Besides using the position of the points, details can be distinguished using their colour [36].

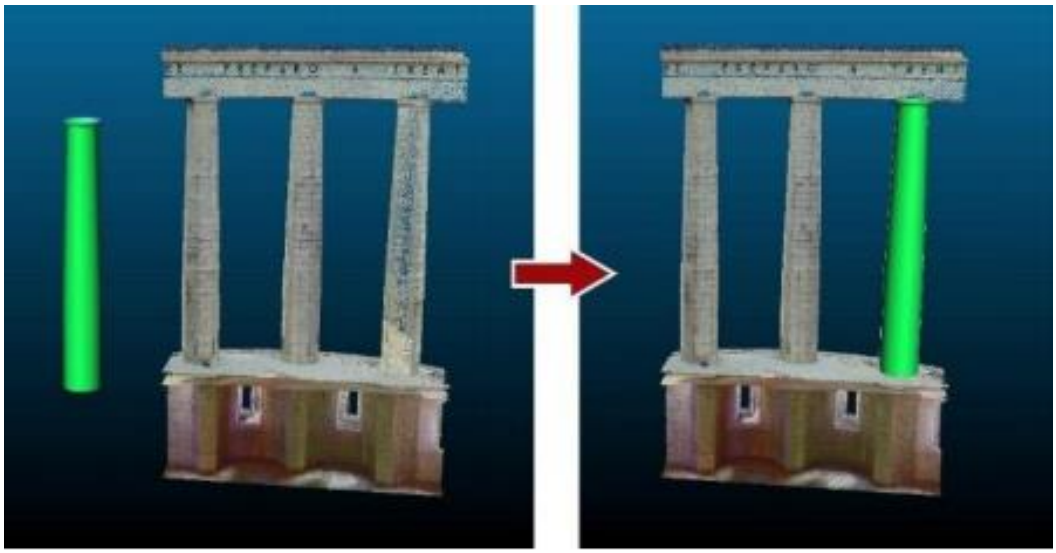

***Figure 7. Segmentation of 3D point cloud by geometric primitive [19]***

This is how the RANSAC algorithm works [29]:

```
Min P ← Minimum points to estimate the parameter s of the mathematical model
I T ← Calculate the minimum number of iterations. Equation 1
Cur I T ← 0
Best SolutionSet ← ∅
CandidatesT oSet ← ∅
P ← Points sample
while Cur I T < I T do
      Consensus Set ← "Min P" random points of P and estimate the model parameter
      s
      CandidatesT oSet ← P – Consensus Set
      while size(CandidatesT oSet) > 0 do
            r Point ← random point from CandidatesT oSet
            if |distance(r Point, model parameter s)| ≤ T H R E S H O L D then
                  Consensus Set = Consensus Set + r Point
            end if
            CandidatesT oSet = CandidatesT oSet – r Point
      end while
      if size(Consensus Set) > size(Best SolutionSet) then
            Best SolutionSet ← Consensus Set
      end if
      Cur I T ← Cur I T + 1
end while
Adjust parameter s to Best SolutionSet if possible
R E T U R N parameters
```

***Code 2. RANSAC algorithm [29]***

### 2.2.4 Hybrid segmentation technique

It consists of combining more than one technique with the objective to have better results than just using one of them. Sometimes one technique is not enough for getting the desired results.

In 2016 Hamid-Lakzaeian and Laefer [37] proposed a new method for segmenting automatically laser scanning data for decorative urban buildings. The method combined octree indexing and RANSAC algorithms, which were used before separately but not in conjunction. This technique resulted very efficient and scalable regardless of the facades grade of ornamentation or non-recti-linearity.

Lin Li [38] combined NDT (Normal Distribution Transformation) and RANSAC algorithms for plane detection in an application for automatically reconstruct indoor environments from unorganized point clouds obtained by laser scanners. The aim of this method was to avoid the false plane recognition that frequently happens using RANSAC. It resulted more reliable and accurate than the original RANSAC, executing even faster.

### 2.2.5 Machine learning segmentation

**Segmentation based on K-means clustering.** K-means is a method to group or classify, with the objective of partitioning a set of 3D points in K groups where each observation belongs to the group whose average value is the closest [19]. It was first presented by James McQueen in 1967 [39]. After that, it has been used many times. Savkare [40] segmented blood cells from microscopic images using K-means clustering. Ahammed Muneer and Paul Joseph [41] also used this method for medical purposes as detecting brain tumours in Magnetic Resonance Imaging (MRI).

**Segmentation based on hierarchical clustering.** These techniques calculates representative characteristics for every point of the cloud. The geometrical and radiometric features are: point position, residuals of best fitting surface, locally approximated surface normals, point reflectance… Normally, a hierarchical decomposition is made, splitting data and creating subdivisions iteratively until each subset consists of a one object [42].

## 2.3 Knowledge-based systems

### 2.3.1 Definition

A knowledge-based system (KBS) is a computer program, which generates and utilizes knowledge from different sources, data and information. These systems help to solve problems, especially complex ones, by using Artificial Intelligence. They support humans when facing problems, making decisions and taking actions. Knowledge-based systems are also known as Expertise Systems [43].

KBS is a system to represent knowledge explicitly with tools like ontologies and rules rather than implicitly through code as conventional programming. A knowledge base system has three basic components: a knowledge base, an inference engine and a user interface. The knowledge base contains the facts about the reality. The inference engine represents logical affirmations and conditions expressed by IF-THEN statements [41], [44].

In the 1970s, AI researchers developed a knowledge-based system that was able to diagnose and recommend treatments for blood infections, the program was called MYCIN [46]. This required the knowledge of an expert physician and the program reasoned based on conditions and facts.

### 2.3.2 General features of knowledge-based systems

The major features of a KBS are:

- It includes the knowledge of an expert, containing information, structures and rules.
- It is programmed to solve problems as humans, making conclusions based on facts and adopt the solution to the problem.
- It can deal with incomplete information, guessing the answer with the information available or asking for more info.
- It has a user interface that shows the containing information or request information if needed.
- Its development can be economical and effective, reusing different structures, rules and problem-solving methods in other applications [43].

### 2.3.3 Components of a knowledge-based systems

The main components of a KBS are:

- A knowledge base that contains information, structures and rules of a specific area. It can be used as a repository.
- Working memory is responsible for temporarily holding information available for processing [45].
- An inference engine or reasoning engine applies logical rules to the knowledge base and deduces new knowledge.

Nowadays, most systems also have these two elements:

- An interface to connect with users or other computer systems.
- An editor that allows changing knowledge base to domain experts.

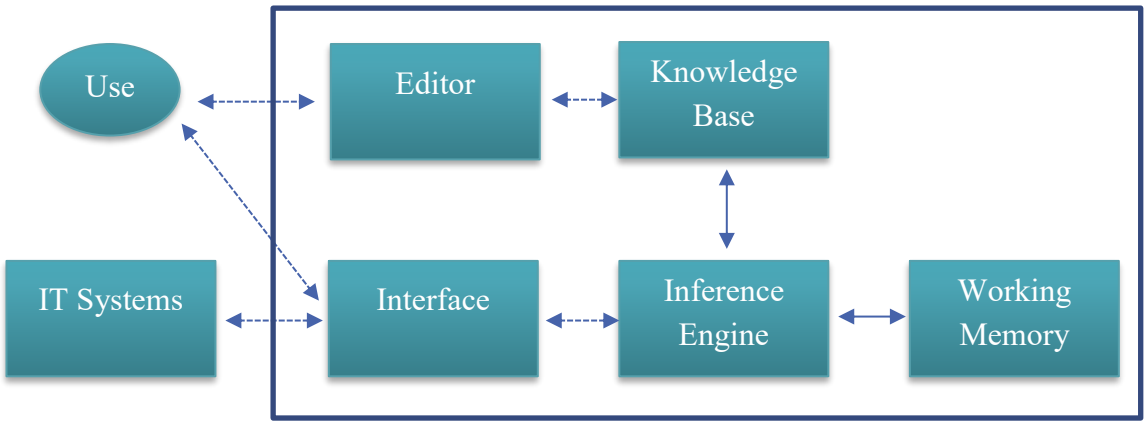


*Figure 8. The architecture of a basic Knowledge-Based System [43]*

### 2.3.4 Chaining

Rules can be matched in two ways:

- The if-part is the Left-Hand Side (LHS), which is also called the antecedent. It consists of one or more of condition elements. The representation of the conditions may be categorized for simple problems, integer/real intervals or combination of these for more complex problems [41].
- The then part -which is called the Right-Hand Side (RHS), consequent or action consists of a number of actions [41].

There are two types of reasoning based on the previous methods:

- In forward chaining, the data of the working memory is matched to the left-hand side of the rules, and this process is repeated until a conclusion is made.
- Backward chaining operates the other way around, it starts with a conclusion and sees if the left-hand sides are present in the working memory. In this way, some candidate solutions can be proposed and discard the ones that do not fit the facts [43].

Forward chaining method is suitable for solving most of the problems when the details of the answer are unknown, for example, when designing a product or creating a plan. Backward chaining normally is better, when choosing from a list of possible solutions like classifying something as a thing or other.

### 2.3.5 Categories of KBS

There are five main types in KBS [46].

1. Expert systems
2. Neural networks (NNs)
3. CASE-based reasoning
4. Genetic algorithms

5. Intelligent agents
6. Data mining

**Expert Systems**

The expert system is the pioneer of KBS and the most popular [47]. It contains the knowledge of one or more experts, being capable of offering intelligent advice or taking a decision about a processing function. A desirable feature would be that the system would justify its own mode of reasoning. A way of achieving this characteristic is a code based on rules [46].

We can deduct from the definition that the system uses knowledge and a way of storing this knowledge. It must have an inference engine for reasoning and get a conclusion. Finally, an expert system should be able to act like a human expert, taking decisions in a certain field [46].

These systems are useful in the following situations [47]:

- When an expert is not available.
- To store the expertise for the future or to copy the expertise.
- When intelligent assistance is needed for problem-solving or decision making.
- When more than one expert's knowledge has to be gathered in one platform.

**Neural networks**

Artificial neural networks (ANNs) are software tools designed to estimate relationship in data, it means relating or mapping raw data to its features. NN can be programmed to classify, estimate, simulate, and predict the process of generating data.

An ANN is a hardware implementation or software that attempts to simulate the information processing capabilities of its biological equivalent. The principal properties of an ANN are quite different from a conventional computer, these are the features:

- adaptive learning
- self-organisation
- error tolerance
- real-time operation
- parallel information processing.

NNs has many advantages like the ability to adapt to new kind of problems or their robustness. They are very useful solving problems that conventional methods cannot. They are more used with to dealing with less structured problems.

On the contrary, they do not have a good performance on solving complex problems for humans, as arithmetic problems or large volume data processing. Although, there are other methods for these tasks that can complement ANN. They have also difficulties when expressing the explanations for the decision that were taken. For good performance, they

need a high amount of data, which part of the data is used for training and another part for ensuring accuracy.

**Case-based reasoning**

Case-bases reasoning (CBR) is an automated reasoning and machine learning technique. It aims to solve a new problem using a database of old solved problems, noticing a similarity with one or several of them and adopting a solution for the actual problem. Therefore, it depends on past experience. In many aspects, case-based reasoning systems are different to other AI techniques. It is able to use concrete and previously experienced problems instead of using general domain knowledge as in expert systems. An important feature is that when a problem is solved, it can be added to the knowledge, learning and increasing experience to solve future problems [48].

CBR is recommended to use when expert's knowledge is too extensive or when the theory of the domain is very poor. They can work when the model used for solutions are not clear.

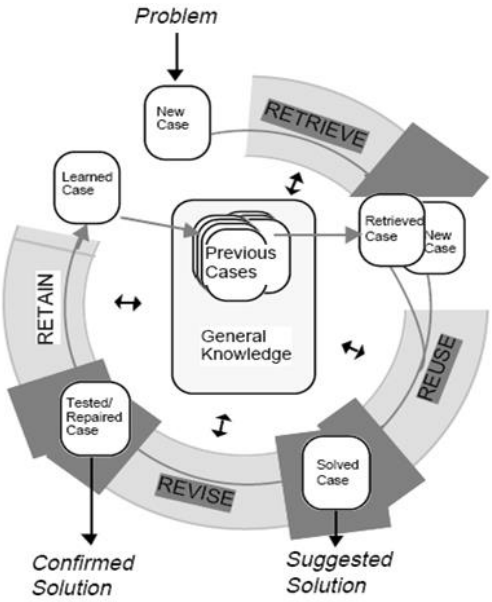


***Figure 9. CBR Cycle [49]***

**Genetic Algorithms**

Genetic algorithms (AGS) are adaptive methods that can be used to solve search and optimization problems. They are based on the genetic process of living organisms. Throughout the generations, the populations evolve in the nature in accordance with the principles of the natural selection and the survival of the strongest. Genetic algorithms are able to create solutions to problems in the real world. The evolution of these solutions towards optimal values of the problem depends to a good extent on an adequate codification.

Genetic algorithms are used to solve many large problems including [46]: scheduling,

transportation and layout and circuit design.

**Intelligent agents**

Intelligent agents are entities like robots or computer programs that realize useful functions or have unique characteristics. For being recognized as an intelligent agent, the entity has to be able to [46]:

- perceive the environment
- understand their purpose
- interact with their environment
- react to changes in the environment
- make decisions and learn from their experiences

**Data mining**

Data mining is a set of methods and technologies that permit to explore big databases in an automatic or semi-automatic way. The general objective of the data mining process is to extract information from a data set and transform it into an understandable structure for later use.

It uses practical statistics and in some cases, search algorithms close to Artificial Intelligence and neural networks.

### 2.3.6 Knowledge representation

Knowledge representation and reasoning is an Artificial Intelligence area with the objective of representing knowledge in a way that eases to inference from that knowledge. Generally, a kind of logic is used to provide a formal semantics of how the functions of reasoning are applied to the symbols of the domain, in addition, to provide the operators like quantifiers, modal operators, etc. This, with a theory of interpretation, gives meaning to the sentences in logic.

When designing a representation of knowledge, we have to make choices across a number of design areas. The most important decision to make is the expressiveness of the representation of knowledge. The more expressive it is, saying something is easier and more compact. However, the more expressive a language is, the more difficult it is to derive inferences automatically from it. There are five different roles that define knowledge representation [50].

The first role consists on acting as replacement of real objects in an intelligent matter. In order to interact with real world, internal representation becomes the substitution for making the reasoning. Real world tasks are replaced with operations on representations. All the knowledge representations are approximations of real models, which can be similar to the reality but not perfect, leading to inaccurate reasoning [51].

