POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

# Anomaly Detection in Point Clouds with Local Polynomial Approximation & Intersection of Confidence Intervals

Tesi di Laurea Magistrale in
Computer Science and Engineering

Author: **Riccardo Nobili**

Student ID: 946763
Advisor: Prof. Giacomo Boracchi
Co-advisors: Diego Stucchi, Luca Magri
Academic Year: 2021-22

*Alla mia famiglia e a chi c'è sempre stato.*

# Abstract

Nowadays point clouds are a very powerful instrument that allows to have very detailed scans of objects and surfaces of every kind, from the smallest and common objects to big and complex infrastructures like buildings, bridges and roads. Thanks to the great precision of the scanners, they are able to catch the smallest details and geometrical features and so they can offer a very simple and easily understandable representation of the scanned objects. Such technology has a wide range of applications and the scanning of infrastructure for inspection purposes is probably the most common one. Routine checks on buildings and the monitoring of their conditions are a critical task, but necessary in order to prevent possible catastrophic failures. Thanks to the accuracy of such scans, it is possible to precisely capture the underlying geometry of objects and to use point clouds to search for defects and identify malformations on the analyzed surfaces.

In this work, we exploit a technique mainly used for image denoising and signal restoration, called Local Polynomial Approximation using Intersection of Confidence Intervals (LPA-ICI), to identify defects on surfaces represented as point clouds; in particular, surfaces like columns and walls have been analysed to find possible damages and defects with very good results. The developed method is able to identify different right rectangular prisms over the point cloud, called Directionl Neighbours (DN), whose size is dependent on whether they are in areas characterized by sharp features or not. Based on how many DNs they belong to, our algorithm assign an anomaly score to each point: the less the number of DNs it falls into, the higher its anomaly score will be.

The importance of working in this field relies on the fact that techniques like the one described in this thesis are quite innovative: there are a lot of works dealing with the search of defects and malformations in 2D representations (images) of objects, but very few on their 3D representation. This is probably due to the lack of comprehensive 3D datasets that are explicitly designed for the unsupervised detection and localization of anomalies. In addition to this, such technique may allow to speed up and make safer operations involved in the searching of defects on structures that usually requires people to work in dangerous situations, potentially exposed to hazardous conditions.

**Key words:** Point cloud, Anomalies, Unsupervised, Local Polynomial Approximation, Intersection of Confidence Intervals, Anomaly score

# Abstract in lingua italiana

Oggigiorno le point cloud sono uno strumento estremamente potente che permette di avere scannerizzazioni piuttosto dettagliate di oggetti e superfici di ogni tipo, dai più piccoli e comuni oggetti alle grandi infrastrutture come palazzi, ponti e strade. Grazie alla loro elavata precisione, gli scanner utilizzati sono in grado di catturare tutti i più piccoli dettagli e caratteristiche geometriche. Questo permette di offrire una rappresentazione molto semplice e facilmente comprensibile degli oggetti scansionati. Tale tecnologia ha una vasta gamma di applicazioni e la scansione di infrastrutture a scopo di ispezione è probabilmente la più comune: i controlli di routine sugli edifici e il monitoraggio delle loro condizioni sono un compito critico, ma necessario al fine di prevenire possibili guasti catastrofici. Grazie all'accuratezza di tali scansioni è possibile catturare con precisione la geometria sottostante agli oggetti e utilizzare le point cloud per ricercare difetti, come crepe e malformazioni, sulle superfici analizzate.

In questo lavoro di tesi abbiamo sfruttato una tecnica utilizzata principalmente per il denoising di immagini e la ricostruzione di segnali, ossia la Local Polynomial Approximation using Intersection of Confidence Intervals (LPA-ICI), per identificare difetti su superfici rappresentate come point cloud. In particolare sono state analizzate superfici tra cui colonne e muri col fine di ricercare possibili difetti, portando ad ottenere ottimi risultati. Il metodo sviluppato è in grado di identificare diversi parallelepipedi sulla point cloud analizzata, chiamati Directionl Neighbours (DN), le cui dimensioni dipendono dal fatto che questi si trovino o meno in aree caratterizzate da geometrie spigolose. In base al numero di DNs in cui sono coinvolti, ad ogni punto viene assegnato un anomaly score: minore è il numero di DNs in cui un punto ricade, maggiore sarà il suo anomaly score.

L'importanza di lavorare in questo ambito si basa sul fatto che la tecnica descritta in questa tesi è piuttosto innovativa: ci sono molti lavori che si occupano della ricerca di difetti e malformazioni in rappresentazioni 2D (immagini) di oggetti, ma molto pochi su quella 3D. Questo è probabilmente dovuto alla mancanza di dataset di point cloud esplicitamente progettati per il rilevamento e la localizzazione non supervisionata delle anomalie. Oltre a questo, tale tecnica può permettere di velocizzare e rendere più sicure le operazioni di ricerca di difetti su strutture che di solito richiedono al personale ad-

detto di lavorare in situazioni pericolose, riducendo così la loro esposizione a situazioni potenzialmente critiche.

**Parole chiave:** Point cloud, Anomalie, Non-supervisionato, Local Polynomial Approximation, Intersection of Confidence Intervals, Anomaly score

# Acknowledgements

First of all I would like to thank Professor Boracchi for supporting me and for always being ready to give me advice and new ideas. I would also like to thank my supervisors Diego Stucchi and Luca Magri for their support and commitment shown towards me.

Finally, I would like to express my deep gratitude to my family. Whatever you have done or said has brought me to where I am today. Thank you for always supporting me in every occasion and decision.

Milan, April 28, 2022

*Riccardo Nobili*

# Contents

# 1 | Introduction

## 1.1. General overview

Point clouds are a particularly useful tool to faithfully represent the surfaces of objects and spaces. One of their peculiarities is that they are able to define surfaces without having to resort to meshes, like triangular ones, or mathematical formulas. Usually 3D laser scanners and LiDAR (Light Detection and Ranging) are exploited to obtain samples of the object's surface in order to have a simple and very detailed representation: they calculate how long it takes for beam of lights to hit an object or surface and reflect back, computing in this way the distance of the surface from the laser scanner. Each point in the point cloud can hold information, called components, which contains a value that describes the point. Components may include X, Y, and Z coordinates as well as information about the intensity, color, time, and many more.


Figure 1.1: An example of a 3D laser scanner (left), a LiDAR scanner (center) and a portable 3D laser scanner (right)

Point clouds are used wherever objects need to be precisely captured and digitized and can be applied in different fields thanks to their versatility. In the automotive industry, for instance, they can be used for the design and optimization of production parts, helping in the creation of new prototypes. In the construction industry they are used to scan buildings for different purposes, like helping in the creation of Building Information Models

(BIM) [1] or to perform routine condition assessment [2] and inspections after extreme events that could have damaged the structures [3, 4]. Non Destructive Tests (NDT) techniques are generally used to check the state and condition of structures: they are based on methods that allow analysis without altering what is being analyzed. These techniques are very useful and offer a great level of accuracy in identifying defects and damages thanks to the use of technologies that, for example, are based on ultrasonic waves, x-rays and magnetic fields. However the biggest drawbacks of these advanced technologies are based mainly on their very high cost and the need to adequately train technicians on the use of such tools. Among the NDT techniques there is also the so called Visual Testing (VT) which consists in the simple observation of the object being analyzed by an expert. Differently from the other NDT techniques, Visual Testing has clearly reduced costs as it does not require the need to use particular technologies, but at the same time it requires that there is great competence on the part of the technician who is performing the visual inspection and that they have to be physically close to what is being analyzed. This, in the case of the analysis of damaged structures, could represent a risk for the subject that could be exposed to hazardous conditions. Moreover, the results of the analysis could be influenced by the person who performs the analysis itself, introducing a subjective component that could be very dangerous especially in the field of structural health monitoring.

Point clouds offer in this context a great alternative to these technologies, especially considering that in recent years there have been great improvements in their ability to capture the smallest details and in their accuracy. In fact, they allow to have a digitized representation of what needs to be analyzed without the need to incur high costs and allow technicians to scan at long distances, keeping themselves away from potentially dangerous locations and without excessively reducing the quality of the final result. In addition to this, another advantage of point clouds in this context is that when there is the need to scan buildings such as schools, hospitals or sporting venues, they are not required to be shut down, being point clouds a non-intrusive way to measure building or object properties. The search for defects is clearly not only of interest to the civil sector: any object, from the simplest and most everyday to the most complex and technical, can be scanned to create a digitized version via a point cloud. This is very useful when there is the need to carry out quality checks on the shape of the objects that are produced, looking for possible malformations and anomalies that can cause malfunctions in the product in question [5].

## 1.2. Thesis structure

This thesis is structured as follow:

- **Chapter 2** presents the problem formulation
- **Chapter 3** presents the state of the art in the field of the detection of anomalies and defects on the surface of objects represented as point clouds.
- **Chapter 4** presents a background knowledge regarding the theory of the Local Polynomial Approximation and the Intersection of Confidence Intervals.
- **Chapter 5** presents the description of the approach that has been followed, focusing the attention on the implementation of the model.
- **Chapter 6** presents the result of the application of the developed algorithm on some point clouds either synthetic or obtained through a laser scanner.
- **Chapter 7** describes the final conclusions and possible future applications and developments.

# 2 | Problem formulation

In this chapter we provide a formal description of the problem addressed in this thesis. In particular the attention will be focused on the definition of the kind of data provided as input to the algorithm, its output, the goals/desiderata regarding what the algorithm should do and finally the challenges that have to be faced.

## 2.1. The input

The input of the developed algorithm is a noisy point cloud $P = \{p_i, i = 1, \dots, I\}$ of the form

$$p_i = q_i + \eta_i \qquad\qquad p_i, q_i, \eta_i \in \mathbb{R}^3 \tag{2.1}$$

where $q_i$ is a point that belongs to the noise-free point cloud $Q$ and $\eta_i \sim \mathcal{N}(\mu, \Sigma)$ is the i.i.d. zero-mean ($\mu = [0, 0, 0]^T$) Gaussian white noise with $\Sigma = \sigma^2\mathbf{1}$ where $\mathbf{1}$ is the 3x3 identity matrix.

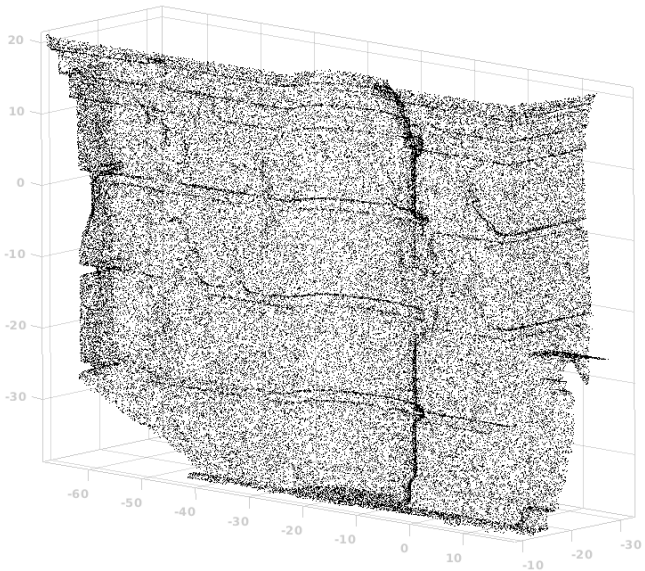



Figure 2.1: The point cloud representation of a damaged wall

Each point belonging to the point cloud is identified by a column vector with 3 components that represent its position in the space. The representation defined in (2.1) highlights the idea that each point that belongs to the point cloud $P$ can be seen as a noisy sample of the surface $S$ identified by the noise-free point cloud $Q$.

## 2.2. The output

The algorithm should be able to provide, for each point $p_i \in P$, a score $a_i$ that specifies how anomalous it is. It is reasonable to think that this allows to identify the areas that probably contain some sort of damage as these should be characterized by points with a high anomaly score.

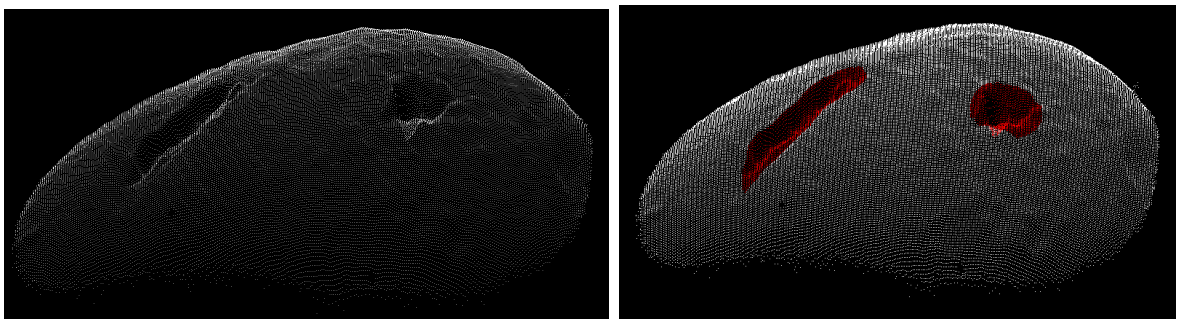


Figure 2.2: The point cloud representation of the surface of a damaged potato. On the left the input point cloud. On the right the input point cloud where the identified defects have been colored.

