Universidade de Aveiro
2022

Bruno Campos Cardoso Leal Neto

Three-Dimensional Shape Recognition for a Laser Vibrometer Scanner

Reconhecimento de Forma Tridimensional para um Sistema de Varrimento de um Vibrómetro Laser

Universidade de Aveiro
2022

Bruno Campos
Cardoso Leal Neto

Three-Dimensional Shape Recognition for a Laser Vibrometer Scanner

Reconhecimento de Forma Tridimensional para um Sistema de Varrimento de um Vibrómetro Laser

Trabalho de Projeto apresentado à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob orientação científica de Rui António da Silva Moreira, Professor Auxiliar, do Departamento de Engenharia Mecânica da Universidade de Aveiro, e de Vítor Manuel Ferreira dos Santos, Professor Associado com Agregação do Departamento de Engenharia Mecânica da Universidade de Aveiro.

Esta dissertação teve o apoio dos projetos UIDB/00481/2020 e UIDP/00481/2020 - Fundação para a Ciência e a Tecnologia; e CENTRO-01-0145 FEDER-022083 - Programa Operacional Regional do Centro (Centro2020), através do Portugal 2020 e do Fundo Europeu de Desenvolvimento Regional.

Agradecimentos / Acknowledgements

Em primeiro lugar gostaria de agradecer aos meus orientadores por toda a ajuda e apoio prestados durante o desenrolar deste projeto. Um agradecimento especial a toda a minha família que sempre me apoiou e que esteve do meu lado durante todo o meu percurso académico. Gostaria de agradecer a todos os meus amigos que também tiveram um papel importante tanto no desenvolver deste documento como em todo o meu percurso académico. A todas as pessoas que desenrolaram um papel importante nesta fase da minha vida, sem especificamente mencioná-las, elas sabem quem são, o meu profundo obrigado.

Keywords Laser Vibrometry, Galvanometer, Vibration, Laser Scanning, Matlab

Abstract Laser vibrometry is a technique that, among several possibilities, allows modal analysis of light and thin structures. To make this analysis possible, the capture of a set of points is initially necessary. The devices present in the market, besides being expensive, present a rudimentary obtaining of points, usually done manually by the user. Not presenting the flexibility required for research, it is planned in this project to develop a system of automatic acquisition of these points for a laser vibrometer scanner. Given this need, a system was created to capture points through the information acquired by a Microsoft Kinect camera. In a Matlab environment, these points were converted into point clouds and processed through an application, in which the user can define their final configuration. This work presents the results obtained during the analysis of two different types of geometries through the application developed.

Palavras-chave Vibrometria Laser, Galvanómetro, Vibrações, Varrimento Laser, Matlab

Resumo

A vibrometria laser é uma técnica que, entre várias possibilidades, permite a realização de análises modais de estruturas leves e finas. Para tornar esta análise possível, a captura de um conjunto de pontos é inicialmente necessária. Os dispositivos presentes no mercado, para além de dispendiosos, apresentam uma obtenção de pontos rudimentar, normalmente feita de forma manual pelo utilizador. Não apresentando a flexibilidade exigida para investigação, está previsto neste projeto desenvolver um sistema de aquisição automático destes pontos para um scanner de um vibrometro laser. Dada esta necessidade, foi criado um sistema de captura de pontos através da informação adquirida por uma câmara da Microsoft Kinect. Num ambiente Matlab, estes pontos foram convertidos em nuvens de pontos e processados através de uma aplicação, na qual o utilizador pode definir a sua configuração final. Este trabalho apresenta os resultados obtidos durante a análise de dois tipos diferentes de geometrias através da aplicação desenvolvida.

Contents

List of Tables

Intentionally blank page.

List of Figures

List of Code Listings

Intentionally blank page.

Chapter 1

Introduction

1.1 Motivation

Studying the dynamic behaviour of light structures (cars, aircraft, wind turbine blades) commonly uses experimental modal analysis or operational mode analysis techniques. It is imperative to measure the dynamic response of a structure when subjected to a known excitation or due to its operation. In this type of testing, it is usual to use accelerometer networks to measure the structure response. However, the laser vibrometry technique presents a robust implementation in various industrial applications due to its speed, reduced analysis requirements and negligible interference on the structure. It uses a velocity measurement system based on the Doppler principle, allowing the laser beam to measure the illuminated point speed.

The main advantage when using laser vibrometry is the non-contact measurement. However, as a disadvantage, it can only measure the velocity at a single point (where the laser beam is incident). Some scanning systems are available in the market [1], but these are very expensive and require a time-consuming setup since the user must draw a complete mesh of points based on an image captured by an attached camera. The systems available on the market only present the flexibility and versatility necessary for research involving large investments.

1.2 Problem Description

Although laser vibrometry can be used in vibration analysis in general, it provides a valuable tool for experimental modal analysis of large structures. Experimental modal analysis is a method that describes a structure in terms of its natural characteristics (frequency, damping, and mode shapes).

A standard setup for experimental modal testing and measuring the response due to excitation requires sensor technology (force transducers, accelerometers, cameras, or non-contact laser vibrometers), data acquisition, and a computer for monitoring and analyzing the measured data [2]. When using a laser vibrometer, a single point can be measured; therefore, a scanning system able to sequentially change the measurement point is a valuable tool to reduce the response time of the acquisition task. An automatic point acquisition system would no longer require the user to create the measurement points necessary for using the laser vibrometry technique, simplifying the whole process.

1.3 Objectives

In order to use a laser vibrometer with a scanning feature, it is mandatory to develop an automatic technique to define the measurement mesh with minimal user intervention. This project aims to develop an adaptive scanning system based on 3D shape recognition and 2D contour recognition techniques. For this purpose, a Microsoft Kinect system provides three-dimensional object recognition and identification of the projected geometry. It is supposed to achieve these primary objectives:

- Recognition system capable of tracking the two and three-dimensions of objects under analysis and their contours;
- Graphical user interface allowing data visualization and point cloud editing;
- Automatic measurement point data generation for the laser vibrometer;

These systems give a set of points for a scanner device in an automatic measurement setup. They will be integrated into a single control environment to implement in a programming language. It is expected to achieve a prototype system and consequent experimental validation.

1.4 Document Structure

The present document is divided into six chapters. The current chapter introduces the problem, its objectives, and motivation. Chapter 2 describes the state of the art and published works developed and carried out within the same scope. It also aims to give the reader a better perception and contextualization of the project. Chapter 3 discusses experimental infrastructure and equipment, such as software and hardware used in the project. Chapter 4 reports the proposed solution and the applied methodology. Chapter 5 shows how the tools used in the project were implemented. Chapter 6 describes the experiments and results arising from the tactics of Chapter 4. Chapter 7 sums up all the conclusions drawn from this project and proposes a future work plan.

Chapter 2

Related Work and State of the Art

As technology advances, cameras and models for object detection develop. This chapter will explore how the emergence of these new technologies has influenced the industry and how this topic complements measurement systems. Related work will also be discussed in this chapter.

2.1 Image Acquisition and Processing

Depth cameras are systems that simultaneously capture color and depth information of a scene, resulting in dense point clouds or triangulated meshes [3]. During the development of this project, a camera will capture all the environment and objects shapes, colors, and depth.

Over the years, RGB-D (depth sensor) cameras, like the Microsoft Kinect, greatly impacted Computer Vision, like robotics and image processing. These cameras provide dense depth estimations and color images at a high frame rate, pushing forward several research fields such as 3D reconstruction, camera localization and mapping, gesture and object recognition, and bilateral filtering, among others [4]. One of the most common devices which use this technology is the Microsoft Kinect sensor. The first versions of Kinect cameras work around the Structured Light Principle. Light sectioning and coded structured light projection methods are typical methods used to obtain three-dimensional shapes by projecting light information on objects and using this information for their measurement [5]. Structured light is the projection of a known sequence of light patterns onto an object, which gets deformed by its geometric shape, as Figure 2.1 displays.

Fundamentally, structured-light triangulation requires addressing two basic questions: what patterns to project onto the scene, and how to compute projector-camera stereo correspondences from the images captured [7]. A specific projector or light source, modulated by a spatial light modulator, generates 2D structured illumination. Then the object is observed through a camera from a different direction as three-dimensional information is obtained by analyzing the distortion of the observed speckle pattern, i.e., the disparity from the original projected pattern, causing depth information extractable [6].

A speckle pattern is a random intensity pattern resulting from the mutual interference of many waves of the same frequency but with different phases and amplitudes. It can be observed when a coherent light, such as a laser beam, is reflected or transmitted from a rough surface [8]. Figure 2.2 shows how the Microsoft Kinect produces this pattern.

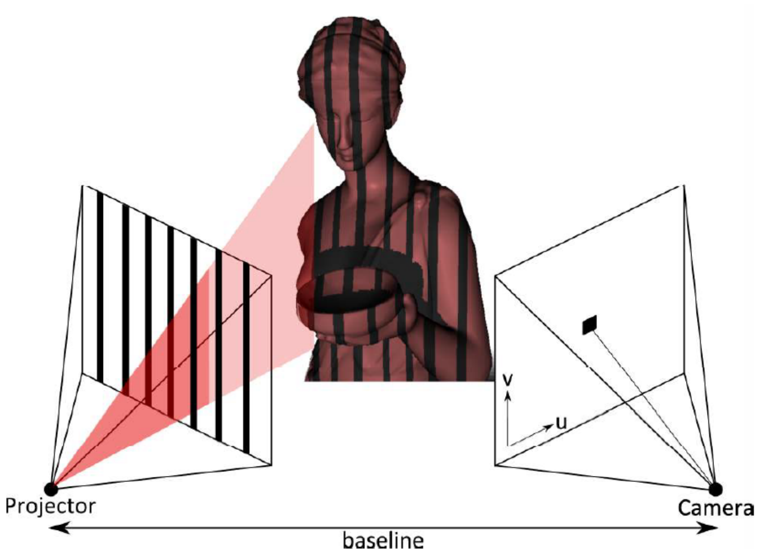


Figure 2.1: Principle of structured light based systems [6].

The infrared (IR) emitter and IR depth sensor distinguish the depth of objects in the field of view of the RGB camera [9]. The IR emitter casts an IR speckle dot pattern into the 3D scene while the IR camera captures the reflected IR speckles [10]. The Microsoft Kinect comprises multiple sensors, an RGB camera, an IR emitter and IR depth sensor, a microphone array for speech recognition and a tilt motor to track the user movements [9]. Figure 2.3 shows the hardware configuration of the Microsoft Kinect sensor.

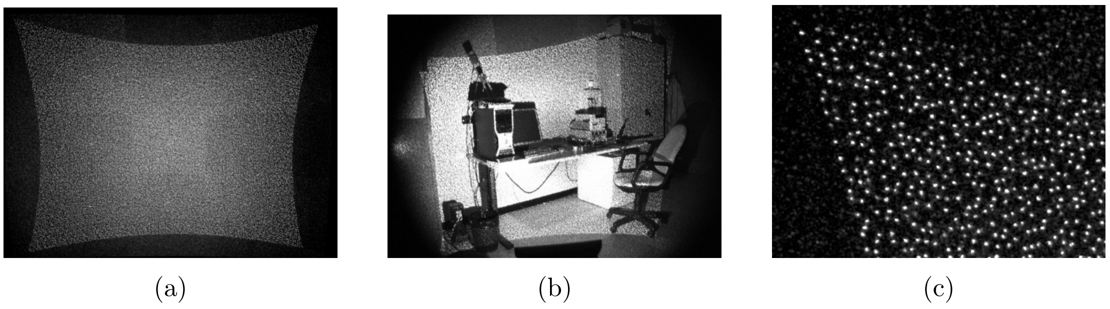


Figure 2.2: The projected pattern on a flat wall (a), a generic scene (b), and a closer view of the speckle pattern (c) [8].

The depth value is encoded with grey values; the darker a pixel, the closer the point is to the camera in space. The black pixels indicate that no depth values are available for those pixels [11]. This might happen if the points are too far (and the depth values cannot be computed accurately) and are too close (there is a blind region due to limited fields of view for the projector and the camera) [11]. It can also happen if the points are in the cast shadow of the projector (there are no IR dots) or reflect poor IR lights (such as hairs or specular surfaces).

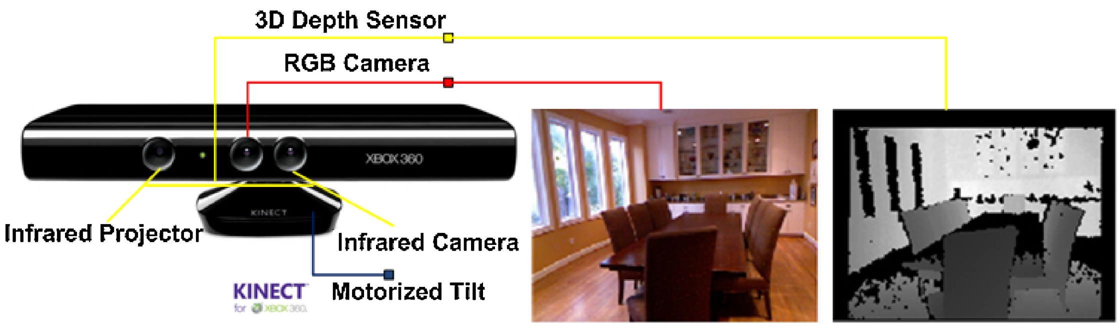


Figure 2.3: Hardware configuration of Microsoft Kinect, and two image samples captured [10].

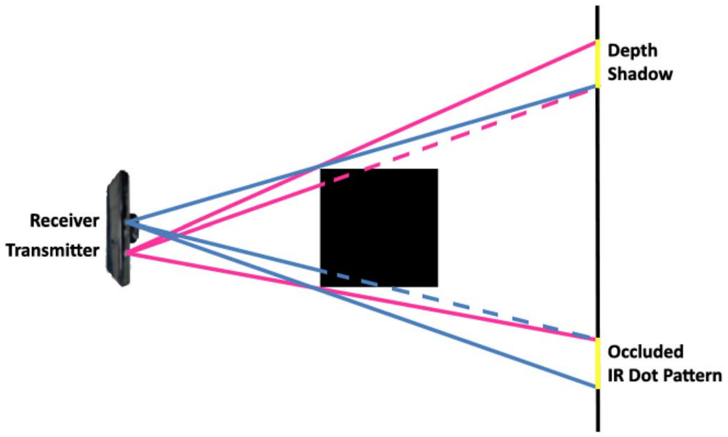


Figure 2.4: A schematic provided to explain the cause of IR dot occlusions and depth shadows [12].

2.2 Laser Vibrometry

A laser vibrometer is a device that allows the user to point a laser beam at a target surface to measure its velocity level in amplitude and phase. It can detect displacement, with a resolution on the order of nanometers, by analyzing the difference in phase between a measurement beam reflected off the object and a reference beam [13]. Experimental modal analysis of light and thin structures uses this non-contact, optical measuring technique that allows non-contact and non-destructive testing and evaluation of the dynamic behaviour of mechanical structures, while providing a fast and flexible measurement performance [14]. Based on the Doppler effect principle, these devices allow point velocity measurement of a vibrating structure through the Doppler phase shift between the incident light and scattered light returning to the measuring instrument [15].

The beam splitter divides the light into the reference and measurement beams. The reference beam points directly at the photodetector, while the measurement beam reaches the test object, where the moving surface scatters the light [16]. Depending on the object's velocity, the backscattered light changes in frequency and phase. Therefore, motion characteristics keep contained in the backscattered light. The superposition of this light with the reference beam creates a modulated detector output signal, revealing the Doppler shift in frequency. This non-contacting transducer will complement the accelerometer when hot, light or rotating surfaces are involved [17].

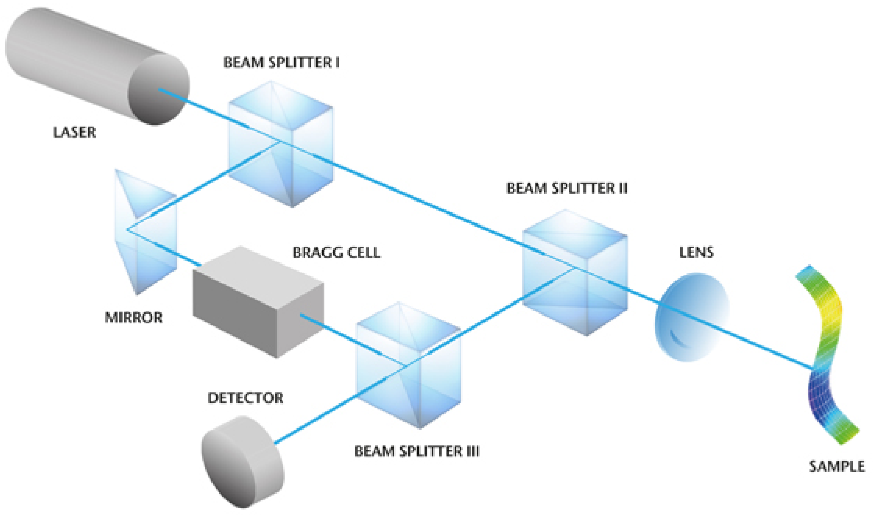


Figure 2.5: Basic measurement principle of vibrometry and setup of a laser Doppler vibrometer [16].

Laser Doppler Vibrometers (LDV) with a scanning system head use a scanner that deflect the laser beam horizontally and vertically. These scanners, responsible for guiding the laser use a pair of mirrors and rotating galvanometers and are controlled by external analog voltage signals. Laser beams can scan different positions on the plane according to different rotation angles of the two plane mirrors about the X and Y axes [18].

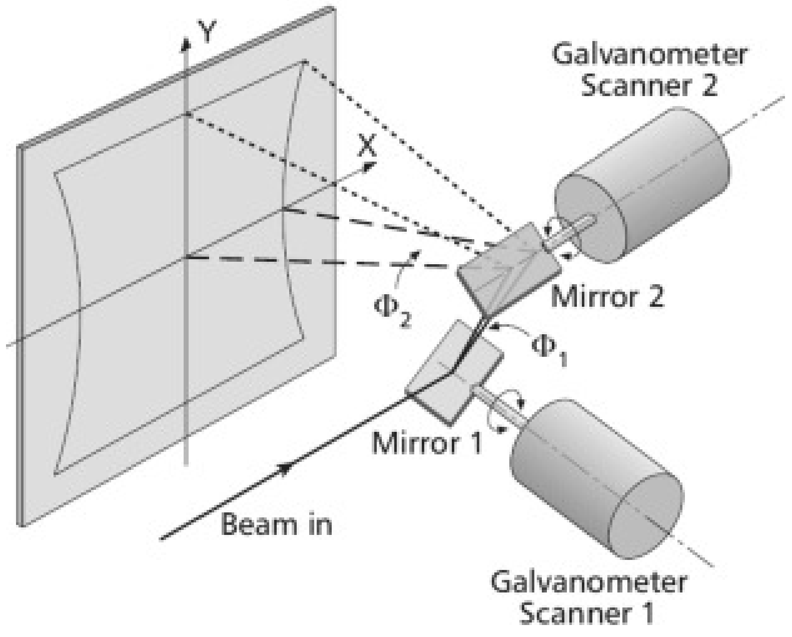


Figure 2.6: Laser scanner mounting configuration [19].

Computer-generated digital commands ultimately control the scanners. Consequently, these commands depend on the scan angles that deflect the laser beam. Through software, analog signals are sent to the LDV, which, in turn, sends the signal to the respective scanner, performing its control [20].

2.3 Industrial Framework

In industrial research and development, studying objects with extensive shapes and sizes, ranging from entire car bodies, airplane parts, and engines to hard drive components, requires a comprehensive and usually time-consuming dynamic characterization task [21]. Laser vibrometry is gaining an important position within this analysis scope. It is vital to measure the dynamic response of the structures when subject to controlled excitation or under the effect of vibration due to regular operation.

Laser vibrometers are predominantly used in numerous applications: automotive, aerospatial, medical, biological, ultrasonic, microstructural, and manufacturing, mainly for vibration measurements, system identification, testing, and damage detection.

The PSV-400 Scanning Vibrometer [22], in Figure 2.7, is a powerful data acquisition platform, from Polytec, that can effortlessly integrate into the engineering workflow.

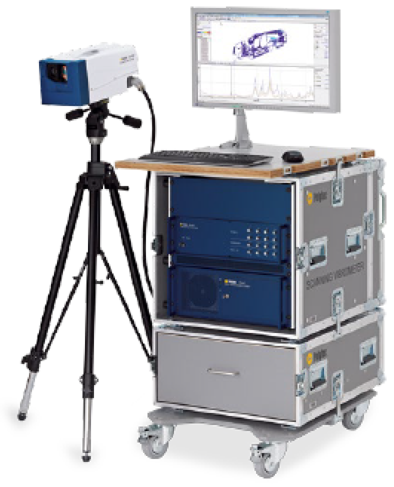


Figure 2.7: Polytec OFV-400 scanning vibrometer [22].

PSV-400 can be configured to meet the application data measurement requirements optimally. A wide range of velocity and displacement decoders combined with matching acquisition hardware guarantees the highest performance. The scanning laser vibrometer head has a built-in video camera and a high-precision scanning unit, which detects the laser beam. Since to start a measurement, placing the scanning head from 80mm up to 30m in front of the object is necessary, the scanning vibrometer head is on a tripod for easy setup. The object will then be visible on the monitor in real time.

2.4 Project Context

This project will begin by creating a controlled scanning system, where a Microsoft Kinect will automatically capture the object points to be analyzed. An automated and sequential scanning will be performed for possible control of the data acquisition system, becoming the primary goal of this work.

Concerning future work, it intends to construct a functional prototype of an automatic scanner for the laser vibrometer existing in the laboratory (Polytec OFV-505/OFV-5000) to continue previously developed work [23].

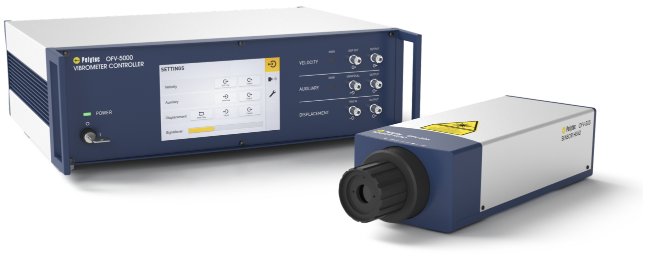


Figure 2.8: OFV-5000 vibrometer controller [24] and OFV-505 sensor head [25].

The Polytec OFV-505, Polytec OFV-series single-point laser vibrometer, is a standard sensor head with multiple lens options featuring stand-off distances ranging from 100mm-300m. This sensor head integrates an interferometer, laser and imaging optics in stable housing. It is also compatible with the Polytec OFV-5000 Modular Vibrometer, capable of reaching its highest signal quality when combined [24].

Tables 2.1 and 2.2 show the primary specifications to operate correctly with the sensor head and the modular vibrometer. Table 2.1 contains the general specifications for the Polytec OFV-505 sensor head. It is possible to verify the laser type and class, its wavelength, and minimum and maximum stand-off distances.

General specifications	
Laser type	Helium Neon (HeNe)
Laser class	Class2, < 1 mW, eye safe
Laser wavelength	633 nm, visible red beam
Focus	Auto, remote and manual focus
Minimum stand-off distance	100 mm
Maximum stand-off distance	300 m
Weight	3.4 kg
Dimensions [W×H×L]	120×80×345 mm

Table 2.1: Polytec OFV-505 specifications. [25]

Table 2.2 contains the prevailing specifications for the Polytec OFV-5000 modular vibrometer. The analog and digital signal outputs, frequency range, and maximum velocity are listed.

General specifications	
Analog signal outputs	BNC, ± 10V: Velocity signal Displacement signal AUX output DSP output (velocity with DSP filter)
Digital signal output	S/P-DIF standard 24 bit 48/96 kSa/s
Frequency range	DC to 24 MHz
Max velocity	± 10 m/s
Power supply	100...240 VAC ±10%, 50/60 Hz
Power Consumption	max. 100 VA
Weight	10kg
Dimensions [W×D×H]	450×360×150 mm

Table 2.2: Polytec OFV-5000 specifications. [24]

Intentionally blank page.