The second role of a knowledge representation is making a set of ontological commitments about the real world. Expressing the relevant information and ignoring the rest [51].

The third role of a representation is as a fragmentary theory of intelligent reasoning. Each representation shows how humans infer in an intelligent way. Besides, for each particular representation the system can deduce a set of conclusions, this set use to be very big and without any limitation [51].

The fourth role of a knowledge representation is as a medium for efficient computation. To this end, every representation provides some orientation for organizing knowledge with the aim to be most efficient for computation.

The fifth role is a medium for human expression. Humans recover, organize and store knowledge in the knowledge base of an intelligent system. For that, humans need to interact with computers to create and communicate knowledge [51].

Several languages can express representation of conceptual models. These languages have many different characteristics in terms of computational complexity, facility of use and expressiveness. They can divided into these types: vocabularies defined using natural language; object-based knowledge representation languages such as frames, rule-based representation and languages based on predicates expressed in logic [51].

Natural language can be used to represent knowledge in AI systems. The advantage of using this representation method is that it is extremely expressive and it is widely used as knowledge representation choice. On the contrary, the syntax and the semantics are complex and not fully understood. In addition, there is little uniformity in the structure of the sentences and normally ambiguous[52], [53].

A frame-based system provides greater structure than the natural language. They are a knowledge structure that describes a specific object and contains several slots, which have facts, characteristics or objects specifications. Attributes can be accessed rapidly and without being computed constantly. The properties of the relations are easy to describe [54].

An alternative to frames is logic, formal logic is becoming popular as knowledge representation language because it is practical. There are many types of formal logic and they are suitable for machine implementation. Every logic has a clear syntax that determines how the sentences are built, a semantic that determines the meaning of the sentences and an inference procedure defining the sentences derived from other sentences. The reasoning based on formal logics tends to evolve from facts for such assertions. In formal logics, it is complicated some type of human reasoning like assumption, likelihood, belief, disbelief, doubt, desires, etc. [46].

Rule based systems are the simplest form or AI having an appropriate syntax for representing the If–Then structure of rules. Rule based systems copy the behaviour of human reasoning depending on expert systems for problems solving. Instead of representing the knowledge in a declarative and static way link a set of truths, the system represents the knowledge like a group of rules that says how to act in different situations [55].

# 3. METHODOLOGY

In this chapter, it is explained the way of approaching the problem that is presented in this thesis and the use cases for developing an application for 3D reconstructions of technical installations for building rehabilitation. First of all, the final idea was developed, an application that was able to reproduce three-dimensionally a technical installation with accurate, providing the information of sanitation elements such as, pipes. This information includes the dimensions of each pipe as well as the data where can be bought. This data contains the price of each product from different suppliers in addition to the location of the company and the delivery time. The aim of this app is to ease the work for rehabilitations.

## 3.1 Approach

### 3.1.1 3D scanning and formatting

Firstly, how to build the 3D model without having the planes defining the building must be determined. There are many ways of building a 3D model, it can be done manually by using CAD applications but nowadays we should take advantage of technology and use them for our purpose. The use of 3D scanners is spreading and they are very suitable for analysing a scene or collecting data of the shape of an object.

To obtain an accurate 3D scanned data, the scene must be captured from different positions. The reason for this is that some areas cannot be scanned from a certain position because there are objects between them and the scanner. The result of the scanning are several files containing point clouds that need to be merged to have the complete room analysed with precision. The fusion of the different files is made using a specialized software for this type of data (Autodesk Recap).

The next step should be to get the point cloud data in a desired file extension for the processing of it. In this case, as I use the PCL(Point Cloud Library) library, the input file format should be converted to .pcd file format. This library is based on C++ programing language and it is chosen because it has many tools for point cloud data processing.

Processing of large areas of point clouds demands a lot of time and resources, affecting the quality of the result. Ideally, the application should work taking as an input the whole scene. In this case, for reducing tests time and complexity, the sample used was a part of the wall from the FASTory Lab in order to use the whole data. The sample selected included a couple of columns, a wall with different levels of deepness and many pipes with different radius, length and orientation.

With the aim of reducing the processing time, the density and number of points are reduced by using a voxel grid approach. A slighter sample file is obtained, but with enough information for getting the shape of the scene.

### 3.1.2 Filtering

This downsampled data includes the walls and the pipes that we are interested in for obtaining the 3D model, but it also includes all the elements that are between the scanner and the walls such as machines and tables. Elements that we are not focusing on and should be removed for detecting the structural elements precisely.

To remove these elements, the next steps are followed:

1. A region growing segmentation method is applied, with the aim of merging points that are close enough in terms of the smoothness constraint. The result of this segmentation is a set of clusters, where each cluster is a set of points that are considered to be a part of the same smooth surface. In this part, it is stablished a low angle threshold, for detecting plane objects. The minimum cluster size is set for preventing clusters with very few points.
2. In each cluster, planes are searched using RANSAC algorithm with a plane model. The objective of this is to reject the clusters that are not recognized as planes.
3. The clusters that are considered planes are merged in the same point cloud and the remaining point cloud is stored for future processing to find pipes.
4. A Euclidean cluster segmentation method is applied to the point cloud that has been created by grouping the plane clusters. This segmentation method merges points that are below a distance threshold defined by the user in different clusters.
5. The clusters that are bigger than a certain percentage of the total are considered part of the wall, so they are linked in the same cloud. This cloud is the one that is going to be analysed for obtaining the shape of the wall.

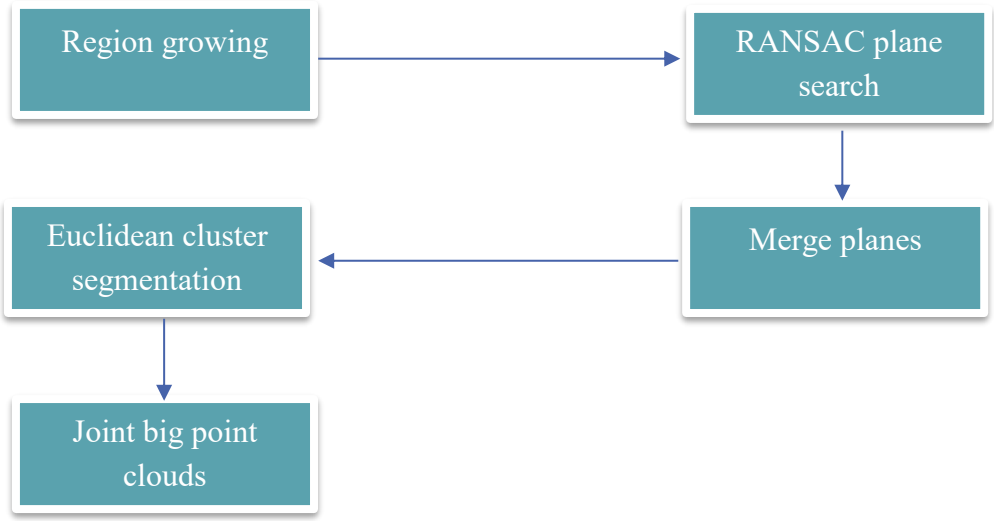


***Figure 10. Filtering process***

With this procedure, there are removed the points that are not part of the wall and we prevent having points that can affect the detection of the wall parameters. There are other ways to remove noisy points like statistical analysis techniques, as well as, conditional or radius outlier removal. The conditional filtering removes all indices in the given input cloud that do not satisfy one or more given conditions. Whereas, the radius outlier removal, removes all indices in its input cloud that do not have at least some number of neighbours within a certain range. These procedures have not been used because they are more suitable to filter isolated noisy points but not a set of points.

### 3.1.3 Planes detecting

After cleaning the point cloud of unnecessary points, the planes that compose the wall have to be identified. There are many ways of segmentation methods that can be used to detect planes. These segmentation techniques are explained in the theoretical background and for this purpose, a model fitting method is chosen because, it fits the points into a mathematical plane model, so we can represent a plane with the following equation:

$$ax + by + cz + d = 0 \tag{1}$$

There are many algorithms that are suitable for model fitting, in this work the RANSAC algorithm is used as it is implemented in the PCL library and is able to detect the planes efficiently and accurately, giving ,as a result, the coefficients of the identified plane and segmented cluster.

First of all, we identify all the planes and divide them. All the points that fit a plane model are gathered in the same cluster, regardless of the distance between the points, so a wall that is cut by a column is identified as the same plane. Then the plane that has the biggest amount of points is considered as the wall.

After that, the planes that are parallel to the wall are segmented in different clusters depending on the distance between points. For example, a wall that is cut by a column will be separated into clusters.

Then, these planes are classified into three groups: planes parallels to the walls, horizontal planes and, perpendicular planes to the wall and the horizontal planes.

The intersection of three planes results in a point, this is how the vertices of each plane are calculated. The vertices are calculated by intersecting a plane with the planes that delimit it from the sides, above and below.

With the vertices of a plane, we calculate the dimensions of a plane (width and height) and the middle point of it. These values are going to use for making the visualization of the 3D model.

### 3.1.4 Pipes detection

Once the structural elements are detected, the information of the pipes must be collected. The aim is to obtain the geometrical parameters of each pipe, so the application can use these data to make a visualization and to provide information about the pipes purchase.

This process is held in many steps:

1. Region growing algorithm is applied to segment objects that have the same smooth surface. This method has been applied before in the cleaning process, but in this time, the angle threshold is bigger than in the previous one. Increasing the angle threshold, the algorithm segments objects that have more curvature like pipes. The input file in this method is the remaining cloud of the cleaning process that was identifying and merging planes.
2. After segmenting the cloud in objects, RANSAC algorithm with a cylindrical model is applied, giving, as a result, the coefficients that define a cylinder, such as, radius, orientation… and point cloud identified as a cylinder. The search of the cylinders is made in certain directions, specifically, parallel to the walls but horizontal, vertical and normal to the wall. Specifying the axis of the RANSAC algorithm, the search is faster and is able to identify more cylinders than without defining the direction to search.
3. The RANSAC algorithm sometimes separate the same cylinders in different clouds so if the clouds and coefficients are compatible, these cylinders are joint and considered as a unique pipe.
4. Once the cylinders are identified, they are separated in different clouds if the distance between a set of points is bigger than a threshold. The reason for doing this is to separate a pipe that it is in both sides of a column, that bypasses the column but does not cross the column and it is detected as the same cylinder.
5. Finally, the length of each cylinder is computed by measuring the distance between the two farthest points in the direction of the cylinder.

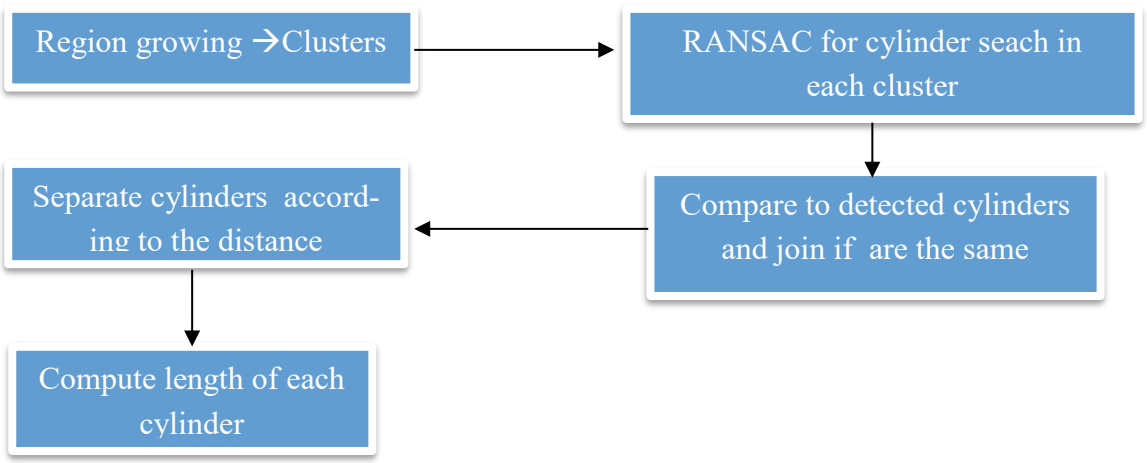


***Figure 11. Pipe detection***

This pipe detection approach is able to recognize the pipe as an object and inside this pipes, the cylinders that compound them. This data is used for making the visualization.

### 3.1.5 Visualization and data storage

Once all the parameters of the point cloud data are extracted, they have to be stored somehow to have the information available at any moment without having to execute the whole program again. In this case, all the data is saved in an ontology model. Ontologies help us to structure information and create a relationship between this data. There is also the possibility to use a database but ontologies are more suitable for reasoning.

In this model, there are defined the parameters of the walls, pipes and the relation among them as well as the information about the market. The ontology file is uploaded to an application on a local server. To populate this model, the information collected in the program is sent to this application.

When building the 3D model of the real scene, we should consider the weight and the smoothness of the visualization. An application that can be opened with any web browser is convenient, making it lightweight. For the previous reasons, the Three js library is chosen.

Now that the ontology model has the information of the geometries of the pipes and walls, the information is taken from there to build the 3D model, representing the walls as planes and the pipes as cylinders.

As described in the Fig. 25 the ontological model has five classes: *column, pipe, product, supplier* and *wall.* The *column* class defines the walls or planes that form a column, it has an object property called *collection_of*. Secondly, *pipe* classes include many data properties defining the geometry of a pipe, matching a product with an object property called *matches*. Thirdly, *product* concept contains the information of a product. It has an object property called *provided_by* that links the product with the suppliers. Next, *supplier* classes has information about its delivery time, location… Finally, the *wall* classes have all the geometrical data of each plane detected in the program. The next class diagram represents the classes created for this system.