Formally:

$$a_i = \mathcal{A}_P(p_i)$$

where $p_i \in P$ and $\mathcal{A}_P : \mathbb{R}^3 \to [0, 1]$ is a function that, given the spatial position of a point $p_i$ belonging to the point cloud $P$, returns a score $a_i \in [0, 1]$. The anomaly score of each point depends on the point cloud $P$ the point belongs to and, specifically, to the number of points that constitute it. This dependence will be properly explained in section 5.5.

## 2.3. The goals and desiderata

The final objective of the developed algorithm is to detect with high fidelity the damaged areas of a surface represented as point cloud. In order to do this the algorithm exploits the LPA-ICI technique to assign to each point an anomaly score which has to be high for the

points belonging to damaged areas and low for all the others. In order to properly evaluate the performance of the algorithm, it is necessary to use some test point clouds and see how the algorithm performs on them. The ideal condition would be if the points of the dataset were labelled: in this situation it would be possible to calculate the performance of the algorithm referring to classical detection metrics. Particular attention should be paid, in the case of the search for defects, to false negatives: it would be more dangerous not to identify a defective area than to mark as defective an area that in reality is not.

## 2.4. The challenges

One of the first biggest challenges that we have to face is the lack of specific point cloud datasets for damage detection. In fact most of the available datasets are unlabelled point clouds that represent structures with defects and malformations. In order to test the performance of the algorithm it has been necessary to manually insert the labels to the points of some point clouds in order to compute some performance metrics. However, only recently, a useful dataset for anomaly detection in point clouds has been published and we have been able to use it to test the developed algorithm. Another element that represents a challenge within this context is the fact that it is not known a priori whether the point cloud given as input to be analyzed actually has damages and defects or not, leading to the necessity of using a completely unsupervised approach. An additional challenge to face is the one regarding the large number of points that make up a point cloud, especially if high-precision scanners that generate thousands of points are leveraged. This large amount of data to be analyzed requires the development of an algorithm efficient enough to process them in a sufficiently short time. Last but not least, the developed method requires to estimate the variance $\sigma^2$ of the noise of the point cloud that is analyzed and its surface sample density $\delta$.

# 3 | Related work

When talking about searching for surface defects and damages two possible approaches can be adopted: the first one is based on a geometric analysis and the second one on machine learning and deep learning. In this chapter we present the state of the art in finding defects in point clouds, focusing the attention on both the possible strategies.

## 3.1. Anomaly detection via geometric analysis of the surface

To carry out manufacturing quality assurance and control, laser scanners are frequently used to acquire points on the surface of the object, thus generating a point cloud from which a 3D representation can then be obtained using meshes. The obtained model can then be compared with the reference CAD model to verify the actual quality of what has been produced [6]. Son et al. [7] has exploited laser scanners to produce scan paths with different view angles, depth of field and occlusion and the quality of the obtained point cloud has been evaluated by checking the differences with respect to the CAD model. The quality control of the analyzed object can be also carried out during its production phase: Holzmond et al. [8] has presented a "certify-as-you-build" quality assurance system in additive manufacturing that compares the geometry of the printed object with the computer model to detect print errors in situ. Almamou et al. [9] has performed quality control of a steel bridge during its construction by using 3D point cloud technology, comparing it with the CAD model.

The measurement of quality carried out in this way does not concern only the manufacturing and production of objects. In particular in the construction industry there is widespread use of the so-called Building Information Modelling (BIM), i.e. a process involving various technologies (laser scanners in particular) that allow to obtain a 3D representation of the places, usually rooms, that are being scanned. The quality assurance in this context varies depending on whether it is performed on fabricated parts or on BIM: in the former case there is usually a design model that works as a ground truth to measure

how close the fabricated model is with respect to it, while the latter aims to identify the quality of the BIM with respect to the point cloud exploited to generate the model itself. In both cases, however, the point cloud plays a key role in the process since the better the quality of the point cloud, i.e. its ability to capture the geometry of reality, the better the result of the quality assessment. In this area, Anil et al. [10] propose a method to perform a deviation analysis in which they evaluate the quality of the 3D BIM generated by a point cloud by measuring how much it deviates from the point cloud itself. The same idea is applied also in the manufacturing field exploiting color-coding of geometric differences between the point cloud and the model to find defects and malformations [11].

The search for defects plays a fundamental and extremely critical role when it comes to infrastructure as it is necessary to properly assess the extent of damage to avoid possible disasters due to subsidence. The use of technologies such as Terrestial Laser Scanners (TLS), which constitute a simple and nondestructive method for the high-accuracy three-dimensional mapping of buildings and structures, is of enormous utility in this field thanks to their ability to capture even the smallest details of the surface. They are also able to obtain precise scans even at a safe distance from the structure. Defects on a surface can be identified by exploiting the difference in curvature between the areas without damages and those containing defects: Teza et al. [12] propose a method that, exploiting the scan obtained by TLS, calculates the Mean and Gaussian curvature of the surface and performs a piecewise comparison of the distributions; in this way, due to the fact that in the areas containing defects the distributions change in an evident way, the algorithm is able to adequately identify such defective areas. Another popular technique for identifying defects on a surface, closely related to curvature analysis, is to analyze the surface normals at different points, considering how much they diverge from a reference normal: the idea is based on the fact that defects tend to be characterized by sharp features that have normals to the surface with directions clearly different from those of defect-free areas. Erkal et al. [13] propose a method to identify different types of defects such as cracks, corrosion, ruptures and spallings in a point cloud by analyzing how much the direction of the normal computed at each point differs from reference one. In this study they identify three possible reference normals, depending on the available data: the first one is the normal to the surface of an area close to the considered point calculated using the k nearest points, the second one is the normal of an object for which its skeleton is known, while the third is the normal lying on the conjunction between a reference point (centroid) and the point currently analyzed.

Jovanvčević et al. [14] have developed a method that combines information about differences in curvature and direction of normals to identify defect areas in aircraft surfaces

represented as point clouds. For each point they take into consideration the normal to the plane tangent to the surface defined by the point cloud. This plane is obtained through the least-square plane fitting estimation algorithm considering the k nearest neighbours of the point in question. Curvature estimation, on the other hand, is calculated by exploiting the eigenvalues obtained during the tangent plane estimation. One by one each point is then considered as a seed and all other points are included in the cluster identified by that seed only if the differences between the normal and curvature of that point with respect to those of the seed are below a certain threshold. This leads to obtain, at the end of the computation, a large cluster considered as background and many small clusters in the regions that should contain defects and malformations. Torok et al.[15] have developed a crack-detection algorithm applied on Structure-from-Motion-derived point cloud for post-disaster damage detection. Their approach involves meshing the point cloud with the Poisson reconstruction method, rotating the object to align it with the vertical direction of the global reference system and finally calculating the relative angle between each triangulated normal vector and the reference vector, considering as defects those mesh elements for which the relative angle exceeded a certain threshold. A similar approach has been adopted by Kim et al. [16] to localize the spalling of a flat concrete block: the point cloud is rotated so that the vertical direction is aligned with the global one and for each point the normal vector is calculated using the covariance matrix obtained by PCA. In addition to calculating the relative angle between each normal and the reference normal, to define whether a vertex belongs to a defective area or not, the distance of each point from the reference plane estimated by the point cloud is also taken into account. Another approach that exploits the distance of each point with respect to the reference plane in order to detect defects in a point cloud is the one proposed by Valença et al. [17]. In their approach the mean and standard deviation of all distances are calculated and each vertex is classified as damage if its distance from the reference plane is larger than two standard deviations.

Another work that makes use of the relative angle between the normal vector of each point and the reference normal to identify structural damages is the one proposed by Erkal and Hajjar [13]. In this case the reference normal is estimated by identifying the skeleton structure of the point cloud and the detection results are further improved through the information retrieved from the color and intensity of each point in the point cloud. Mohammadi et al. [18] propose a method that exploits different approaches to check if a point in a point cloud belongs to a damaged area, computing three distinct geometrical surface descriptors. In particular one of these three descriptors is the approximation of the surface variation, computed using the eigenvalues of the covariance matrix of all the

vertices and their neighbours. The second geometrical surface descriptor is the variation of the vertex normal vector with respect to a reference vector, obtained using the underlying geometry of the point cloud. The last geometrical descriptor is the curvature variation within the point cloud in each principal direction. For all of these descriptor the PDF function is estimated and a point is considered as belonging to a damaged area if all three surface features had classified it a surface anomaly, depending on the PDF.

One last approach worth mentioning is the method proposed by Suchocki et al. [19] that allows to reduce the size of the dataset obtained exploiting TLS, focusing mainly on the areas most likely to contain defects. To down-sample the dataset they exploit both geometric changes of the surface and laser intensity values to improve the detection accuracy of defects which correspond to changes in the physicochemical properties of the surface.

## 3.2. Anomaly detection via Machine Learning

The search for defects falls within the scope of application of Machine Learning algorithms for segmentation and classification. In recent years, these algorithms have reached high levels of accuracy bringing greater interest in the search for new methods and techniques for the identification of malformations mainly in two areas, namely manufacturing and health monitoring of structures. Machine learning algorithms need a training phase in order to identify the best weights to make subsequent predictions: to do this they rely on training datasets related to the area of interest from which the most useful information for learning are extrapolated. The datasets that interest the Structural Health Monitoring however contain mainly, if not completely, data related to defect-free structures and this leads to have unbalanced datasets, not ideal for supervised learning. For this reason most of the machine learning algorithms exploited in this sector adopt an unsupervised approach. This approach involves the use of datasets without labels so that it is the algorithm itself that should be able to find patterns in the data, trying to distinguish the areas that contain defects from the unscathed ones. Obviously this type of learning presents various drawbacks among which the fact that the training phase exploits raw data without any prior knowledge and that the time needed to train the algorithm is usually much longer than the one required in a supervised approach.

In the work of Khoa et al. [20] they first perform dimensionality reduction on vibration data measured by different sensors and then healthy and damaged patterns are learned using either a supervised approach with Support Vector Machine or an unsupervised one with one-class Support Vector Machine depending on which data are available for the damaged states. Similarly, Santos et al. [21] propose a method to extract damage-sensitive features upon a time-series measured from an array of accelerometers and then perform damage detection using algorithms such as one-class support vector machine, support vector data description, kernel principal component analysis and greedy kernel principal component analysis. Gui et al. [22] has presented three optimization algorithms (grid-search, partial swarm optimization, and genetic algorithm) based on support vector machines for damage detection. In the work produced by Ghiasi et al. [23] a new strategy for structural damage detection is proposed using least square support vector machines based on a new combinational kernel function called thin plate spline Littlewood–Paley wavelet kernel to perform damage detection.

Closely related to the world of machine learning is the one of deep learning: the use of deep neural networks is quite common in the 2D world especially for classification and segmentation. In recent years the use of neural networks has been extended to the 3D world and in particular in the point clouds. Pixel and voxel-based methods are become very popular and so 3D point cloud datasets are usually converted into 3D-voxel grids or images in order to use in deep learning algorithms. An example of a neural network applied directly to point clouds is PointNet [24], a Convolutional Neural Network capable of performing classification, part segmentation, and semantic segmentation of objects represented as point clouds. Nasrollahi et al. [25] in their work adapt PointNet to detect surface defects using point cloud datasets from scanning bridge surfaces, reaching good results even if they exploited a small dataset for the training phase.

As mentioned above, the use of machine learning algorithms for defect detection is also of great interest in the manufacturing sector [26]. In this field, machine learning methods that work with 2D information (images) [27, 28, 29, 30] are more widespread than the ones working with 3D data: this is mainly due to the fact that only recently great progress has been made in 3D data processing techniques, obtaining high quality data without incurring excessive costs. Li et al. [31] have demonstrated how using algorithms such as Bagging of Trees, Gradient Boosting, Random Forest, K-nearest Neighbors and Linear Supported Vector Machine to search for defects on additively manufactured objects represented as point clouds far outperformed the z-difference based method, i.e. the distance along z-axis of the horizontal aligned source point cloud from the target point cloud. Trybałet al. [32] have developed an approach to identify defects on conveyor belts represented as

point clouds obtained by TLS, combining multiple methods to provide high robustness to the methodology. After performing dimensionality reduction via PCA, algorithms such as XGBoost Classifier, RANSAC and DBScan are used to perform detection of damaged areas in the belt top cover and identification of edges' imperfections.

# 4 | Background knowledge

In this section we will present the fundamental concepts regarding Local Polynomial Approximation and Intersection of Confidence Intervals. These two methods are at the basis of this thesis and it is therefore necessary to provide a clear and detailed explanation so that it will be possible to fully understand the work that has been done. The material for section 4.1 is mainly taken from [33], [34] and [35].