Chapter 3

Experimental Infrastructure and Equipments

As the project developed, various pieces of equipment and software served as a tool to carry on with it. This chapter briefly describes the hardware equipment and software to manipulate and visualise the data.

3.1 Hardware

Regarding hardware, the Microsoft Kinect camera was the tool used. This device provides data acquisition for the developed systems simply and affordably.

3.1.1 Microsoft Kinect

In order to acquire and analyse images, a Kinect camera from Microsoft was applied in this project. Since it is small-sized, Microsoft Kinect rapidly recognized its usefulness as a 3D depth sensor within many third-party applications [12]. It has applications for medical purposes, language learning, and even partying outdoors. Figure 3.1 shows an example of this device.

Figure 3.1: Kinect V1.

As shown in Figure 3.2, this device contains an RGB color video camera, a depth sensor, and a microphone array. The Kinect sensor lets the computer directly sense the players depth (third dimension) and the environment.

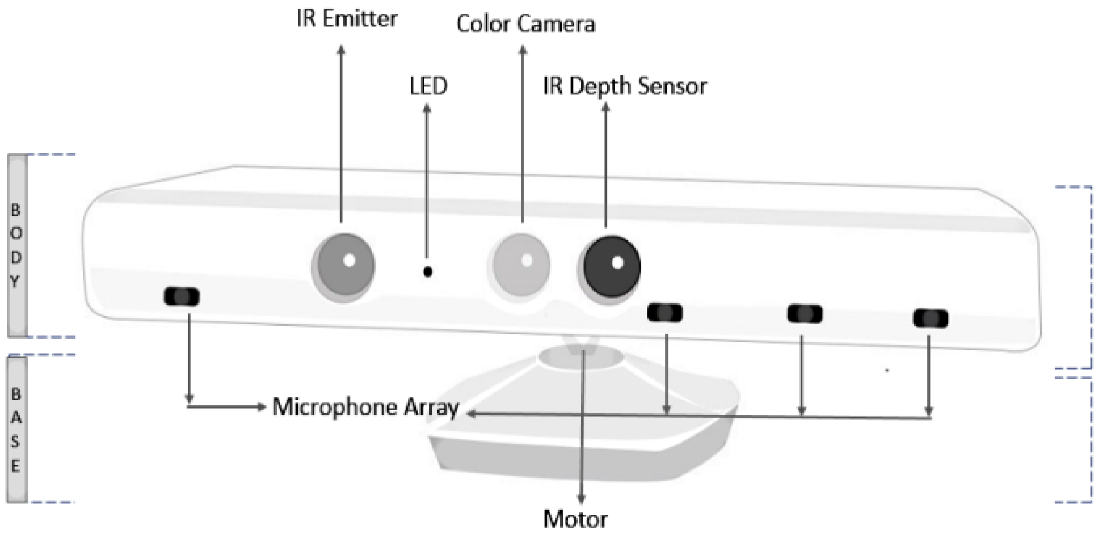


Figure 3.2: Schematic for Kinect V1. [9]

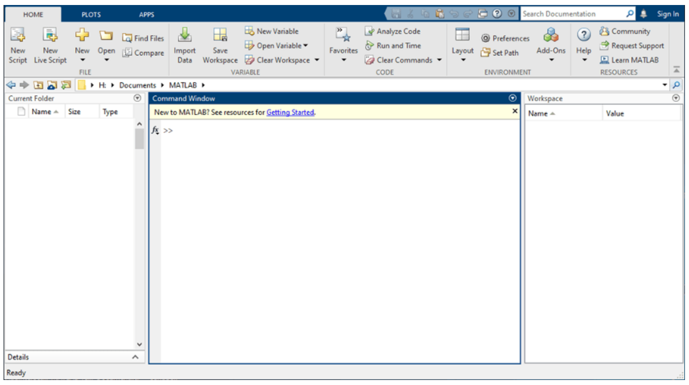
The camera of the device detects the three color models (red, green, and blue) and body shapes to differentiate it from other things. It has a pixel resolution of 640 x 480 and a frame rate of 30 fps. It also contains a stack of signal processing hardware that recognizes all the data generated by the cameras, IR light, and microphones. By combining the output from these sensors, a program can track and identify objects in front of it, determine the direction of sound signals, and isolate them from background noise. The IR projector helps creating a 3D image of the environment.

3.2 Software

The Matlab tools were used to carry out all the system's programming, which provided a large library allowing the Kinect control - offering all the manipulation of the system in a favourable environment for research.

3.2.1 Matlab

The matrix-based Matlab language is the most natural way to express computational mathematics. It can run analyses on larger data sets and scale up to clusters and clouds. Built-in graphics make it easy to visualize and gain insights from data. These Matlab tools and capabilities are rigorously tested and designed to work together. Matlab code can be integrated with other languages, enabling deploy algorithms and applications within web, enterprise, and production systems.



New to MATLAB? See resources for Getting Started.

```
fx >>
```

Figure 3.3: Matlab home page.

In order to implement all the code in the Matlab environment, it was necessary to install several toolboxes. These toolboxes were:

- Image Acquisition Toolbox Support Package for Kinect For Windows Sensor [26];
- Image Processing Toolbox;
- Image Acquisition Toolbox;
- Computer Vision Toolbox;

3.2.2 App Designer

App Designer is an interactive development environment for designing application layouts and programming its behavior. App Designer allows the creation of apps and the control of various visual components to lay out the graphical user interface (GUI) design.

To develop the data manipulation and visualisation application, App Designer was selected for building apps in Matlab since GUIDE (GUI Design Environment) will be removed in future releases of Matlab [27] and replaced by App Designer.

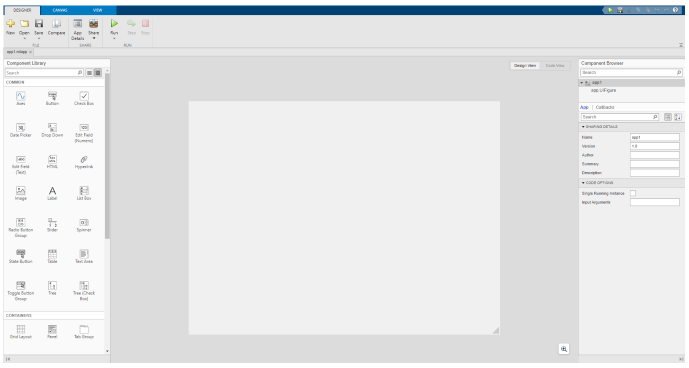


Figure 3.4: Matlab AppDesigner home page.

Intentionally blank page.

Chapter 4

Proposed Solution and Methodology

The following chapter discusses the solution proposed for the project realization and the methodology applied. At first, a solution overview presented for solving the problem shows the connection between each element and the solution workflow. Next, and through examples and explanations, all image acquisition functioning is made clear. Finally, it explains how all the methods converge into a Matlab application for data manipulation and visualization.

4.1 Solution Overview

To better understand and perceive the solution, Figure 4.1 displays a flowchart showing the steps in the proposed solution.

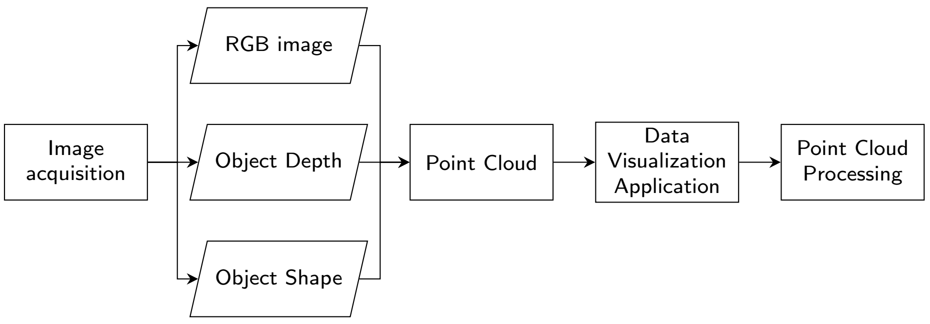


Figure 4.1: Proposed solution flowchart.

As presented in Figure 4.1, image acquisition is the initial stage of the whole project. In this stage, a Kinect camera will capture RGB and depth images of the environment and the object under analysis. This way, detecting the object's shape and depth is made possible. In a Matlab application, with the collected data, it is possible to create and process the point clouds. With this done, it is expectable to reach a final point cloud configuration for the laser vibrometer scanner.

4.2 Image Acquisition

4.2.1 Kinect Data

To be able to create a cloud of points, it is necessary to capture RGB and depth images from the Kinect. For that, it requires a connection to the RGB and the depth camera, responsible for capturing RGB and depth images, respectively. Creating a video object for both cameras was the next step. With both images and objects created, the pcfromkinect [28] function can create a point cloud of the environment captured by the Kinect camera, as Figure 4.2 shows.

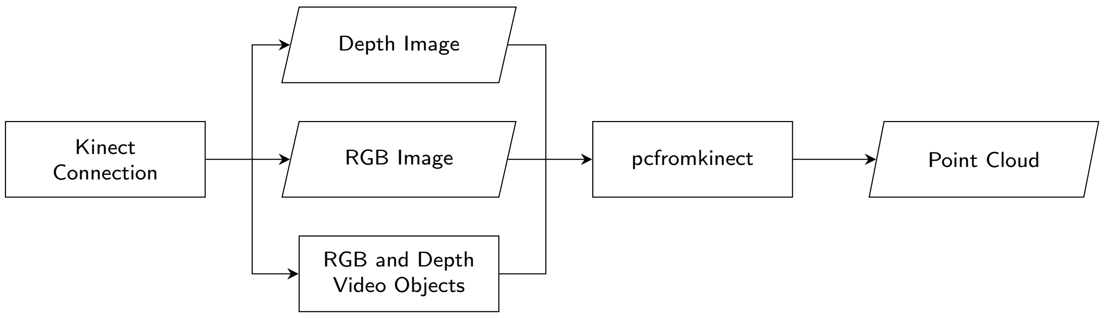


Figure 4.2: Point cloud flowchart.

Generating a target area allows excessive data removal from the environment point cloud. Extracting all data inside this area creates a new point cloud only containing the object. This technique will extract the object under analysis into a new point cloud where it can be processed and treated much easier, as Figure 4.3 shows.

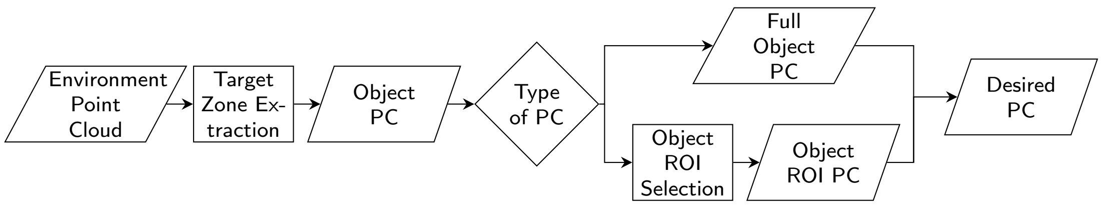


Figure 4.3: Point cloud filtering flowchart.

The number of points inside this new point cloud will reduce without losing the object geometry, so fewer points will be exported to the scanner controller.

4.2.2 Contours

The identified contours provide the object's shape under analysis and facilitate the laser's zone of action. With the object point cloud created, it was possible to remove its third dimension (depth), projecting the object onto a bidimensional space.

With the two-dimensional projection, we proceeded in the following steps to obtain the contours, as Figure 4.4 shows. After following the previous method, detecting the object's silhouette makes it possible to visualize its shape.

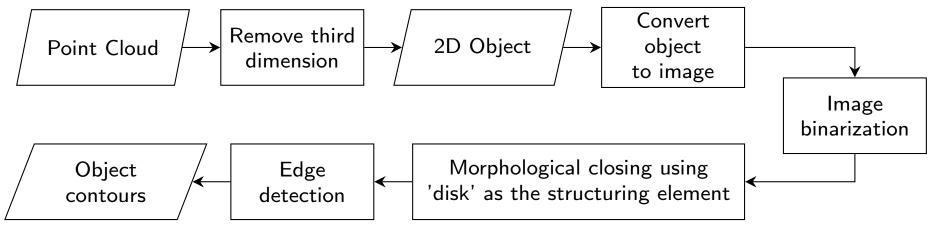


Figure 4.4: Contour flowchart.

4.2.3 Point Cloud Downsampling

Since the points collected would get exported, several methods tested the point cloud reduction while preserving the object's geometry, thus easing the exportation process. Choosing one method from Figure 4.5 enables the downsampling process of the point cloud created. The three methods exemplified in Figure 4.5 reduce the number of points in the cloud, and each method performs a different reduction process. This reduction decreases the number of points exported to the scanner controller.

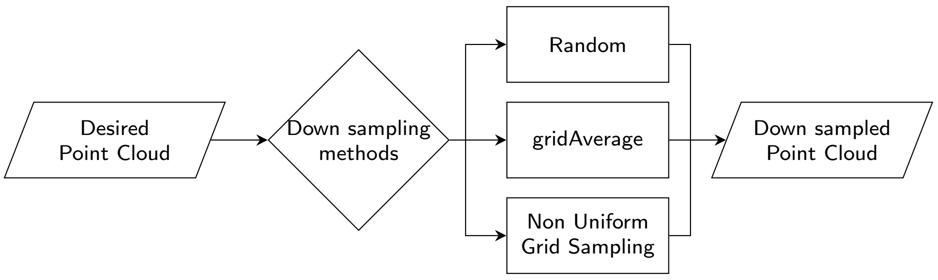


Figure 4.5: Point cloud downsampling flowchart.

The resulting point cloud will then be processed in a specific tab inside the developed application to edit the integrity of the downsampled point cloud.

4.3 AppDesigner Interface

With all the strategies defined in the steps previously demonstrated, it is necessary to implement them in the Matlab environment. Conceiving an application through a graphical user interface provided by Matlab makes that possible. Thus, we can then break up the application into two significant steps:

- Image and data acquisition;
- Processing and treatment of the point cloud;

4.3.1 Image and Data Acquisition

In this application section, the user can acquire and visualize a vast amount of data captured from the Kinect camera. Data such as video streams, environment point clouds, object point clouds and object contours can be analyzed.

The first step in operating the application is to start the video streams. These streams will capture RGB and depth images to create the environment point cloud. With the environment point cloud created, it is now possible to insert the target window limits (X, Y and Z) to extract the object for analysis. The application allows the user to manipulate the object cloud, allowing the selection of a specific region for analysis or leaving the entire object to be processed by the next steps of the application. Once the user has finished selecting the area of the object to be analyzed, it is possible to remove its contours through the application.

In Figure 4.6, a flowchart summarizes all the earlier steps.

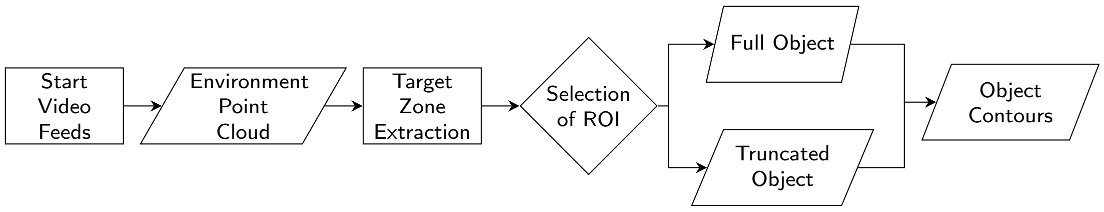


Figure 4.6: Image Acquisition Section Flowchart.

4.3.2 Point Cloud Processing and Treatment

With the point cloud created in 4.3.1, we now move on to the analysis and processing of this cloud. This application section essentially allows point reduction. Therefore, it allows the user to cut out areas of the cloud for point reduction and eliminate, densify and compress the number of points in zones predefined by the user, or even a combination of all these options.

To make it easier to understand, the flowchart in Figure 4.7 shows the direction to follow for the treatment and processing of the point cloud.

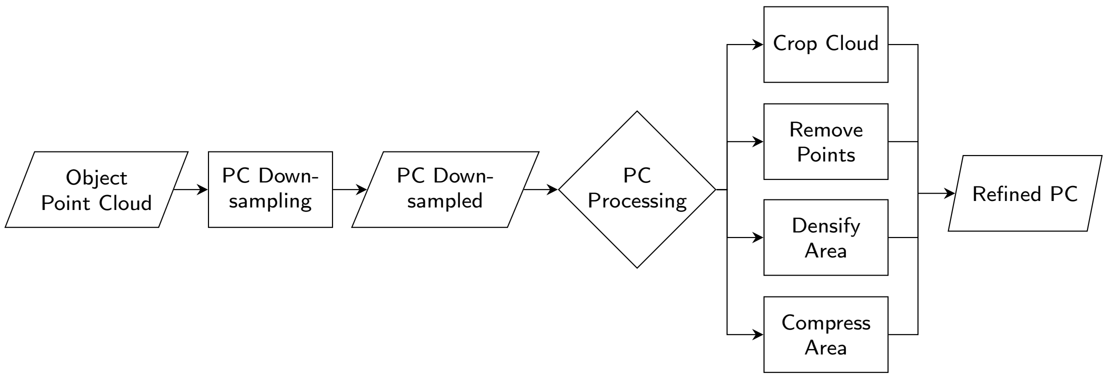


Figure 4.7: Point Cloud (PC) Processing and Treatment Flowchart.

Chapter 5

Tool Implementation

In the current chapter, are known the steps that made possible the development of the system that will construct and process the point matrices. Here it will be demonstrated how the implementation of the tools used was performed.

5.1 Acquiring Data with Kinect

Initially described in subsection 4.2.1, acquiring an RGB and depth image was required to create a point cloud. However, a video object for RGB and depth devices was also needed before acquiring those images.

5.1.1 Video Objects

To create the video object, the videoinput [29] function was implemented, and as described in Code 5.1, both objects were created.

```
%% CAMERA CONNECTION
% RGB CAM
RGBDev = videoinput('kinect',1,'RGB_640x480');

% DEPTH CAM
DEPTHDev = videoinput('kinect',2,'Depth_640x480');
```

Code 5.1: Matlab code to create RGB and Depth video objects.

The inputs of the functions described in Code 5.1 allow the access to the RGB and depth cameras, with 640×480 pixel resolution, thus creating the necessary video objects.

5.1.2 Images Acquisition

With both video objects created, it was possible to command the Kinect to start capturing images. Both images acquired by the Kinect use the Matlab function getsnapshot [30], which takes a single frame from the video objects. Code 5.2 describes the Matlab function to get the desired frames. These two functions acquire both images: the image from the RGB camera and the depth matrix data from the depth device. It was only necessary to insert the video objects as input to get the images shown in Figure 5.1, used to create the environment point cloud.

```
%% ACQUIRE IMAGES FROM BOTH VIDEO OBJECTS
RGBFrame = getsnapshot(RGBDev);
DEPTHFrame = getsnapshot(DEPTHDev);
```

Code 5.2: Matlab code to obtain both RGB and Depth frames.

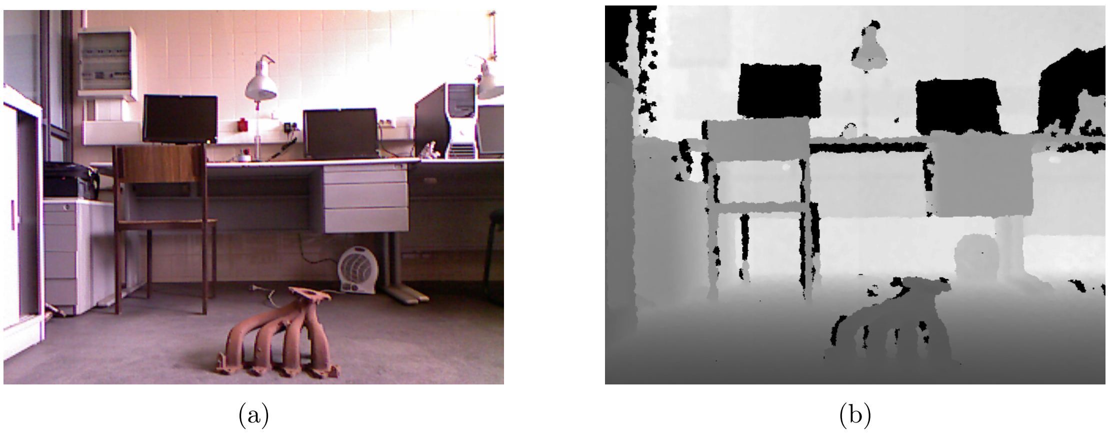


Figure 5.1: RGB image (a) and depth data (b) acquired by Kinect

The frame acquired in Figure 5.1 (b) shows the depth values of each point in a grayscale range, which means that darker colors represent points closer to the Kinect. Pitch black pixels are values the Kinect camera could not capture, either by being out of range, on reflective or transparent surfaces or simply by the Kinect malfunctioning.

5.1.3 Environment Point Cloud

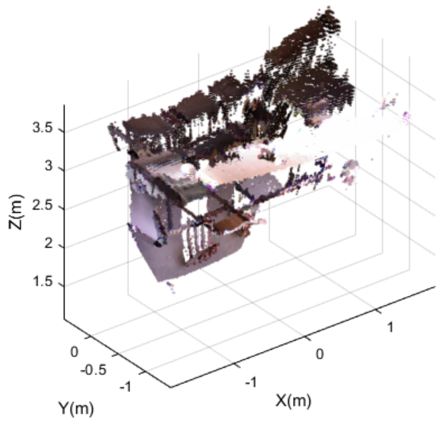
It was necessary to implement the pcfromkinect [28] function to create the point cloud, which only needs one video object, the depth video object, and both frames captured, as Code 5.3 shows.

```
%% CREATE POINT CLOUD
PtCloud = pcfromkinect(DEPTHDev,DEPTHFrame,RGBFrame);
```

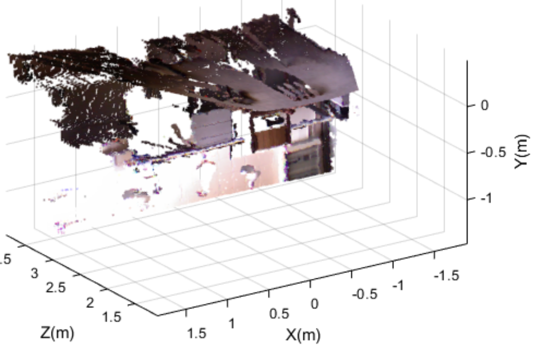
Code 5.3: Matlab code to create point cloud.

After creating the point cloud, it is necessary to visualise it. It is possible to visualise the point cloud using the pcshow [31] function. Here is required to introduce some input arguments to visualise the point cloud correctly. The first input is the point cloud data. Then it is necessary to define the vertical axis and its direction to represent the field of view of the Kinect correctly. In Figure 5.2 (a), the point cloud is rotated and does not represent the real scenario captured by Kinect, and in Figure 5.2 (b), the point cloud is upside down from what it should look. It is, therefore, essential to define the correct representation referential.

Succeeding the code implementation in Figure 5.3 (b), the point cloud is in the desired position and shows what the Kinect camera is acquiring. Figure 5.3 (a) shows some negative values for both the X and Y axis since they represent their position from



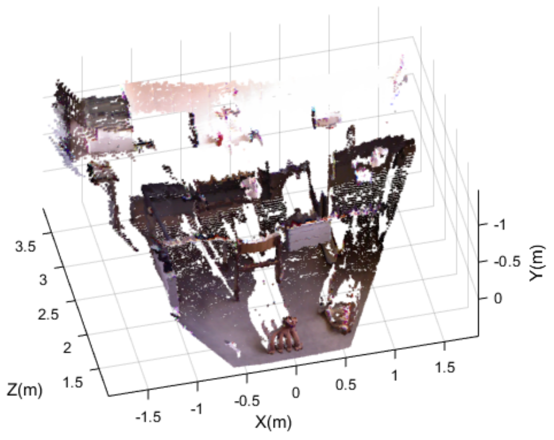
(a)



(b)

Figure 5.2: Point clouds without any input arguments besides the point cloud created (a) and with Y axis assigned as the vertical axis (b).

the centre of the Kinect. Once the direction of the Y axis is changed, its representation changes according to the updated referential.