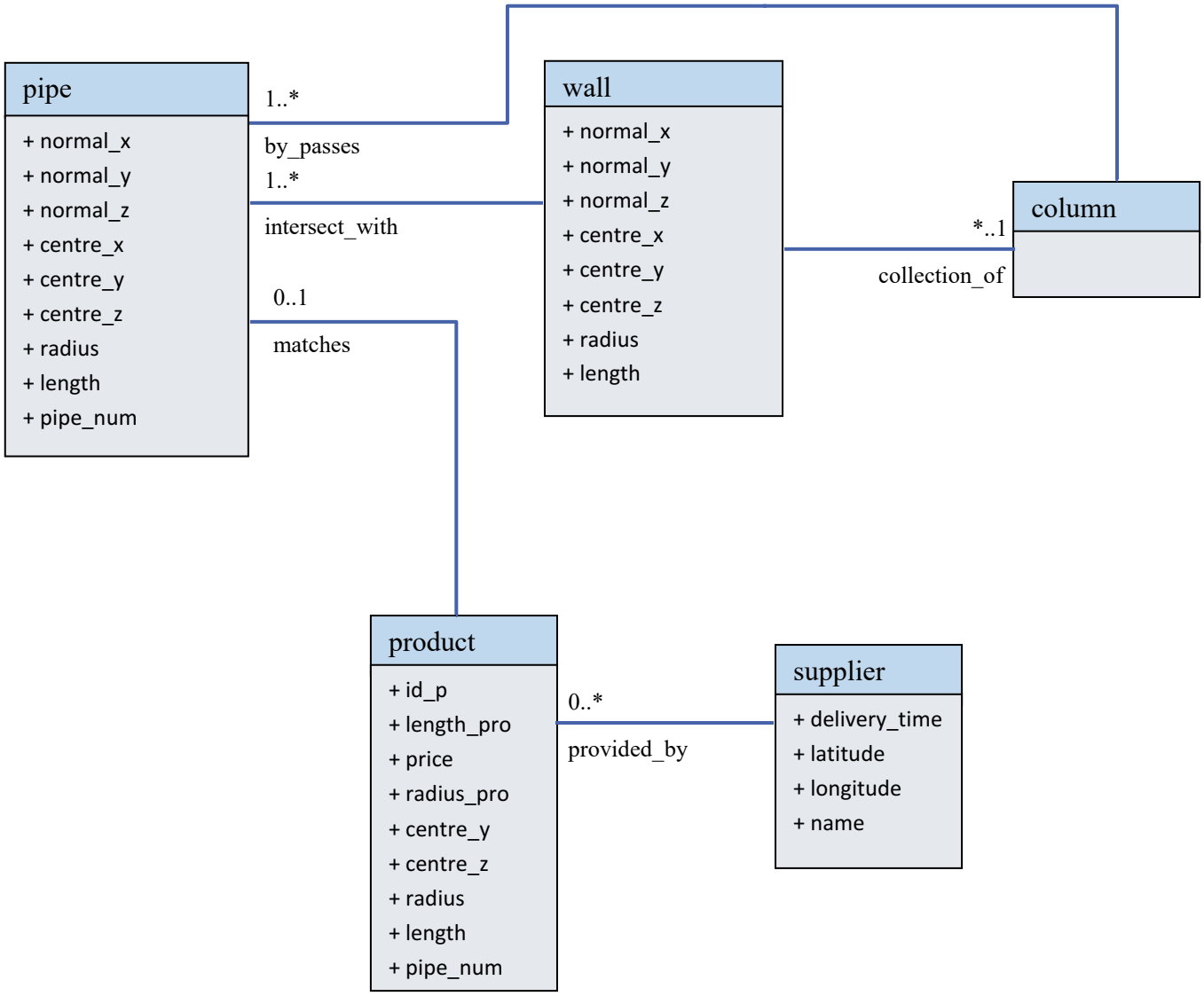


***Figure 12. Class diagram***

With the information extracted from the model, there can be represented straight segments of a pipe. To build the intersection between this segments, the situation of each segment has to be considered and take a conclusion in order to make the correct type of intersection.

Finally, the application shows the information of each cylinder when we click on it. The parameters of the pipes are compared to the products that have been defined manually in the ontology model. If there is a match between a product and a pipe, the purchase information is shown.

## 3.2 Tools

For the developing of this thesis many tools have been used. The first software used is Autodesk Recap, for assembling the point clouds of different scans. Second, with CloudCompare software the format of the point cloud data is converted to the desired one that is used in PCL library.

The code is developed in C++, JavaScript and HTML. C++ programming language is used because the PCL library, which has many point cloud segmentation and filtering tools, is C++ based. So the processing of the scanned data is done using PCL library. The web application is written in JavaScript +HTML because Three js library is used for developing the 3D model. Three js is a lightweight library written in JavaScript to create and display animated 3D computer graphics in a web browser and can be used in conjunction with the canvas element of HTML5, SVG or WebGL.

# 4. IMPLEMENTATION

In this section, the development of this thesis is explained. The developing of this thesis is not completely automatized, it uses some programs until the last result is obtained. For our tests, the FASTory lab, which is located in the Tampere University of Technology, was scanned with a 3D laser scanner. The conversion of the scanned data is done manually using CloudCompare and Autodesk Recap. After that, the structure's shape and pipes search are done in C++ programs. Each step of the process has been programmed in different C++ projects to simplifying the error detection and the coding. Finally, the visualization is developed using JavaScript and HTML.

## 4.1 Scanned data to PCD file format

To scan a scene accurately several scans are made changing the position of the scanner. This reduces the interferences of the objects that are in the middle of the scene and the scanner. Each scan produces a file, in our case, the output of the scanner is a .fls file from a FARO laser scanner. Some photos of the scene are taken to merge with the point cloud, capturing the texture of the scene.

Autodesk Recap is a software for assembling different point clouds and photogrammetry of a scene automatically, permitting the user to navigate in a virtual scene and use the point cloud for creating 3D models. A new project is created and all the point clouds are assembled, creating one point cloud of the whole scene. Then this cloud must be saved in a suitable file for analysing it. This software has many export file formats: E57, PTS, PCG and RCP/RCS.

PCL library is used for the processing of the point cloud, so the file format must be converted to one of these formats: PCD, PLY, CSV or VTK. As the Autodesk Recap output file formats do not match with any of the PCL library ones, the conversion process is divided into two steps:

1. Save the file in PTS file format.
2. Open the file in CloudCompare and export it as a PCD file.

The following graph resumes the process explained for format conversion.

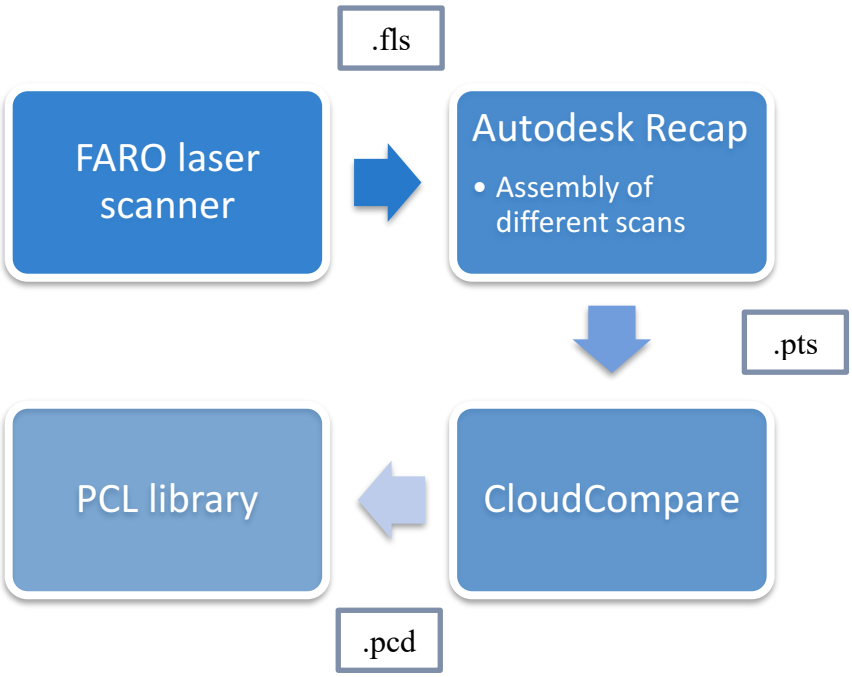


***Figure 13. Scan data to pcd file format conversion***

CloudCompare is a 3D point cloud and meshes processing software and it is compatible with many file formats. Using this program, where are going to select the part of the cloud for the testing, a wall including columns and many pipes.

***Figure 14. Sample file for tests***

Once the pcd file is obtained, the processing of the data starts. PCL (Point Clouds Library) is a C++ library for point cloud processing. This library has many tools for filtering, registration, segmentation, visualization and more. It is very suitable for this thesis' purpose because there are many segmentation methods already implemented and many tutorials to learn the use of it. The developed program is written in C++ programming language to

use the PCL library.

The exported file form CloudCompare, has a big amount of points (2.466.183 points), so to reduce the time of execution of the programs, the density of the points is decreased. Through a class called VoxelGrid from PCL this task is completed. It is possible to change the density of the output file by changing the seat leaf size of x, y and z axis.

```
// Create the filtering object
pcl::VoxelGrid<pcl::PCLPointCloud2> sor;
sor.setInputCloud(cloud);
sor.setLeafSize(0.01f, 0.01f, 0.01f);
sor.filter(*cloud_filtered);
```

***Code 3. Downsampling***

After applying this filter to the sample file, a less dense cloud is get (473.689 points). The difference between the number of points of the unfiltered and filtered cloud is huge, but the loss of information does not affect the processing.

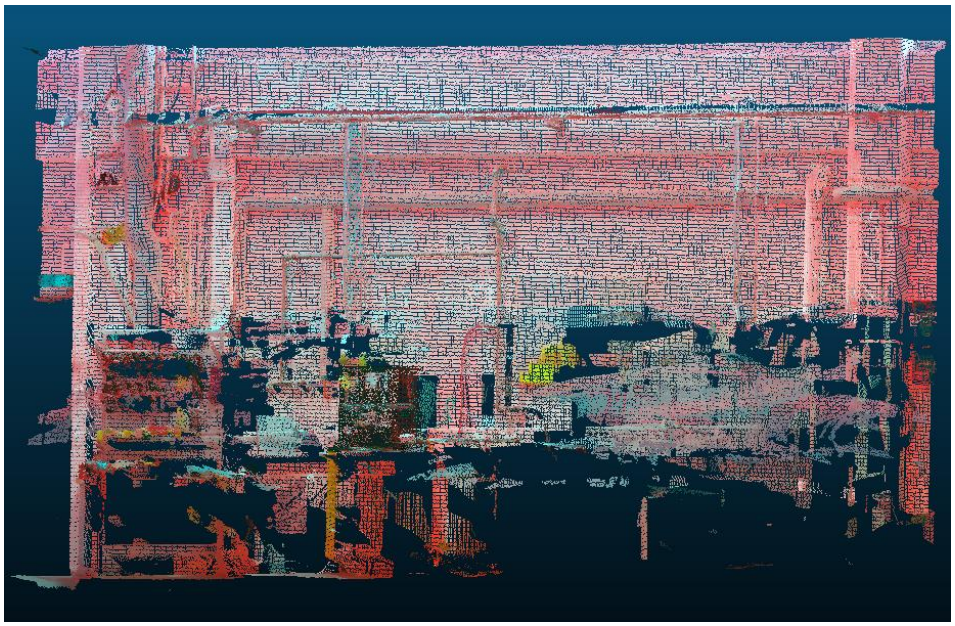

***Figure 15. Sample file after filtering***

## 4.2 Filtering

The filtering process removes the elements that are not part of the structures. The next photo shows many machines in front of the selected wall. These machines produce interferences in the scanned data, avoiding collecting the shape of the wall from a certain height down.


***Figure 16. Test scene***

This process starts segmenting the cloud using the `pcl::RegionGrowing` class. For that, it is necessary to compute the normal vector of each point with the `pcl::NormalEstimation` class. Next, region growing class is created, where we can define the input cloud, the smoothness and curvature threshold. As in this step, the intention is to find for plane objects, the smoothness angle is low.

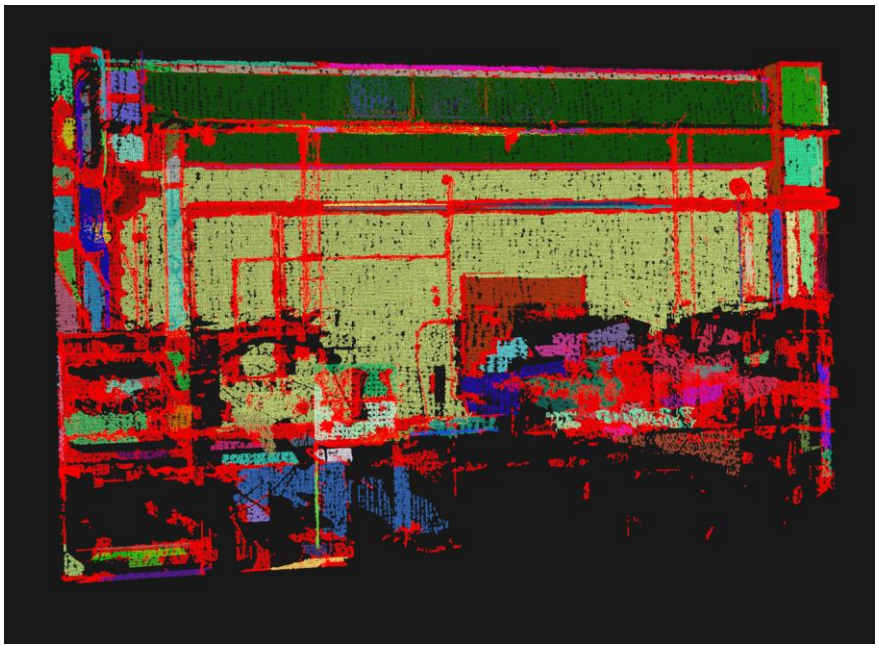

***Figure 17. Region growing segmentation, smoothness threshold: 3.0***

Each cluster is shown in different colour. The points that are in red are the points that are

not part of any cluster or the points number of the cluster is too low.

After segmenting the cloud in different clusters, each cluster is analysed using `sac_segmentation` and `sac_model_plane` classes. A cloud (*planes_cloud*) is created to concatenate clouds, which fit plane models. If in a cluster, a plane that is a high percentage of the cluster is detected, the cluster is added to the *planes_cloud*.

The clusters detected as a plane are extracted from the original cloud, giving, as a result, a point cloud with no planes. This cloud will be the input for the pipe detection.

The following code explains how the *planes_cloud* and the *cylinder_cloud* are created.

```
{N} ← Compute cloud normals
{Reg} ← Create region growing object
{Reg}← set input cloud {Cloud}, set normals {N}
{Reg} ← set smoothness (3.0) and curvature threshold (1.0)
{Clusters}← Calculate clusters
For i = 0 to size {Clusters} do
      {Ci}← Extract cloud from {Clusters}(i)
      Search planes in {Ci} using RANSAC
      If plane detected do
            {Planes_Cloud}← {Planes_Cloud}+ {Ci}
{Cilinder_cloud}← Clusters that are not plane
Save {Planes_Cloud}
```

***Code 4. Filtering algorithm***

Once the cloud composed by planes are obtained, the points that are far from the walls are removed, using `pcl::EuclideanClusterExtraction` class. This class segments the cloud based on the distance between points. The clusters that are big enough are saved and concatenated to get a point cloud that is just part of the wall.

The following diagram resumes the cleaning process.