## 4.1. LPA-ICI

The combination of Local Polynomial Approximation and Intersection of Confidence Intervals has been proposed to address problems related to signal reconstruction, in particular for applications in image processing. The Local Polynomial Approximation is a non-parametric technique that allows to perform estimations using a polynomial data fit on the information contained within a sliding window, also called bandwidth or scale, which is a key parameter of this method. To find which is the best size for the bandwidth we apply the Intersection of Confidence Intervals technique: this algorithm allows to compare the estimates obtained through the LPA with different scales and selects the best one. In this way it is possible to identify the largest area close to the point of estimation that contains data for which the the local polynomial approximation assumptions fit well.

### 4.1.1. Local Polynomial Approximation in 2D

The observation model is defined as:

$$z(x) = y(x) + \eta(x) \tag{4.1}$$

where $z(x)$ is the observed image, $x \in X$ is the pixel index, $y(x)$ is the unknown original noiseless image and $\eta$ is the noise that, for the sake of simplicity, is assumed to be Gaussian noise $\eta \sim \mathcal{N}(0, \sigma^2)$. The goal is to obtain $\hat{y}(x)$, a good estimate of $y(x)$, given the sensed image $z(x)$ and $\eta$.

The non parametric approaches that can be used when performing denoising works either with a local method or a non-local one. The local approximation makes use of weights that depend on the distance $||x_0$ - $x_s||$ between the estimation point $x_0$ and the observation point $x_s$. For the non-local methods instead, the weights used to estimate $y_0$ depends on $y_s$ with some relation typically close to $||y_0$ - $y_s||_2$. Non-parametric approaches can be further divided depending on how many estimations are performed. In particular in the pointwise non-parametric approach the estimation of the noise-free signal $\hat{y}(x_0)$ is computed only for the considered point $x_0$, while in the multipoint approach such estimation is performed also for all the points $x_s$ involved in the estimation of $\hat{y}(x_0)$.

An example of local pointwise technique is the local pointwise weighted averages: given the pixel $x_0 \in X$, the estimate in $x_0$ is

$$\hat{y_h}(x_0) = \sum_{x_s \in X} w_h(x_0 - x_s) z(x_s) \tag{4.2}$$

where

$$w_h = \{w_h(x)\} \quad \text{s.t.} \quad \textstyle\sum_{x \in X} w_h(x) = 1$$

This can be seen as the 0-th order polynomial that performs least square fit

$$\hat{y_h}(x_0) = argmin_C \sum_{x_s \in X} w_h(x_0 - x_s)(z(x_s) - C)^2 \tag{4.3}$$

The weights in the Mean Square Error (MSE) are determined by the averaging window $w_h$ and the parameter $h$ scales the window with respect to a basic window $w$

$$w_h(x) = w(x|h)$$

Local Polynomial Approximation (LPA) is a non-parametric, local and pixelwise denoising algorithm. It defines linear filters that perform pixelwise polynomial fit on a certain neighbourhood. This algorithm determines the polynomial expression $p_{h,m}$ of a fixed order $m$ that fits in the best way the observations close to a fixed pixel, trying to minimize the weighted difference between such polynomial in the points $x_s$ around to $x_0$ and the observed value of the signal $z(x_s)$ in the same points [35]

$$p_{h,m} = argmin_{p \in P_m} \sum_{x_s \in X} w_h(x_0 - x_s)(z(x_s) - p(x_s))^2 \tag{4.4}$$

The estimation of the signal in $x_0$ is the value of this polynomial in $x_0$

$$\hat{y_h}(x_0) = p_{h,m}(x_0)$$

The LPA estimate can be also seen as the result of a convolution operation against discrete kernels $g_h$

$$\hat{y_h}(x_0) = (z \circledast g_h)(x_0) \tag{4.5}$$

These kernels can be computed starting from the order of the polynomial fit and its support as it will shown in Section 5.3.

LPA algorithm, being a local approach, requires for the estimation of the signal in each point the neighborhood of such point. The ideal neighbourhood is the one that constitutes the support of the pointwise minimal-MSE kernel estimators and it strictly depends on the local geometry of the signal that is usually unknown; this means that the size and shape of this ideal neighbourhood varies according to the considered point.

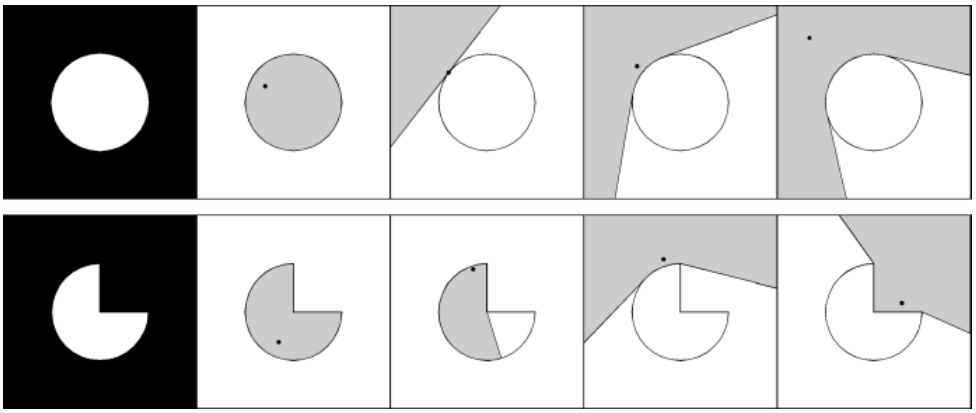


Figure 4.1: [33] Ideal neighborhood (grey area) of the estimation point (black dot). It has to be noticed how this neighbourhood is made of points similar to the considered one

Therefore the ideal estimation neighbourhood has to be approximated in some way: the best approach is to use a sectorial model that is able to minimize the differences with respect to the ideal one.

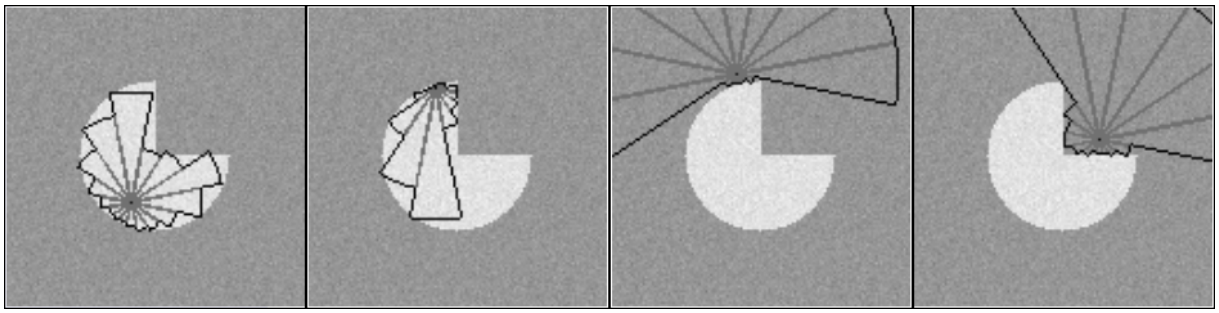


Figure 4.2: [33] Sectorial model to approximate in the best way the ideal estimation neighbourhood

There is therefore the need to define the size of each sector and in order to do this we apply the Intersection of Confidence Intervals.

### 4.1.2. Intersection of Confidence Intervals

The best way to obtain a good discretization of the ideal neighbourhood is by using a set of directional LPA kernels $\{g_{\theta,h}\}_{\theta,h}$ where $\theta$ defines the orientation of the kernel support and $h$ determines its scale.

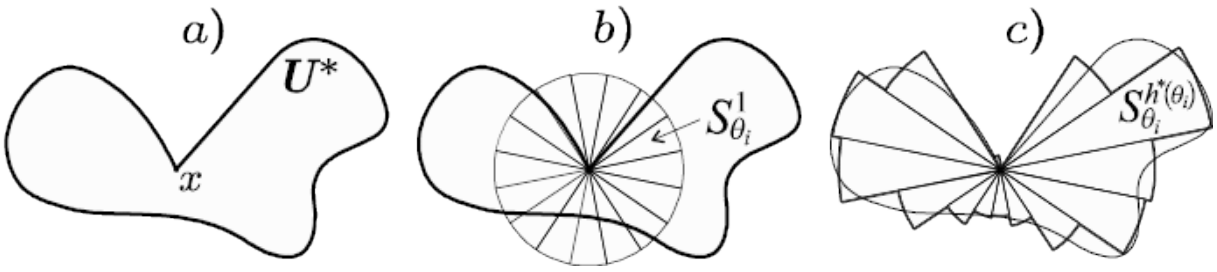



Figure 4.3: [33] **a)** Ideal neighbourhood, **b)** Directional kernels of length l, **c)** Discrete adaptive neighbourhood

The parameter $h$ is fundamental because it controls the amount of smoothing in the estimation: the larger it is the smaller will be the variance $\sigma^2$ (less noisy output), but the larger will be the bias $m$, and vice versa. Therefore this parameter controls the tradeoff between bias and variance in the LPA estimate $\hat{y}_h(x) = (z \circledast g_h)(x)$ and the goal of the ICI rule is to find, for each LPA kernel, the value for this parameter that better approximate the ideal one.

Let's start by analyzing the bias and variance of the LPA estimate [33]:

$$m_{\hat{y}_h(x)} = y(x) - (y \circledast g_h)(x) \tag{4.6}$$

$$\sigma^2_{\hat{y}_h(x)} = \sigma^2 ||g_h||_2^2 \tag{4.7}$$

For the MSE of the estimate it holds that

$$\begin{aligned} MSE(x) &= \mathbf{E}\left\{(y-\hat{y})^2\right\} = \mathbf{E}\left\{(\hat{y} - \mathbf{E}\left\{\hat{y}\right\} + \mathbf{E}\left\{\hat{y}\right\} - y)^2\right\} = \\ &= \mathbf{E}\left\{(\hat{y} - \mathbf{E}\left\{\hat{y}\right\})^2\right\} + 2\mathbf{E}\left\{(\hat{y} - \mathbf{E}\left\{\hat{y}\right\})(\mathbf{E}\left\{\hat{y}\right\} - y)\right\} + \mathbf{E}\left\{(\mathbf{E}\left\{\hat{y}\right\} - y)^2\right\} \end{aligned}$$

The middle term has a factor $\mathbf{E}\left\{\hat{y} - \mathbf{E}\left\{\hat{y}\right\}\right\} = \mathbf{E}\left\{\hat{y}\right\} - \mathbf{E}\left\{\hat{y}\right\} = 0$.
Thus the MSE can be written as

$$\begin{aligned} \mathbf{E}\left\{(y-\hat{y})^2\right\} &= \mathbf{E}\left\{(\hat{y} - \mathbf{E}\left\{\hat{y}\right\})^2\right\} + \mathbf{E}\left\{(\mathbf{E}\left\{\hat{y}\right\} - y)^2\right\} = \\ &= \mathbf{E}\left\{(\hat{y} - \mathbf{E}\left\{\hat{y}\right\})^2\right\} + (\mathbf{E}\left\{\hat{y}\right\} - y)^2 = \\ &= m^2_{\hat{y}_h(x)} + \sigma^2_{\hat{y}_h(x)} = l_{\hat{y}_h(x)} \end{aligned}$$

The goal is to find the value of $h$ that minimizes $l_{\hat{y}_h(x)}$: this will provide an ideal estimate of the real value $y(x)$.

It is possible to write, after some computation, the expressions for the variance and the bias as explicit functions of the scale parameter $h$. In particular [33]

$$\bar{m}_{\hat{y}_h(x)} = ah^\alpha \qquad\qquad \sigma^2_{\hat{y}_h(x)} = b^2 h^{-2\beta} \tag{4.8}$$

where $\bar{m}_{\hat{y}_h(x)}$ is the upper bound on the modulus of the bias. Accordingly, the upper bound of the MSE is

$$MSE(x) < \bar{l}_{\hat{y}_h(x)} = \bar{m}^2_{\hat{y}_h(x)} + \sigma^2_{\hat{y}_h(x)} = a^2 h^{2\alpha} + b^2 h^{-2\beta}$$

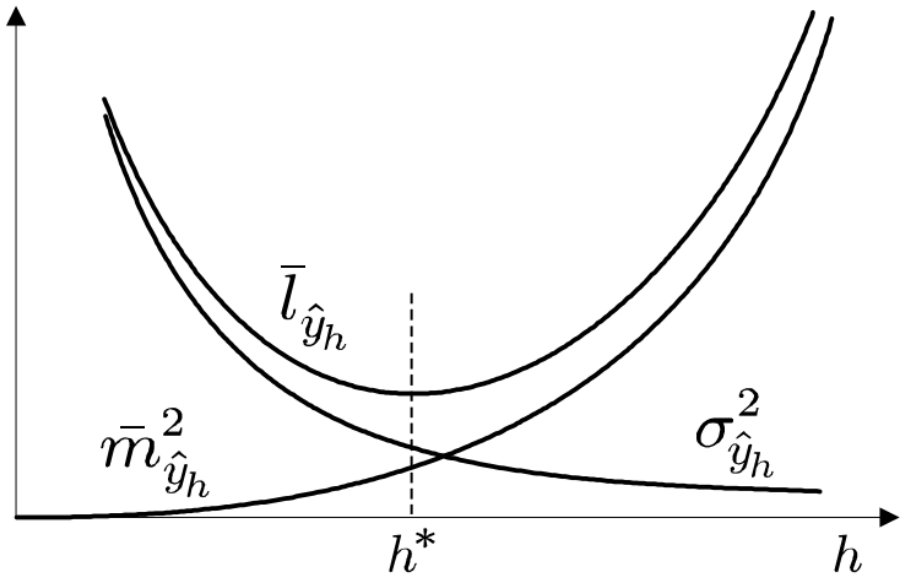



Figure 4.4: [33] Asymptotic bias-variance trade-off. In the plot the variance $\sigma^2_{\hat{y}_h(x)}$, the upper bound for the squared bias $\bar{m}^2_{\hat{y}_h(x)}$ and the upper bound for the MSE $\bar{l}_{\hat{y}_h(x)}$ are represented. $h^*$ is the ideal value for the scale that minimizes $\bar{l}_{\hat{y}_h(x)}$.