(a)

```
% POINT CLOUD VISUALIZATION
figure
pcshow(ptCloud,'VerticalAxis','Y',VerticalAxisDir='Down');
xlabel('X(m)')
ylabel('Y(m)')
zlabel('Z(m)')
```

(b)

Figure 5.3: Point cloud assigning Y as the inverted vertical axis (a) and the Matlab code implementation for the representation of the updated coordinate system (b).

In the code shown in Figure 5.3 (b), "Down" was chosen as the vertical axis direction. If not, the point cloud assumed the configuration depicted in Figure 5.2 (b). Therefore

Figure 5.4 (a) shows the original axis representation of the point cloud, and Figure 5.4 (b) shows the final representation of the Kinect coordinate system.

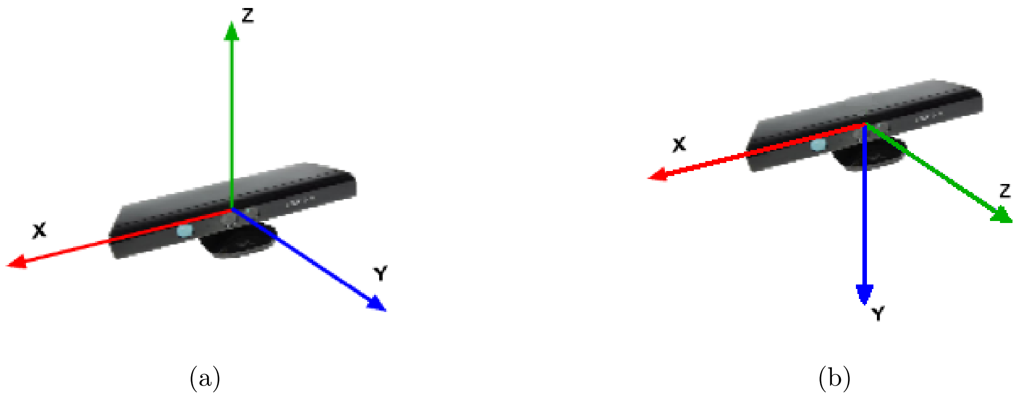


Figure 5.4: Kinect initial coordinate system (a) and corrected coordinate system after code implementation (b).

With the Kinect coordinate system set, the values on the X axis tell if the object is on the right or left side relative to the centre of the Kinect device. Therefore, when the object is on the left side of the Kinect centre, its points have negative values, and positive if the object is on its right side. On the Y axis, negative values mean the object is above the Kinect horizon line and positive values if located below that line. Values on the Z axis correspond to the object's perpendicular distance to the Kinect's vertical plane.

5.1.4 Target Area

Extracting some information is necessary to define the cloud to get the object, excluding all unnecessary information. Inside the point cloud produced in Figure 5.3 (a) is created a cube-shaped target area, where all the points inside are extracted into a new point cloud data matrix. For this task, it is essential to define the limits of this region of interest, as shown in Code 5.4, delimiting the volume occupied by the object.

```
%% DEFINE CUBIC ROI AS TARGET AREA
roi = [Xmin Xmax Ymin Ymax Zmin Zmax];
```

Code 5.4: Six-element array created defining all axis limits.

The function findPointsInROI [32] shown in Code 5.5 finds points in the initial cloud within the region of interest, with the limits defined earlier. The output of this function is a matrix of indexes corresponding to the stored points.

```
% FIND POINTS OF THE ORIGINAL CLOUD WITHIN ROI PREVIOUSLY DEFINED
indices = findPointsInROI(ptCloud,roi);
```

Code 5.5: Matlab function to find points within a region of interest in the point cloud [32].

Once the points are identified, selecting them from the original point cloud is possible. The function select [33] returns a point cloud object containing only the selected points. Setting the limits in Code 5.4 correctly, the selected points in the new point cloud match the object's point cloud. Then, it is possible to obtain the desired point cloud using the code from Code 5.6.

```
% SAVE THE POINTS INTO A NEW POINT CLOUD
ptTargetArea = select(ptCloud,indices);
```

Code 5.6: Matlab function to select points in point cloud [33].

In order to make the target area visible, the cube, formed by the area boundaries, is drawn. Both Code 5.7 (a) and (b) illustrate how the limits defined in Code 5.4 determine the vertices and faces of the cube, respectively.

```
% CREATE THE COORDINATES OF THE CUBE'S VERTICES
p1 = [Xmin Ymax Zmax]';
p2 = [Xmin Ymax Zmin]';
p3 = [Xmax Ymax Zmin]';
p4 = [Xmax Ymax Zmax]';
p5 = [Xmin Ymin Zmax]';
p6 = [Xmin Ymin Zmin]';
p7 = [Xmax Ymin Zmin]';
p8 = [Xmax Ymin Zmax]';
```

(a)

```
% CREATE THE FACES OF THE CUBE
Cube = [p1 p2 p3 p4;
        p4 p3 p7 p8;
        p7 p8 p5 p6;
        p7 p6 p2 p3;
        p6 p5 p1 p2;
        p8 p5 p1 p4];
```

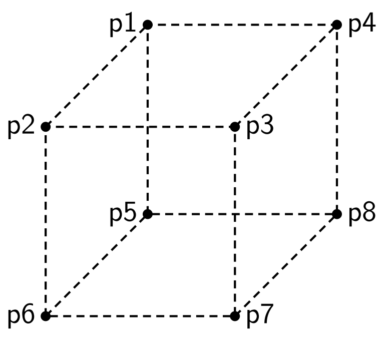
(b)

Code 5.7: Configuration of target area vertices (a) and faces (b).

By applying the code from Figure 5.5 (a), it is possible to display the cube created, specifying the color and transparency of its faces and the line type of its edges. Figure 5.5 (b) exemplifies configured target area.

```
% DISPLAY THE TARGET AREA WITH COLOR AND
    TRANSPARENCY SET
p=fill3(Cube(1,:),Cube(2,:),Cube(3,:),'--w');
p.FaceAlpha = 0.3;
p=fill3(Cube(4,:),Cube(5,:),Cube(6,:),'--w');
p.FaceAlpha = 0.3;
p=fill3(Cube(7,:),Cube(8,:),Cube(9,:),'--w');
p.FaceAlpha = 0.3;
p=fill3(Cube(10,:),Cube(11,:),Cube(12,:),'--w');
p.FaceAlpha = 0.3;
p=fill3(Cube(13,:),Cube(14,:),Cube(15,:),'--w');
p.FaceAlpha = 0.3;
p=fill3(Cube(16,:),Cube(17,:),Cube(18,:),'--w');
p.FaceAlpha = 0.3;
```

(a)



(b)

Figure 5.5: Matlab code to display and configurate the target area (a) and visual representation of that configuration (b).

As a result, in Figure 5.6 (a), the selected points are marked in red, and in Figure 5.6 (b), the target area that delimits the selected points is represented.

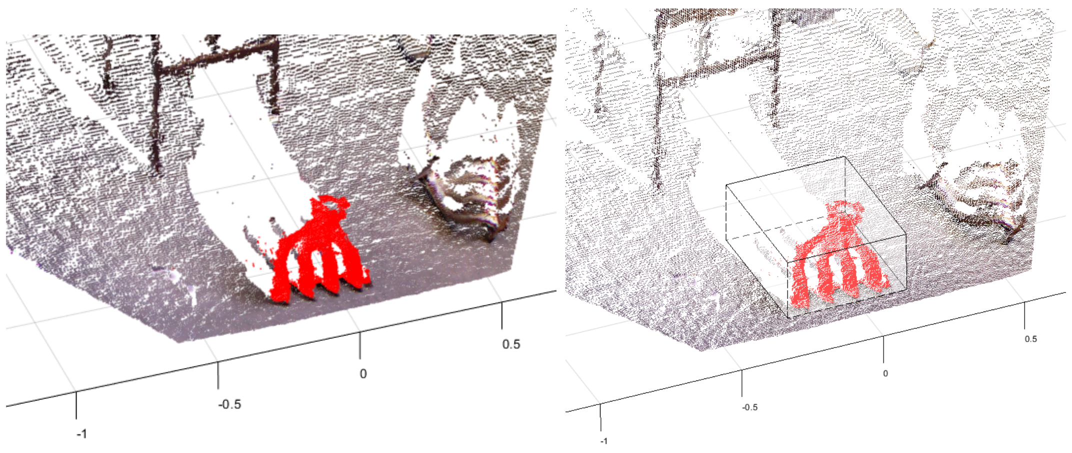


Figure 5.6: Original point cloud with selected points marked in red (a) and the target area deployed (b).

5.1.5 Object Point Cloud

With the help of the target area, it is now possible to extract data from some regions of the cloud into a new data matrix. It is only necessary to define the boundaries of the target area, to create this new point cloud within a specific area. This solution allows easy point extraction and removes any cloud disturbances. In Figure 5.7, it is possible to perceive the target area created after defining the X, Y, and Z limits.

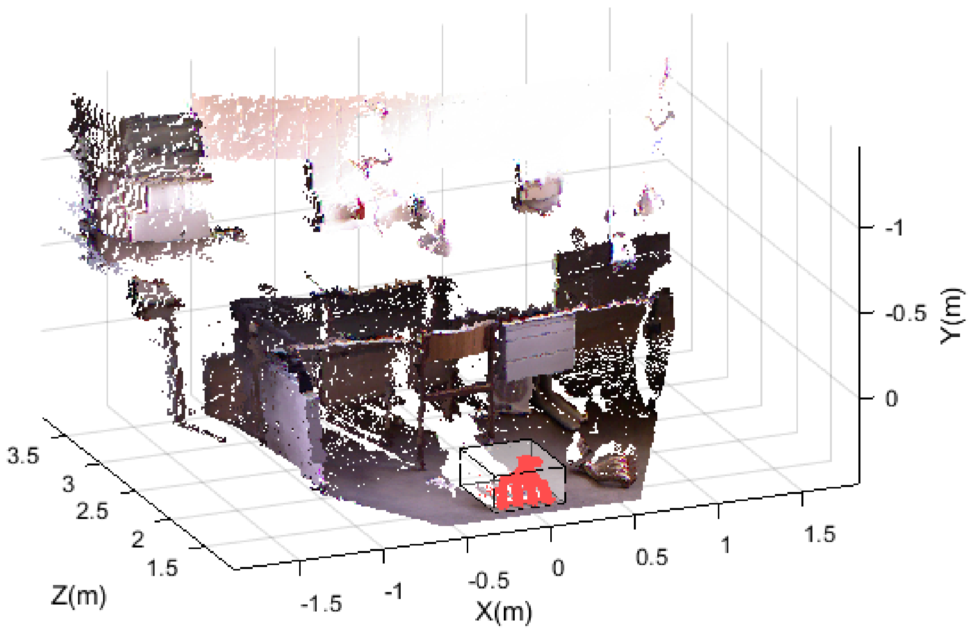


Figure 5.7: Environment point cloud with target area.

After completing the data extraction, the object point cloud is created. The same procedure in Figure 5.3 (b) makes this point cloud visible. Adopting the point cloud created in Code 5.6 as the first and only changeable input is possible to see the object point cloud, as depicted in Figure 5.8.

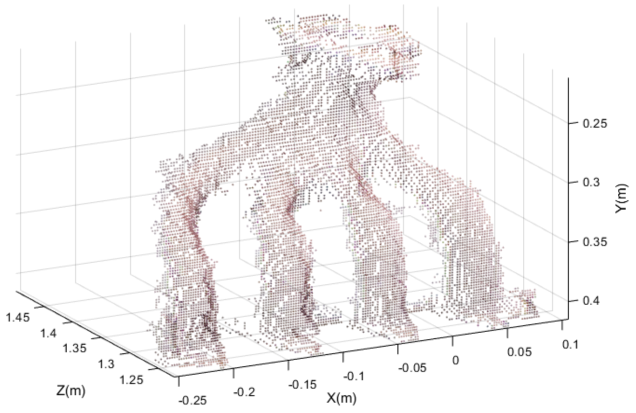


Figure 5.8: Object point cloud extracted from target area.

The points of the object, taken from Figure 5.8, reveal the object's location relative to the Kinect's centre. It means that with these points, it is possible to know the object's approximate position in RGB images similar to Figure 5.1 (a). Thus, to only select a particular part of the object cloud, a figure is opened that shows only the part of the RGB image where the object is, shown in Figure 5.9.

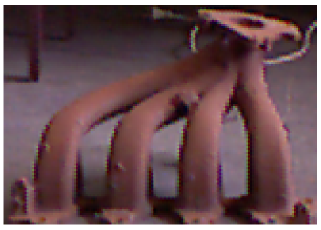


Figure 5.9: RGB image of the object.

On top of this image, it is possible to draw a rectangle to define the area taken from the cloud. Figure 5.10 shows examples of regions of interest created.

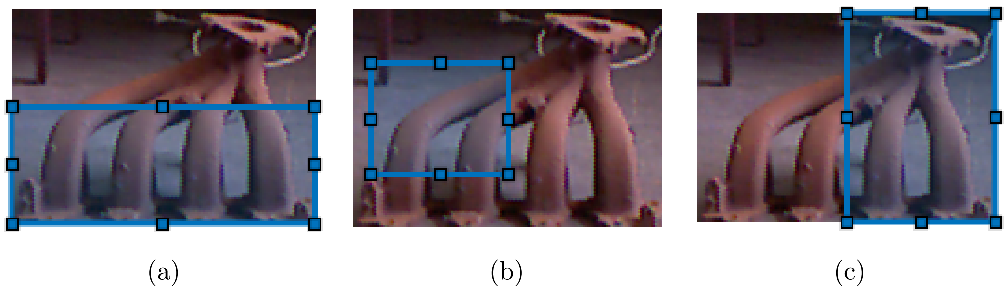


Figure 5.10: Object regions of interest selections (a), (b) and (c).

It is initially necessary to define the rectangle corners to create a new point cloud with only the selected part. Similar to the target area in the environment point cloud, this process takes a part of the original cloud to create a new cloud, as shown in Figure 5.11.

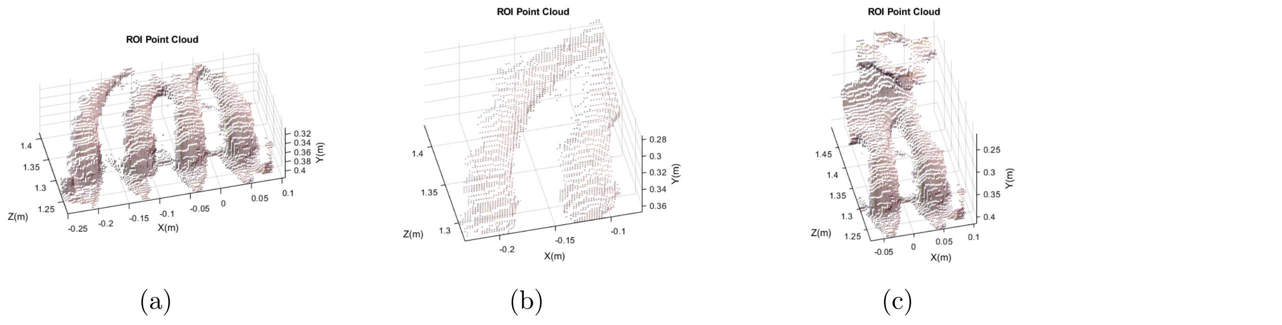


Figure 5.11: Object region of interest point clouds (a), (b), and (c).

5.1.6 Contours

With the point cloud created and processed, it is possible to start the contour treatment. As described in 4.2.2, the third dimension (depth) was removed from the object turning the object into a two-dimensional set of points, as represented in Figure 5.12.

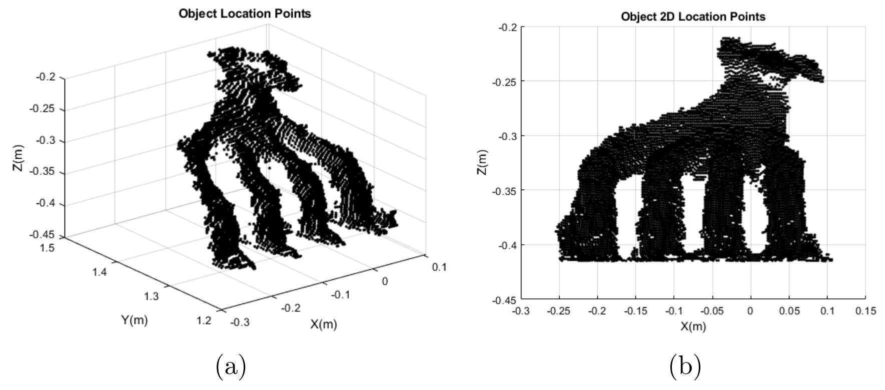
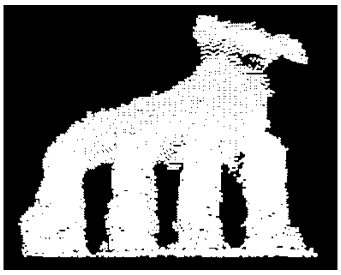


Figure 5.12: Three-dimensional (a) and two-dimensional object points (b).

Now it was necessary to convert the set of points into an image to proceed to the image processing, which enables contour detection. Next, this image was binarized, and its colors inverted, resulting in Figure 5.13.

In order to close all the gaps in the image, the function imclose [34] was executed. This function performs morphological closing on the binary image using a structuring element. In this case, "disk" was used as the structuring element, and its results are visible in Figure 5.14 (b) and (a), respectively.

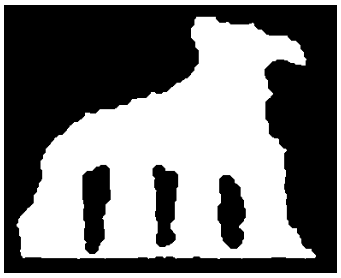
With all the gaps closed, correct contour detection was possible. Here another function is implemented to obtain the contours. The edge function [35] finds edges in two-dimensional grayscale images using an edge detection method. Figure 5.15 (b) shows the function implementation, using canny as the edge detection method, and Figure 5.15 (a) illustrates its results.



```
% IMAGE BINARIZATION AND COLOR INVERSION
ImgBin = imbinarize(im2gray(imgFramePlot));
InverseImg = uint8(255) - im2uint8(ImgBin);
```

(a) (b)

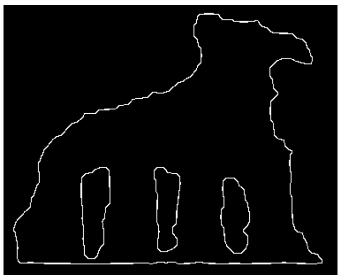
Figure 5.13: Object set of points converted into an image, binarized and with colors inverted (a) and the Matlab code used (b).



```
% MORPHOLOGICAL CLOSING USING DISK
% AS THE STRUCTURING ELEMENT
se = strel('disk',disk_radius);
ImgClosed = imclose(InverseImg,se);
```

(a) (b)

Figure 5.14: Object silhouette after morphological closing (a), and the Matlab code (b).



```
% FIND CONTOURS
ImgEdge = edge(ImgClosed,'canny');
```

(a) (b)

Figure 5.15: Object edge detection (a) using Matlab function (b).

It is also possible to do all these processes using an ROI object point cloud, similar to Figure 5.11, instead of turning the whole object point cloud into a two-dimensional set of points. Then redo the same processes until edges are detected, similar to Figure 5.15 (a). This hypothesis will be discussed and presented in Section 5.2.1.

5.1.7 Point Cloud Downsample

The size of the point cloud, acquired by the Kinect camera and post-processed by the procedures described above, is too large for the final purpose of this project. A complete vibrometry analysis only requires a limited yet representative set of points. Therefore, a strategic and controlled reduction of the cloud of points is essential at this stage. A downsample [36] function is implemented to reduce the number of points acquired and preserve the object structure. Three options were available to downsample the point cloud.

Random Downsampling

By using Code 5.8, it is possible to create a downsampled point cloud using the random method, being the inputs, the object point cloud, the downsample method and, in this case, the percentage. The percentage input in the downsample function specifies the percentage returned from the original point cloud.

```
% CLOUD DOWNSAMPLING USING RANDOM METHOD
ptRndm = pcdownsample(ptObject,'random',percentage);
```

Code 5.8: Matlab code to downsample the point cloud using the random method.

From the analysis of Figure 5.16, random downsampling [36] returns a percentage from the original point cloud.

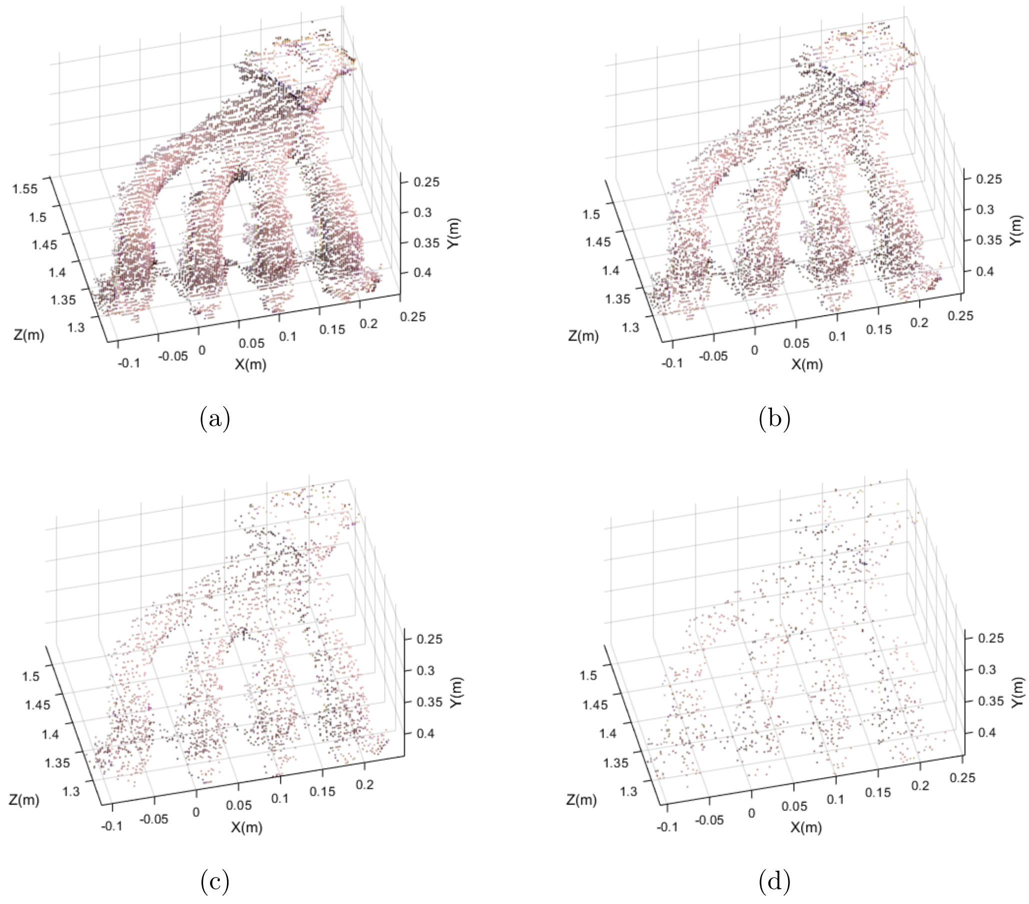


Figure 5.16: Point cloud random downsampled with 75% (a), 50% (b), 25% (c) and 10% (d) of the original point cloud returned.

GridAverage Downsampling

The program lines from Code 5.9 downsample the point cloud using the gridAverage method. This function returns a downsampled version of the input point cloud with points depending on the gridStep size. GridStep is the input of this function and specifies the size of the 3D bounding box.

```
% CLOUD DOWNSAMPLING USING GRIDAVERAGE METHOD
ptGridAvgDownS = pcdownsample(ptObject,'gridAverage',gridStep);
```

Code 5.9: Matlab code to downsample the point cloud using gridAverage method.