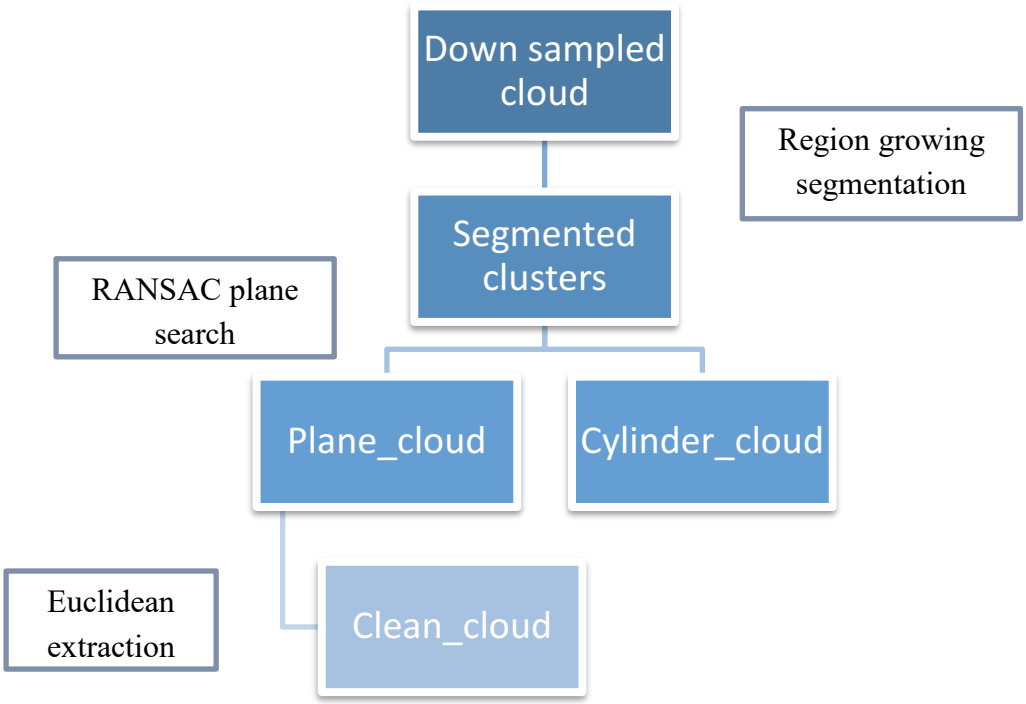


***Figure 18. Cleaning of the point cloud***

## 4.3 Planes detection

For the reconstruction of the scene, the shape and the position of the structure like walls and columns must be calculated. As the walls and columns of our test sample are planes without any curve, the search is done by fitting plane models into these elements.

The clean cloud is the input for the planes searching, avoiding all the unnecessary items. The `sac_segmentation` classes from PCL library are used for the planes searching. The model type of the RANSAC has to be defined to `sac_model_plane`. The coefficients the `sac_model_plane` returns are [**normal_x normal_y normal_z d**]. The first three values determine the direction of the normal vector of the plane and the **d** coefficient represents the offset of the plane. First, all the planes are search and saved separately. The clouds are stored in an array of point clouds of the type `pcl::PointCloud<pcl::PointXYZ>` and the coefficients in another array of the type `pcl::ModelCoefficients`.

The following code explain the steps for segmenting the planes. It is an iterative process that is repeated until the algorithm does not find any plane. When a plane is found, the points that are part of this plane are removed from the input cloud, avoiding detecting the same plane again and again.

```
{N }← Compute plane cloud normals
{seg}← Create RANSAC segmentation object
{seg}← Define plane model
Do
      {seg}← set input cloud {clean_cloud},set normals{N}
      {seg}←set distance threshold
      {inliers} ← search for inliers
      If size{inliers}< 1000 do
            {cloud_plane}(i)← extract inliers
            {clean_cloud} ← remove inliers
            {plane_coefficients}(i)← save model coefficients
While a plane is found
```

***Code 5. Algorithm for plane searching***

Once all the planes are detected, they are classified in different groups depending on their direction. The cloud with the highest number of points is considered as the main wall. All the other planes are or parallel or perpendicular to this plane. Calculating the angle of the normal vector of each plane with the normal of the main wall, the angle between the planes is calculated and afterwards classified into the different groups.

```
{v_main}← main wall normal vector
For i=0 to size{cloud_plane}
      {v_plane}← normal vector of {plane_coefficients}(i)
      {angle}← calculate angle between {v_main} and {v_plane}
      If  {angle}≈ 90 do
            If {v_plane} = vertical do
                  {horizontal}(j)←{cloud_plane}(i)
                  {horizontal_coefficients}(j)←{plane_coefficients}(i)
                  j++
            Else
                  {perperndicular}(k)←{cloud_plane}(i)
                  {perpendicular_coefficients}(k)←{cloud_plane}(i)
                  k++
      If {angle}≈ 0 do
            {wall_parallels}(h)←{cloud_plane}(i)
            {parallels_coefficients}(h)←{plane_coefficients}(i)
```

***Code 6. Algorithm for plane classifying***

In this step, all the planes have been detected but regardless the distance between points. The next step is to separate group of points that are far from other groups of points. In the following picture, a detected plane is shown, and different groups of points can be distinguished.

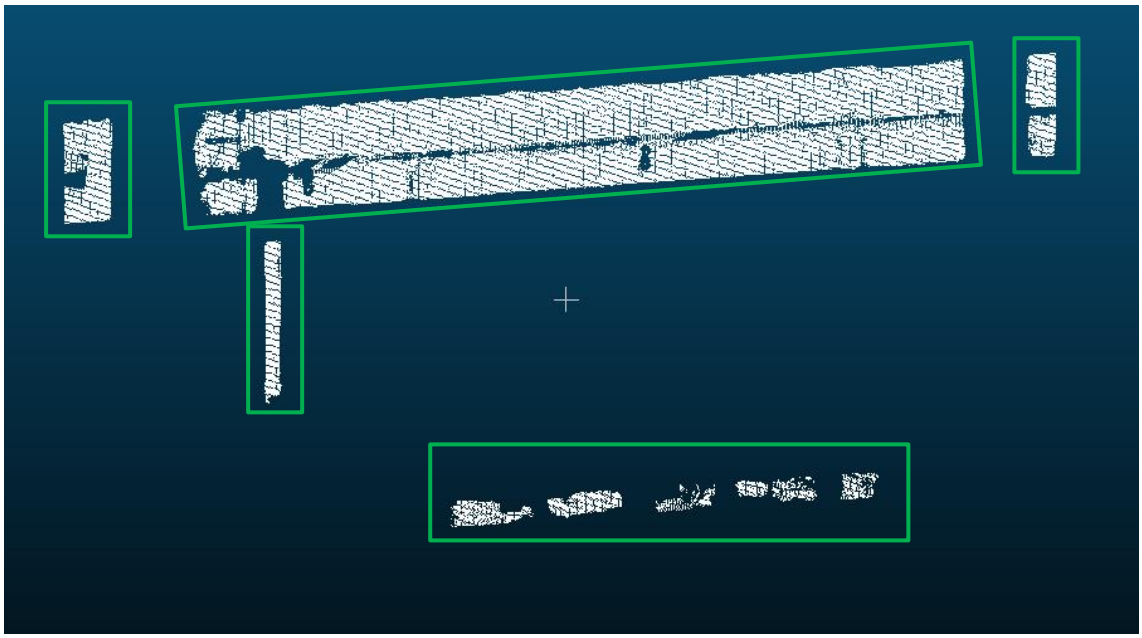

*Figure 19. Point cloud of a plane highlighting sets of points*

The set of points of each plane parallel to the wall are separated using `pcl::EuclideanClusterExtraction` class and saved them in an array.

The next graph shows the steps to follow for getting all the planes segmented having as an input the *clean_cloud* and classified regarding their direction.

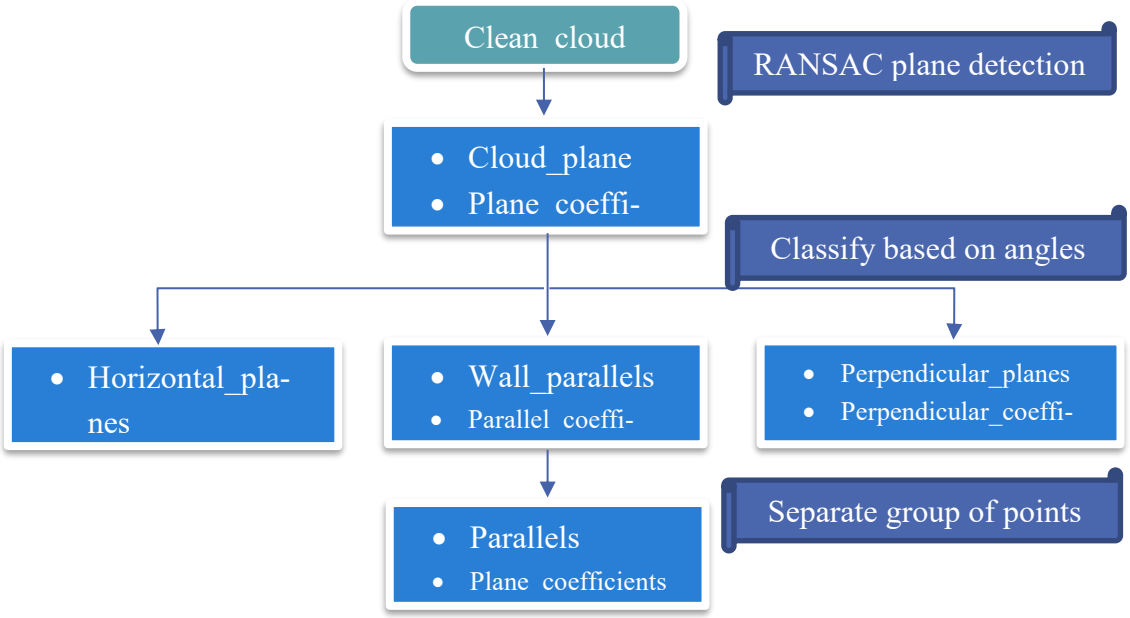


*Figure 20. Plane detection process*

There are many ways of defining a plane, like with the normal vector and the offset value, or with three corresponding points of the plane. For the visualization, the following values are needed: the coordinates of the planes' centre, the orientation, the length and the width.

To calculate these values, each plane parallel to the wall is intersected with the planes that delimit from the sides, from the top and from the bottom. The intersecting points are only calculated if the delimiting planes are close enough to the cloud, this is done dismiss the isolated planes that are detected in the previous process.

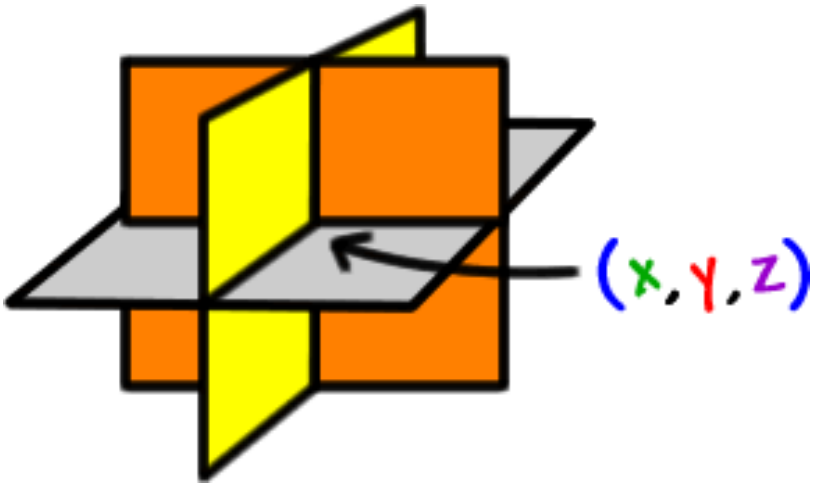


***Figure 21. Planes intersection for calculating points***

After calculating all the intersecting points of each plane, the coordinate of the centre is computed by taking the middle point of the extremes of the plane in two directions. To calculate the height and the width, the distance between the points of the extremes in two directions is measured.

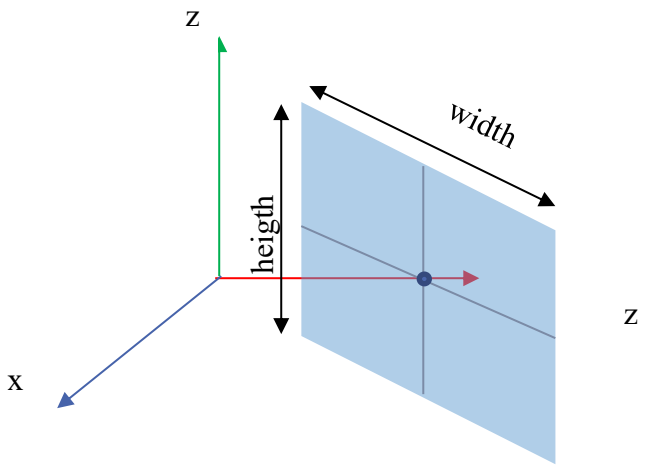


***Figure 22. Calculate the centre of a plane, height and width***

## 4.4 Pipes detection

The input file for the pipes detection is the *cylinder_cloud*, which originates as the remaining cloud of the cleaning process. First, `pcl::RegionGrowing` class is applied for detecting objects. However, this time the smoothness threshold is bigger for detecting objects with bigger curvature.