$h^*$ is the ideal scale that minimizes $\bar{l}_{\hat{y}_h(x)}$:

$$h^* = \arg\min_h \bar{l}_{\hat{y}_h(x)}$$

It can be found by solving

$$\partial_h \bar{l}_{\hat{y}_h(x)} = 0$$

which brings to the expression of the ideal scale $h^*$

$$h^* = \left(\frac{\beta b^2}{\alpha a^2}\right)^{\frac{1}{2\alpha+2\beta}}$$

where $\alpha$ and $\beta$ are constant values [33].

Substituting this in the equations (4.8) and performing the ratio between $\bar{m}_{\hat{y}_{h^*}(x)}$ and $\sigma^2_{\hat{y}_{h^*}(x)}$ we obtain

$$\frac{\bar{m}_{\hat{y}_{h^*}(x)}}{\sigma^2_{\hat{y}_{h^*}(x)}} = a^2 b^{-2} \frac{\beta b^2}{\alpha a^2} = \frac{\beta}{\alpha} = \gamma^2 \tag{4.9}$$

The ratio expressed in the equation (4.9) is a function that increases monotonically with $h$, and so it holds that

$$\bar{m}_{\hat{y}_h(x)} \begin{cases} < \gamma^2 \sigma^2_{\hat{y}_h(x)} & \forall h < h^* \\ > \gamma^2 \sigma^2_{\hat{y}_h(x)} & \forall h > h^* \end{cases} \tag{4.10}$$

The ICI rule has the objective of finding the adaptive scale $h^+(x)$ such that $\hat{y}_{h^+(x)}$ is the closest estimate with respect to the ideal one $\hat{y}_{h^*(x)}$. Let's focus the attention on the total estimation error $|y(x) - \hat{y}_h(x)|$. This is bounded by the sum of the moduli of the bias $m_{\hat{y}_h(x)}$ and the random error $r_{\hat{y}_h(x)} = \mathbf{E}\{\hat{y}_h(x)\}$ - $\hat{y}_h(x)$ which is a normal-distributed random variable $r_{\hat{y}_h(x)} \sim \mathcal{N}(0, \sigma^2_{\hat{y}_h(x)})$

$$|y(x) - \hat{y}_h(x)| \leq |m_{\hat{y}_h(x)}| + |r_{\hat{y}_h(x)}|$$

With probability $p = 1 - \lambda$ it holds that

$$|r_{\hat{y}_h(x)}| \leq \chi_{1-\frac{\lambda}{2}} \sigma_{\hat{y}_h(x)},$$

where $\chi_{1-\frac{\lambda}{2}}$ is a $(1 - \frac{\lambda}{2})$-th quantile of the normal distribution $\mathcal{N}(0, 1)$.

Accordingly, the following inequality holds with probability $p$:

$$|y(x) - \hat{y}_h(x)| \leq |m_{\hat{y}_h(x)}| + \chi_{1-\frac{\lambda}{2}} \sigma_{\hat{y}_h(x)}.$$

For $h \leq h^*$ we obtain from the inequalities (4.10) that

$$|y(x) - \hat{y}_h(x)| \leq (\gamma + \chi_{1-\frac{\lambda}{2}}) \sigma_{\hat{y}_h(x)} = \Gamma \sigma_{\hat{y}_h(x)} \qquad \Gamma = (\gamma + \chi_{1-\frac{\lambda}{2}}) \tag{4.11}$$

which can be also written as

$$\hat{y}_h(x) - \Gamma \sigma_{\hat{y}_h(x)} \leq y(x) \leq \hat{y}_h(x) + \Gamma \sigma_{\hat{y}_h(x)} \qquad \forall h \leq h^*(x)$$

This allows to define the following confidence interval $\mathcal{D}(h)$ for the estimate $\hat{y}_h(x)$:

$$\mathcal{D}(h) = [\hat{y}_h(x) - \Gamma \sigma_{\hat{y}_h(x)}, \hat{y}_h(x) + \Gamma \sigma_{\hat{y}_h(x)}] \tag{4.12}$$

More precisely, for each $h \leq h^*$ with a certain probability $p$ it can be stated that $y(x) \in \mathcal{D}(h)$. The width of this interval is a monotonically decreasing function of $h$ according to the expression of $\sigma_{\hat{y}_h(x)}$ defined in (4.8): the more $h$ increases the more it reduces.

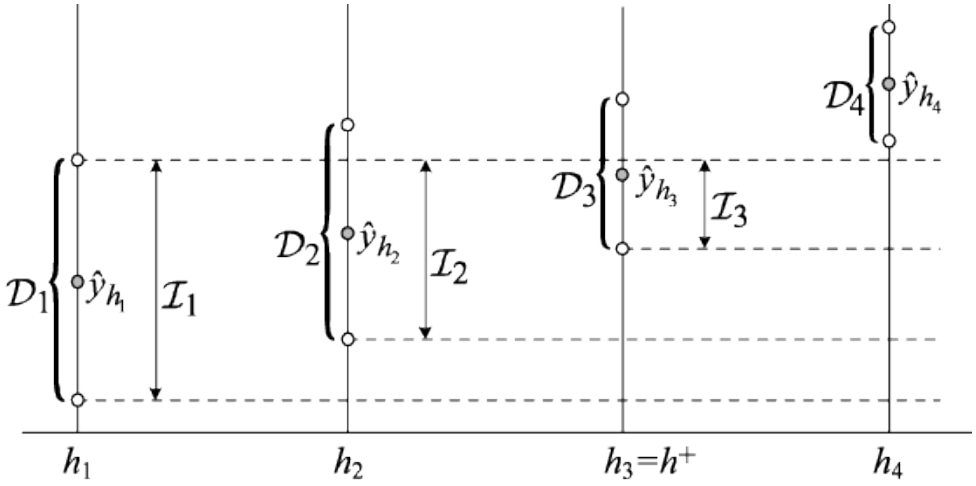



Figure 4.5: [33] The ICI rule. $h_3$ is selected as $h^+$ because it is the largest $h_j \in H = \{h_1, h_2, h_3, h_4\}$ for which all the $\mathcal{D}(h_i)$ with $h_i \leq h_j$ have a point in common.

Let's consider an increasing set of scales $H = \{h_1, ..., h_J\}$ where $h_1 < ... < h_J$ and the corresponding estimates $\left\{\hat{y}_{h_j}(x)\right\}_{j=1}^{J}$ which are normal-distributed random variable with variance $\sigma^2_{\hat{y}_{h_j}}(x)$. Remembering that for each $h \leq h^*$ with a certain probability $p$ the real value $y(x) \in \mathcal{D}(h)$, it can be said that with probability $p'$ all the $\mathcal{D}(h_j)$ with $h_j \leq h^*(x)$ have a point in common. Let's call $j^+$ the largest among the indexes $j$ such that all the $\mathcal{D}(h_i)$ with $i \leq j^+$ have a point in common. Notice that $h_{j^+} \geq h^{*-} \triangleq max\{h_j : h_j \leq h^*(x)\}$ and in particular all $\mathcal{D}(h_j)$ with $h_j \leq h^{*-}$ have a non empty intersection. Obviously the best value that can be selected for $h$ among all the $h_j \in H$ is $h^{*-}$ which is unknown; however, thanks to the fact that the confidence intervals shrink as $h_j$ increases, the ICI algorithm allows to find $h_{j^+}$ for which its corresponding estimate $\hat{y}_{h^+}(x)$ is ensured to be in a certain range from the true signal $y(x)$.

In particular $y(x) \in \bigcap_{h_j \leq h^{*-}} \mathcal{D}(h_j)$ and so, according to (4.11), $|y(x) - \hat{y}_{h^{*-}}| \leq \Gamma\sigma_{\hat{y}_{h^{*-}}(x)}$. Similarly, since $\mathcal{D}(h^{*-}) \bigcap \mathcal{D}(h^+) \neq \varnothing$, it holds that $|\hat{y}_{h^{*-}} - \hat{y}_{h^+}| \leq \Gamma\sigma_{\hat{y}_{h^{*-}}(x)} + \Gamma\sigma_{\hat{y}_{h^+}(x)}$. So at the end it can be said that:

$$|y(x) - \hat{y}_{h^+}| \leq 2\Gamma\sigma_{\hat{y}_{h^{*-}}(x)} + \Gamma\sigma_{\hat{y}_{h^+}(x)} \leq 3\Gamma\sigma_{\hat{y}_{h^{*-}}(x)} \tag{4.13}$$

If $H$ is sufficiently rich, $h^{*-} \simeq h^*$ and so $\sigma_{\hat{y}_{h^{*-}}(x)} \simeq \sigma_{\hat{y}_{h^*}(x)}$ which means that the adaptive estimate selected by the ICI rule $\hat{y}_{h^+}$ is at most $3\Gamma$ times the ideal deviation $\sigma_{\hat{y}_{h^*}(x)}$.

In conclusion the ICI rule can be stated as follow:
*Consider the intersection of confidence intervals* $\mathcal{I}_j = \bigcap_{i=1}^{j} \mathcal{D}_i$ *and let* $j^+$ *be the largest of the indexes* $j$ *for which* $\mathcal{I}_j$ *is non-empty,* $\mathcal{I}_{j^+} \neq \varnothing$ *and* $\mathcal{I}_{j^+ +1} = \varnothing$*. Then the adaptive scale* $h^+$ *is defined as* $h^+ = h_{j^+}$ *and the adaptive-scale kernel estimate is therefore* $\hat{y}_{h^+}(x)$*.*

The ICI rule can be applied to select the adaptive scale-kernel estimate $\hat{y}_{d,h^+}(x)$ given a set of LPA kernels $\left\{g_{d,h_j}\right\}_{j=1}^{J}$ and the estimate $\hat{\sigma}^2$ of the variance of the noise related to the observation (4.1) for each direction $d$.

After having obtained the best estimate for each direction the estimation of the signal in $x$ is obtained by fusing all of them with a convex linear combination where the weights are determined by the inverse of the directional estimates variance $\sigma^2_{\hat{y}_{h^+}(x)}$ (4.7): in this way the lesser noisy estimates will impact more thanks to the larger weights assigned to them

$$\hat{y}(x) = \sum_d \lambda_d(x)\hat{y}_{d,h^+}(x) \qquad\qquad \lambda_d(x) = \frac{\sigma^{-2}_{\hat{y}_{d,h^+}(x)}}{\sum_d \sigma^{-2}_{\hat{y}_{d,h^+}(x)}}$$

# 5 | The proposed algorithm

In this chapter we propose an explanation of the followed approach and a description of the developed algorithm. The attention will be focused on the theory of the Local Polynomial Approximation and Intersection of Confidence Intervals applied to 3D data, in particular to point clouds, and how such theory has been used to find damages and malformations.

## 5.1. The Local Coordinate System

The Local Coordinate System (LCS) is a system of coordinates that expresses the position of each point relatively to the position of another one taken as origin. The construction of the LCS is performed starting from the identification of the K nearest neighbours of each point $p_i \in P$: these points are necessary for the computation of the three principal components through the Principal Component Analysis (PCA). The $1^{st}$, $2^{nd}$ and $3^{rd}$ principal components obtained for the considered point $p_i$ are three versors, i.e. column vectors of unit norm, and are denoted respectively as $c_i$, $d_i$ and $e_i$. By construction, the plane identified by the components $c_i$ and $d_i$ is more or less locally parallel to the surface identified by the point cloud in $p_i$. However this is not a precise parallelism due to the exploitation of the PCA which allows to have a rough approximation of such surface, but this is enough for obtaining good results with the algorithm. Similarly the third component $e_i$ can be seen as the normal versor of the plane identified by the other two components. In this way a coordinate system which has $p_i$ as the origin of the axes $c_i$, $d_i$ and $e_i$ has been obtain and it is identified as $\mathcal{L}_i$.