GridAverage downsampling [36] uses a box grid filter to downsample the point cloud. Inside this bounding box, defined by GridStep, all points are merged into a single point, as seen in Figure 5.17. When increasing the gridStep value, the bounding box increases; consequently, the number of points merged into a single point increases.

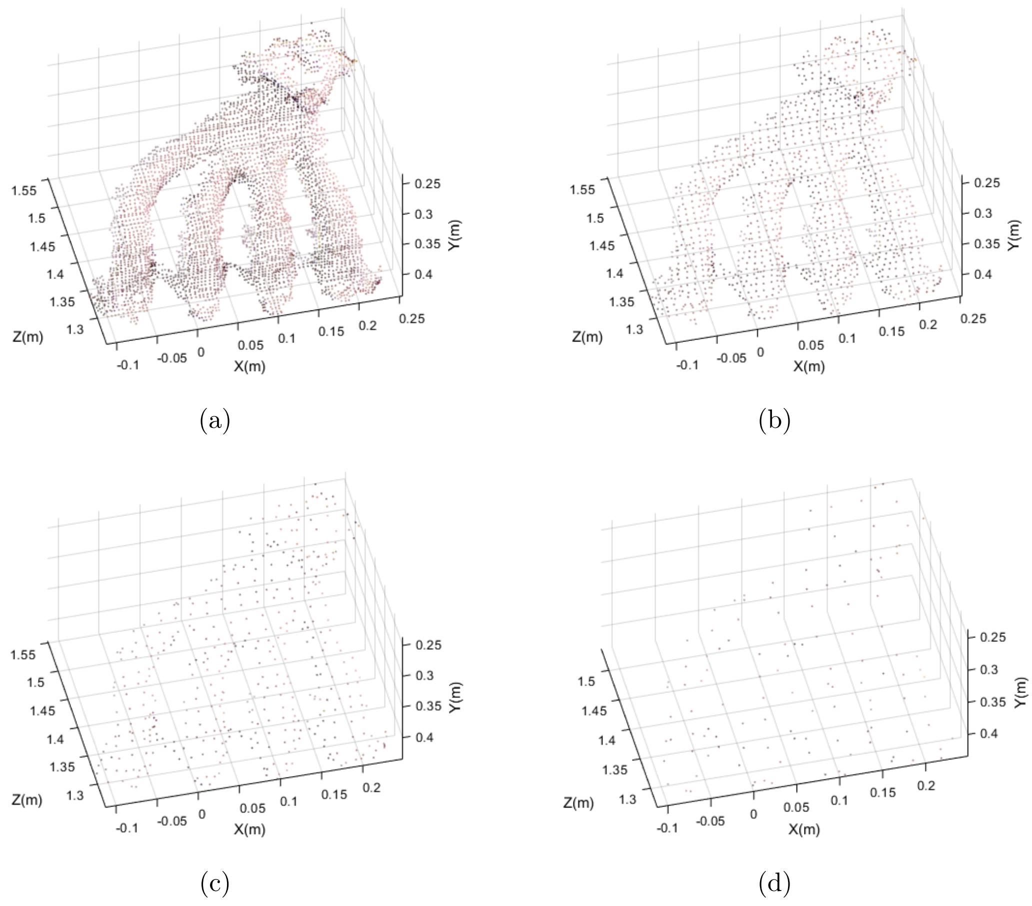


Figure 5.17: Point cloud gridAverage downsampled with box grid filter size of 0.005 (a), 0.01 (b), 0.02 (c) and 0.04 (d).

NonUniformGridSample downsampling

The last method is the NonUniformGridSample, given in Code 5.10, where the function outputs the downsampled point cloud according to the maxNumPoints. The input maxNumPoints defines the maximum number of points inside the box grid filter.

```
% CLOUD DOWNSAMPLING USING NONUNIFORMGRIDSAMPLE METHOD
ptNonUnifDownS = pcdownsample(ptObject,'nonuniformGridSample',maxNumPoints);
```

Code 5.10: Code to downsample the point cloud using NonUniformGridSample method.

NonUniformGridSample downsampling [36] returns a downsampled point cloud using a NonUniform box grid filter. This function draws the box grid filter in each point and randomly selects a single point from each box, as exemplified in Figure 5.18.

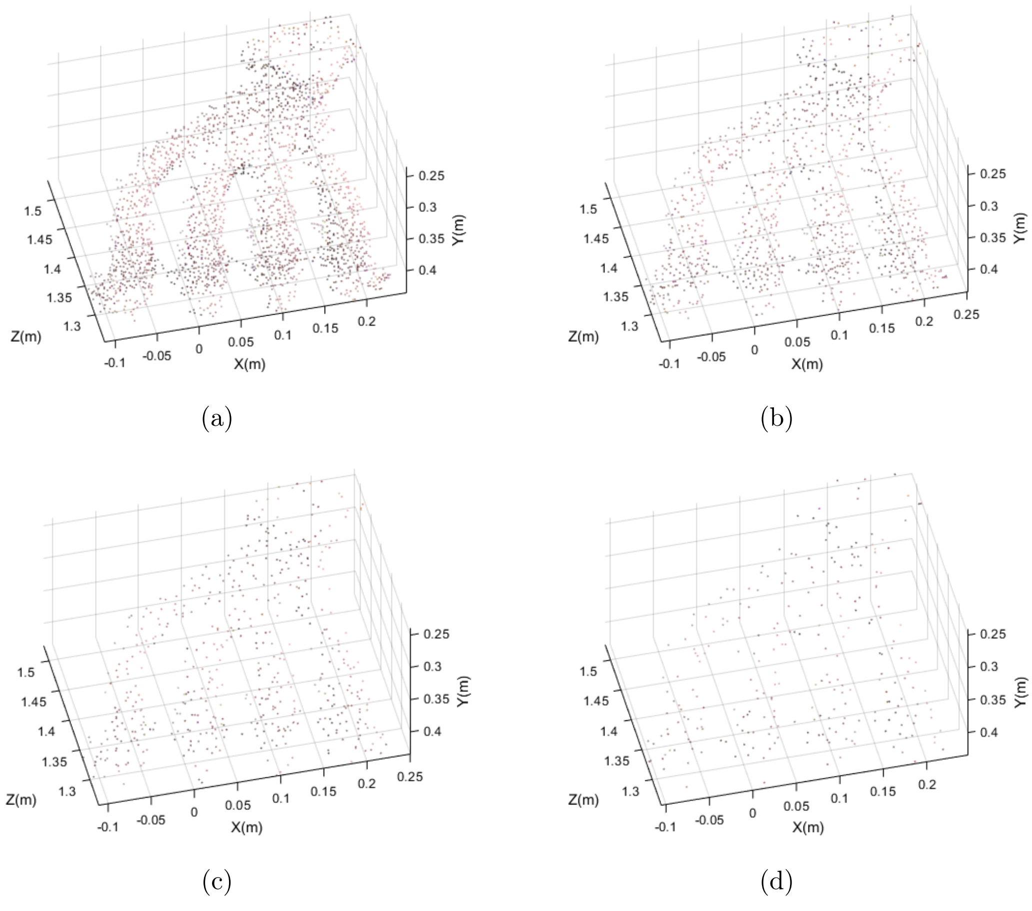


Figure 5.18: Point cloud NonUniformGridSample downsample with maxNumPoints of 6 (a), 10 (b), 20 (c) and 50 (d).

As can be seen, the gridAverage downsample method, in Figure 5.17, downsamples the point cloud so that the object structure remains better preserved. Therefore, it becomes the method to downsample the point cloud. This conclusion is not universal, as it suits better for this case in particular.

5.2 AppDesigner Application

As mentioned in Section 4.3, the app is split in two major tabs: Image Acquisition and Point Cloud Editor.

5.2.1 Image Acquisition Tab

When starting the application, the initial tab by default is "Image Acquisition", Figure 5.19. In this tab, is detected the object and its contours. It is also possible to do the same for a specific object zone and prepare their point clouds for the next tab.

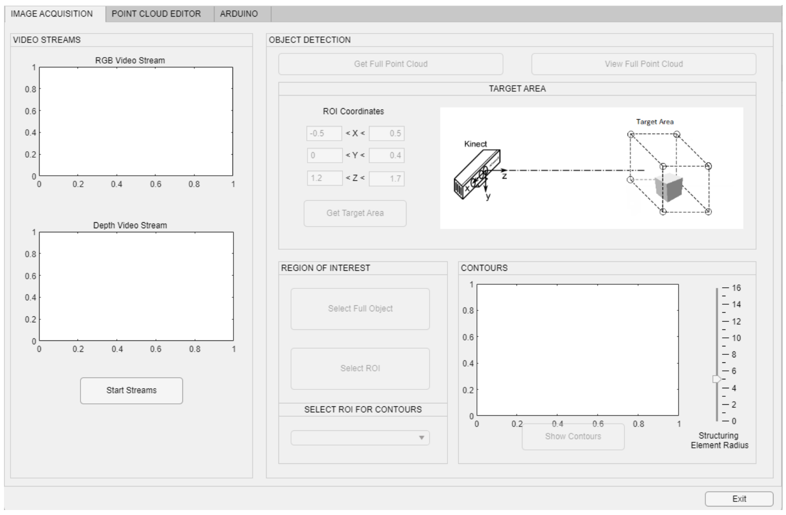
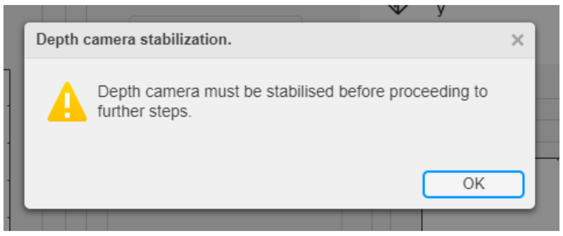


Figure 5.19: Image acquisition section homepage.

As visible in Figure 5.19, the only button available is the "Start Streams" button. This button will turn on the Kinect and display the video streams. After clicking it, a warning will inform the user that the Kinect camera requires time to acquire data correctly, as illustrated in Figure 5.20. If the user pressed the button to get the point cloud too early, it would result in several gaps in the image. The time it takes for the user to read and close the warning is enough for the Kinect to acquire data correctly, fulfilling the purpose of the warning. There was also a timer that only allowed the user to progress after it finished, giving time for the images to stabilize.

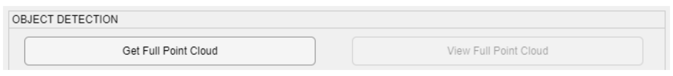

Depth camera stabilization.

Depth camera must be stabilised before proceeding to further steps.

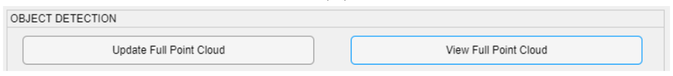
Figure 5.20: Camera warning.

The "Get the Point Cloud" button was only available after the timer finished. If the images are not acquired correctly, the user can always check if the data is correctly acquired by analyzing the video streams.

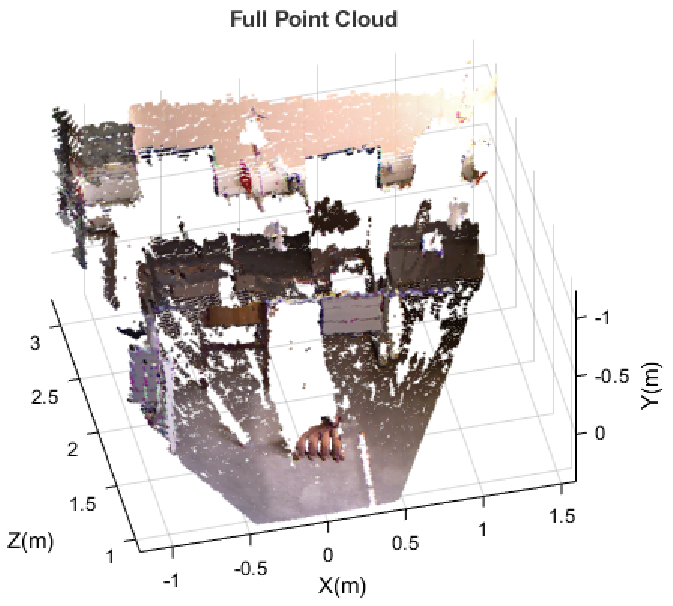
Next, the button that opens a browser shows in a point cloud the information captured by the Kinect, as shown in Figure 5.21 (a). If the cloud generates incorrectly, the user can recreate it using the "Update Full Point Cloud" button, Figure 5.21 (b). When the browser closes, it can reopen without affecting the cloud. With the point cloud created, Figure 5.21 (c), the user can define the target area boundaries for object detection.



(a)



(b)



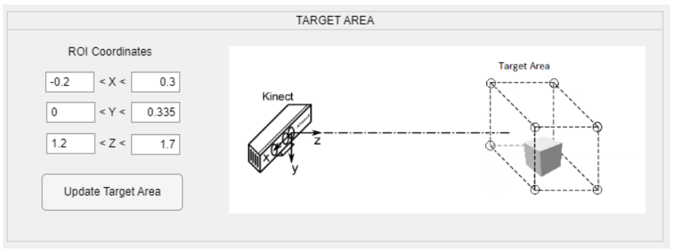
(c)

Figure 5.21: "Get Point Cloud" button available (a), buttons to upgrade and view the point cloud (b), and the point cloud created and displayed in a browser (c).

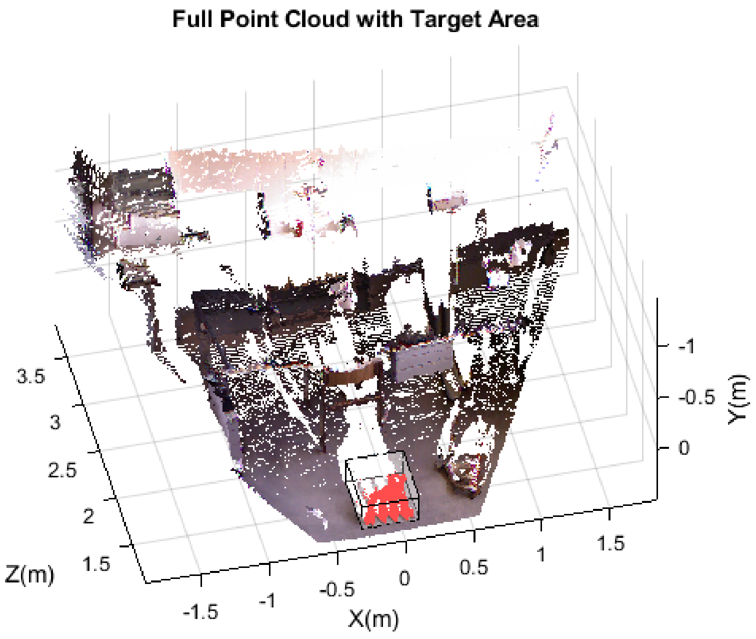
Taking the same course as in subsection 5.1.4, the user can now insert the limit values to create the target volume of analysis, as indicated in Figure 5.22 (a). By analyzing the point cloud, such as Figure 5.21 (c), the user can visualize the range of values where the object is and thus create the target area to extract the object under analysis. The constructed target area is visible in Figure 5.22 (b).

With the target volume defined, extracting the object becomes available to extract the information within the target area. Then, a new figure opens, showing the information extracted by the target area, represented in Figure 5.23.

This action still leaves the browser open to let the user verify the result. At this stage, it is still possible to define only a part of this object. By pressing the button to select a region of interest of the object, an alert dialogue box will inform the user how to define this region, shown in Figure 5.24 (a). An RGD image of the object will also appear, where the user draws the rectangular area of interest, Figure 5.24 (b).



(a)



(b)

Figure 5.22: Target area limits setup (a) and display in the browser (b).

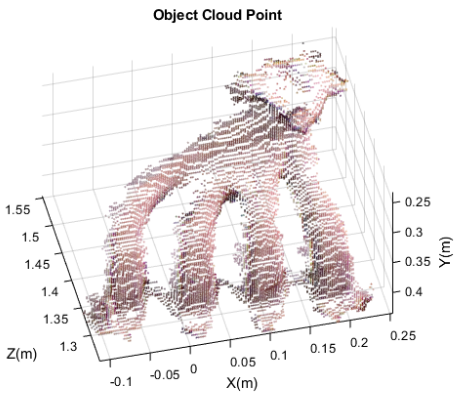
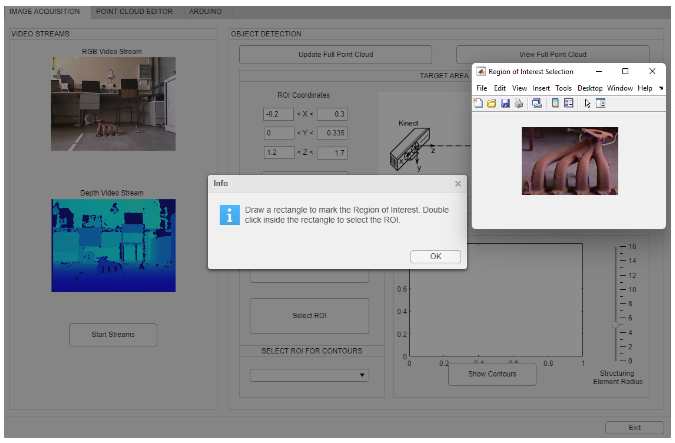
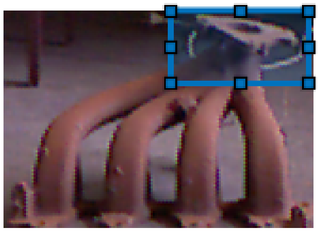


Figure 5.23: Object extracted within target area.

When the region of interest is delimited, it is possible to extract the point cloud of that region. After double-clicking on the defined region of interest, a new figure opens



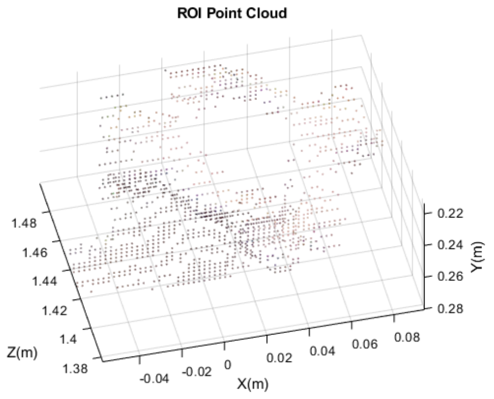
(a)



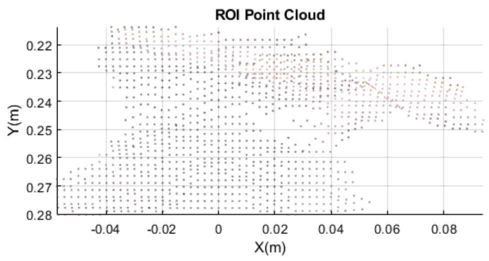
(b)

Figure 5.24: Information about the ROI definition (a) and rectangle draw to create the ROI (b).

containing the extracted point cloud, as shown in Figure 5.25.



(a)



(b)

Figure 5.25: Default (a) and frontal (b) view of the object's ROI

The user now has two clouds of points ready for treatment and processing: the complete object cloud (Figure 5.23) and the ROI cloud (Figure 5.25). A drop menu (Figure 5.26) lets the user choose between the two clouds to identify their contours. This option is only available after extracting the information from Figure 5.22.

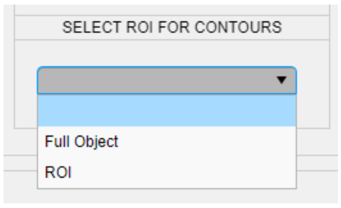
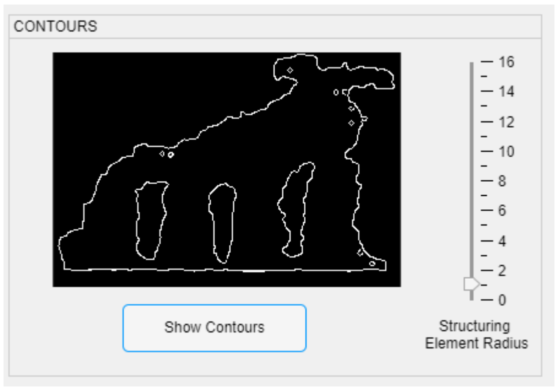
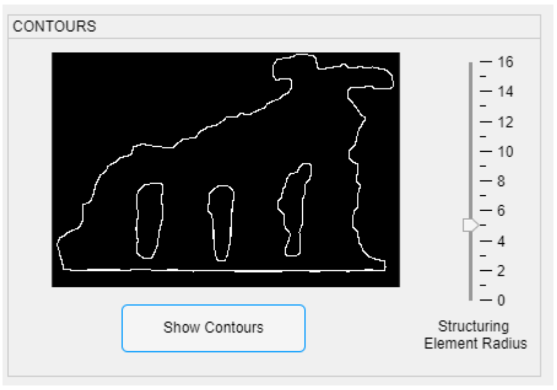


Figure 5.26: Dropdown menu for cloud point selection.

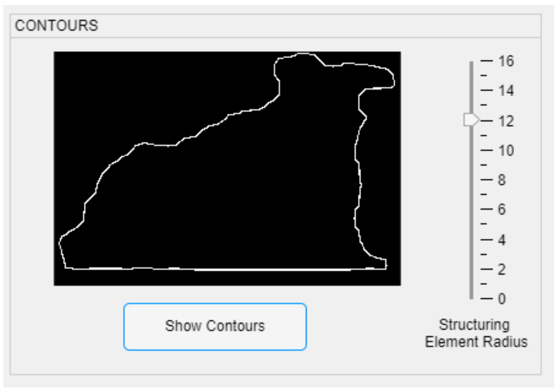
Once the cloud is selected, it proceeds to obtain its contours. Following the method of Figure 4.4, the user can correctly control the dimension of the structuring element to obtain the contours. Figure 5.27 shows examples of the contours obtained for the complete object, with different values for the structuring element dimension.



(a)



(b)

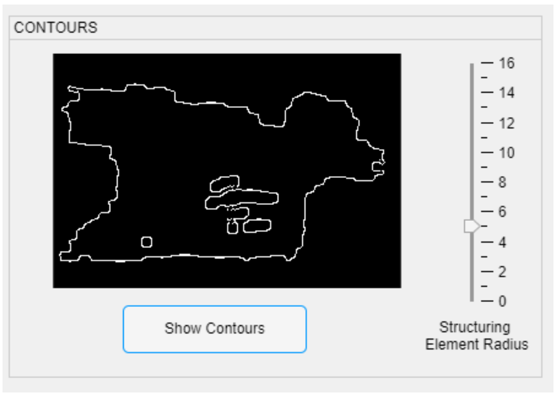


(c)

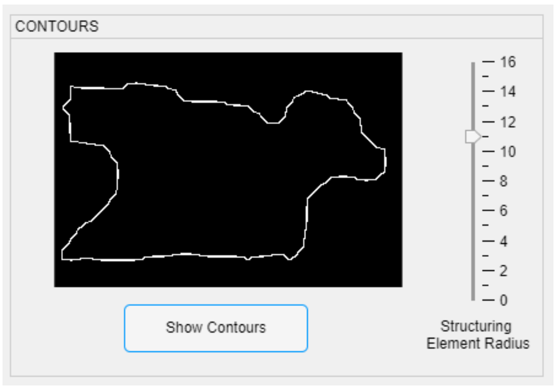
Figure 5.27: Edge detection after morphological closing using disk as structuring element with dimension 1 (a), 5 (b) and 12 (c).

An oversized structuring element can lead to poor contouring since holes in the object will be closed, showed in Figure 5.27 (c). As seen in Figure 5.27, increasing the size of the structuring element closes gaps in the contours. In contrast, as visible in Figure 5.27 (a), a small size can lead to excessive holes appearing in the image, incorrectly representing the object contours. By selecting the ROI cloud, the same process is followed but for a portion of the cloud, as depicted in Figure 5.28.