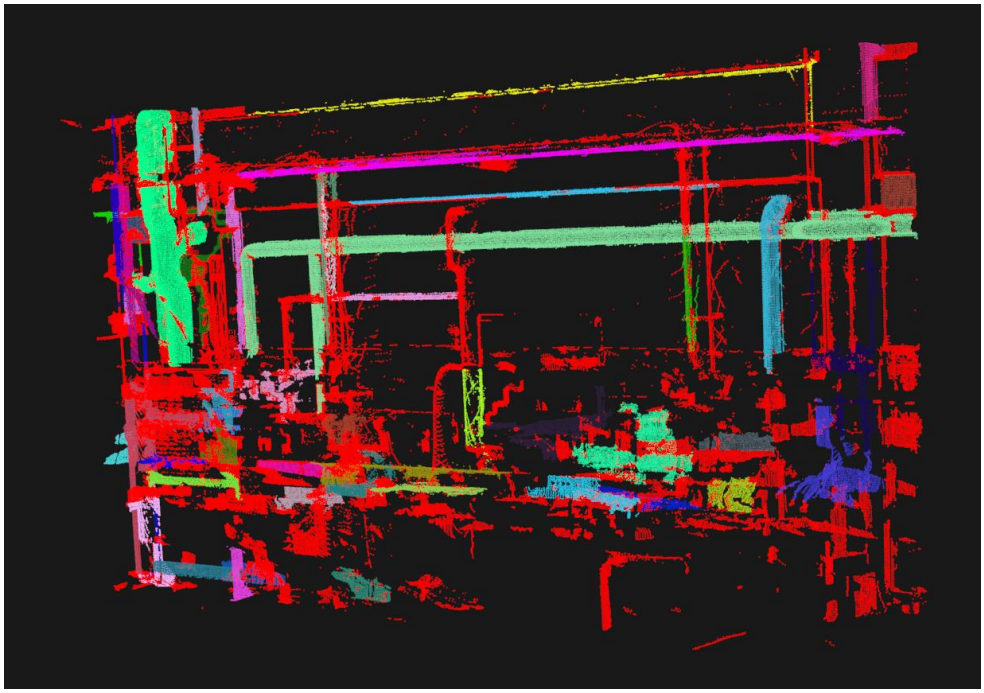

***Figure 23. Region growing segmentation, smoothness threshold: 8.0***

Afterwards, cylinder search is done using `sac_segmentation` class, but this time, the model type of the cylinder is `sac_model_cylinder`. The coefficients that this model type returns are [**point_on_axis.x point_on_axis.y point_on_axis.z axis_direction.x axis_direction.y axis_direction.z radius**]. The first three values define the coordinate of a point from the axis of the detected cylinder. The next three values determine the direction of the cylinder and the last one is the radius.

The RANSAC search using the `sac_model_cylinder` model type, let to define the search axis. Defining the axis, the search is more efficient and faster. In this test, the axis for searching the pipes are horizontal and parallel to the wall, vertical and normal to the wall. Depending on the maximum iteration number, the results vary and increasing the iterations number affects considerably the execution time.

Below the code for detecting the cylinders is presented.

```
{N} ← Compute cloud normals
{Reg} ← Create region growing object
{Reg}← set input cloud {cylinder_cloud}, set normals {N}
{Reg} ← set smoothness (8.0) and curvature threshold (1.0)
{Clusters}← Calculate clusters
{seg}← Create RANSAC segmentation object
{seg}← Define cylinder model
For i = 0 to size {Clusters} do
      {Ci}← Extract cloud from {Clusters}(i)
      Do
            {seg}← set input cloud {clean_cloud},set normals{N}
            {seg}←set axis
            {inliers} ← search for inliers
            If size{inliers}< 300 do
                  {cloud_cylinder}(i)← extract inliers
                  {Ci} ← remove inliers
                  {cylinder_coefficients}(i)← save model coefficients
            If cylinder not found
                  Change axis
      While a cylinder is found on any axis
```

*Code 7. Cylinder search algorithm*

Once all the cylinders are detected, the length of each pipe is computed, measuring the distance of the two extremes of the pipe on its direction.

Figure 24 shows the process to search the pipes.

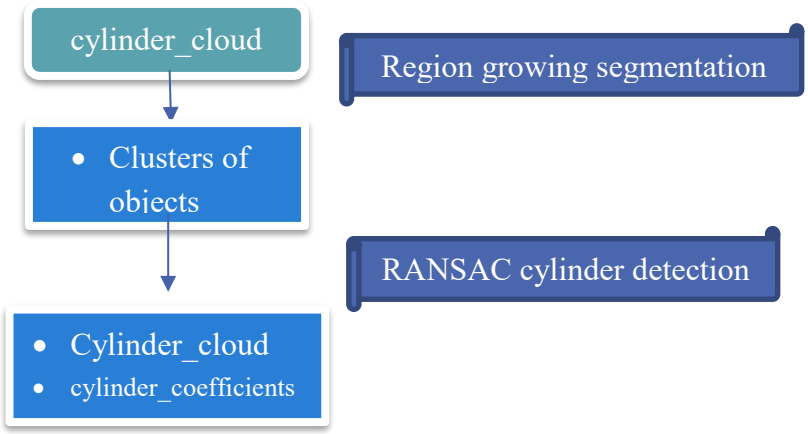


*Figure 24. Pipes detection*

## 4.5 Ontology modelling

A knowledge base system is developed for structuring the information extracted from the shape detection of the structure and elements. Besides, information about products and suppliers is added for giving the user information about pipes purchasing.

An ontology model OWL-based is built as tool for knowledge base system. This model is created using Olingvo that is a graphical user interface application developed at the Tampere University of Technology, which allows the creation, navigation, and edition of

OWL ontologies.

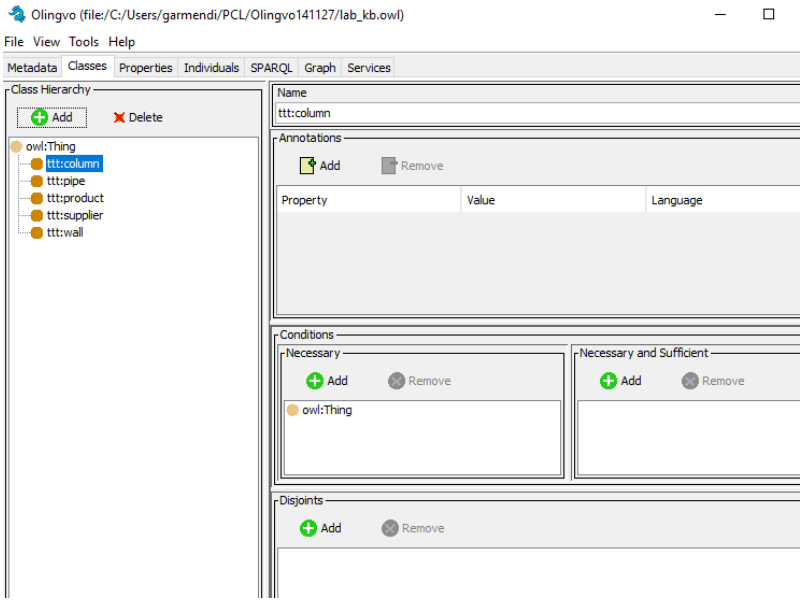


***Figure 25. Olingvo graphical interface***

The product and the supplier classes individuals are generated in Olingvo. For the test program, six products and three suppliers are generated. These data can be updated and extended giving a wide range of variety for purchasing pipes. The Fig. 26 shows the instances of a product.

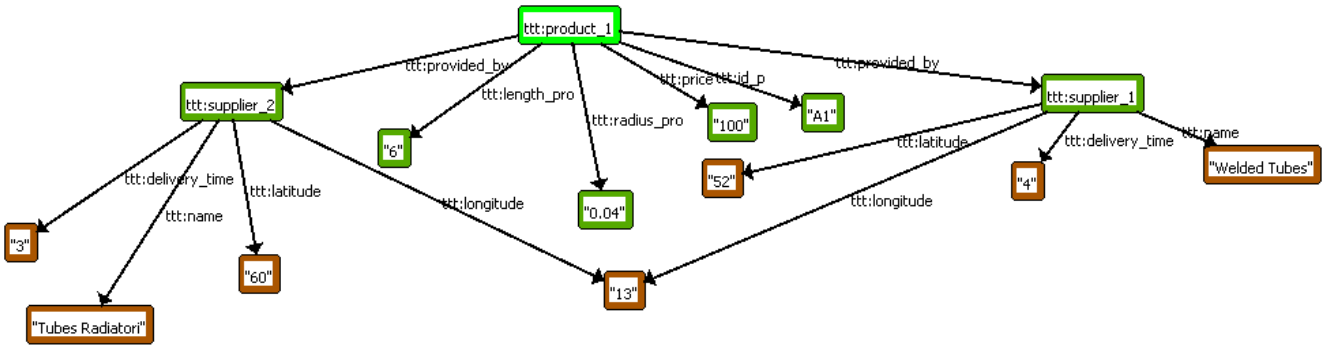


***Figure 26. Instances of product_1***

The information obtained from the C++ program must be stored in the ontological model automatically. The ontological model, which is in an OWL file format is uploaded to Fuseki that is a SPARQL server. It provides REST-style SPARQL HTTP Update, SPARQL Query, and SPARQL Update using the SPARQL protocol over HTTP.

The model is populated with the information of pipes and walls is by the execution of a SPARQL Update query that inserts a new individual of the class *wall* or *pipe* as well as their attributes. The following code shows an example of a query to insert the data of a wall to the model.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ttt: <http://www.ontologies.com/Ontology6613.owl#>
      INSERT DATA {
      ttt:wall_1 rdf:type ttt:wall.
      ttt:wall_1 ttt:n_x   n_x.
      ttt:wall_1 ttt:n_y   n_y.
      ttt:wall_1 ttt:n_z   n_z.
      ttt:wall_1 ttt:c_x   c_x.
      ttt:wall_1 ttt:c_y   c_y.
      ttt:wall_1 ttt:c_z   c_z.
      ttt:wall_1 ttt:width  w.
      ttt:wall_1 ttt:height  h.
}
```

***Code 8. SPARQL Update query***

The connection between the C++ program and the Fuseki server is done by using a POST method that is implemented in C++ REST SDK library. This is a Microsoft project for cloud-based client-server communication.

The parameters that need to be defined in the post method are the url where the Fuseki application is executed *(http://localhost:3032/iii2017/update)*, the body and the content type of the body. The body is the SPARQL query message shown in the Code 8 with a content type of *application/x-www-form-urlencoded.*

## 4.6 Web application

The web application is developed in HTML and Javascript programming languages. For the visualization of 3D models, Three js library is used, it is a cross-browser JavaScript library and Application Programming Interface (API) used to create and display animated 3D computer graphics in a web browser.

The application is an html document where some Three js scripts are loaded and Javacript code is developed for the using the Three js library. *Three.js* and *OrbitControls.js* script are loaded. The first one is to create mesh objects, scenes, lights and more. The second script allows the camera to orbit around a target.

```
<script src="js/three.js"></script>
<script src="js/OrbitControls.js"></script>
```

All the visualization elements, the reasoning to show the information of the pipes and the intersection between pipes is elaborated in a Javascript. First, scene, camera and renderer objects are created. The walls and pipes are drawn as planes and cylinders. For that, the information of each plane and cylinder is needed. The data of each plane and cylinder is taken from the ontological model which has been uploaded to Fuseki and populated from C++ application.

To take the information, an HTTP request is done to the Fuseki application. The HTTP request method is the same as used before, POST method. Now, the information must be extracted and not updated, so the url (*http://localhost:3032/iii2017/query)* and query message change. Code 9 shows an example querying data from a wall.

```
PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
PREFIX ttt: http://www.ontologies.com/Ontology6613.owl#
        SELECT ?predicate ?object
        WHERE{
        ttt:wall_1 ?predicate ?object
        }
```

***Code 9. Retrieving wall data from model***

All the data of the pipes, walls and products is stored in JSON objects. This data is used to create the mesh that is going to be rendered in the web application. To draw an object, first the geometry and the material must be defined. Then the mesh is created with the geometry and the material. Finally, to display a mesh, it must be added to the scene.

The Code 10 shows the function to create and add the pipes to the scene.