To properly define the LCS, it is necessary to express the coordinates of each point belonging to the point cloud with respect to this newly created reference system. The local coordinates in $\mathcal{L}_i$ of each point $p_m \in P$ are denoted as $p_m^{\mathcal{L}_i} = \left[x_m^{\mathcal{L}_i}, y_m^{\mathcal{L}_i}, z_m^{\mathcal{L}_i}\right]^T$ and can

be obtained from the three principal components as

$$p_m^{\mathcal{L}_i} = \begin{bmatrix} c_i \\ d_i \\ e_i \end{bmatrix} (p_m - p_i)$$

It is easy to notice that $p_i^{\mathcal{L}_i} = [0,0,0]^T$, being $p_i$ in the origin of $\mathcal{L}_i$, while the versors are identified as

$$c_i^{\mathcal{L}_i} = [1,0,0]^T$$
$$d_i^{\mathcal{L}_i} = [0,1,0]^T$$
$$e_i^{\mathcal{L}_i} = [0,0,1]^T$$

## 5.2. Directional Neighborhoods

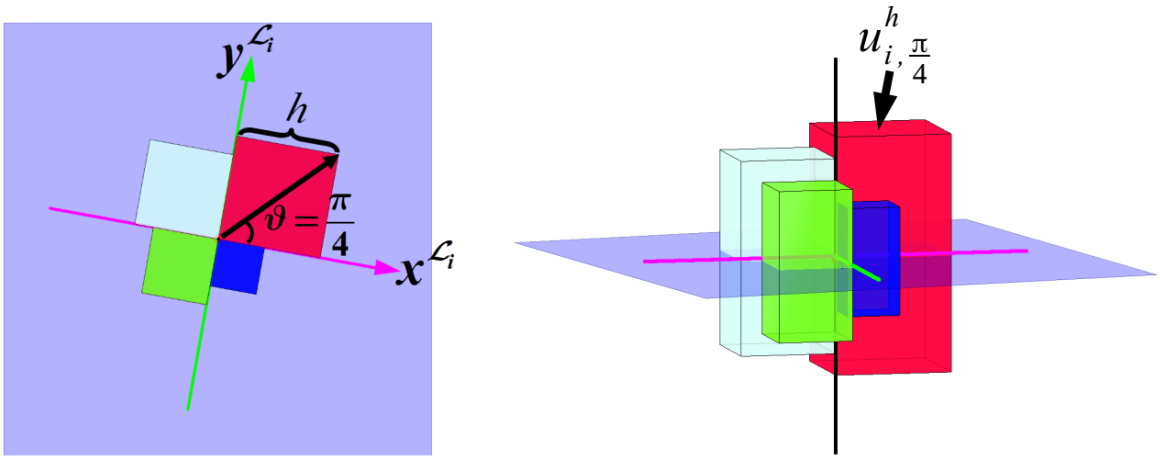



Figure 5.1: [36] An example of the directional neighborhoods of the point $p_i$. Considering the red one on the left image, it has size $h$ and its direction is the one of the first quadrant ($\theta = \frac{\pi}{4}$) on the plane defined by the first two principal axes $x^{\mathcal{L}_i} \equiv c_i$ (purple) and $y^{\mathcal{L}_i} \equiv d_i$ (green) of the LCA. On the right side a 3D representation of all the directional neighborhoods and the third principal axis $z^{\mathcal{L}_i} \equiv e_i$ (black) are represented.

Unlike images, point clouds live in the 3D world and clearly this is reflected also on the Directional Neighborhoods (DN) used in the LPA, which result to be subvolumes of $\mathbb{R}^3$. Each DN is shaped like a parallelepiped which base is a square of dimensions $h$x$h$, where $h$ is the parameter that, like in the 2D case, regulates the dimensions of every DNs. The height of each DN is strictly related to $\sigma$, which is the estimate of the variance of the

noise of the point cloud. Such estimation is performed following the technique defined in the Appendix 1 of [36]. Each DN is developed in height following the direction of the third principal component $d_i$ and this height is set equal to the maximum between $6\sigma$ and $2h$. This is done in order to consider also those noisy points $\pm 3\sigma$ far from the plane identified by the two main axes of the LCS. Similarly to the 2D world case (described in the subsection 4.1.1), the DNs used for point clouds allow the selection of particular points, i.e. those that are in the intersections between the DNs and the point cloud $P$. The adopted approach involves the use of four DNs, one for each quadrant identified by the first two principal components $c_i$ and $d_i$ of the LCS of the current point $p_i$ as shown in Figure 5.1. Each of the four DNs of the considered point $p_i$ is denoted as $u^h_{i,\theta}$ where $\theta$ identifies one of the four possible directions and $h \in \mathbb{R}^+$ denotes its size. Moreover, it results necessary to identify for every $u^h_{i,\theta}$ which points belong to its intersection with the point cloud $P$: let $M^h_{i,\theta}$ be the set of indexes of these points, such that

$$P \bigcap u^h_{i,\theta} = \{p_m, m \in M^h_{i,\theta}\}$$

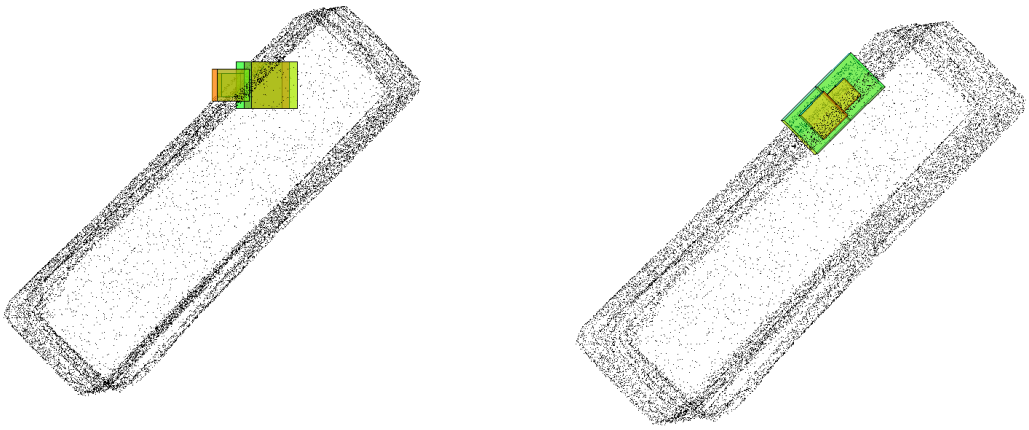


Figure 5.2: An example of the orientations of the DNs of a point $p_i$. The ones on the left are oriented as the Local Coordinate System: it can be clearly seen that this provides a rough estimate of the surface under the point cloud. On the right side the ideal orientation of the DNs.

## 5.3. Local Pointwise Polynomial Estimate

The DNs are fundamental in order to apply the LPA. Thanks to their intersection with the point cloud $P$, it is possible to obtain, for each DN, the points necessary for the fitting of the local polynomial model. In the 2D case presented in Section 4.1.2, the ICI rule needs the estimates of the value of a pixel obtained from the various DNs, in order to identify the best DNs that allow to obtain the best estimate. Similarly, in the 3D case

of point clouds, in order to identify the best DNs, we decide to exploit as estimate to be used in the ICI rule the elevation of the currently considered point $p_i$ with respect to the surface underlying the point cloud. This estimate will not be of interest to us, since we are only interested in the best DNs that lead to it. For this reason the objective of this step is to obtain estimations of the normal elevation of the surface $S$ under the point cloud $P$ at $p_i$ for each direction $\theta$ and for each value of $h$.

Let $g^h_{i,\theta}$ be the LPA-kernel for a particular point $p_i$, direction $\theta$ and size $h$. For each $k = 1, \dots, |M^h_{i,\theta}|$, such kernel is defined as [36]:

$$g^h_{i,\theta}(k) = \phi(k,:)\left(\phi^T\phi\right)^\dagger \phi(1,:)^T$$

where $\dagger$ is the matrix pseudo-inverse, $\phi$ is a matrix consisting of column vectors $\phi_{l,r}$ of length $|M^h_{i,\theta}|$:

$$\phi = [\phi_{0,0}, \dots, \phi_{l,r}, \dots]$$

Each of these column vectors is a bivariate monomial where $l$ and $r$ are integer values such that, for an order-$O$ LPA, the degree of each monomial $\phi_{l,r}$ is limited to $l + r \leq O$. The column vectors are obtained considering the coordinates $x^{\mathcal{L}_i}_m$ and $y^{\mathcal{L}_i}_m$ of each point $p_m$ for which $m \in M^h_{i,\theta}$:

$$\phi_{l,r} = \begin{bmatrix} (x^{\mathcal{L}_i}_{m_1})^l (y^{\mathcal{L}_i}_{m_1})^r \\ \vdots \\ (x^{\mathcal{L}_i}_{m_k})^l (y^{\mathcal{L}_i}_{m_k})^l \\ \vdots \end{bmatrix} \qquad \{m_k\}_{k=1}^{|M^h_{i,\theta}|} = M^h_{i,\theta}$$

The size of the matrix $\phi$ changes depending on the chosen order $O$: for instance in the case in which $O = 1$ the matrix $\phi$ will have size $|M^h_{i,\theta}| \times 3$. Moreover in the adopted approach $m_1 = i$, i.e. the first row of each column vector $\phi_{l,r}$ is related to the current point $p_i$.

Similarly to the 2D case (4.5), the polynomial approximation of the normal elevation of the underlying surface $S$ at $p_i$ with respect to $\mathcal{L}_i$ is defined as [36]:

$$\left(\tilde{z}^{\mathcal{L}_i}_i\right)^h_\theta = \sum_{k=1}^{|M^h_{i,\theta}|} z^{\mathcal{L}_i}_{m_k} g^h_{i,\theta}(k) \tag{5.1}$$

## 5.4. Size selection through the ICI rule

Generally surface defects and malformations are characterized by uneven and non-regular areas, identified by features that tend to be very sharp. By its very nature the ICI is able to define the size of each DN depending on how the points are distributed in the DN itself for its different possible dimensions $h$. Basically, in correspondence of a sharp feature the ideal DN will not expand excessively, i.e. it will not to go beyond the edge itself, because of the large bias that there would be between the points before and after the sharp feature. Indeed if a DN of size $h$ contains points characterized by a high bias the confidence interval related to that particular value $h$ will move away as shown in Figure 4.5. This prevents the DN to expand beyond the edge, thus limiting it not to go beyond the sharp feature.

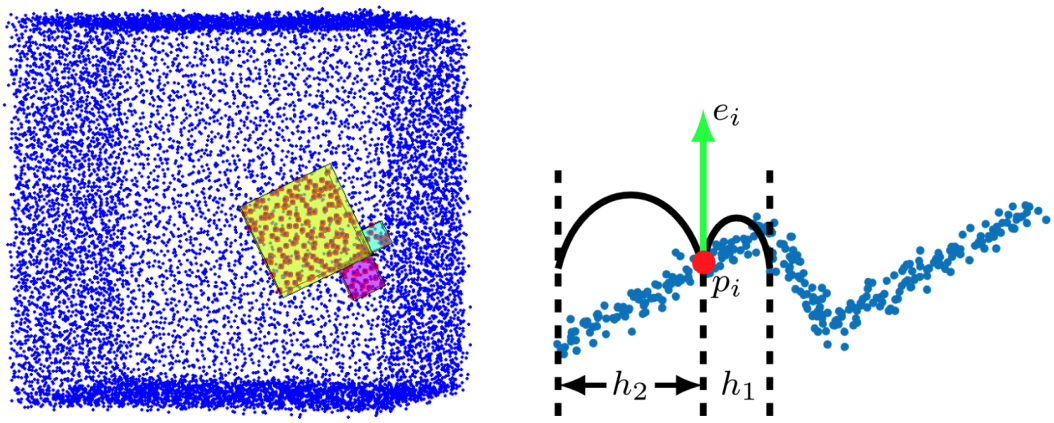



Figure 5.3: [36] On the left an example of the directional neighborhoods of a point close to an edge of a cubic point cloud is shown. It can be seen that the directional neighborhoods near the edge are smaller in size with respect to those extending into flatter areas and this is due to how ICI chooses the sizes for each of them. Specifically, the figure on the right shows in 2D how the size of the directional neighborhood to the right of $p_i$, i.e. the one close to the edge, is $h_1 1$, which is smaller than the size chosen for the directional neighborhood on the left side of $p_i$ which, since it faces a flat area, has a size $h_2 > h_1$, thus allowing it to expand more.

Let $H = \{h_1, \dots, h_J\} \subset \mathbb{R}^+$ be the set of possible values for the size of each DN sorted in ascending order, i.e. $h_1 < h_2 < \cdots < h_J$. Fixed a direction $\theta$, for each $h_j \in H$ the set of estimates $\left\{ \left(\tilde{z}_i^{\mathcal{L}_i}\right)_\theta^{h_1}, \dots, \left(\tilde{z}_i^{\mathcal{L}_i}\right)_\theta^{h_J} \right\}$ has to be computed (5.1) using the same polynomial order $O$ for all of them. The ICI technique works with the confidence intervals associated to each estimate, progressively intersecting them starting from the smallest size $h_1$ until their intersections become empty. In order to compute these intervals it is necessary to define the value of the variance of each estimator. It is interesting to note that this, since $M_{i,\theta}^h$ grows with the increase of $h$, will tend to reduce given the increasing amount of points that will be part of the DN necessary to perform the LPA. Referring to the

equation (4.7), the pointwise variance of each estimate is computed as

$$\left(\sigma_{i,\theta}^{h}\right)^2 = \sigma^2 ||g_{i,\theta}^{h}||_2^2$$

Having defined the variance, the confidence interval $\mathcal{D}_j$ for each $h_j \in H$ can be identified, taking into consideration the equation (4.12), as

$$\mathcal{D}_j = \left[\left(\tilde{z}_i^{\mathcal{L}_i}\right)_\theta^{h_j} - \Gamma\sigma_{i,\theta}^{h_j}, \left(\tilde{z}_i^{\mathcal{L}_i}\right)_\theta^{h_j} + \Gamma\sigma_{i,\theta}^{h_j}\right]$$

where $\Gamma \in \mathbb{R}^+$ is a threshold parameter that influences the choice of the best size defined by the ICI technique: the higher it is the easier will be to find non empty intersections, allowing the algorithm to choose higher values for the size $h$.