Here, after implementing the method described in Section 4.2.2, the points in the ROI cloud are further apart than in the entire cloud. After converting the 2D points to an image, they will be further apart by comparing Figure 5.23 and Figure 5.25. Thus, a short range of values for the structuring element dimension is required to close all the unwanted gaps and open the object holes, making it harder to get the contours correctly.



(a)



(b)

Figure 5.28: Edge detection after morphological closing using disk as structuring element with dimension 5 (a) and 11 (b).

5.2.2 Point Cloud Editor Tab

Concluding the steps previously exemplified, all image acquisition processes ended, and it is now possible to move to point cloud processing and treatment. This section majorly reduces the number of points of both point clouds created. It is also possible to remove points and increase and decrease the point density in a specific area. In this tab, the user can see the result of cloud processing, and a browser will allow the user to check the history of all actions performed throughout the process, Figure 5.29.

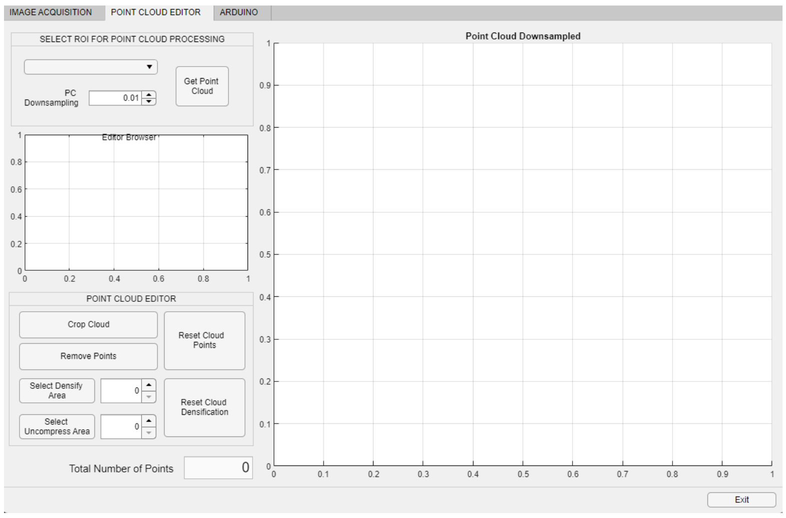
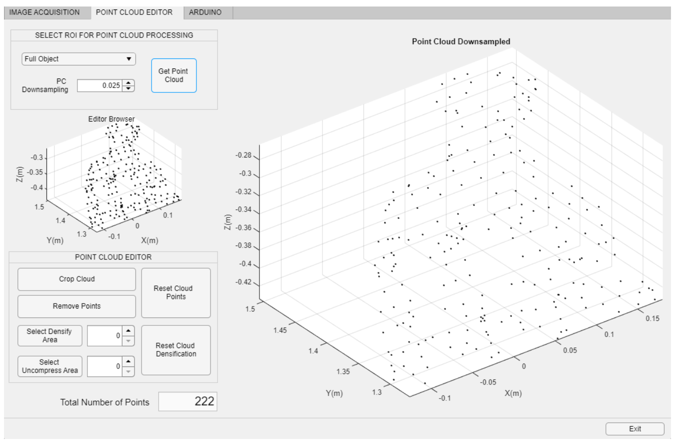


Figure 5.29: Point Cloud Editor Section Homepage.

With the cloud chosen, by pressing the "Get Point Cloud" button, a new point cloud with the reduced points will be shown. Figure 5.30 shows variations of this reduction. The point reduction is applied by establishing the box grid filter dimension and following the chosen method, exemplified in Figure 5.17.

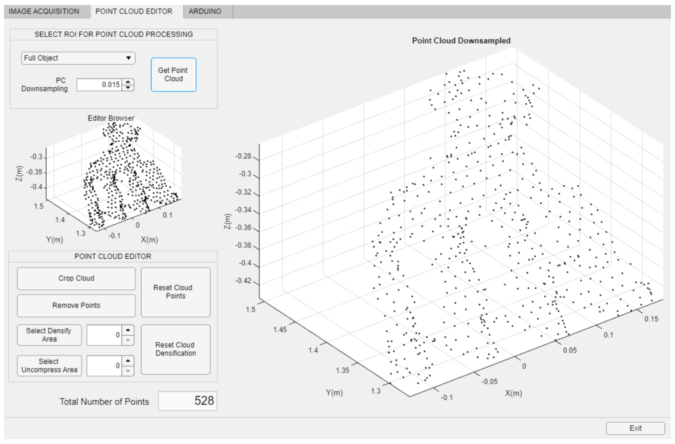
Notice that as this value increases, the number of points decreases. This decrease is due to the size of the box grid filter. By increasing the box size, more points are inside it,



PC Downsampling 0.025

Total Number of Points 222

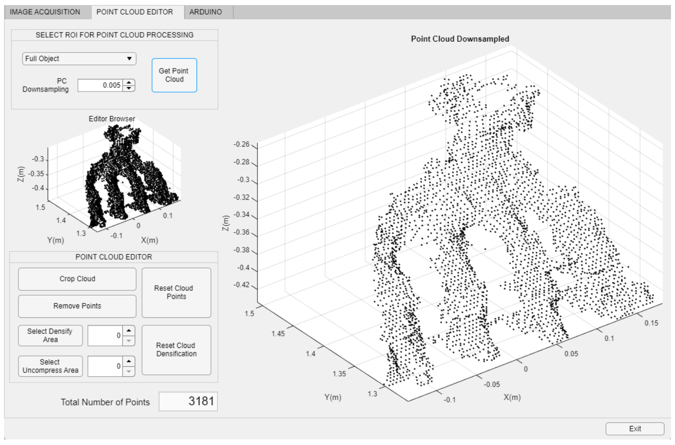
(a)



PC Downsampling 0.015

Total Number of Points 528

(b)



PC Downsampling 0.005

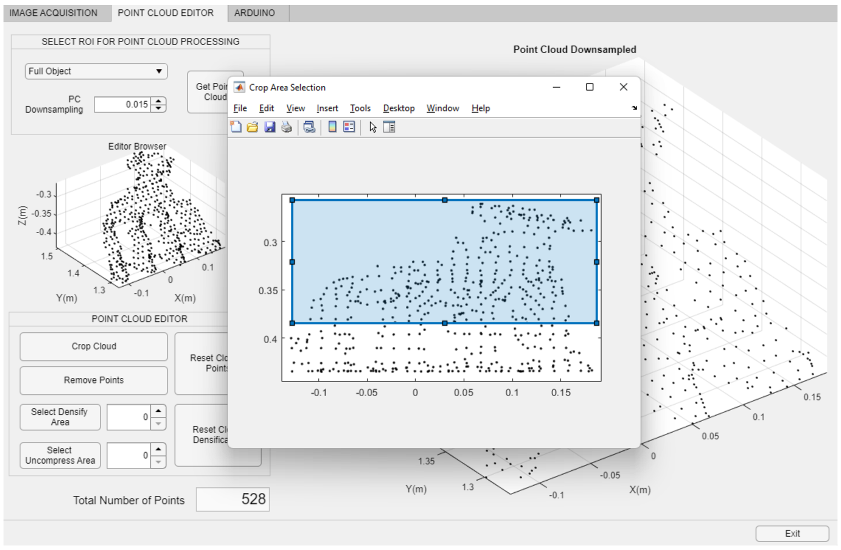
Total Number of Points 3181

(c)

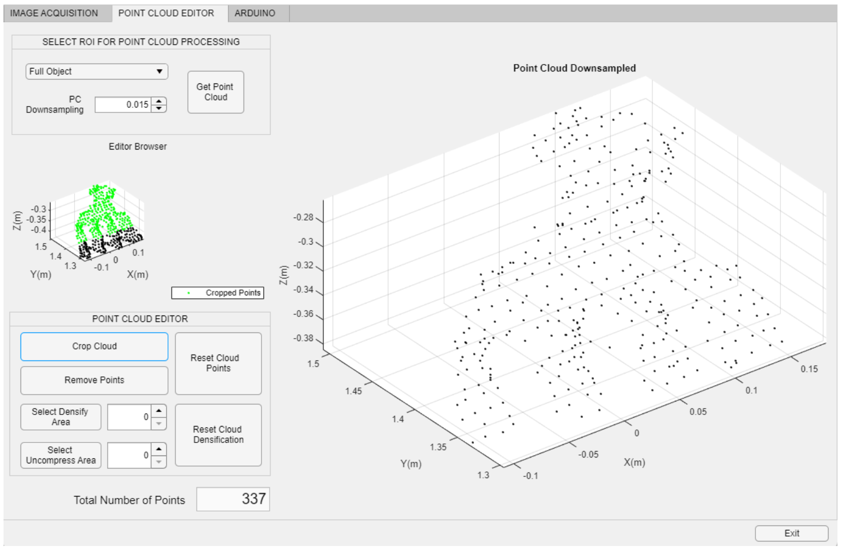
Figure 5.30: Point cloud downsampled with box grid filter size of 0.025 (a), 0.015 (b) and 0.005 (c).

and therefore more points are removed. The maximum number of points is the number of points before any reduction. Along with all changes, a counter shows the total number of points in the cloud.

By choosing the value for the point reduction, the user can choose several options to edit the point cloud, as shown in Figure 4.7. The first option for editing the cloud is the "Crop Cloud" option, in which, by selecting an area, similar to the process of 5.24, the user can define an area of the cloud to be removed, as illustrated in Figure 5.31.



(a)

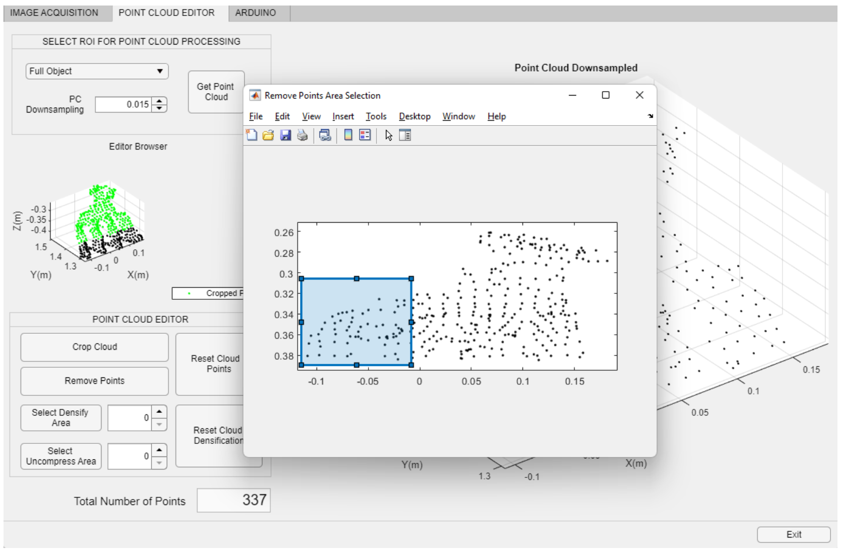


(b)

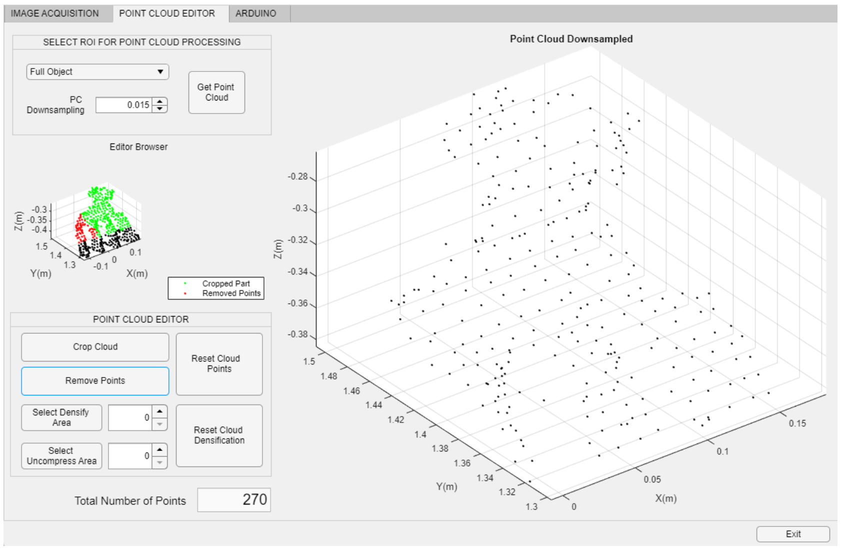
Figure 5.31: Area selected to crop the downsampled cloud (a) and its result (b)

Thus, the new cloud will appear after selecting the region of interest. Visible in Figure 5.31 (b), the browser saves the changes marked in green for the cropped area and the number of total points after the change.

The user can also remove points from a limited area of the point cloud. Previously selected the "Crop Cloud" option, and by clicking on the "Remove Points" button, a new figure displays the point cloud created in Figure 5.32 (a), allowing the user to draw the area to remove the points.



(a)



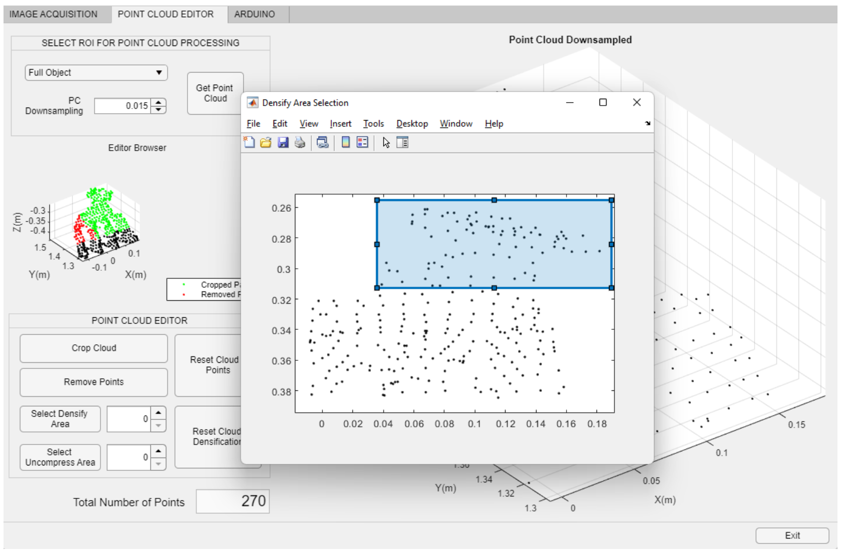
(b)

Figure 5.32: New figure displaying the two-dimensional points of the cloud and the area selected to remove points (a) and its result (b).

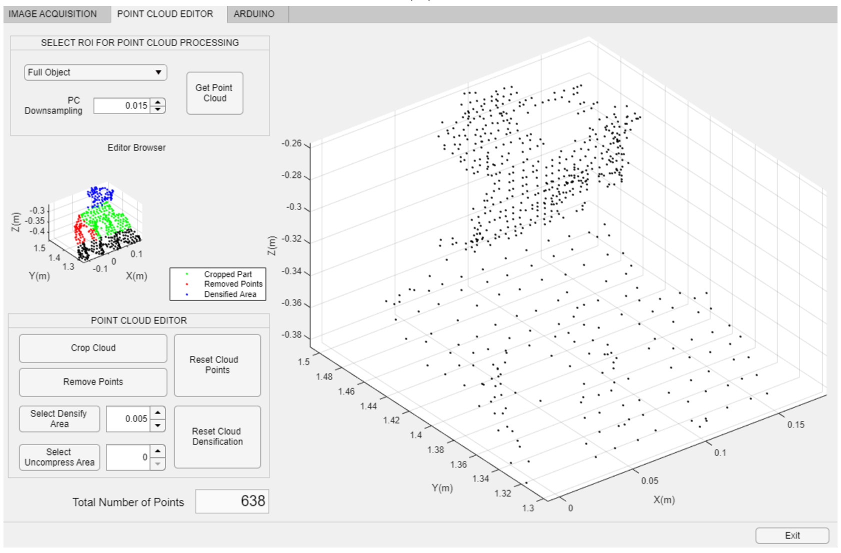
As shown in Figure 5.32 (a) and (b), the points within this area are removed, displaying the new point cloud in the application. The browser is also updated to show the history of these changes, marked in red for the removed points, Figure 5.32 (b).

After performing the crop and removing points, it is possible to modify the point density in user-defined areas. As in the previous steps, a new window will appear with the edited cloud and will be ready for the user to select the area to modify its density, as shown in Figure 5.33 (a).

Selecting the area will increase the density of the points inside it. This density can never be greater than the density in Figure 5.23 nor Figure 5.25 since the added points are recovered from the initial cloud of points. From the value chosen in Figure 5.30, the



(a)

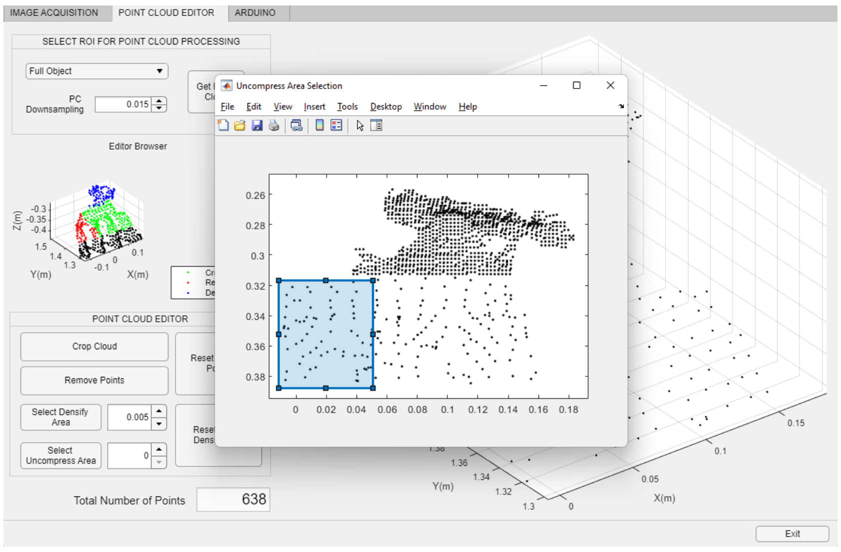


(b)

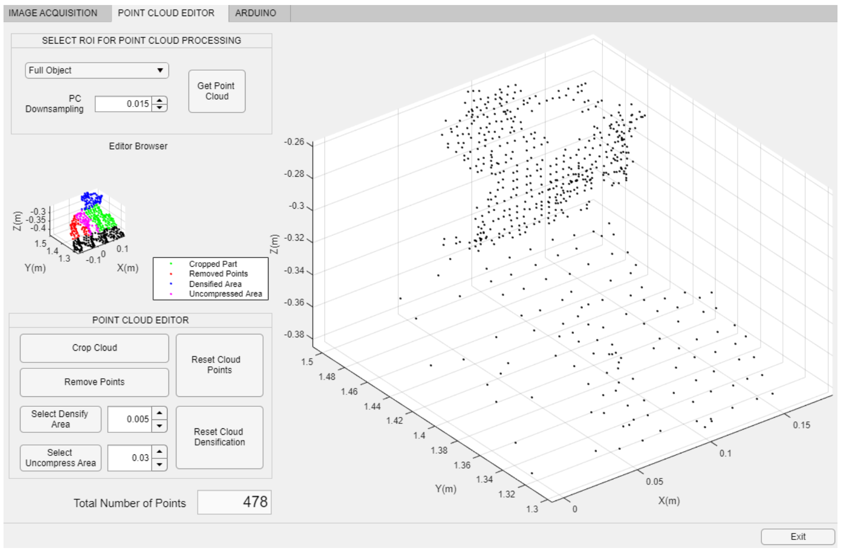
Figure 5.33: New figure displaying the previous edited point cloud and with the area drawn to increase density points (a) and the densification result (b).

user will decrease it, thus increasing the point density, resulting in the point cloud shown in Figure 5.33 (b). This process is similar to Figure 5.17, in which increasing the box grid filter decreases the point density, and decreasing it increases its density. All density changes will be saved and displayed in the editor browser, marked in blue in Figure 5.33 (b).

With the density of the points increased, it is also possible to do the opposite. It will be shown again in a new figure, the point cloud with all the previous changes, where the user again defines the processing area, Figure 5.34 (a). Once the area is selected, all the points will suffer a reduction in their density, as shown in Figure 5.34.



(a)

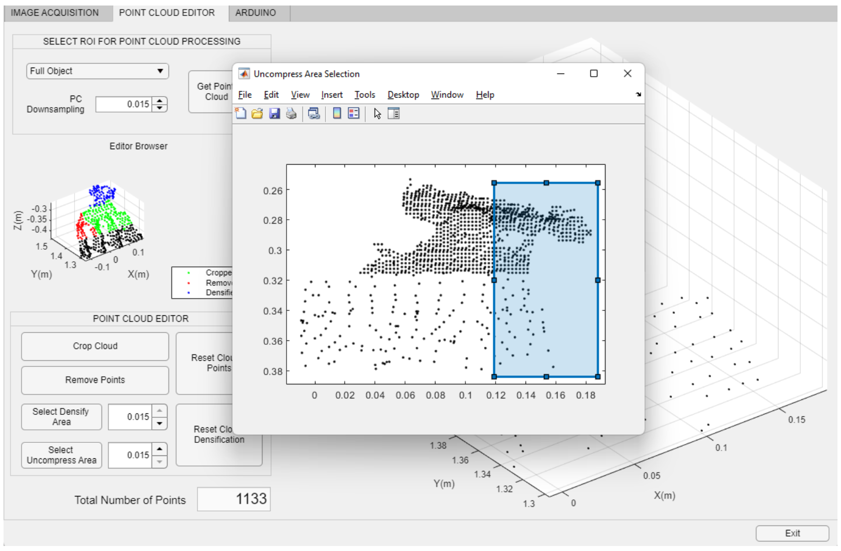


(b)

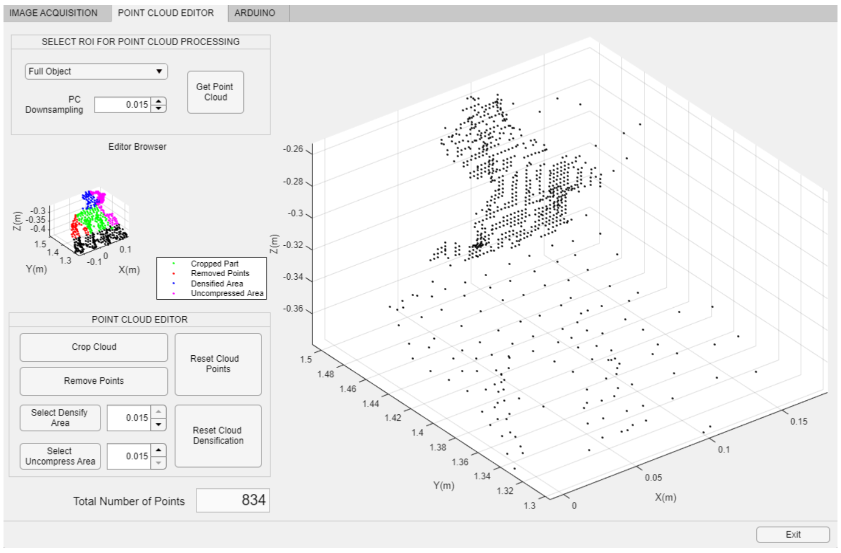
Figure 5.34: Figure displaying the area selected in the point cloud (a) and the result of the density decrease (b).

Resembling the densification process, the user will increase the value chosen in Figure 5.30 to proceed with the density reduction. From the value chosen during the downsampling, the user will increase it, decreasing the point density. This process results in the point cloud shown in Figure 5.34 (b), and the changes in the editor browser are saved as well and marked in magenta color in Figure 5.34 (b).

It was also tested whether it would be possible to combine density configurations, i.e., decrease density in zones where its point density had already been increased and vice versa. By drawing the area shown in Figure 5.35 (a), two object parts were selected with different point densities. The points selected will be replaced by points from the initial cloud in the same area, and a new density will be attributed according to the user choice, as shown in Figure 5.35 (b).