```
function creategeometry(pipes)
{
    for (i=0;i<pipes.length;i++)
    {
    var material = new THREE.MeshLambertMaterial({color: 0xcFFFF5, side: THREE.DoubleSide});
        geometry[i]=new THREE.CylinderBufferGeometry(pipes[i].r,pipes[i].r,pipes[i].l,32);
        cylindermesh[i]=new THREE.Mesh(geometry[i],material);
        cylindermesh[i].name= i;
        if (pipes[i].nz!=1)
        {
            cylindermesh[i].rotateX(Math.PI / 2);
            cylindermesh[i].rotateZ(2.23249);
        }
        cylindermesh[i].postion.set(pipes[i].cy,pipes[i].cz,pipes[i].cx);
        scene.add(cylindermesh[i]);
    }
}
```

***Code 10. Function for pipe creation***

The geometry of a cylinder is created using *THREE.CylinderBufferGeometry*, where the radius and the length is defined. For the geometry of a plane, *THREE.PlaneBufferGeometry* is used, defining the height and the width of the plane.

The positioning of an object is done by using the function *object.position.set(x,y,z).* Where the x, y and z are the position of the object's centre. To orient the object *object.ro-*

*tateX(angle)*, *object.rotateY(angle)* and *object.rotateZ(angle)* can be used. These functions rotate the object in local axis.

Once the scene is done, the application includes a button that when pressing on it, the application makes the intersections between the pipes. As in the pipe detection, first the pipe as an object was segmented and after that, the cylinders were searched inside this object, it is known which cylinders are part of the same object. There are many types of intersections depending on the position of one pipe respect another one. These possible intersections where calculated based on this works sample file cases:

1. Two perpendicular pipes that cross forming a T. In this case a pipe is enlarged to the centre of the other pipe.
2. Two perpendicular pipes forming an L that their extremes need to be joint. A torus is added joining the extremes of the two pipes.
3. Two parallel pipes but that the distance between their axis is bigger than a value and are bypassing a column. A perpendicular cylinder is added between the two pipes and two tori are added to joint this cylinder two the two pipes.
4. Two parallel pipes but that the distance of the axis is smaller than a certain value. A tube is added with a spline curvature approximated to the extremes of the pipes.

When clicking on a pipe, the information of the pipe, the product that matches with the pipe and the suppliers of it are shown in a panel. For providing accurate information, reasoning is needed. The selected pipe is compared with the products catalogue selecting, selecting the product if the radiuses are equal and the length is similar. Each products information is obtained from the Fuseki server where the owl file is loaded.

A product class has an attribute called *provided_by* that links a product with its suppliers. When the product has been matched with a pipe, the Javascript program gets the information of the suppliers querying to Fuseki and prints it in the web application's info panel.

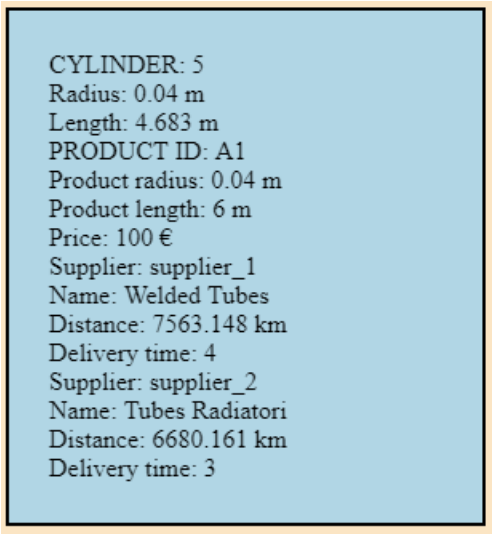


***Figure 27. Cylinder's info panel***

The system is divided into many parts, the C++ program, Fuseki server and the web application. The Fig. 28 shows the architecture of the system. The C++ program processes the scanned data and obtains the information of each element of the scene. The ontological model is loaded in Fuseki, the model is updated from the C++ program using SPARQL over HTTP Update. The web application queries information from the model in Fuseki using SPARQL for the model creation.

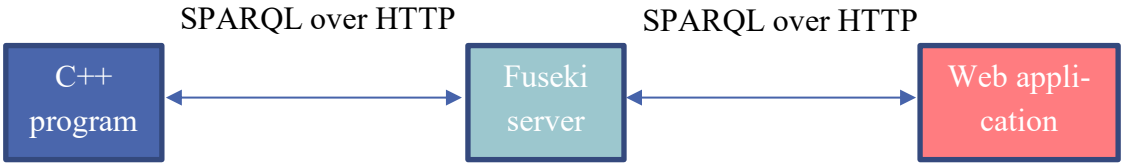


***Figure 28. Architecture of the system***

The next activity diagram explains step by step the process to build the web application. In the C++ program the steps to obtain the scenes information are: filtering, plane and cylinder searching and finally populating the ontological model. The ontological model is done in Olingvo and uploaded to Fuseki. The Fuseki applications takes the updating and querying requests and respond to the client. The web application asks data to Fuseki and recieves an answer which is processed to build the 3D model and the provide replacement information of the market.

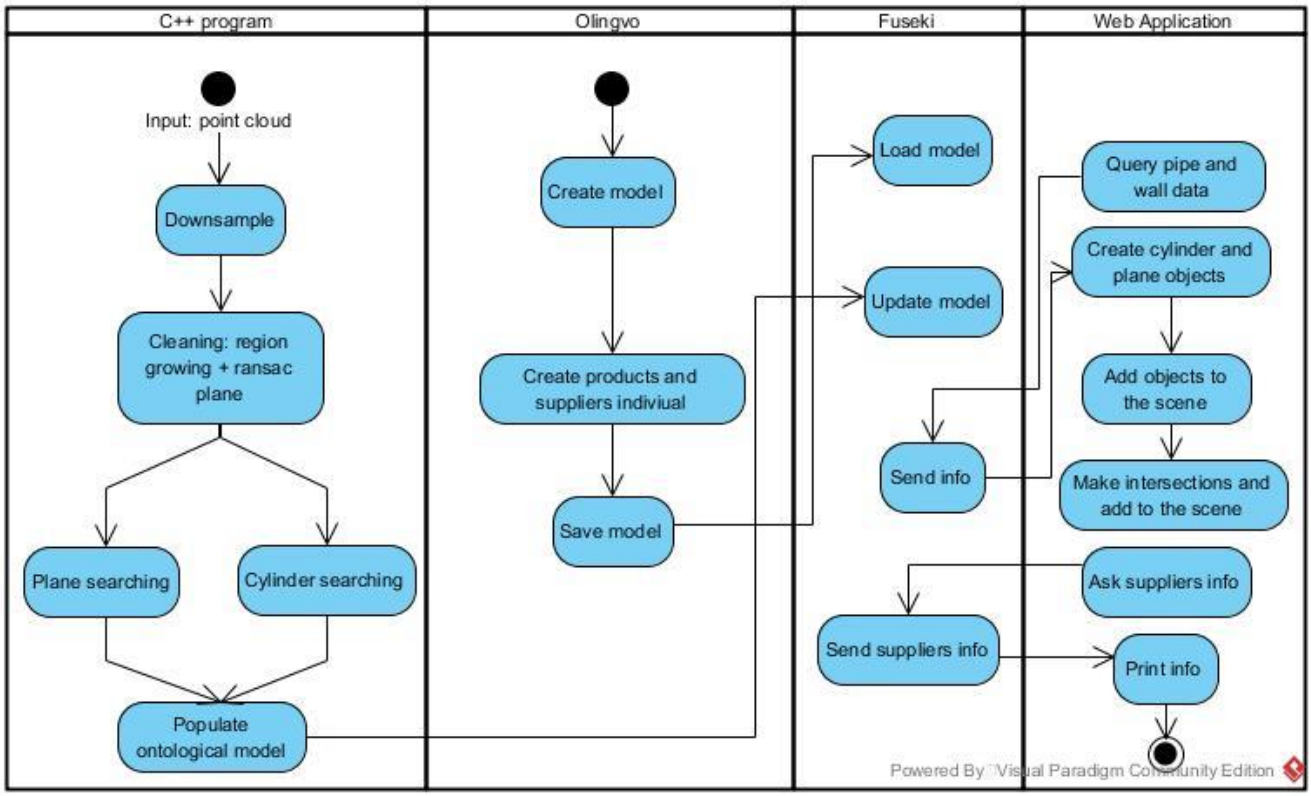


***Figure 29. Activity diagram***

# 5. RESULTS

The goal of the cleaning process is to remove all the elements that are not structural like machines or tables. The shape of the structure and the pipes could be detected without cleaning the point cloud of unnecessary points. It could be possible to apply the RANSAC algorithm but the execution time would be higher and unwanted elements would be detected. Some conditions must be established to accept or reject the detected items.

In the Fig. 30 the point cloud after the cleaning process is shown. Comparing to the Fig. 11, you see that pipes or other items have been removed. Facilitating the recognition of walls.

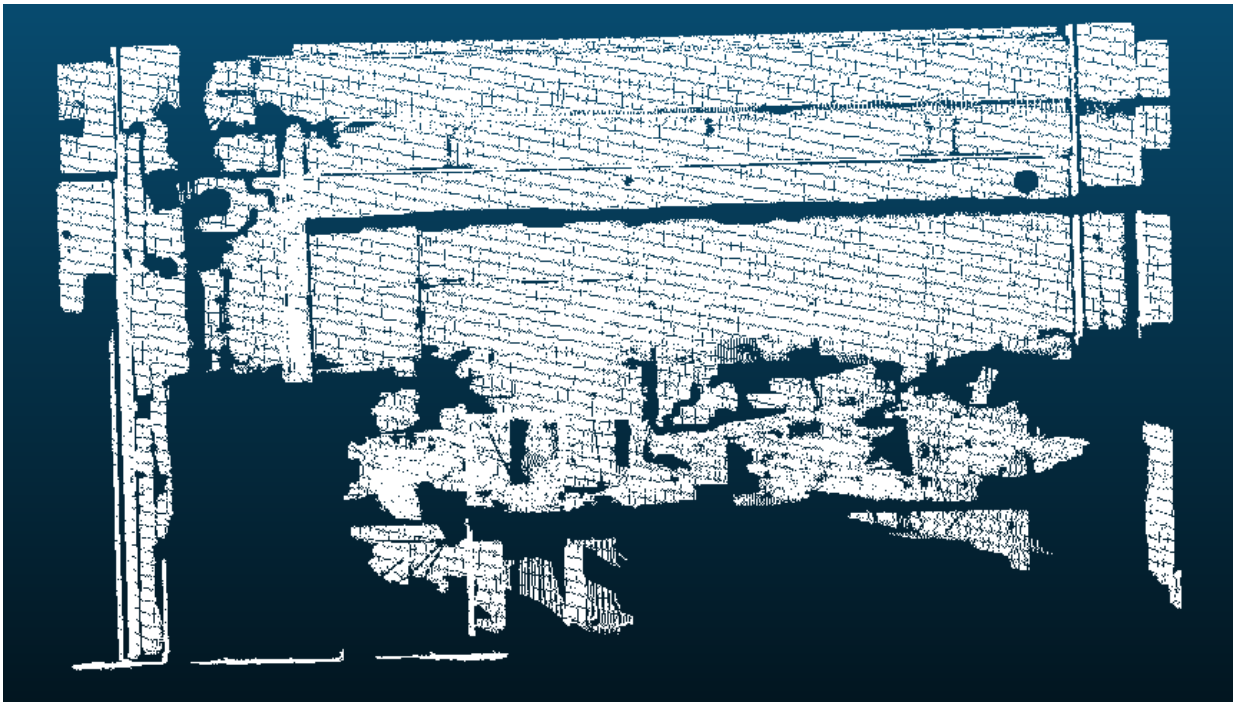

*Figure 30. Clean point cloud*

*Figure 11. Clean point cloud*

The remaining point cloud contains the pipes and the elements that have been extracted from the cleaned cloud. The Fig. 31 cloud is more suitable to search for pipes, improving the search time and obtaining better results. The RANSAC algorithm can be applied directly in this cloud, but there is a risk to obtain false positives and recognize cylinders that in the reality are not. Moreover, the pipes are recognized as objects by segmenting the cloud using region-growing algorithm. After applying RANSAC it is known which cylinder belongs to which tube.

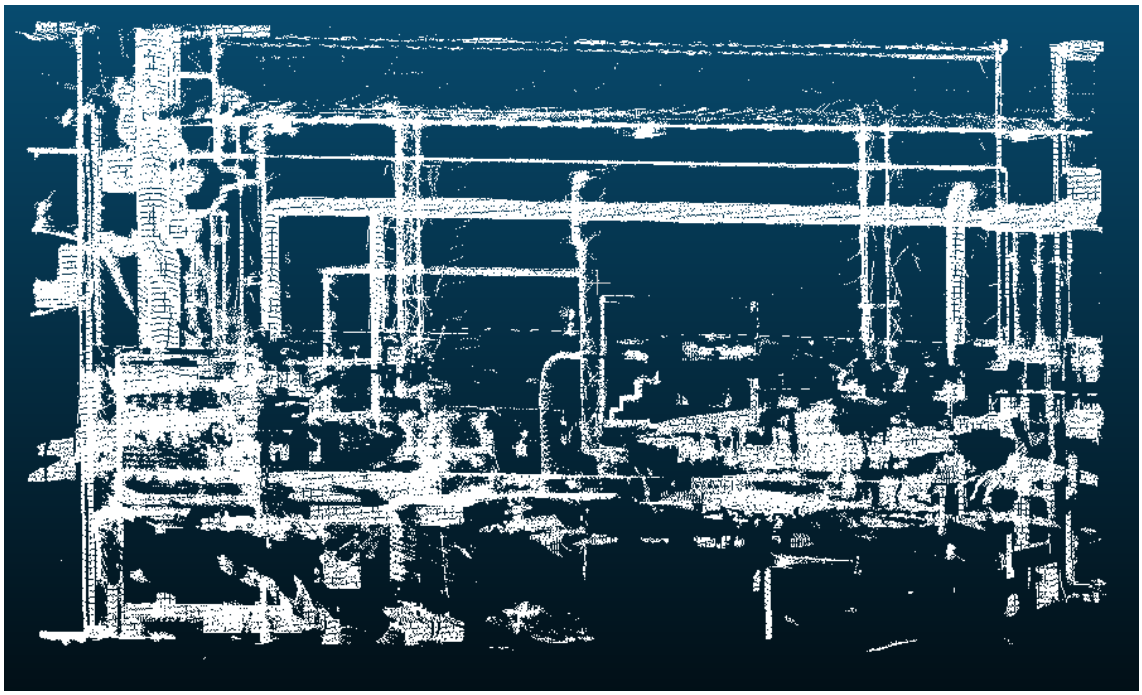

*Figure 31. Point cloud for cylinder search*

In the cleaning process, first region growing and then RANSAC is applied to search for planes. These planes could be used to determine the shape of the structure, but there are too many planes detected and most of them unnecessary. Establishing some conditions planes would be accepted or rejected. This approach is not followed and a plane search is carried out using RANSAC algorithm. The Fig. 32 shows the results after searching the planes. Each detected plane is shown in a different colour.

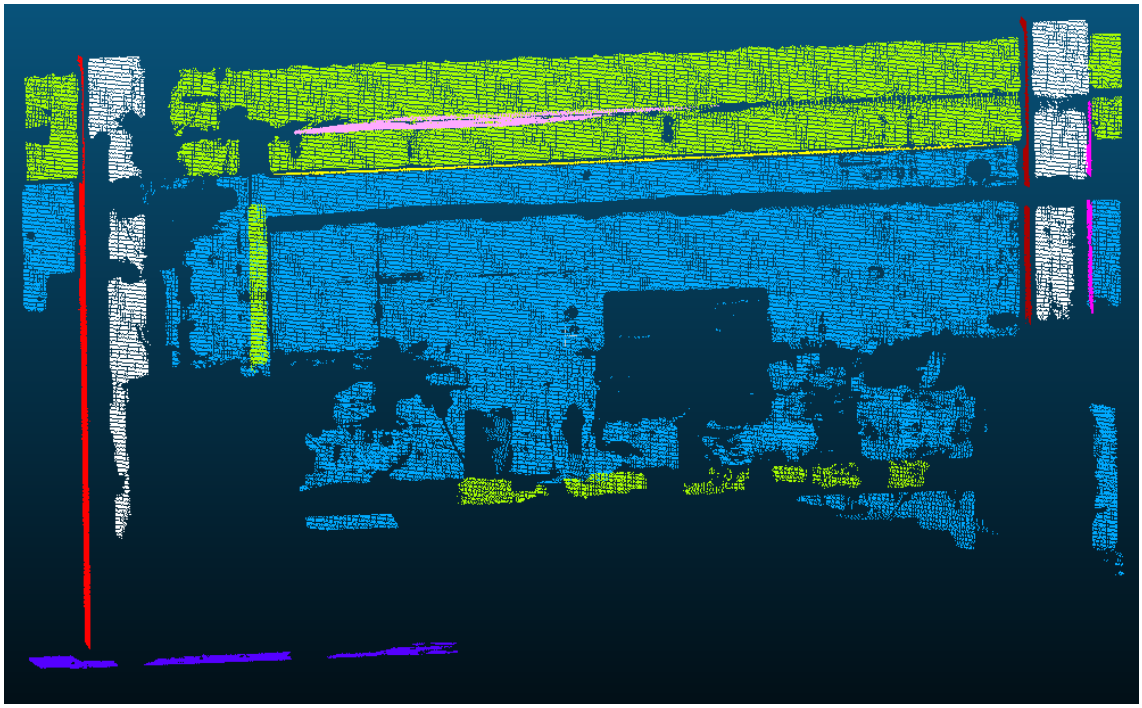

***Figure 32. Segmented planes***

For an accurate recognition, some parameters must be adjusted, like the weight of the angular and the distance of each point to the model. Depending on the density of the point cloud and the profundity of the points, these values will change. In a point cloud with similar features, the same method with the same configurations could be applied.

In the cylinder detection, the program is not able to identify cylinders with small radius. This is because the cylinders with small radiuses are represented with a low number of points comparing to the big ones. RANSAC algorithm parameters should be adjusted to identify them. The Fig. 33 shows the identified cylinders.

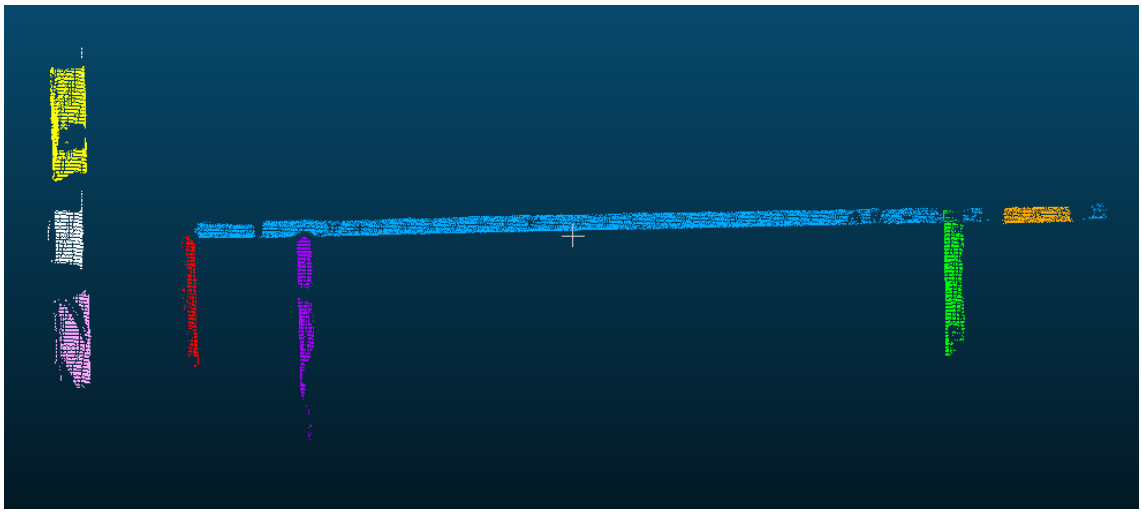

*Figure 33. Detected cylinders*

A negative point of the RANSAC cylinder search algorithm is that it cannot detect the curvatures of a pipe. For searching the curvatures of a pipe, the volumes of these parts should be approximated to a mathematical model, which is not easy and are not implemented in the PCL library used in this work. It would be possible to consider the remaining points of the cylinder search as intersections and convert them to meshes. However, if the points just represent the visible part of the pipe to the scanner there would be missing points and reconstruction of these meshes would be required.

The Three js library gives facilities to build a web application with 3D content. The 3D model could be done in different ways, directly using the Three js library or creating a model like .obj for example. Saving the model in a certain file format would enable to open this file in many software for visualizing it or to modify it.

Three js is suitable to create animated 3D scenes, it is lightweight and smooth. In this case, when the button to intersect cylinders, the scene is updated modifying pipes or adding new objects.