As explained in subsection 4.1.2, the ICI is able to identify $h^+$ among all the $h_j \in H$, which is the largest size for the considered direction $\theta$ of the current directional neighborhood before the intersection is empty. The corresponding directional neighborhood is identified as $u_{i,\theta}^+$.

## 5.5. Anomaly score

The last step of the developed approach consists in assigning to each point of the point cloud a score that tends to be high when this point is located in a defective area. Thanks to this approach, the areas that constitute defects or malformations on the surface of the object will be characterized by points that have a high anomaly score score. On the contrary, points belonging to defect free regions will be characterized by a low anomaly score.

The idea is to define a function $\mathcal{A}_P$ that takes as input a point $p_i \in P$ and returns a value $a_i \in [0, 1]$ that represents the anomaly score of the point. Formally

$$a_i = \mathcal{A}_P(p_i)$$

where $p_i \in P$ and $\mathcal{A}_P : \mathbb{R}^3 \to [0, 1]$ is a function that, given the spatial position of a point $p_i$ of the point cloud $P$, returns a score $a_i \in [0, 1]$.

For each point $p_i \in P$ four DNs $u_{i,\theta}^+$ have been obtained, one for each direction $\theta$. As mentioned earlier these DNs are such that they tend not to expand beyond the sharp features and this property is exploited in order to compute the anomaly score of each point. The damaged areas of a surface are represented by sharp features and this causes the directional neighborhoods of the points belonging to such areas to assume rather

small dimensions, not having the possibility, by construction, to expand excessively. On the contrary, the directional neighborhoods of points that are in defect-free areas, with few or even without any sharp features, tend to expand as much as possible.

The bigger the DN is, the larger will be the number of points belonging to the intersection between the point cloud and the DN itself: it can be observed that the points belonging to areas without defects will tend to be included in many DNs given their ability to expand as much as possible in such areas. Conversely, points that belong to defective areas will fall into a rather limited number of them. Thus, the lower the number of different DNs a point falls into, the higher must be its anomaly score. This highlights a relationship of inverse proportionality between these two values. For this reason, we adopt as anomaly score $a_i$ of a point $p_i \in P$ the reciprocal of the number of DNs in which $p_i$ falls into.

Such number, and consequently its anomaly score, is related to the point cloud $P$ and, specifically, to the number of points that constitute it. In particular, the points of a big point cloud are more likely to fall into a higher number of DNs than those of a point cloud consisting of fewer points. Consequently, the anomaly scores related to a large point cloud will tend to be higher than those of a smaller one.

# 6 | Experimental results

In this chapter we present the point cloud datasets that were used to test the ability of the developed algorithm to identify anomalous areas and the results that were obtained.

## 6.1. Parameters settings

| **Parameter** | **Description** |
|---|---|
| $\sigma^2$ | The variance of the Gaussian noise added to each point of the point cloud |
| $\mu$ | The mean value of the Gaussian noise added to each point of the point cloud |
| $K$ | Number of nearest neighbors to consider when computing the LCS |
| $O$ | Order of the polynomial for the LPA |
| $\Gamma$ | Threshold parameter that influences the choice of the best size defined by the ICI technique |
| $H$ | Set of the possible sizes $h_j$ from which the ICI technique will choose the best one |

Table 6.1: Parameters of the developed algorithm with their description

Table 6.1 shows all the parameters of the algorithm that must be defined to work properly. After various tests we have decided to assign to most of these parameters a fixed value for every test carried out: indeed we have observed that it has been possible to obtain very good results just fixing such parameters and modifying few of the others. In particular Table 6.2 shows the values that we have decided to assign to each parameter.

| Parameter | Chosen value |
|---|---|
| $\sigma^2$ | 0.001 |
| $\mu$ | 0 |
| $K$ | 80 |
| $O$ | 2 |

Table 6.2: Chosen values for some parameters for the performed experiments

Moreover, we have decided to initialize the values of the sizes belonging to the set $H$ always in the same way:

$$H = \left\{ \frac{3(\sqrt{2})^0}{\sqrt{\delta}}, \frac{3(\sqrt{2})^1}{\sqrt{\delta}}, \frac{3(\sqrt{2})^2}{\sqrt{\delta}}, \ldots, \frac{3(\sqrt{2})^5}{\sqrt{\delta}}, \frac{3(\sqrt{2})^8}{\sqrt{\delta}} \right\} = \left\{ \frac{3(\sqrt{2})^n}{\sqrt{\delta}}; n = 0, 1, \ldots, 5, 8 \right\}$$

where $\delta$ is the estimated surface sample density of the point cloud [33] and it provides a normalization of these values with respect to the surface density. Such initialization, which adopts an exponential rule, has been chosen because it allows to double the base area of each directional neighborhood at each next scale, which is a standard approach in multiscale and multiresolution signal processing [36]. Moreover the minimum value $h_1 \in H$ allows the base surface of each directional neighborhood to be at least $9/\delta$ wide so that, even in the smallest directional neighborhood, there are at least 9 points on average, allowing to get good results using the LPA kernel [36]. On the other hand the maximum value in $H$ provides a directional neighborhood that can expand widely in defect-free areas.

## 6.2. Simple synthetic test dataset

The search for defects and anomalies in point clouds is a relatively new and little explored area: differently from the 2D world, for which there are many datasets to test the different developed models, in the 3D world of point clouds this does not happen. In fact, at the beginning of this work there were no datasets of point clouds characterized by defects and anomalies. For this reason we have had to create our own dataset of point clouds necessary to test the effectiveness and performance of the developed algorithm.

One option to obtain a point cloud useful for this purpose is to purchase specific instru-

ments, such as laser scanners, and scan surfaces that contain defects and deformations. Due to the high cost of this instrumentation we decided to adopt a different approach that allows to obtain suitable point clouds, although their quality is not as high as what we would obtain using high precision scanners. The first step for the creation of this dataset is to exploit 3D models that have surfaces characterized by defects and malformations. We decided to create very simple 3D models that would allow to make some initial tests on the performances. In particular, we used Blender as an application to create 3D models in .obj format that represent surfaces with simple deformations, such as scratches and bulges.

In order to adequately test the performance of the developed algorithm, we manually identify the defective areas of the created models, coloring them in blue and the remaining surface in white. In this way it is possible to define a ground truth to evaluate the performance of the algorithm through metrics such as precision, recall, accuracy, False Negative Rate, F1 and AUC. A complete definition of these metrics is presented in Appendix A.

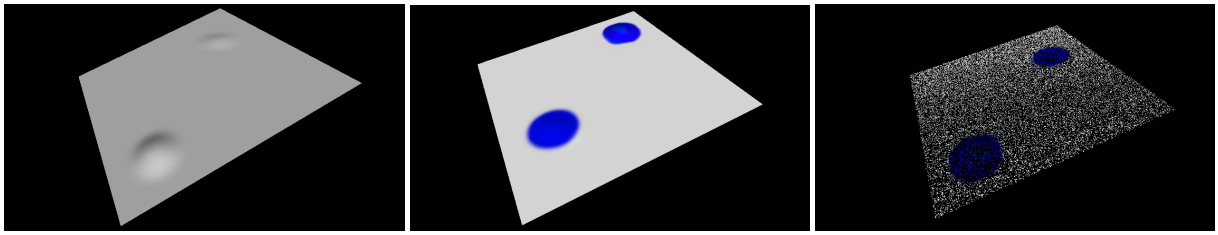


Figure 6.1: An example of a surface with two dents generated with MeshLab (left), the version where the defects are highlighted with a blue color (center) and the point cloud obtained performing a sampling operation over the colored surface (right)

The next step is to create a point cloud starting from the previously defined models. The scanners used to obtain high-precision point clouds exploit lasers that hit the surface, returning the spatial coordinates of the contact points: this operation can be seen as a sampling of the surface of the scanned object. In order to obtain a point cloud from the 3D models we have followed an approach similar to the one just described. Through a program called MeshLab it has been possible to import each 3D model and then sampling its surface, thus obtaining a representation in the form of point cloud for each of them as shown in figure 6.1. This technique, however, allows to obtain a "perfect" point cloud, without any type of inaccuracy that, even if minimal, characterizes the real scans made by laser scanner. In order to make the scan more similar to reality, Gaussian noise (with variance and mean value defined in Table 6.2) is added to each point to simulate what

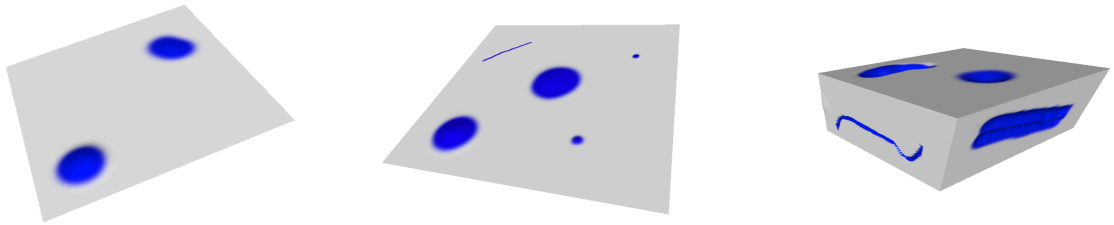


Figure 6.2: An example of a surface with two dents generated with MeshLab (left), the version where the defects are highlighted with a blue color (center) and the point cloud obtained performing a sampling operation over the colored surface (right)

could be the noise that characterizes the point clouds obtained using real-world scanning tools.

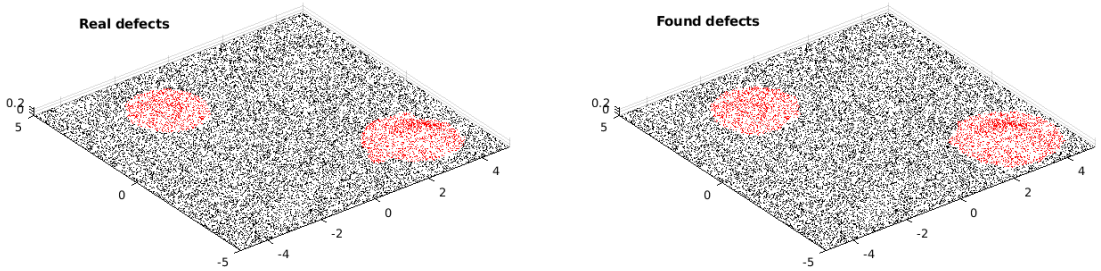
We performed some initial test to verify the performance of the algorithm by exploiting point clouds in which points belonging to faulty zones are colored in blue. For these first tests 3 point clouds have been generated with the methodology described previously and each of them has different characteristics. The first one is represented in the Experiment A of Figure 6.3, that is a point cloud obtained from a flat surface with 2 bulges and it has 20000 points. The second is a point cloud with 20000 points that was obtained from a flat surface with 4 bulges of various sizes and a scratch represented in the Experiment B of Figure 6.3. The last point cloud, composed by 40000 points, is shown the Experiment C of Figure 6.3 and represents a 3D object with bulges of different sizes, holes and scratches. These point clouds were given one at a time as input to the algorithm and, in order to test the performance of the algorithm, six metrics are considered: Accuracy, Precision, Recall, F1-score, False Negative Rate and AUC.

However, in order to calculate these metrics it is necessary to define when a point is considered belonging to a defective area or not. We have chosen to identify a point as really belonging to a defect if its anomaly score is less than a certain threshold $T$. However we have not been able to find a formula that would allow to identify a value for this threshold that would best identify all the defects. For our experiments we have decided to assign to $T$ the value that balances the False Positive Rate and True Positive Rate when defining the ROC curve, i.e. the one closet to the left upper corner of the ROC.