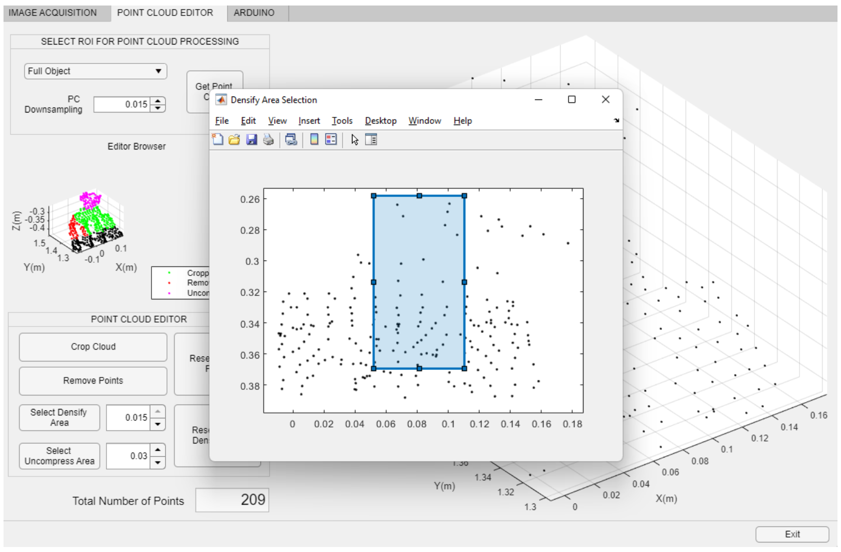
(a)



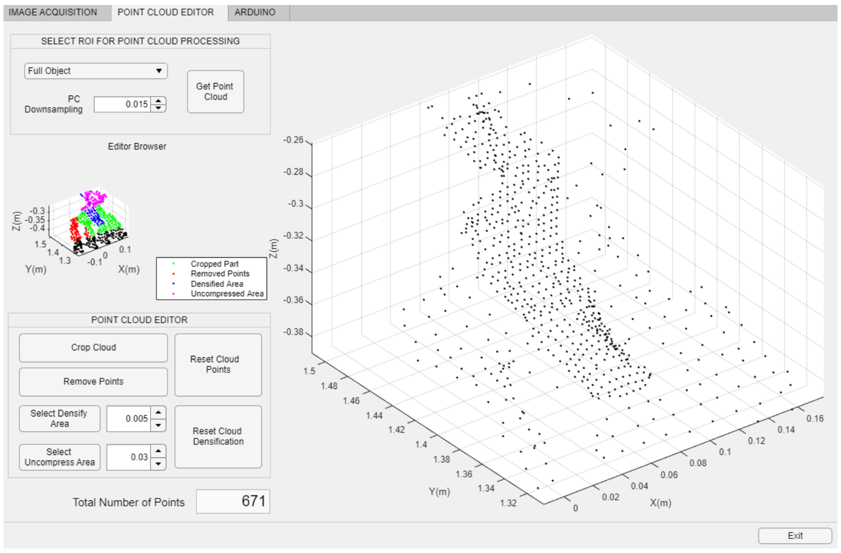
(b)

Figure 5.35: Density mixed area selected (a) to proceed to its density reduced (b).

The same was tested to increase the density of zones with initial and reduced densities. The points inside the area selected in Figure 5.36 (a) were again extracted from the cloud with initial density. Their new density will vary depending again on the value of density increase. The result of this new density is shown in Figure 5.36 (b). The browser in the "Point Cloud Editor" tab saves all changes made by the user. Thus, the browser shows where this action occurs when editing the cloud. Initially, and after selecting the "Get Point Cloud" button, the browser shows the same points as the downsampled cloud (Figure 5.30).



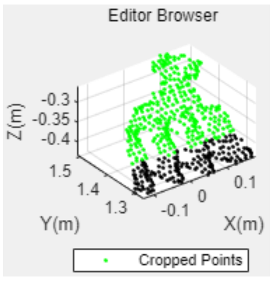
(a)



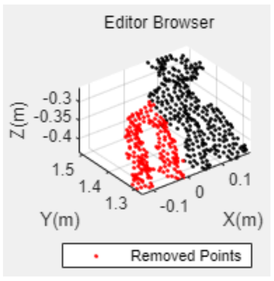
(b)

Figure 5.36: Density mixed area selected (a) to proceed to its density increased (b).

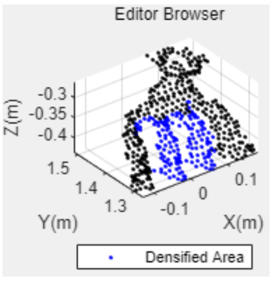
After selecting the areas for processing, as in Figure 5.31 (a), Figure 5.32 (a), Figure 5.33 (a) and Figure 5.34 (a), the points to be processed are known. Depending on their color, these points will be shown in the browser and represent the different actions taken, as seen in Figure 5.37.



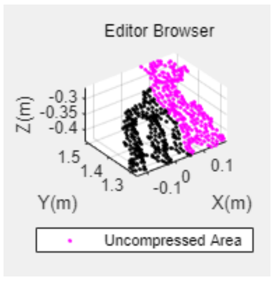
(a)



(b)



(c)



(d)

Figure 5.37: Editor browser individually displaying crop (a), remove points (b), increase (c) and decrease density (d) options with the colors green, red, blue and magenta associated, respectively.

Unlike the "Image Acquisition" task, which was mandatory to follow a sequential procedure, actions do not follow a sequential process in this tab, allowing the user to choose any option individually. In "Image Acquisition", the application prohibits access to the target area until the creation of the initial cloud, and obtaining the contours is denied prior to the object cloud, for example. The "Point Cloud Editor" tab gives freedom of action to the user, choosing the combination of options for processing the cloud as desired or just remaining with the initial point reduction.

Chapter 6

Experiments and Results

This chapter describes all the tests performed during the project and their results. The results described are all obtained through the application developed in the Matlab environment.

6.1 Image Acquisition

As it was possible to acquire images and depth information, two different geometries were used to test the capacity of the developed system and the device used. Figure 6.1 contains several images obtained from the different geometries.

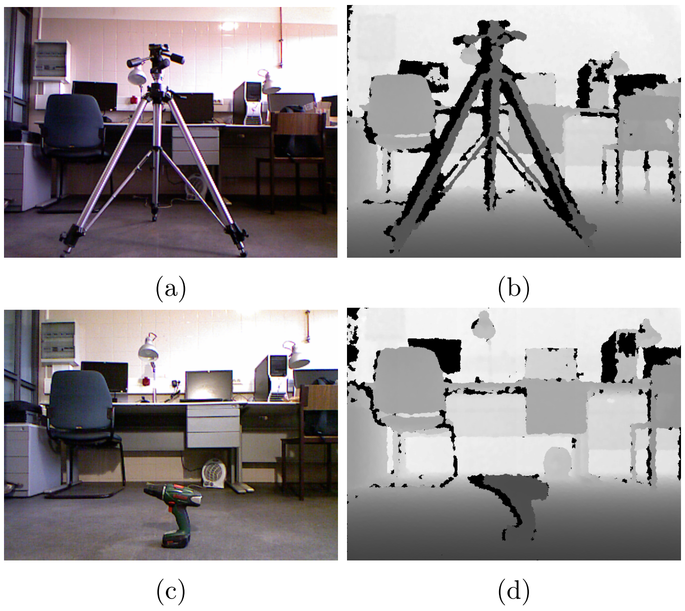


Figure 6.1: RGB image (a) and depth data (b) of the first geometry, and a RGB image (c) and depth data (d) of the second geometry used.

The depth data of the first geometry, in Figure 6.1 (b), allowed the verification of the Microsoft Kinect response to reflective surfaces. On the other hand, the second geometry information, in Figure 6.1 (d), allowed the analysis of small and complex shapes.

Comparing the depth data of both geometries, the presence of several black zones is more noticeable in Figure 6.1 (b) than in Figure 6.1 (d). These zones represent incorrectly acquired information due to the reflection of the pattern drawn by Microsoft Kinect.

6.2 Point Clouds

With the RGB images and the depth data from both geometries, it is now possible to create the respective point clouds. Figure 6.2 shows the clouds created with the information obtained by the Microsoft Kinect.

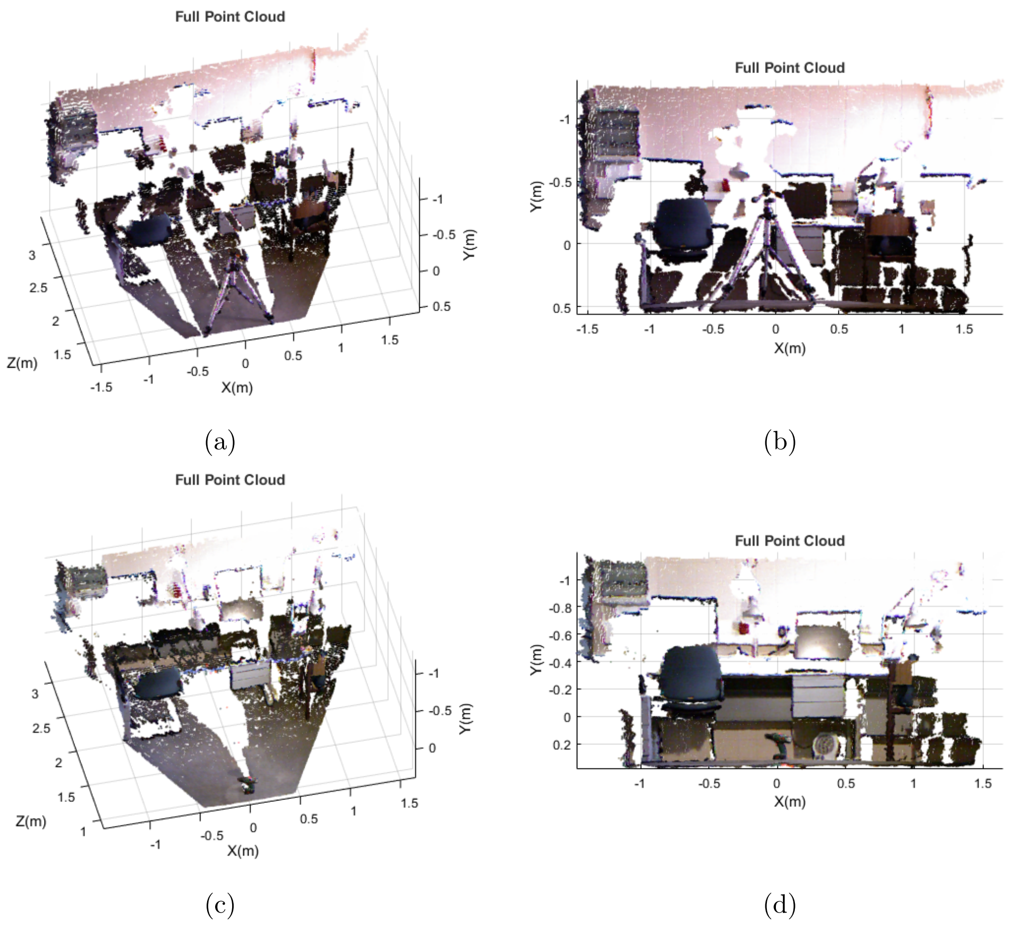


Figure 6.2: Default (a) and (c), and frontal view (b) and (d) of the point cloud generated

Obstacles between the object and the Microsoft Kinect will cover the object, making it impossible to collect its data. Therefore, several areas of the point clouds do not have information. In addition, the black regions in the depth data also produce zones without information in the point cloud.

Now that the cloud has been created, both geometries were extracted, removing the excess information. Then, by defining the target area, and entering its boundaries, as exemplified in Section 5.2.1, the clouds of the two geometries were created, as shown in Figure 6.3.

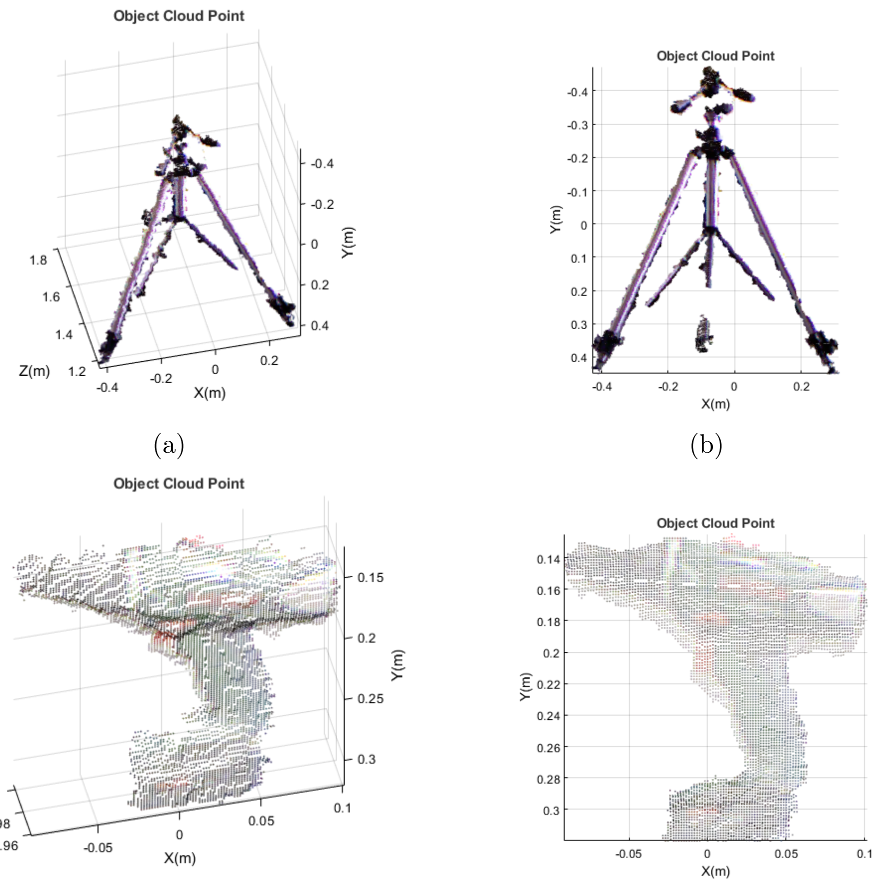


Figure 6.3: Point cloud for the first (a) and (b), and for the second geometry (c) and (d).

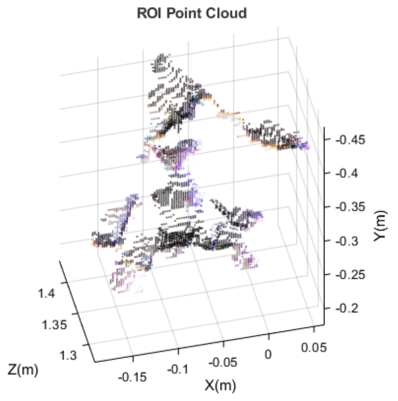
By analysing Figure 6.3 (a) and (b), it can be seen that Kinect has difficulties acquiring information projected onto reflective surfaces. These difficulties lead to the non-acquisition of points that pertain to the object, as it will be possible to observe in more detail in Figure 6.4 (c) and (d).

The application also allows the selection of a part of the object for post-processing. As explained in Section 5.2.1, a region of interest was defined within the object cloud, thereby creating a new cloud. Figure 6.4 shows the regions of interest created and the resulting clouds for each geometry. The defined area delimits the cloud of the complete object, extracting the data to create the ROI cloud. In Figure 6.4 (b), (c), (d) and (e), the clouds extracted by the defined areas are represented. As in Section 5.1.5, it is possible to know where the points to extract are located through the limits of the area drawn on the RGB image, exemplified in Figure 6.4 (a) and (b). In Figure 6.4 (c) and (d), it is then possible to see the points not captured by the Kinect in greater detail. In Figure 6.4 (e) and (f), it is represented the ROI cloud for the second geometry.

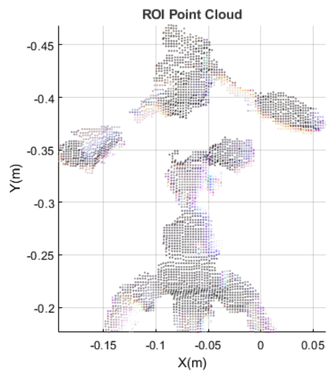
The ROI cloud allows better visualization of the different types of curvature of the second geometry, concluding that Kinect does not encounter difficulties detecting these types of zones. The clouds of both geometries are then ready to proceed by detecting and processing their contours.

(a)

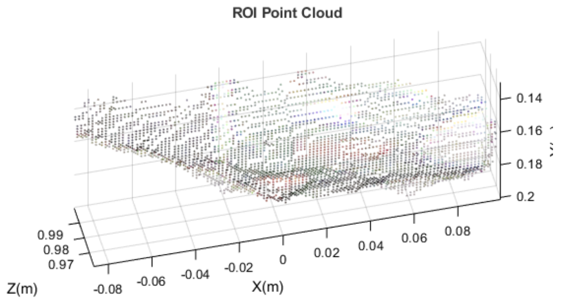
(b)



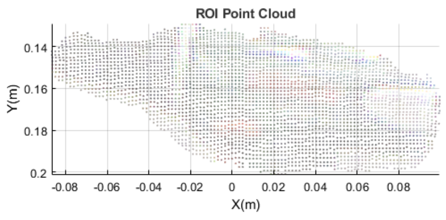
(c)



(d)



(e)



(f)

Figure 6.4: Areas selected (a) and (b) to create the ROI clouds (c), (d), (e) and (f) for both geometries geometries.

6.3 Contours

With the two clouds available (the whole object and ROI), it is required to select one of them to trace its contours. Figure 6.5 and 6.6 show variations of the detected contours obtained based on the procedures mentioned in Section 5.1.6.

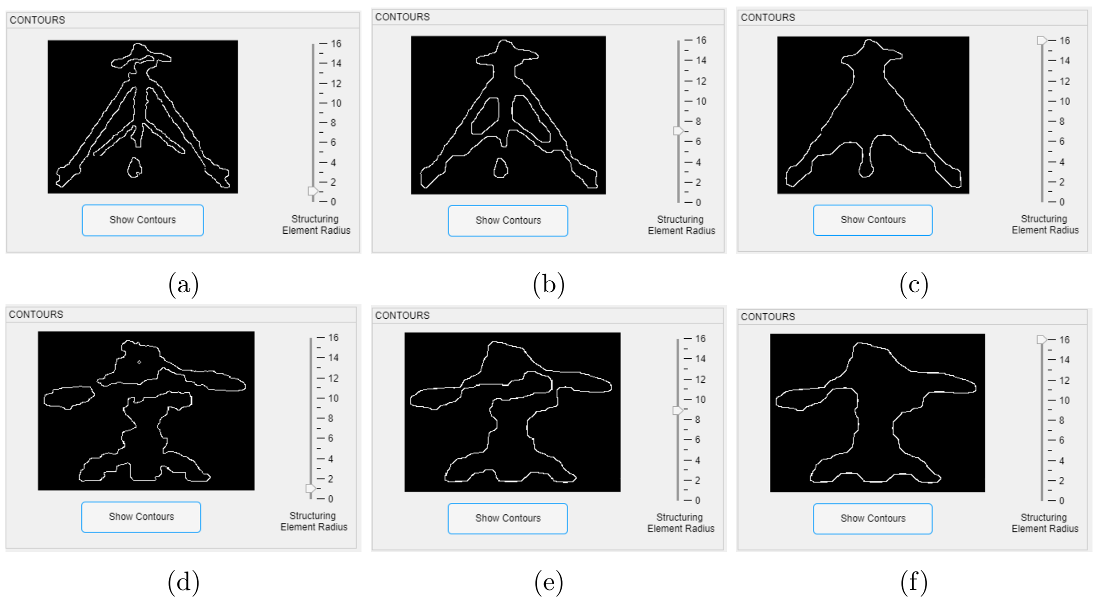


Figure 6.5: Different contour detection for the full object (a), (b) and (c), and for the ROI (d), (e) and (f), for the first geometry.

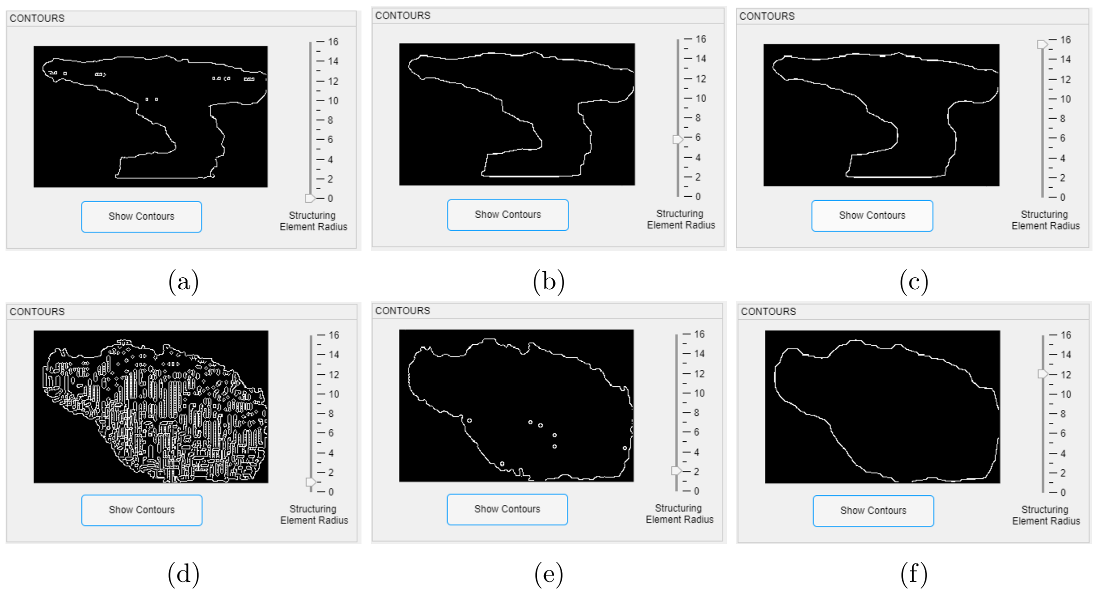


Figure 6.6: Different contour detection for the full object (a), (b) and (c), and for the ROI (d), (e) and (f), for the second geometry.

The contours of Figure 6.5 and 6.6 were obtained by varying the size of the structuring element. With the increase of this value, possible holes in the object can be covered. Also, the decrease in this value leads to the appearance of unwanted flaws in the contours, such as holes. The fact that the user can obtain the contours by varying the value of the structuring element manually rather than in an automated manner allows the user to choose which configuration of the contours benefits it the most.

6.4 Point Cloud Processing

With the part responsible for image acquisition completed, the next step is to process the point clouds. In this section, an action area must be predefined to perform all actions except the downsample.

6.4.1 Point Cloud Downsample

Initially, the first point reduction is performed using the method to downsample the point cloud, explained in Section 5.1.7. Variations of the point reduction for both geometries are obtained and shown in Figure 6.7 and Figure 6.8.