The scene created out of the data was obtained from the scanning and processing of it is quite accurate. The shape of the structure is perfectly digitized, representing all the walls and columns.

Pipes are represented as cylinders and comparing the virtualized pipes to the real ones, the conclusion is that digital pipes have a precise position, length and radius.

The Fig. 34 shows the web application. The interface contains the digitized 3D model, an info panel in the top left corner to print the information of the pipes and a button in the right top corner for making the intersection between the cylinders.

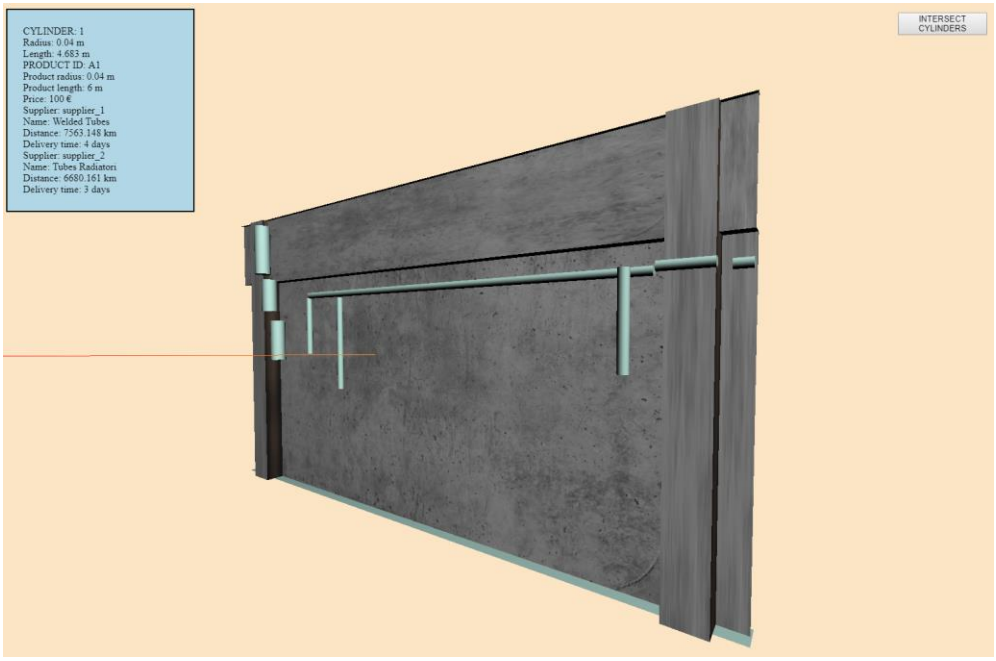


***Figure 34. Web application interface***

In the Fig. 35, we can compare the real scene and the virtual one. All the machines are removed and some elements are missing like small pipes, grids, but the main structure and main pipes are shown.

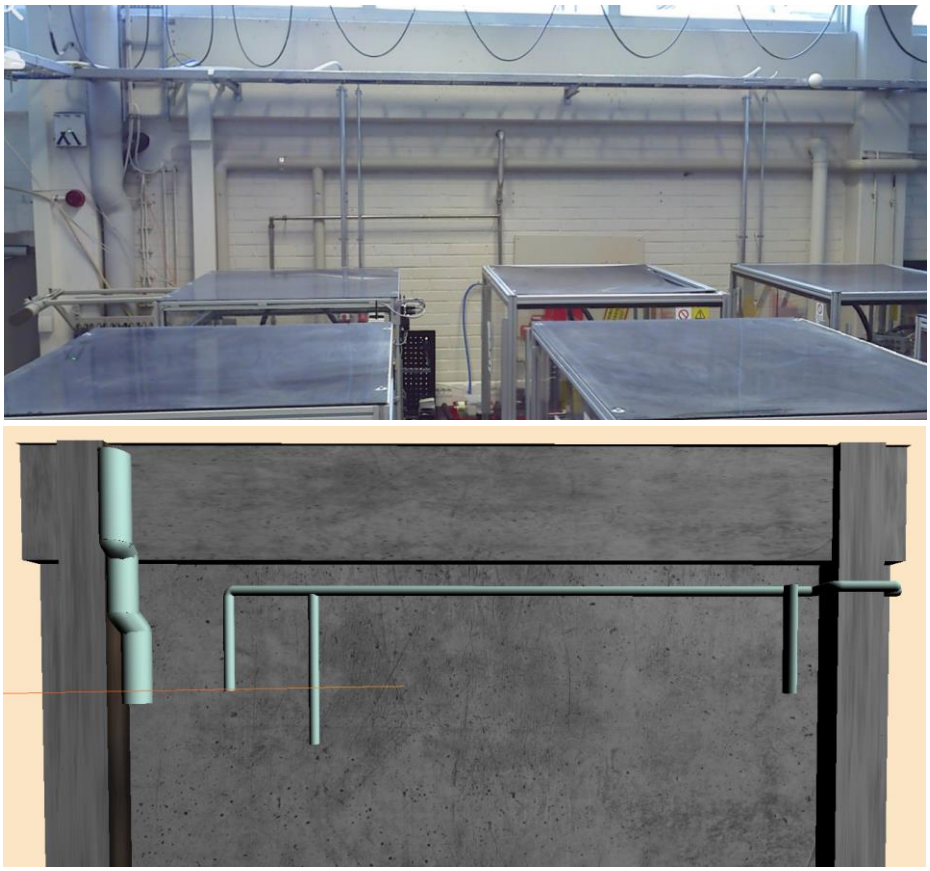

***Figure 35. Real scene and 3D model***

In the Fig. 34 the model is shown without making the intersections of the pipes. After the button is clicked these are made and represented in the scene. The four cases explained in the implementation part are exposed in the following figures.

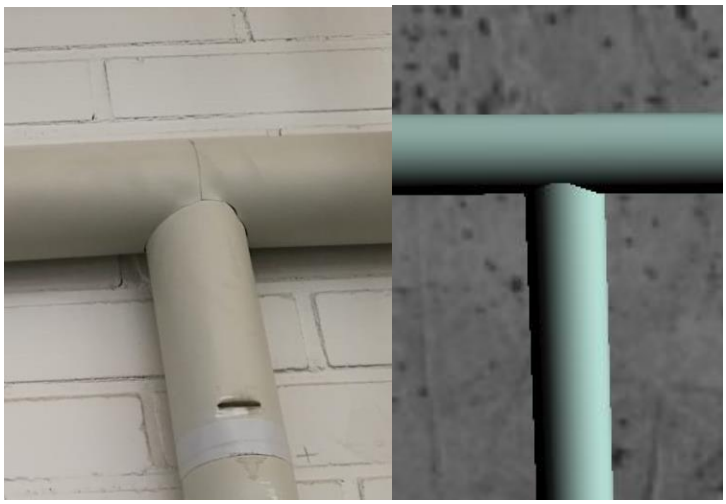

***Figure 36. Intersection type T***

Fig. 36 shows two perpendicular pipes forming a T shape. The vertical pipe is enlarged until the centre of the horizontal axis.

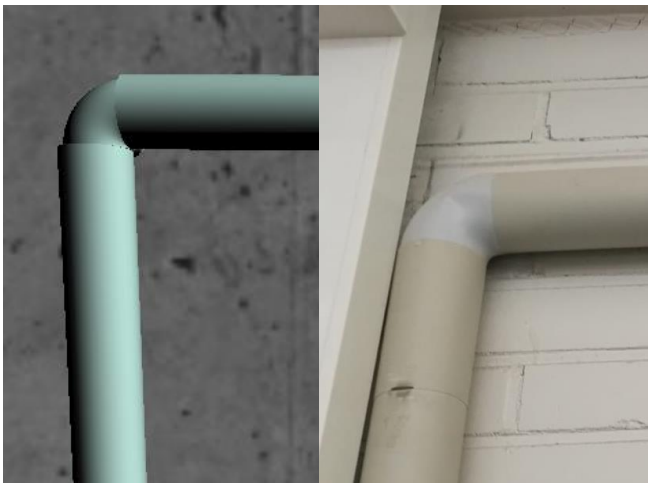

***Figure 37. Intersection type L***

Two perpendicular pipes forming an L are shown in Fig. 37. A torus is added for joining the two tubes.

*Figure 38. Intersection of two parallel pipes, bypassing a column*

Fig. 38 shows the intersection of pipes that are bypassing a column, the intersection is done by adding two perpendicular pipes in the normal direction of the wall and joining these pipes with the others using torus geometry.

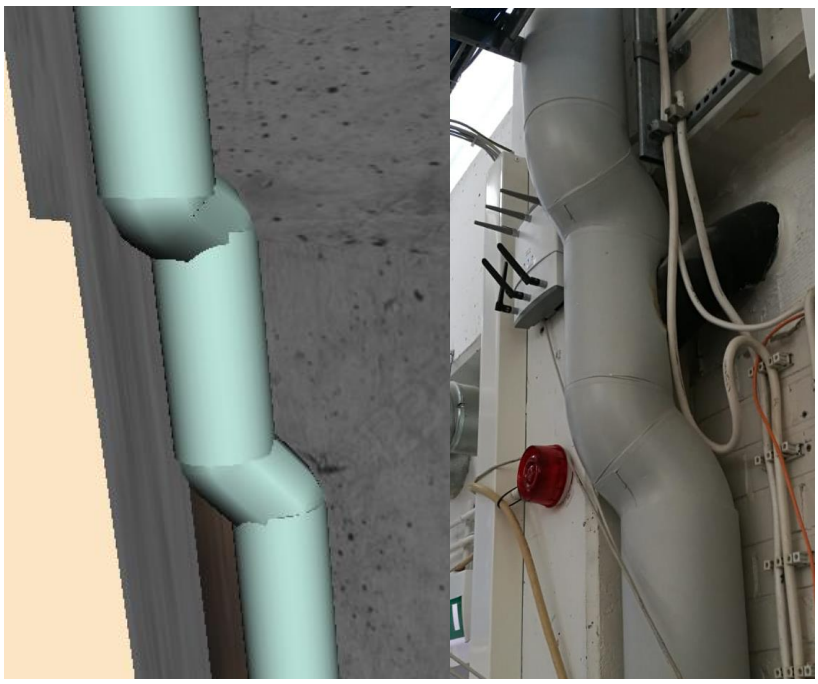

*Figure 39. Intersection of parallel pipes, spline*

The intersection between two parallel pipes is done in Fig. 39. The intersecting tube is approximated to a spline curve, which goes from the extremes of two pipes. The real intersection is not as the one of the web application. The 3D models joint is an approximation of the real one that has like two elbows.

The information for purchasing a replacement of a pipe benefits the user showing the price of the it and the information of the supplier such as, the distance and the delivery time of the supplier as well as products radio and length. The Fig. 40 and Fig. 41 show the purchase information of two pipes. These two pipes have the same radius but different length, so the mathching product is different as well as the suppliers.

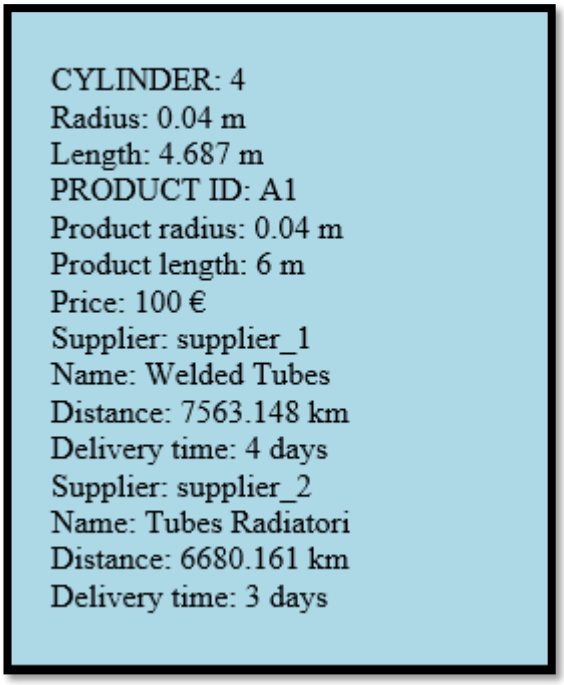


***Figure 40. Cylinder 4 purchase information***

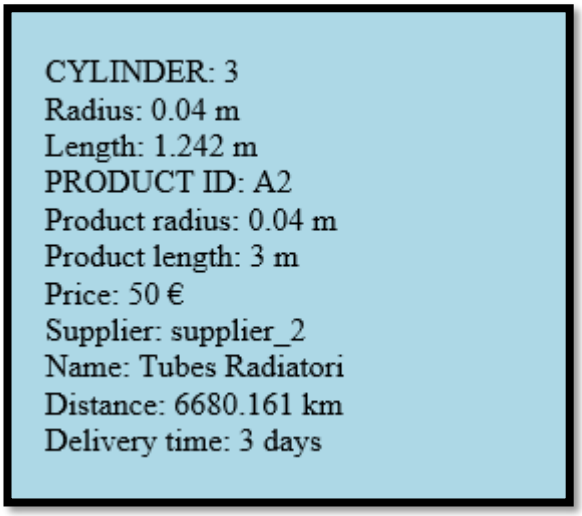


***Figure 41. Cylinder 3 purchase information***

# 6. CONCLUSION

A knowledge-driven method for 3D reconstruction of technical installations in building rehabilitation is presented in this thesis. This chapter discusses the completion of the aspects that have been presented in the introductory chapter, explaining the factors that can affect the final result.

Digitizing a scene can be done in many different ways using many techniques. In this thesis, a 3D laser scanner was used to capture the shape of the FASTory Lab. It is a consistent and a fast method to catch the scene but big objects like machines can create interferences in the point cloud data. When scanning a scene, the position of the scanner must be taken into account to obtain the most accurate possible data. In this case, the scanner, must focus on all the pipes and walls. The obtained point cloud data was missing points that were behind the machines, that affected in the computing the length of a pipe.

The method presented in this paper can be applied to other scenes or point clouds. Depending on the features of the point cloud, it may be needed to adjust some parameters like the distance threshold of the points when searching for planes or cylinders. These parameters affect the model fitting segmentations, over or under segmenting the scene.

The detection of pipes with small radius has not been achieved. This kind of pipes are represented with a low number of points, making difficult the fitting of the points to a cylinder model.

The 3D model that was built from the data obtained from the processing of the point cloud was quite accurate, representing all the walls and pipes as in the real scene. However, as the points that were behind the machines were not captured perfectly, the length of a pipe is shorter than in reality.

Furthermore, the current proposal developed in this work has many advantages. Allows the user to access the application from any platform without installing, updating or maintaining any software. A simple click to execute the web browser allows them to run the application.

For future development or for another application using the data that has been stored in the ontological model. The architecture of the application allows to make modifications or to update it easily as the program is divided into many modules.

The final application can be very useful for reconstruction works, providing the user information of the pipes that must be replaced.

## 6.1 Future work

The present thesis has explained to the reader the potential of digitizing a real scene. Nowadays, digitizing and connecting every system to the cloud is becoming very popular. Having a virtual model allows working remotely without visiting the scene and simplifying the tasks. The proposal that is suggested in this thesis offers a lot of possibilities for further development and expansion.

First of all, the scene recognition and modelling should be improved to capture the entire room and not just a wall. A whole room could be virtualized with just scanning it in order to build the model manually. All the pipes should be detected, including the small ones and regardless the direction. The intersections could be fitted to geometrical mathematical models for RANSAC algorithm search.

The texture and colour of the scene can be used for the identification of the elements in the scene. The visualization can be done with the original textures, approximating the model to the reality.

Furthermore, the database of the products and suppliers could be updated with real products and suppliers, showing the user a wide variety of products. Having real time information of each supplier would be a further step, allowing the user to decide immediately and avoiding asking the supplier if a product is available or not.

The web application could be executed on a server, where the scanned data would be uploaded. The program would do all the recognition of the elements and the modelling. The program would return to the user the 3D model to navigate on it and also, the information of the pipes.

Finally, the 3D model could be saved on a CAD file, enabling to open the model on CAD programs, where §it could be modified and extract the desired information.

# REFERENCES


[1] Wolfgang BOEHLER and Andreas MARBS, “3D SCANNING INSTRUMENTS.” .

[2] R. Schmidt, A. Zimmermann, M. Möhring, S. Nurcan, B. Keller, and F. Bär, “Digitization – Perspectives for Conceptualization,” in *Advances in Service-Oriented and Cloud Computing*, 2015, pp. 263–275.

[3] H. Hirsch-Kreinsen, “Digitization of industrial work: development paths and prospects,” *J. Labour Mark. Res.*, vol. 49, no. 1, pp. 1–14, Jul. 2016.

[4] A. Mathys, J. Brecko, and P. Semal, “Comparing 3D digitizing technologies: What are the differences?,” in *2013 Digital Heritage International Congress (DigitalHeritage)*, 2013, vol. 1, pp. 201–204.

[5] A. Núñez Andrés, F. Buill Pozuelo, J. Regot Marimón, and A. de Mesa Gisbert, “Generation of virtual models of cultural heritage,” *J. Cult. Herit.*, vol. 13, no. 1, pp. 103–106, Jan. 2012.