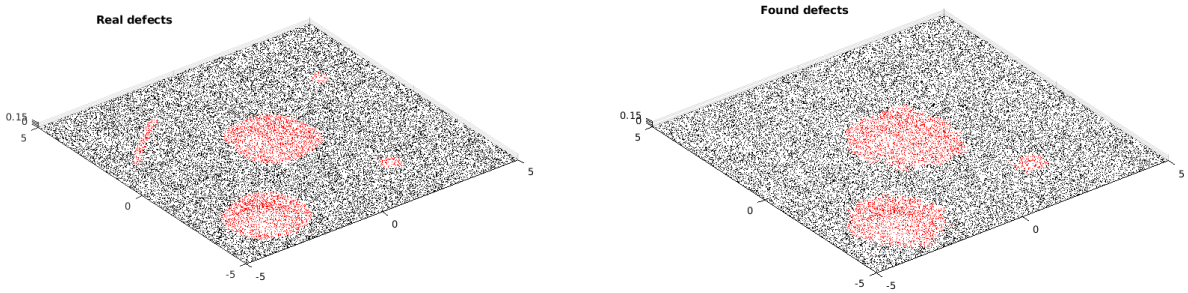
### 6.2.1. Synthetic dataset results

Below we report the results of the experiments performed on the synthetic dataset. For each of them we show the ground truth and the output point cloud, the values of the computed metrics and the graphs of the various AUC.

**Surface with 2 bulges (Experiment A)**




**Surface with 4 bulges of different dimensions and a scratch (Experiment B)**




**3D object with defects of different kind (Experiment C)**

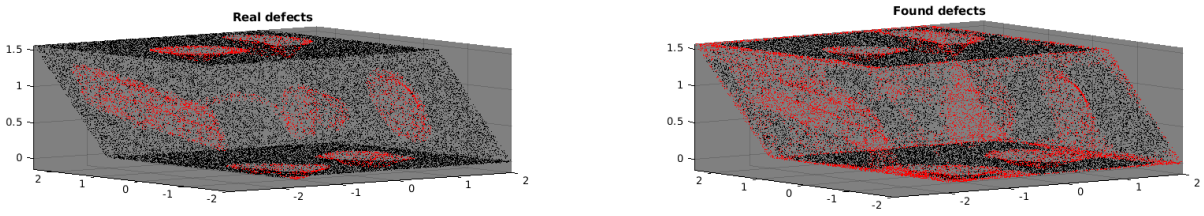



Figure 6.3: The input point cloud with the real defects colored in red (left) and the output point cloud with the found defects colored in red (right), while the not-anomalous points are colored in black. From top to bottom we show the point cloud of the surface with two bulges (A), the point cloud of the surface with four bulges of different dimensions and a scratch (B) and the point cloud of the 3D object with defects of different kind (C).

| Name | Γ | S | F1 | AUC | Acc | Prec | Rec | FNR | Time $[s]$ |
|---|---|---|---|---|---|---|---|---|---|
| **A** | 4 | 3 | 0.8583 | 0.9931 | 0.9636 | 0.7919 | 0.9368 | 0.0632 | 194 |
| | 4 | 11 | 0.8134 | 0.9880 | 0.9517 | 0.7458 | 0.8943 | 0.1057 | 61 |
| **B** | 4 | 3 | 0.8244 | 0.9496 | 0.9492 | 0.7395 | 0.9312 | 0.0688 | 190 |
| | 4 | 11 | 0.8581 | 0.9417 | 0.9612 | 0.8068 | 0.9164 | 0.0836 | 70 |
| **C** | 4 | 3 | 0.5874 | 0.9016 | 0.7698 | 0.4342 | 0.9076 | 0.0924 | 600 |
| | 4 | 11 | 0.5913 | 0.8753 | 0.8042 | 0.4745 | 0.7844 | 0.2156 | 195 |

Table 6.3: For each experiment this table shows the chosen values for the parameter Γ, the value of the stepsize **S**, the values of the different metrics and the time required by the algorithm to find the defects for the point cloud with 2 bulges

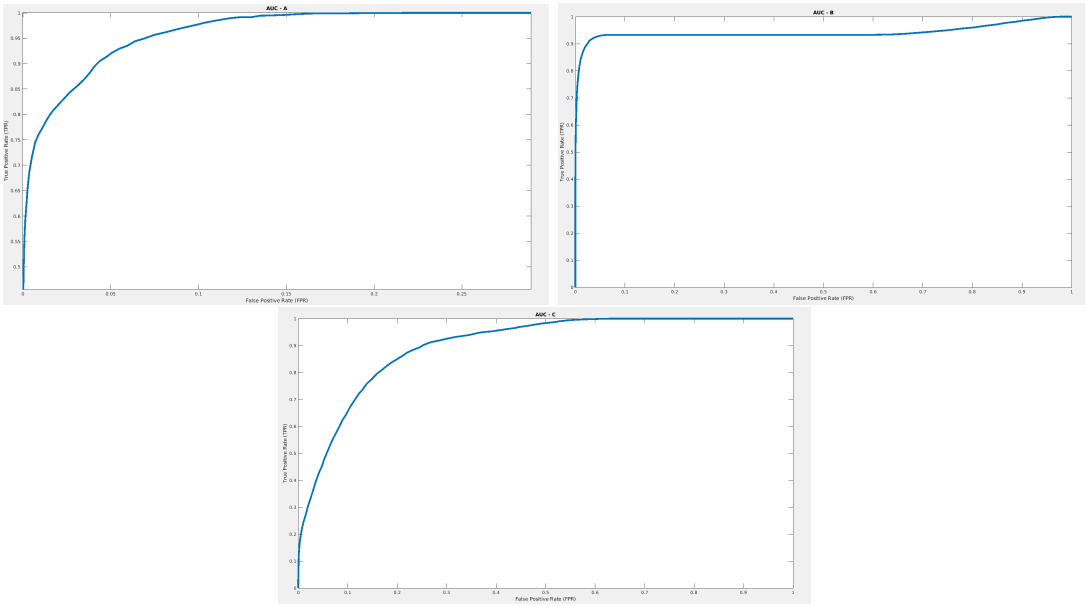


Figure 6.4: The AUC of each experiment. In particular on the top left corner the AUC for Experiment A, on the top right corner the AUC for Experiment B and on the bottom the AUC for Experiment C.

What can be seen from these experiments is that the algorithm is able to adequately identify the defects on each surface, obtaining very good results considering the different metrics. It can also be observed that the performances of the algorithm are slightly better when using a smaller stepsize (**S**). This is due to the fact that, the higher the number of points that are considered, the higher the number of DNs that will be generated. This causes more points to fall into at least one DN, resulting in a more accurate anomaly

score for each point. The drawback of having a small stepsize is obviously to have a high execution time, caused by the increase of the number of points for which we have to define the DNs.

In the Experiment B it can be seen that the algorithm is not able to detect the section of the scratch and the area containing the smallest defect. This is because these defects consist of points that are slightly detached from the underlying surface and this, due to the presence of noise, makes the algorithm misidentify them as defect-free areas. On the contrary, the more prominent defects are easily detected by the algorithm.

The algorithm also behaves well when the point cloud of an object that has a certain depth is given as input, like in the Experiment C. It is interesting to note that in this case the algorithm considers as points belonging to anomalous zones even those that are on the edges of the object. This is caused by the fact that the algorithm considers as anomalies those areas characterized by sharp features, like edges are. Indeed, referring to the figure 5.3, it can be observed that the DNs tend not to go beyond the edges: this means that the points in correspondence of such regions fall within a few number of DNs and consequently assume a rather high anomaly score. This is the reason why, in this experiment, a low Precision is obtained, which also influenced the F1.

## 6.3. Civil structure models

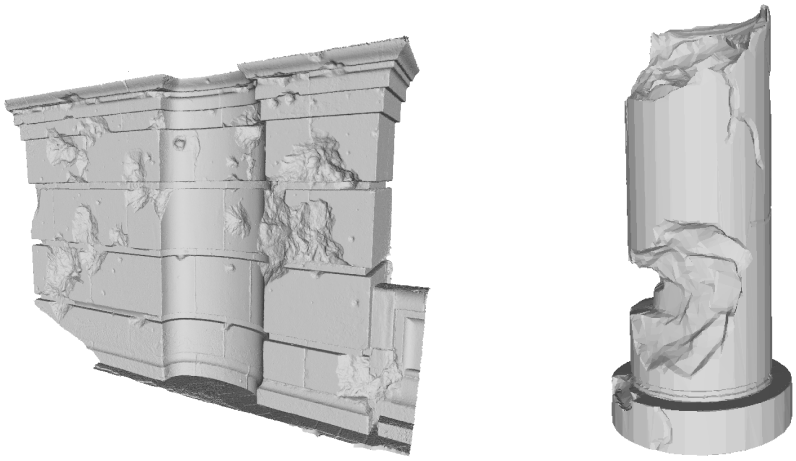


Figure 6.5: Two 3D models from which we extracted the point clouds used to test the performance of the developed approach. The choice fell on models like these because they are characterized by damages and malformations that the algorithm should be able to identify

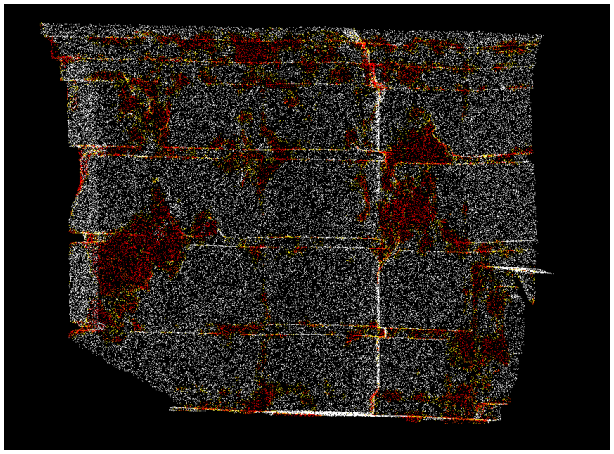
In addition to the simple custom models created to perform some initial tests on the performance of the algorithm, we have decided to use some CAD models representing

structures such as walls of buildings and columns with various damages and deformations. We exploit this type of models because the civil sector is one in which the search for defects and damages plays a rather critical role in determining the health of the structures. These models (all files in .obj format) have been obtained from the site Sketchfab.com [37], which provides free 3D models. Similarly to what has been done for the custom 3D models, the point clouds are obtained by sampling the surfaces of these models.

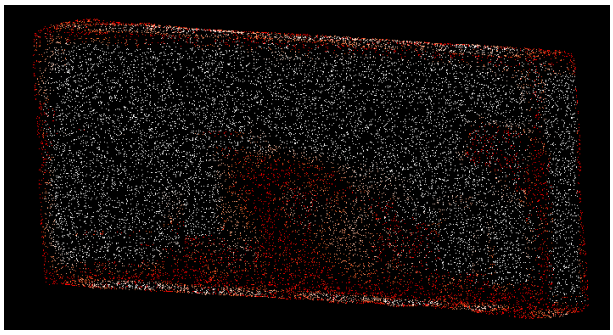
Since these models are unlabelled we cannot quantitatively measure the performance of the proposed algorithm. Therefore, in the following we report a qualitative analysis of the output of the model, where anomalous points are colored in red.

For each experiment, we report on the left the 3D model from which the point cloud given as input to the algorithm has been generated, while on the right the result of the application of the algorithm on the input point cloud. In particular the point cloud that is represented is the one given as input in which each point is colored red in the case in which it has been identified as belonging to a defective area by the algorithm. The parameters specified in 6.2 are also initialized in the same way in these experiments. Moreover, for each of them the number of points that make up the point cloud (**Count**), the stepsize **S** and the parameter $\Gamma$ that have been exploited are shown.