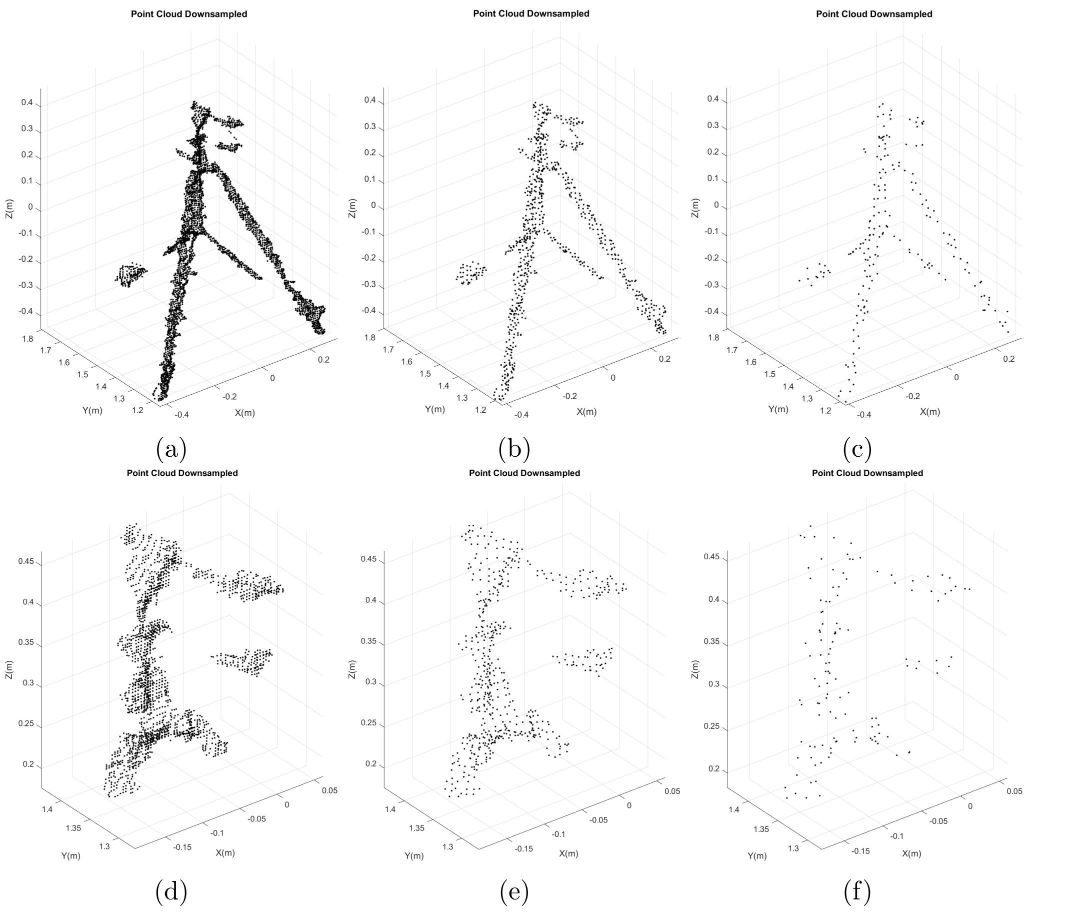


Figure 6.7: Variations of the downsample for the full object (a), (b) and (c). Variations of the downsample for the ROI object (d), (e) and (f). Both variations from the first geometry.

Similar to Section 5.2.2, as the value of gridStep is changed, the number of points in the selected cloud changes. The higher the value, the more significant the point reduction is. By analyzing Figure 6.7 (c) and Figure 6.8 (c), it is noticeable that a reduced gridStep number will reduce more points from the cloud than the reductions made in Figure 6.7 (a) and Figure 6.8 (a), which used a lower gridStep value. All operations in this application section are saved and displayed in the browser, as shown in Figure 6.9.

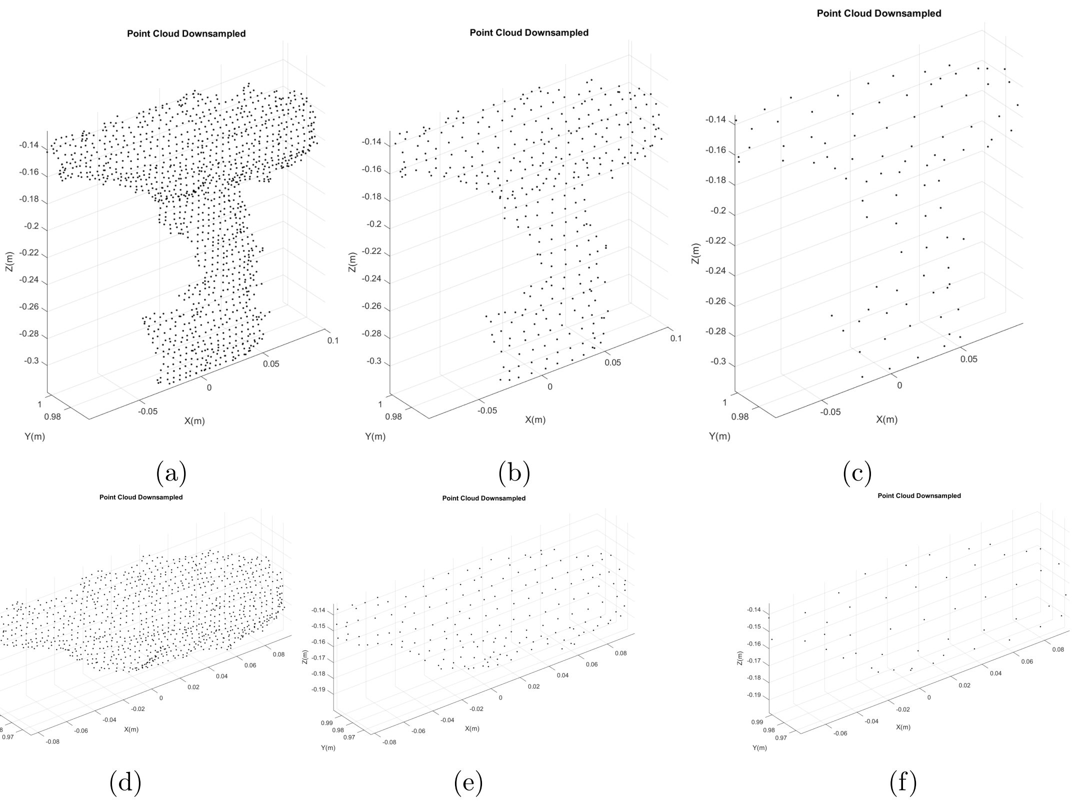


Figure 6.8: Variations of the downsample for the full object (a), (b) and (c). Variations of the downsample for the ROI object (d), (e) and (f). Both variations from the second geometry.

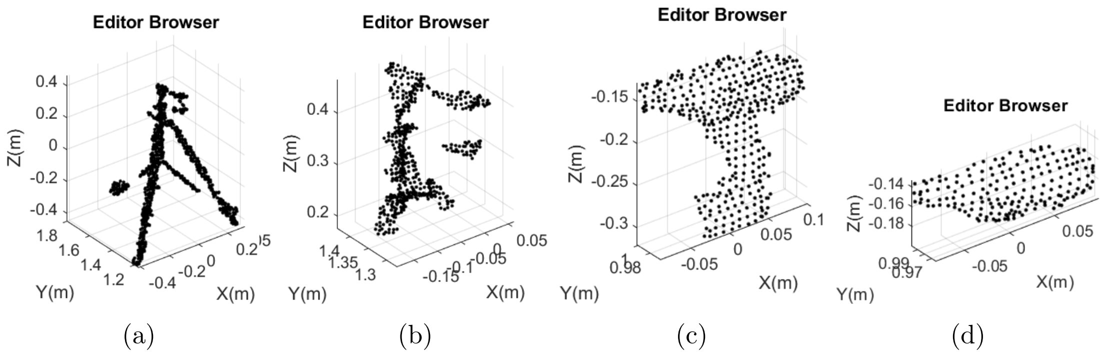


Figure 6.9: Editor browser for the first geometry object (a) and ROI (b), and for the second geometry object (c) and ROI (d)

A color representing each action is added to the browser, letting the user know which action was applied and where it was performed. Once the initial point reduction has been chosen, it proceeds to reduced cloud processing. From this part of the application, operations such as cropping the point cloud, removing points, and increasing and reducing point density are possible. All these operations do not occur successively, i.e., the user can decide which operations want to do regardless of the order of action.

6.4.2 Crop Point Cloud

The user may now want to select only a portion of the reduced cloud for processing. By pressing the "Crop Cloud" button, the user selects an area of the reduced cloud, as shown in Figure 6.10 and Figure 6.11, to be processed further.

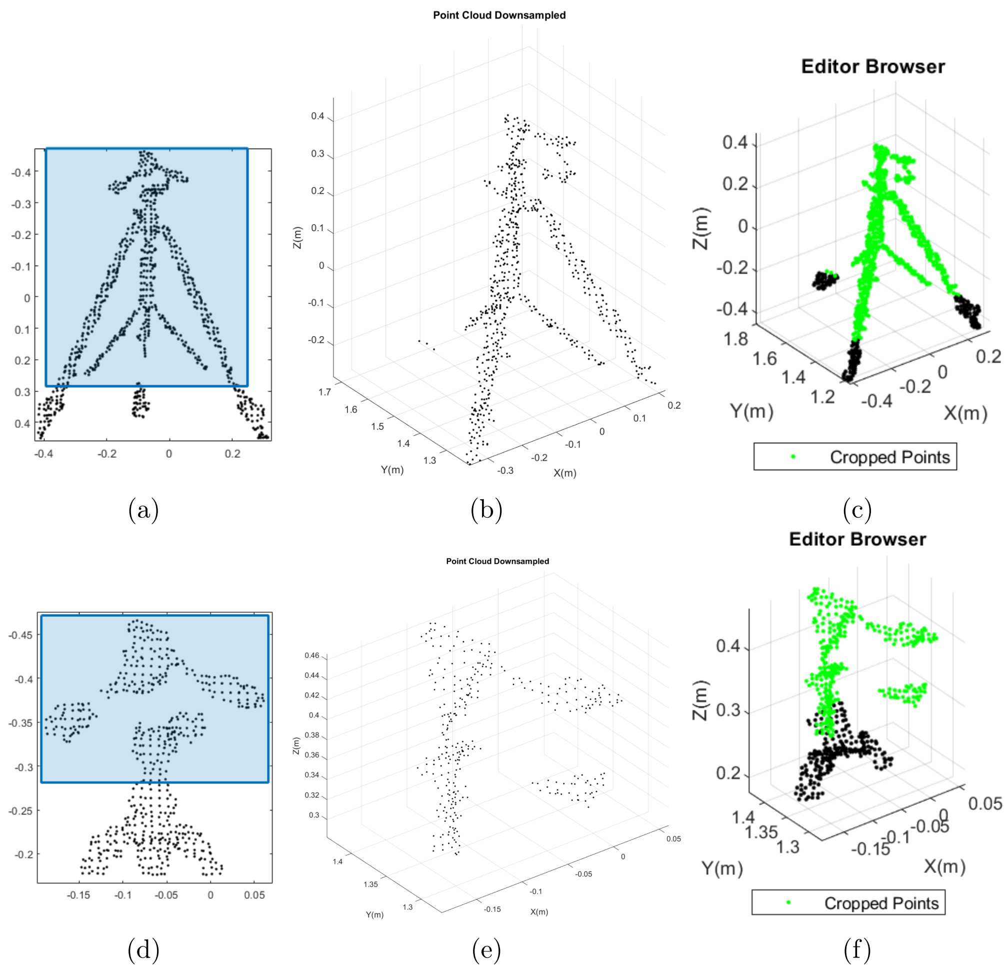


Figure 6.10: Areas selected on the object (a) and ROI (d) to crop the point cloud, their resulting point cloud (b) and (e), and the browser showing both operations performed (c) and (f), respectively and regarding the first geometry.

The areas selected in Figure 6.10 (a) and (d) and Figure 6.11 (a) and (d) give the location of the points inside. These points are then extracted from the cloud to create the resulting clouds, shown in Figure 6.10 (b) and (e) and Figure 6.11 (b) and (e). The browsers are also updated, showing the operation performed and the region where it acted, as shown in Figure 6.10 (c) and (f) and Figure 6.11 (c) and (f). By performing this action, only the selected part of the cloud will be used for further processing.

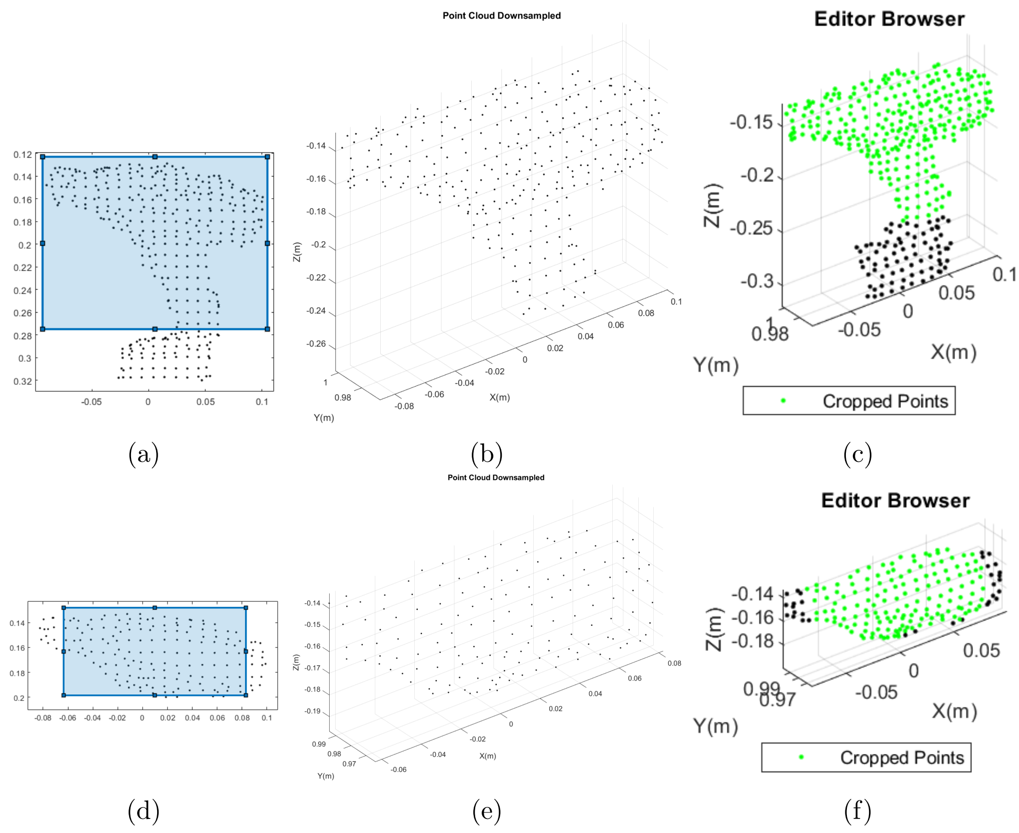
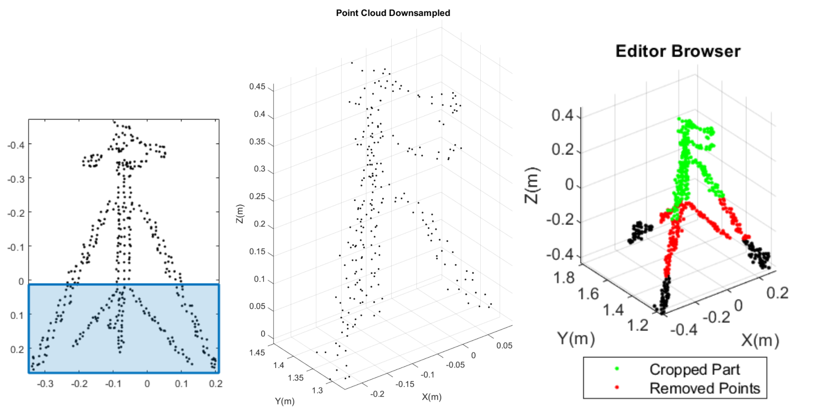


Figure 6.11: Areas selected on the object (a) and ROI (d) to crop the point cloud, their resulting point cloud (b) and (e), and the browser showing both operations performed (c) and (f), respectively and regarding the second geometry.

6.4.3 Remove Points

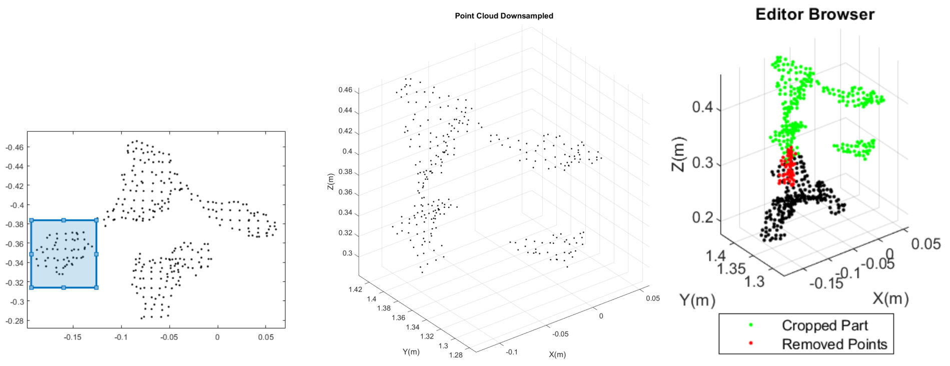
It is possible to define zones in the point cloud to remove points by selecting the option to remove points. The selected points will be removed to create a new cloud, as shown in Figure 6.12, Figure 6.13, Figure 6.14 and Figure 6.15.

This operation is responsible for removing unwanted points in the cloud. By selecting the areas, the points are removed, and the new point cloud is updated. Note that as the crop option has been previously selected, only the cloud resulting from the crop option will be shown when defining the areas for point removal. The browser is also updated to show the changes made, as demonstrated in Figure 6.12 (c), Figure 6.13 (c), Figure 6.14 (c) and Figure 6.15 (c).



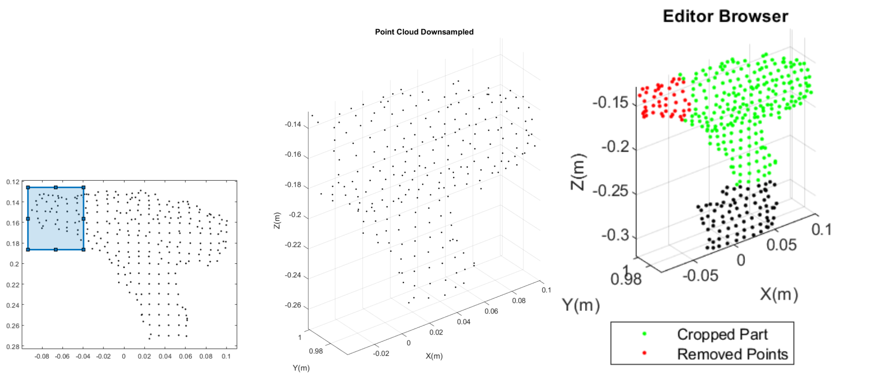
(a) (b) (c)

Figure 6.12: Area selected on the object (a) to remove points, the reduced point cloud (b), and the browser showing the operations performed (c), regarding the first geometry.



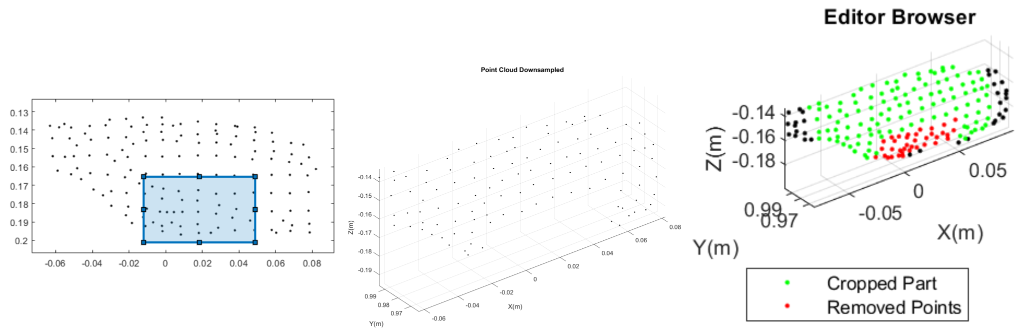
(a) (b) (c)

Figure 6.13: Area selected on the ROI (a) to remove points, the reduced point cloud (b), and the browser showing the operations performed (c), regarding the first geometry.



(a) (b) (c)

Figure 6.14: Area selected on the object (a) to remove points, the reduced point cloud (b), and the browser showing the operations performed (c), regarding the second geometry.

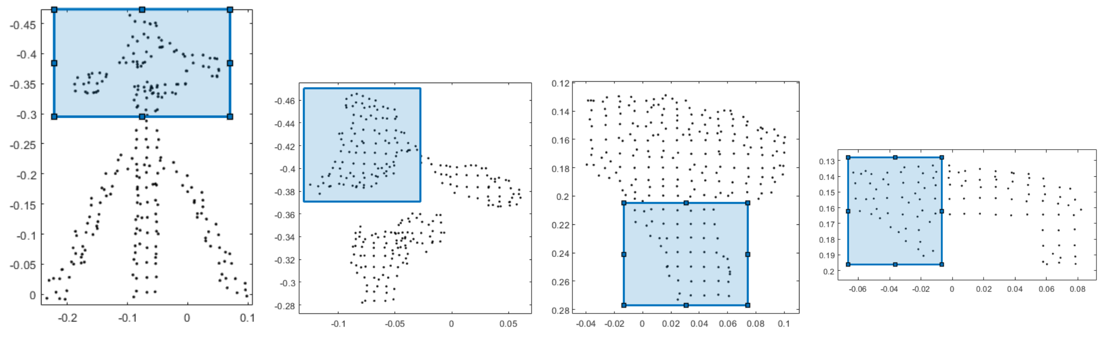


(a) (b) (c)

Figure 6.15: Area selected on the ROI (a) to remove points, the reduced point cloud (b), and the browser showing the operations performed (c), regarding the second geometry.

6.4.4 Increase Point Density

Defining the action area, it is also possible to increase the point density, and, as the remove points option has been selected, the removed points will not be shown. Figure 6.16 shows the areas where point densities will be increased in both geometries.



(a) (b) (c) (d)

Figure 6.16: Areas selected in the first geometry (a) and (b), and in the second (c) and (d), to increase point density.

After selecting the action areas for the first geometry, as shown in Figure 6.16, (a) and (b), the point density was varied. It is possible to see in Figure 6.17 this variation for the complete object and the ROI of the object of the first geometry. The point density increases by decreasing the grid box size, as explained in Section 5.2.2. It also tested variations in the density increase in the clouds of the second geometry. Using the selected areas in Figure 6.16 (c) and (d), density variations were obtained within the areas, as shown in Figure 6.18.

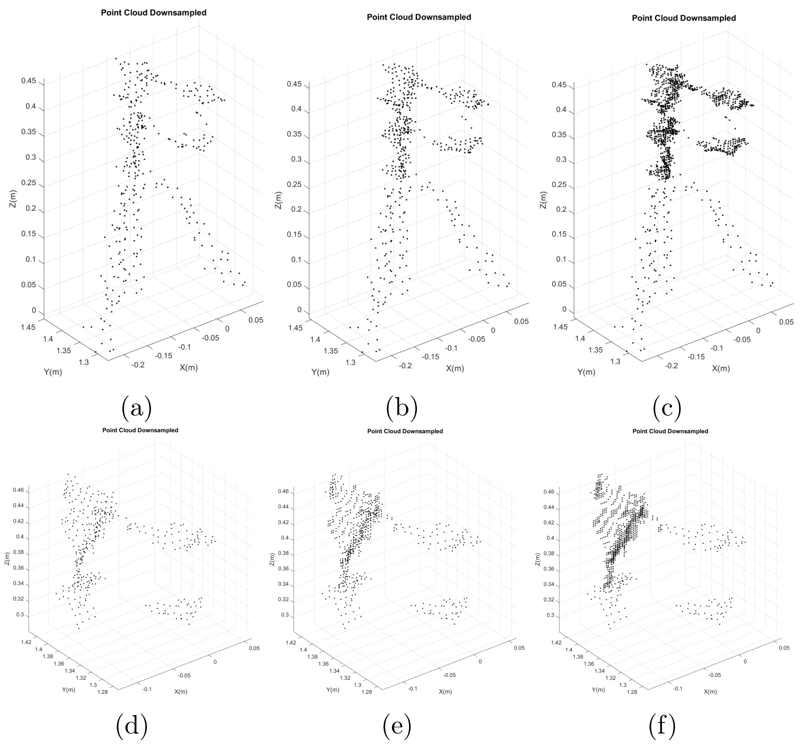


Figure 6.17: Variations of the point density increase for the full object (a), (b) and (c). Variations of the point density increase for the ROI object (d), (e) and (f). Both variations from the first geometry.