[6] W. BOEHLER and A. MARBS, “3D SCANNING INSTRUMENTS,” p. 4.

[7] G. Pavlidis, A. Koutsoudis, F. Arnaoutoglou, V. Tsioukas, and C. Chamzas, “Methods for 3D digitization of Cultural Heritage,” *J. Cult. Herit.*, vol. 8, no. 1, pp. 93–98, Jan. 2007.

[8] L. Gomes, O. Regina Pereira Bellon, and L. Silva, “3D reconstruction methods for digital preservation of cultural heritage: A survey,” *Pattern Recognit. Lett.*, vol. 50, pp. 3–14, Dec. 2014.

[9] S. Zhang, “High-speed 3D shape measurement with structured light methods: A review,” *Opt. Lasers Eng.*, vol. 106, pp. 119–131, Jul. 2018.

[10] V. Carbone, M. Carocci, E. Savio, G. Sansoni, and L. De Chiffre, “Combination of a Vision System and a Coordinate Measuring Machine for the Reverse Engineering of Freeform Surfaces,” *Int. J. Adv. Manuf. Technol.*, vol. 17, no. 4, pp. 263–271, Jan. 2001.

[11] OR3D, “What is 3D Scanning? - Scanning Basics and Devices,” *OR3D*. .

[12] J. Lee and S. Zlatanova, Eds., *3D geo-information sciences*. Berlin: Springer, 2009.

[13] “FARO Focus | FARO SPAIN, S.L.U.” [Online]. Available: https://www.faro.com/es-es/productos/construccion-bim-cim/faro-focus/. [Accessed: 04-May-2018].

[14] H. Radvar-Esfahlan and S.-A. Tahan, “Robust generalized numerical inspection fixture for the metrology of compliant mechanical parts,” *Int. J. Adv. Manuf. Technol.*, vol. 70, no. 5–8, pp. 1101–1112, Feb. 2014.

[15] X.-T. Yan, B. Eynard, and W. J. Ion, Eds., *Global design to gain a competitive edge: an holistic and collaborative design approach based on computational tools*. London: Springer, 2008.

[16] L. Ladický, O. Saurer, S. Jeong, F. Maninchedda, and M. Pollefeys, “From Point Clouds to Mesh Using Regression,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 3913–3922.

[17] Q. Li, R. Xiong, S. Huang, and Y. Huang, “Building a dense surface map incrementally from semi-dense point cloud and RGBimages,” *Front. Inf. Technol. Electron. Eng.*, vol. 16, no. 7, pp. 594–606, Jul. 2015.

[18] A. Nguyen and B. Le, “3D point cloud segmentation: A survey,” in *2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, 2013, pp. 225–230.

[19] E. Grilli, F. Menna, and F. Remondino, “A REVIEW OF POINT CLOUDS SEGMENTATION AND CLASSIFICATION ALGORITHMS,” *ISPRS - Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. XLII-2/W3, pp. 339–344, Feb. 2017.

[20] D. Bazazian, J. R. Casas, and J. Ruiz-Hidalgo, “Fast and Robust Edge Extraction in Unorganized Point Clouds,” 2015, pp. 1–8.

[21] "Line segment extraction for large scale unorganized point clouds - ScienceDirect." [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271615000362. [Accessed: 04-May-2018].

[22] H. Ni, X. Lin, X. Ning, and J. Zhang, "Edge Detection and Feature Line Tracing in 3D-Point Clouds by Analyzing Geometric Properties of Neighborhoods," *Remote Sens.*, vol. 8, no. 9, p. 710, Sep. 2016.

[23] P. J. Besl and R. C. Jain, "Segmentation through variable-order surface fitting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 2, pp. 167–192, Mar. 1988.

[24] J. P. Mills and J. H. Chandler, "ISPRS Commission V Symposium: Image Engineering And Vision Metrology," *Photogramm. Rec.*, vol. 22, no. 117, pp. 94–96, Mar. 2007.

[25] A. Khaloo and D. Lattanzi, "Robust normal estimation and region growing segmentation of infrastructure 3D point cloud models," *Adv. Eng. Inform.*, vol. 34, pp. 1–16, Oct. 2017.

[26] "Documentation - Point Cloud Library (PCL)." [Online]. Available: http://pointclouds.org/documentation/tutorials/region_growing_segmentation.php#region-growing-segmentation. [Accessed: 23-May-2018].

[27] J. Illingworth and J. Kittler, "A survey of the hough transform," *Comput. Vis. Graph. Image Process.*, vol. 44, no. 1, pp. 87–116, Oct. 1988.

[28] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Commun ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[29] A. Hidalgo-Paniagua, M. A. Vega-Rodríguez, N. Pavón, and J. Ferruz, "A Comparative Study of Parallel RANSAC Implementations in 3D Space," *Int. J. Parallel Program.*, vol. 43, no. 5, pp. 703–720, Oct. 2015.

[30] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N. J. Mitra, "GlobFit: Consistently Fitting Primitives by Discovering Global Relations," p. 11.

[31] C. Papazov and D. Burschka, "An Efficient RANSAC for 3D Object Recognition in Noisy and Occluded Scenes," in *Computer Vision – ACCV 2010*, 2010, pp. 135–148.

[32] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for Point-Cloud Shape Detection," *Comput. Graph. Forum*, vol. 26, no. 2, pp. 214–226, Jun. 2007.

[33] O. Chum and J. Matas, "Matching with PROSAC — Progressive Sample Consensus," 2005, vol. 1, pp. 220–226.

[34] P. H. S. Torr and A. Zisserman, "MLESAC: A New Robust Estimator with Application to Estimating Image Geometry," *Comput. Vis. Image Underst.*, vol. 78, no. 1, pp. 138–156, Apr. 2000.

[35] S. Barnea and S. Filin, "Segmentation of terrestrial laser scanning data using geometry and image information," *ISPRS J. Photogramm. Remote Sens.*, vol. 76, pp. 33–48, Feb. 2013.

[36] F. Hamid-Lakzaeian and D. F. Laefer, "An Integrated Octree-RANSAC Technique for Automated LiDAR Building Data Segmentation for Decorative Buildings," in *Advances in Visual Computing*, 2016, pp. 454–463.

[37] L. Li, F. Yang, H. Zhu, D. Li, Y. Li, and L. Tang, "An Improved RANSAC for 3D Point Cloud Plane Segmentation Based on Normal Distribution Transformation Cells," *Remote Sens.*, vol. 9, no. 5, p. 433, May 2017.

[38] J. MacQueen, "Some methods for classification and analysis of multivariate observations," presented at the Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, 1967.

[39] S. S. Savkare, A. S. Narote, and S. P. Narote, "Comparative Analysis of Segmentation Algorithms Using Threshold and K-Mean Clustering," in *Intelligent Systems Technologies and Applications 2016*, 2016, pp. 111–118.

[40] K. V. A. Muneer and K. P. Joseph, "Performance Analysis of Combined k-mean and Fuzzy-c-mean Segmentation of MR Brain Images," in *Computational Vision and Bio Inspired Computing*, Springer, Cham, 2018, pp. 830–836.

[41] F. Kharbat and H. Ghalayini, "New Algorithm for Building Ontology from Existing Rules: A Case Study," in *2009 International Conference on Information Management and Engineering*, 2009, pp. 12–16.

[42] R. T. Ng and J. Han, "Efficient and Effective Clustering Methods for Spatial Data Mining," p. 12.

[43] N. Milton, "Knowledge Technologies," *ArXiv08023789 Cs*, Feb. 2008.

[44] R. G. Smith, "Knowledge-Based Systems: Concepts, Techniques, Examples," p. 84.

[45] P. Shah and A. Miyake, *Models of Working Memory – An Introduction*. 1999.

[46] D. Gašević, D. Djurić, and V. Devedžić, *Model driven architecture and ontology development*. Berlin: Springer, 2006.

[47] R. Akerkar and P. Sajja, *Knowledge-Based Systems*. Jones & Bartlett Publishers, 2010.

[48] G. Paliouras, V. Karkaletsis, and C. D. Spyropoulos, Eds., *Machine learning and its applications: advanced lectures*. Berlin ; New York: Springer, 2001.

[49] "Case-Based Reasoning VS Reinforcement Learning – Part 1 – Introduction," *Omar's Brain*, 09-Feb-2010. .

[50] R. Davis, H. Shrobe, and P. Szolovits, "What Is a Knowledge Representation?," *AI Mag.*, vol. 14, no. 1, p. 17, Mar. 1993.

[51] S. L. Kendal and M. Creen, *An introduction to knowledge engineering*. London: Springer, 2007.

[52] W. M. Mohammed *et al.*, "Generic platform for manufacturing execution system functions in knowledge-driven manufacturing systems," *Int. J. Comput. Integr. Manuf.*, vol. 0, no. 0, pp. 1–13, Nov. 2017.

[53] B. R. Ferrer, W. M. Mohammed, and J. L. M. Lastra, "A solution for processing supply chain events within ontology-based descriptions," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 2016, pp. 4877–4883.

[54] S. Iarovyi, W. M. Mohammed, A. Lobov, B. R. Ferrer, and J. L. M. Lastra, "Cyber-Physical Systems for Open-Knowledge-Driven Manufacturing Execution Systems," *Proc. IEEE*, vol. 104, no. 5, pp.

1142–1154, May 2016.

[55] C. Grosan and A. Abraham, “Rule-Based Expert Systems,” in *Intelligent Systems*, Springer, Berlin, Heidelberg, 2011, pp. 149–185.