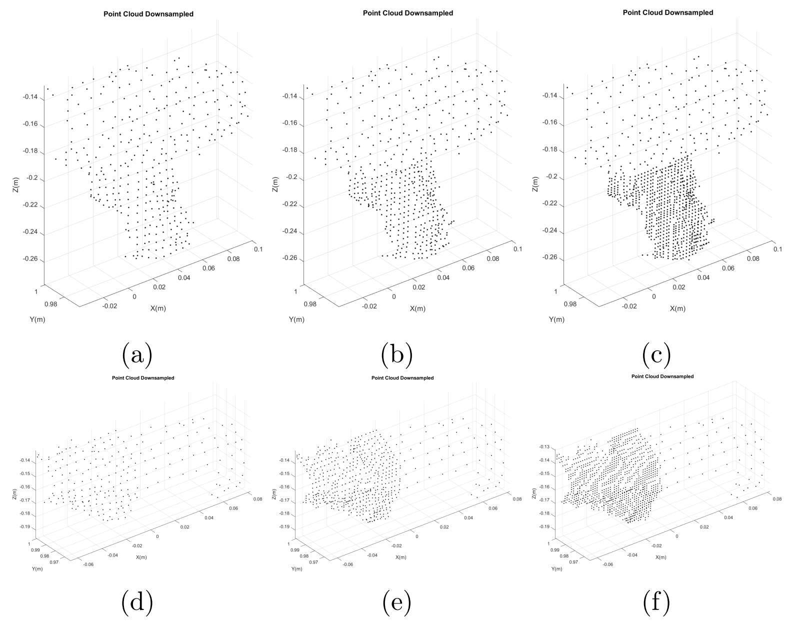
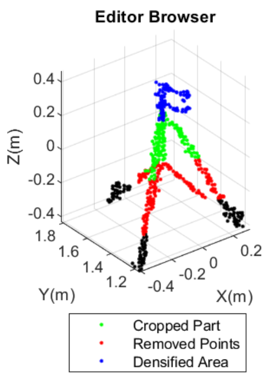
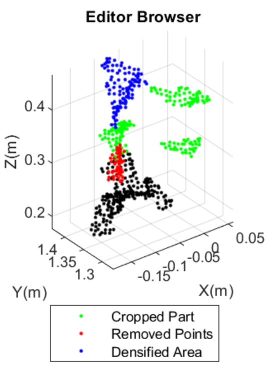


Figure 6.18: Variations of the point density increase for the full object (a), (b) and (c). Variations of the point density increase for the ROI object (d), (e) and (f). Both variations from the second geometry.

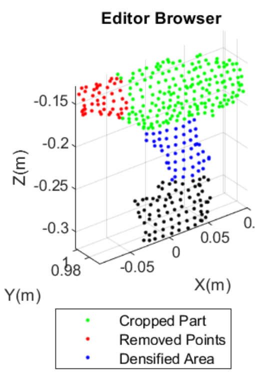
After increasing the point density, the browser recorded this interaction. Figure 6.19 shows the browser, with the remove points operation marked, for the changes made to both geometries.



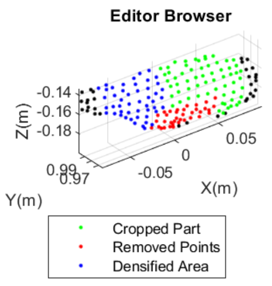
(a)



(b)



(c)

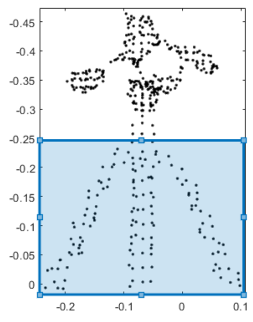


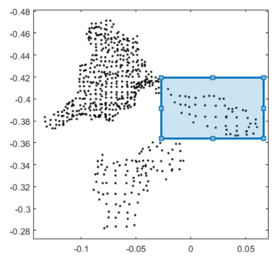
(d)

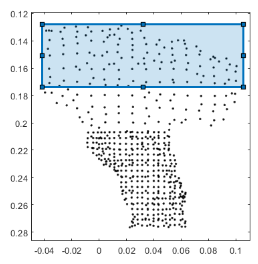
Figure 6.19: Editor browser showing the area with increased point density in the full object (a) and ROI (b) from the first geometry. Editor browser showing the area with increased point density in the full object (c) and ROI (d) from the second geometry.

6.4.5 Decrease Point Density

To reduce the point density is again necessary to define the area where the reduction will occur. Figure 6.20 shows the areas where the point density reduction will happen in both clouds of both geometries. When defining these areas, the area where an increase in density had happened is also visible.


(a)


(b)


(c)

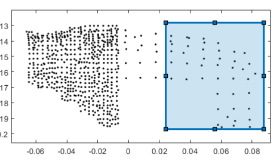

(d)

Figure 6.20: Areas selected in the first geometry (a) and (b), and in the second (c) and (d), to decrease point density.

The reduction varied for the first geometry clouds to test point density decrease, as shown in Figure 6.21. In contrast to increasing the density, increasing the grid box size is necessary to reduce the point density. Through the selected areas in Figure 6.20 (c) and (d), new point clouds were created with the respective reduced point density zone. Figure 6.22 shows these reductions in both clouds from the second geometry.

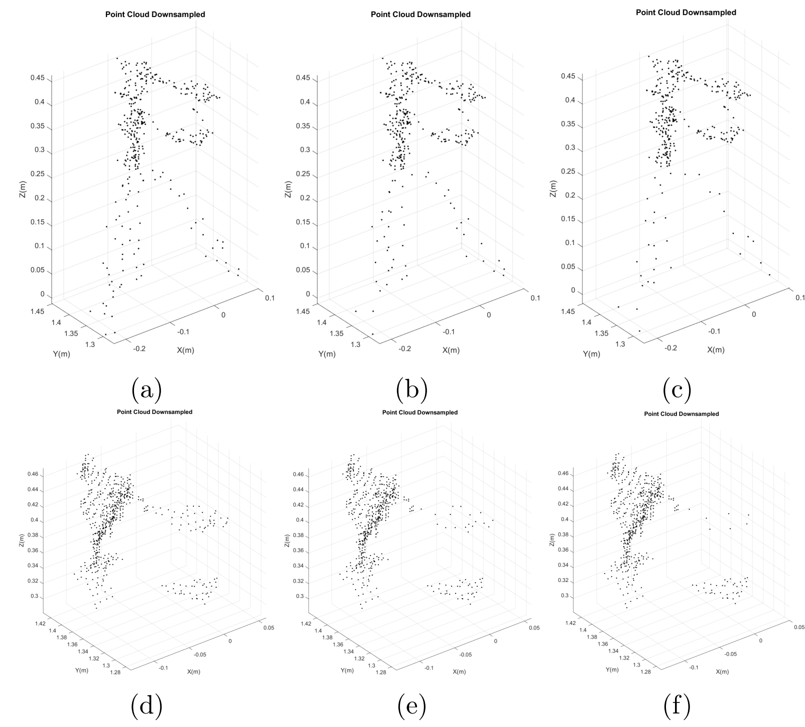


Figure 6.21: Variations of the point density decrease for the full object (a), (b) and (c). Variations of the point density decrease for the ROI object (d), (e) and (f). Both variations from the first geometry.

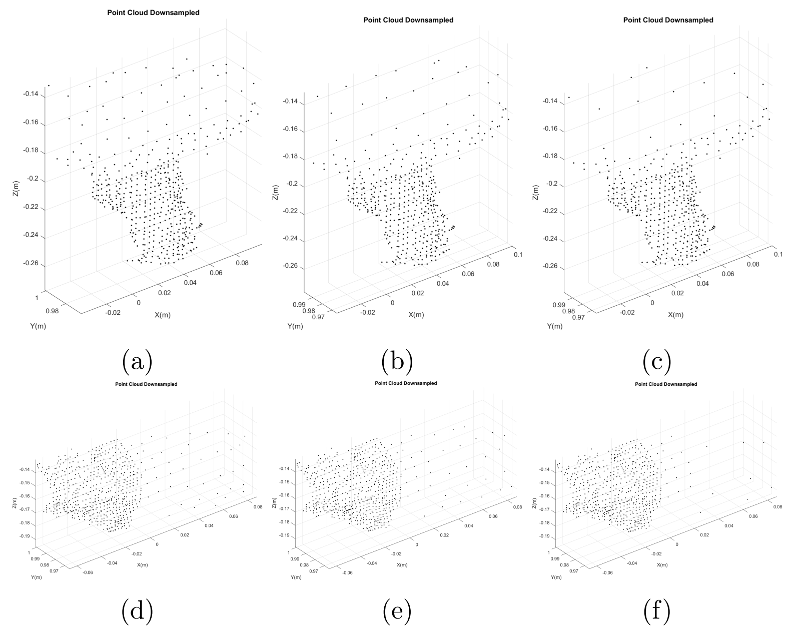


Figure 6.22: Variations of the point density decrease for the full object (a), (b) and (c). Variations of the point density decrease for the ROI object (d), (e) and (f). Both variations from the second geometry.

This action is again recorded in the browser, which displays in pink the area of points where a point density occurred, visible in Figure 6.23.

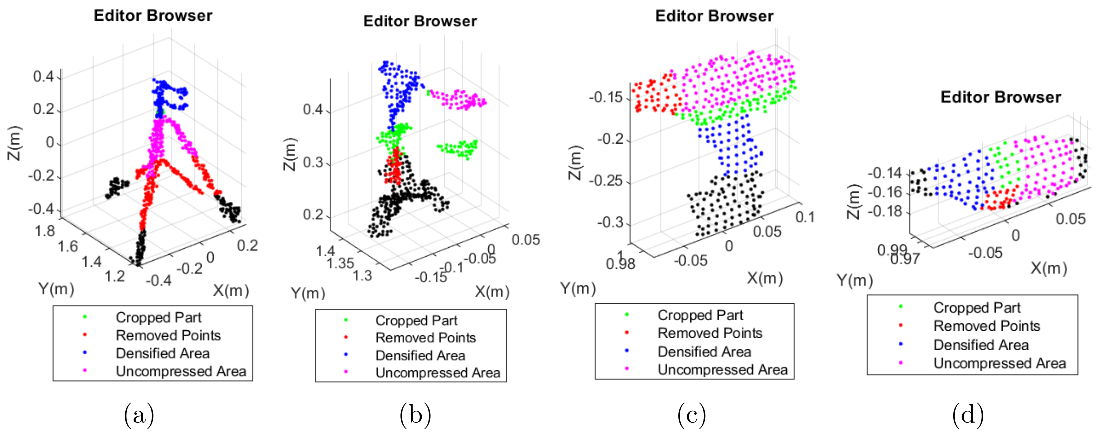


Figure 6.23: Editor browsers showing the areas with decreased point density in both clouds from both geometries.

6.4.6 Density Combinations

It was also necessary to test mixed-density areas, i.e. reduce the point density in an area previously increased and vice versa. Figure 6.24 shows an example of a density reduction in a previously increased area concerning the first geometry.

As explained in Section 5.2.2, the points selected in Figure 6.24 (a) and (d) will be replaced by points from the original cloud. A new point density will be assigned to it, resulting in the clouds shown in Figure 6.24 (b) and (e). In Figure 6.24 (c) and (f) is visible the updated browsers with the respective changes. Regarding the second geometry, Figure 6.25 shows the results of the density combination.

The areas in Figure 6.25 (a) and (d) were defined in order to encompass areas with density mixing. Selecting these points results in the clouds shown in Figure 6.25 (b) and (e). It is possible to observe these changes in Figure 6.25 (e) and (f) browsers.

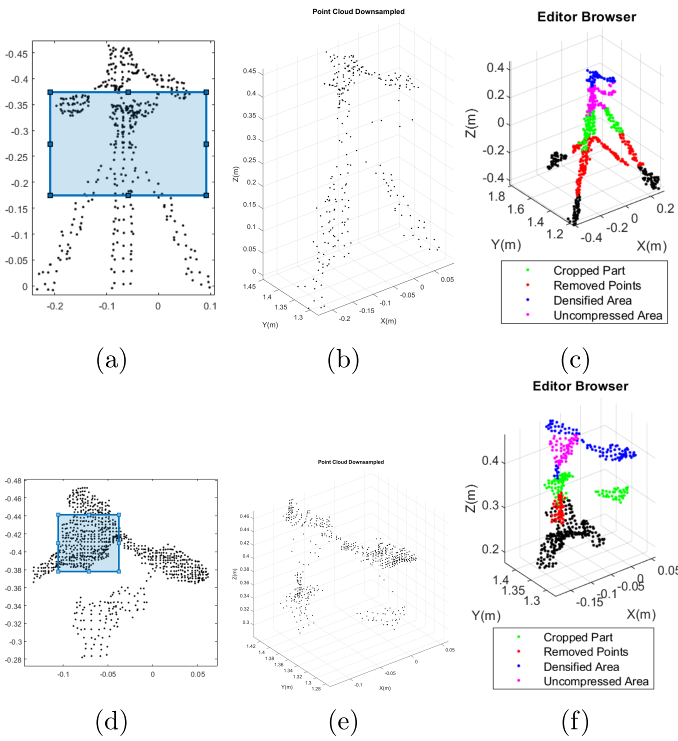


Figure 6.24: Object (a) and ROI (d) areas selected to decrease point density in mixed-density areas, their results (b) and (e), and browsers updated (c) and (f), respectively and regarding the first geometry.

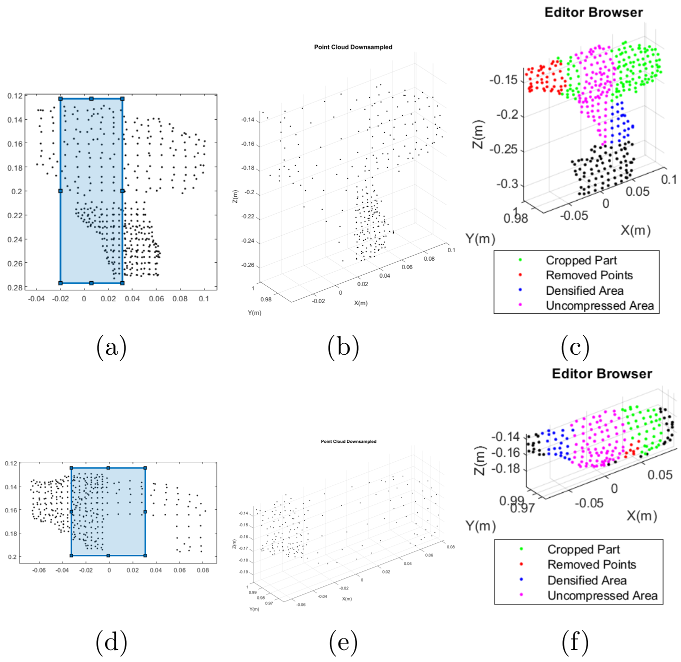


Figure 6.25: Object (a) and ROI (d) areas selected to obtain a decrease of point density in zones with previously altered densities, their results (b) and (e), and browsers showing which operations were made (c) and (f), respectively and regarding the second geometry.

Chapter 7

Conclusions

7.1 Work Conclusion

A prior definition of which points are to be scanned is necessary to perform Lased-based Vibrometry. For this purpose, an application was developed in which the user can define the object and its contours and extract point clouds.

The pillar of the work relies on the information captured by a Kinect sensor. RGB and depth images were the basis for creating all the results shown. From the beginning, the information in depth and RGB images allowed us to create the clouds that molded this project. The biggest challenge in developing the application was the development of its back end, taking a large percentage of the progress of the whole project since several methodologies were tested to achieve the results presented.

Finally, a target area was implemented where the user defined its boundaries, and all the information inside was extracted. Creating this target area changed the entire course of the project's development. As it was possible to extract the information through a three-dimensional cloud, it was also possible to do it in two dimensions following the same logic. Several steps request the user to draw the target areas initially, and then the appropriate treatment is done. This solution proved very basic in the user's view allowing easy and interactive application usage.

After the clouds were adequately obtained, it was possible to do several processing and treatment actions, such as increasing or reducing the density of its points and removing or selecting only parts of the cloud for point scanning. The tab responsible for this configuration shows the total number of points after each iteration, making the user aware of the number of points.

7.2 Future Work

The work developed in this dissertation is apt to implement both changes and improvements. Implementing a second, or even a third, Kinect camera should be considered to improve the information acquired. Adequately spaced and tilted to a certain degree between them allow an improved point cloud formation. Thus the object could be analyzed through an extensive view by the Kinects. Not capturing holes in the object could be prevented, and an increase in the number of points captured is guaranteed.

Improve the handling of the point cloud, mainly allowing the creation of areas of varied configuration. The user would no longer be restricted to rectangular areas and could define triangular, circular, and polygonal areas. Thus, in selecting points for the operations of choosing the ROI, of downsample (crop and remove points), and of editing the density (increase and decrease density), the user would have greater freedom of operation.

Finally, to develop the scanner's control component. The generation of analog signals for each drive of the two galvanometers allows the mirrors to rotate and, consequently, the orientation of the laser. The interaction with the signal acquisition system should also be developed to automate the entire analysis process.

References

[1] Polytec. "Full field vibrometers." *Available at* `https://www.polytec.com/int/vibrometry/products/full-field-vibrometers` (Visited on 01/05/2022).

[2] Polytec. "Modal Analysis." *Available at* `https://www.polytec.com/eu/vibrometry/modal-analysis` (Visited on 01/05/2022).

[3] A. Deris and I. Trigonis and A. Aravanis and E. K. Stathopoulou. "Depth cameras on UAVs: a first approach." *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, vol. 42, pp. 231-236, 2017.

[4] Oliver Wasenmüller and Didier Stricker. "Comparison of Kinect V1 and V2 depth images in terms of accuracy and precision.", 2016.

[5] Idaku Ishii and Kenkichi Yamamoto and Kensuke Doi and Tokuo Tsuji. "High-speed 3D image acquisition using coded structured light projection.", pp. 925-930, 2007.

[6] Hamed Sarbolandi and Damien Lefloch and Andreas Kolb. "Kinect range sensing: structured-light versus time-of-flight Kinect.", *Computer Vision and Image Understanding*, 2015.

[7] Parsa Mirdehghan and Wenzheng Chen and Kiriakos N Kutulakos. "Optimal Structured Light à la Carte.", pp. 6248-6257, 2018.

[8] Fabio Menna and Fabio Remondino and Roberto Battisti and Erica Nocerino. "Geometric investigation of a gaming active device." *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 8085, 2011.

[9] Choubik Youness and Mahmoudi Abdelhak. "Machine learning for real time poses classification using Kinect skeleton data." *Journal Proceedings - Computer Graphics, Imaging and Visualization: New Techniques and Trends, CGiV 2016*, pp. 307-311, 2016.

[10] Han, Jungong and Shao, Ling and Xu, Dong and Shotton, Jamie. "Enhanced Computer Vision With Microsoft Kinect Sensor: A Review." *Journal IEEE Transactions on Cybernetics*, vol 43, 2013.

[11] Zhang, Zhengyou. "Microsoft Kinect Sensor and Its Effect." *IEEE Multimedia - IEEEMM*, vol.29, pp. 4-10, 2012.

[12] Michael J. Landau and Benjamin Y. Choo and Peter A. Beling. "Simulating Kinect infrared and depth images.", *IEEE Transactions on Cybernetics*, vol. PP, pp. 1-14, 2016.

[13] E. H. Bokelberg and H. J. Sommer III and M. W. Trethewey. "A Six-degree-of-freedom Laser Vibrometer, Part II: Experimental Validation.", *Journal of Sound and Vibration - J SOUND VIB*, vol. 178, pp. 655-667, 1994.

[14] Joline Dank. "Zero mass loading effect and compensation in vibration analysis". *Available at* `https://www.polytec.com/fileadmin/website/vibrometry/pdf/OM_AN_VIB_G_026_Mass_loading_E_52000.pdf` *(Visited on 01/05/2022).*

[15] A. B. Stanbridge and D. J. Ewins. "Modal testing using a scanning laser doppler vibrometer.", *Mechanical Systems and Signal Processing*, vol. 13, 2002.

[16] Polytec Website. "Basic principles of laser Doppler vibrometry." *Available at* `https://www.polytec.com/us/vibrometry/technology/laser-doppler-vibrometry` (Visited on 01/05/2022).

[17] C. J. D. Pickering and N. A. Halliwell and T. H. Wilmshurst. "The laser vibrometer: a portable instrument." *Journal of Sound and Vibration*, vol. 107, pp. 471-485, 1986.

[18] Guangsheng Chen, Yi Zhang, Peiyun Tian. "An error compensation method based on machine vision for laser-processing systems with galvanometers.", *Applied Physics B*, vol. 127, 2021.

[19] Hasan Ahmed and Ali Mohsin and Seung-Chan Hong and Jung-Ryul Lee and Jeong-Beom Ihn. "Robotic laser sensing and laser mirror excitation for pulse-echo scanning inspection of fixed composite structures with non-planar geometries." *Journal Measurement*, vol. 176, pp. 109, 2021.

[20] M. A. Stalne and L. D. Mitchell and R. L. West. "Positional calibration of galvanometric scanners used in laser Doppler vibrometers.", *Journal Measurement*, vol. 28, pp. 47-58, 2000.

[21] Polytec Website. "Areas of application. Non-contact vibration measurment." *Available at* `https://www.polytec.com/int/vibrometry/areas-of-application` (Visited on 01/05/2022).

[22] Polytec. "PSV-400 Scanning Vibrometer." *Available at* `http://hysen.cafe24.com/wp-content/uploads/2019/10/OM_DS_PSV-400_2011_05_E.pdf?ckattempt=1` *(Visited on 01/05/2022).*

[23] Gomes, Rafael Martins. "Automatic orientation system for a laser vibrometer.", 2017.

[24] Polytec. "OFV-5000 Vibrometer Controller Datasheet." *Available at* `https://www.polytecstore.fr/polytec_images/documents/oms/om_ds_ofv-5000_e_42346.pdf` *(Visited on 01/05/2022).*

[25] Polytec. "OFV-505 Sensor Head Datasheet." *Available at* `https://www.atecorp.com/atecorp/media/pdfs/data-sheets/polytec-ofv-505.pdf?ext=.pdf` *(Visited on 01/05/2022).*

[26] MathWorks. "Image acquisition toolbox support package for Kinect for Windows sensor." *Available at* `https://www.mathworks.com/matlabcentral/fileexchange/40445-image-acquisition-toolbox-support-package-for-kinect-for-windows-sensor` (Visited on 01/05/2022).

[27] MathWorks. "Comparing GUIDE and App Designer." *Available at* `https://www.mathworks.com/products/matlab/app-designer/comparing-guide-and-app-designer.html` (Visited on 01/05/2022).

[28] MathWorks. "Point cloud from Kinect for Windows." *Available at* `https://www.mathworks.com/help/vision/ref/pcfromkinect.html` (Visited on 01/05/2022).

[29] MathWorks. "Create video input object." *Available at* `https://www.mathworks.com/help/imaq/videoinput.html` (Visited on 01/05/2022).

[30] MathWorks. "Immediately return single image frame." *Available at* `https://www.mathworks.com/help/imaq/getsnapshot.html` (Visited on 01/05/2022).

[31] MathWorks. "Plot 3-D point cloud." *Available at* `https://www.mathworks.com/help/vision/ref/pcshow.html` (Visited on 01/05/2022).

[32] MathWorks. "Find points within a region of interest in the point cloud." *Available at* `https://www.mathworks.com/help/vision/ref/pointcloud.findpointsinroi.html` (Visited on 01/05/2022).

[33] MathWorks. "Select points in point cloud." *Available at* `https://www.mathworks.com/help/robotics/ref/pointcloud.select.html` (Visited on 01/05/2022).

[34] MathWorks. "Morphologically close image." *Available at* `https://www.mathworks.com/help/images/ref/imclose.html` (Visited on 01/05/2022).

[35] MathWorks. "Find edges in 2-D grayscale image." *Available at* `https://www.mathworks.com/help/images/ref/edge.html` (Visited on 01/05/2022).

[36] MathWorks. "Downsample a 3-D point cloud." *Available at* `https://www.mathworks.com/help/vision/ref/pcdownsample.html` (Visited on 01/05/2022).