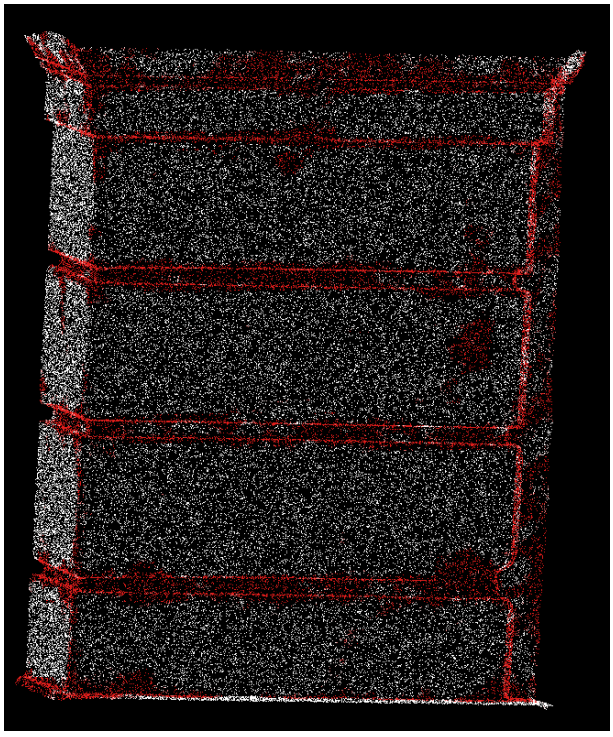
### 6.3.1. Walls



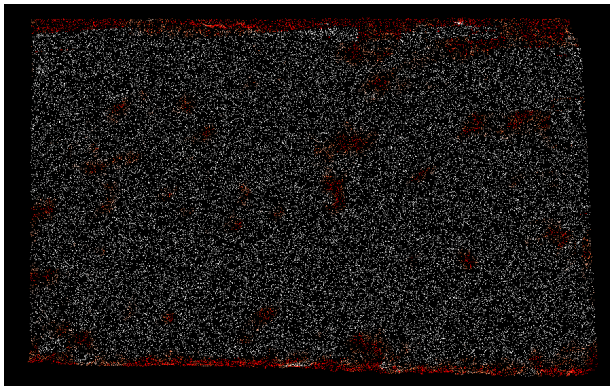
| Count | S | $\Gamma$ |
|---|---|---|
| 80000 | 21 | 5 |



| Count | S | Γ |
|---|---|---|
| 20000 | 1 | 1.2 |



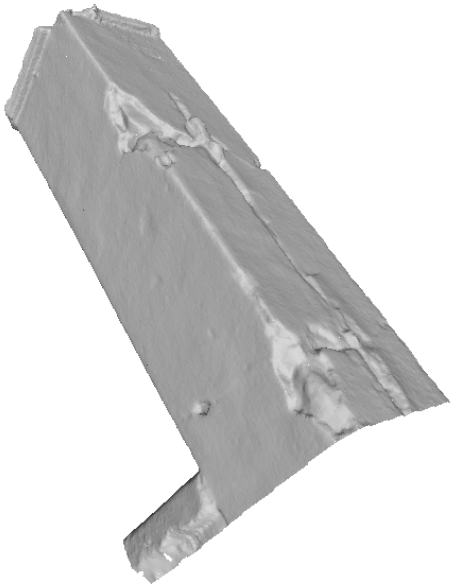
| Count | S | Γ |
|---|---|---|
| 80000 | 21 | 3 |

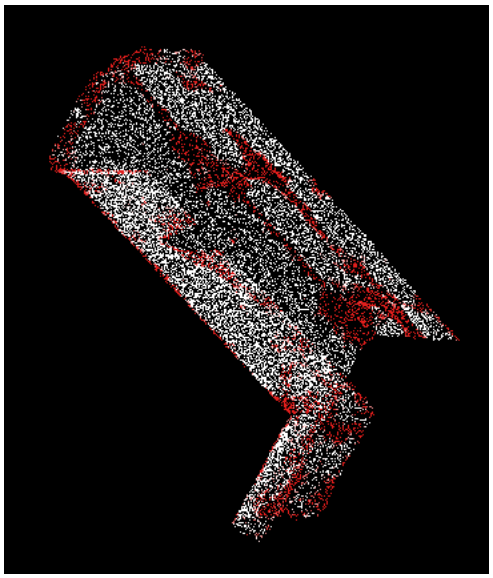


| Count | S | Γ |
|---|---|---|
| 50000 | 5 | 1.2 |

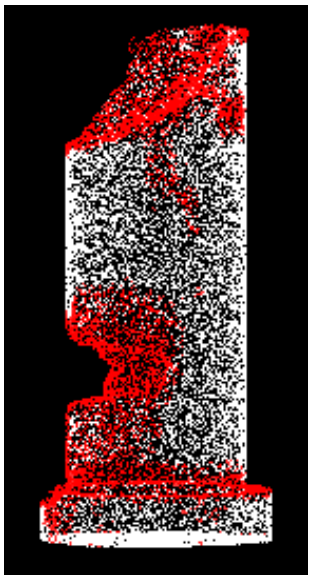
The algorithm adequately identifies most of the damaged areas of each model, whether they are very evident defects (as in the first two examples) or smaller, shallower defects (as in the last two examples). The results that are shown represent the best among the various experiments performed on these point clouds where only the parameters $S$ and $\Gamma$ have been changed. It can be observed that, although they are characterized by a different number of points and very different geometries, the algorithm is able to identify the main defective areas on each of them. Moreover it can be observed that the higher the number of points that make up the point cloud, the higher the parameter $\Gamma$ is. This is due to the fact that the larger the point cloud, the more it is necessary to make the different DNs expand as much as possible and this is done by increasing $\Gamma$ (5.4). However, it can be clearly seen that the algorithm identifies as defective areas also those that are made of sharp features, but are actually part of the structural geometry of the model. Indeed, as explained in the section 6.2.1, they are identified as defective areas because of the inability of the DNs to expand beyond them.

### 6.3.2. Columns

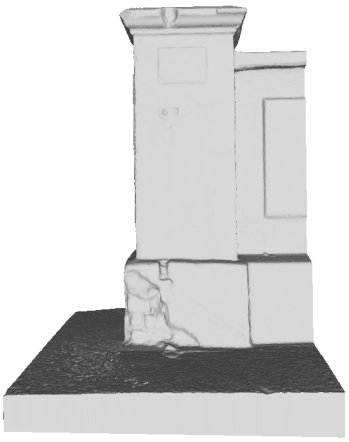


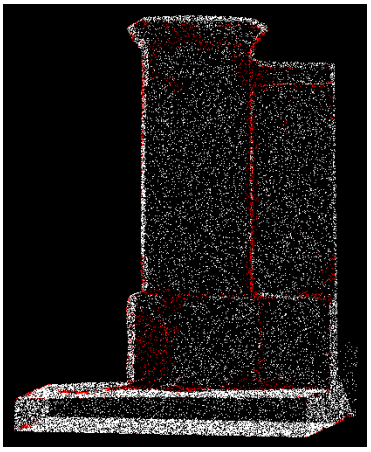


| Count | S | Γ |
|---|---|---|
| 20000 | 2 | 1.2 |

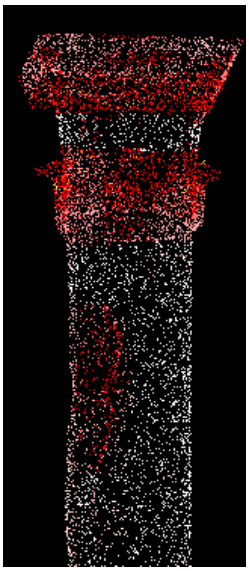


| Count | S | Γ |
|---|---|---|
| 20000 | 3 | 2 |





| Count | Stepsize | Γ |
|---|---|---|
| 50000 | 3 | 1.2 |



| Count | S | Γ |
|---|---|---|
| 15000 | 1 | 1.2 |

For what concerns the models representing columns and pillars the algorithm behaves very well, being able to identify the different defective areas. Particularly interesting are the columns with a cylindrical geometry: these demonstrate how DNs are able to expand

also in areas that are not necessarily flat, but also in the ones that are characterized by a certain curvature. In the last example it can be also seen how the decorations at the top of the column are identified as defective areas: these areas are characterized by sharp features and therefore marked as defects by the algorithm, as mentioned earlier in the Section 6.2.1.

## 6.4. MVTec 3D-AD dataset

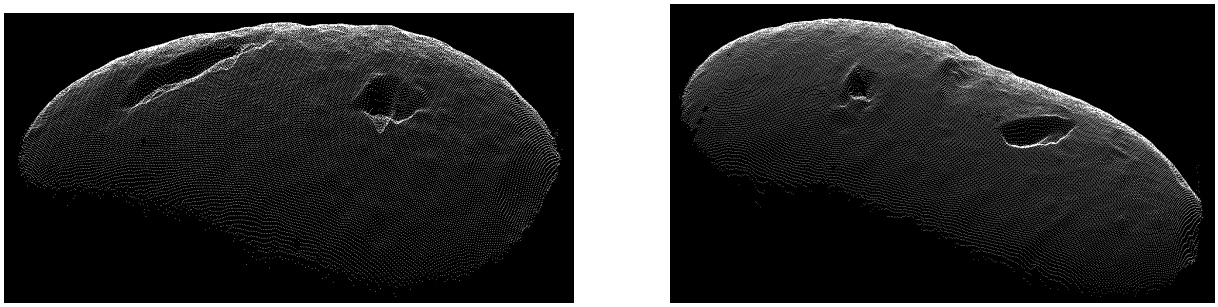


Figure 6.6: Two examples of point cloud of the MVTec 3D-AD dataset representing potatoes

Recently the MVTec 3D-AD dataset [38] has been made available and we tested our algorithm on its point clouds, obtained through an industrial 3D sensor. This dataset contains over 4000 high-resolution scans of 10 different object categories such as potatoes, ropes and tires. Currently this is the only publicly available dataset created specifically for the task of unsupervised anomaly detection and localization.

Bergmann et al. [38] wanted to test some of the few existing techniques for anomaly localization on this dataset. They propose to use the results of these methods as a baseline to compare with the results of other developed approach. Two of the methods they have decided to take into consideration are Voxel f-AnoGAN [39] and their implementation of a convolutional Voxel AE [40]. These methods exploit 3D convolution on voxels data, rather than the 2D convolutions used by the predecessors of these methods. Since these methods, in addition to working with voxels, are able to process depth images they have decided to add also this implementation in the benchmarks. The third and last model used as a baseline is the Variation Models [41] on depth images and voxel data.

The entire MVTec 3D-AD dataset contains about 13.2 GB of data between point clouds, ground truth and RGB images. However, we have decided to work only on a small part of this dataset, namely the point clouds belonging to the category *potato*. This choice is made mainly because this subset of point clouds is the one on which the models used as baseline have generally the worst results. Such results are shown in Table 6.4 and they re-

fer to the application of the described methods on the entire dataset of the category *potato*.

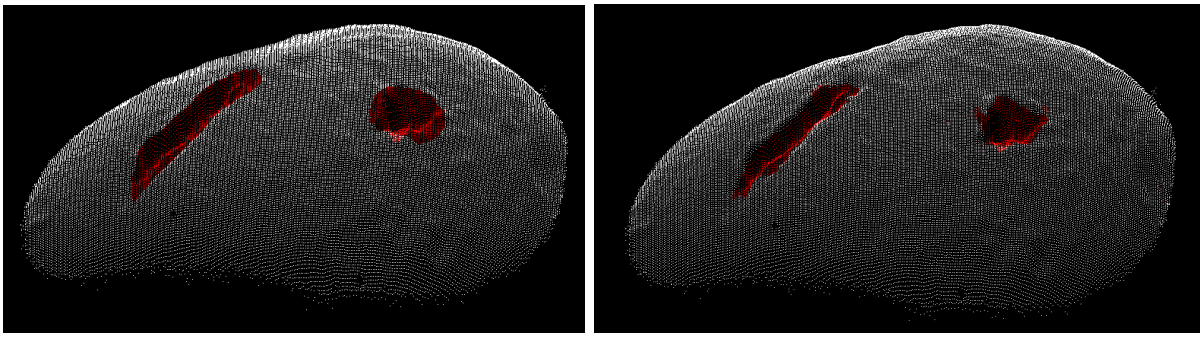
| | | potato |
|---|---|---|
| **Voxel** | **GAN** | 0.427 |
| | **AE** | 0.484 |
| | **VM** | **0.652** |
| **Depth** | **GAN** | 0.489 |
| | **AE** | 0.549 |
| | **VM** | 0.419 |

Table 6.4: AUC values of the benchmarks for the potato dataset. The best result is highlighted in boldface.

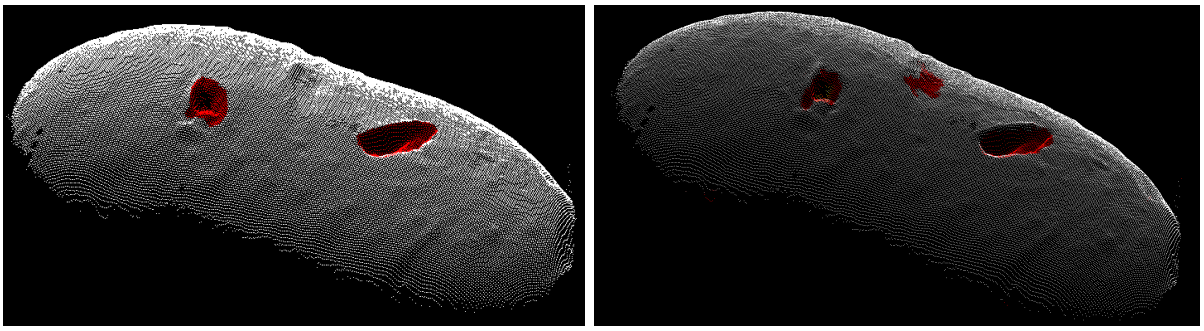
The MVTec 3D-AD dataset contains also the ground truth for each point cloud: this makes this dataset extremely useful as it allows to have an objective feedback on the performance of the algorithm on them. We evaluate the performance of the algorithm computing the F1-score, Accuracy, Precision, Recall, FNR and AUC. Particular attention should be paid to the latter as it is the yardstick with respect to the benchmarks presented in Table 6.4.

In the following for each experiment we show the ground truth (image on the left) and the defects found by the algorithm (image on the right). In Table **??** we specify number of points that make up each point cloud and the time required for the algorithm to identify the defects. Given the size of the point clouds and the irregularities on the surfaces we decide to use $\Gamma = 40$ and $S = 21$ in order to allow the DNs to expand as much as possible even on these rather irregular surfaces and have a good tradeoff between quality of the result and execution time. It is also not necessary to introduce noise on these point clouds ($\sigma^2 = 10e^{-5}$) since they are obtained from real scanners which themselves introduce noise on the obtained points. Finally, the values of the remaining parameters of Table 6.2 remain unchanged for these experiments.

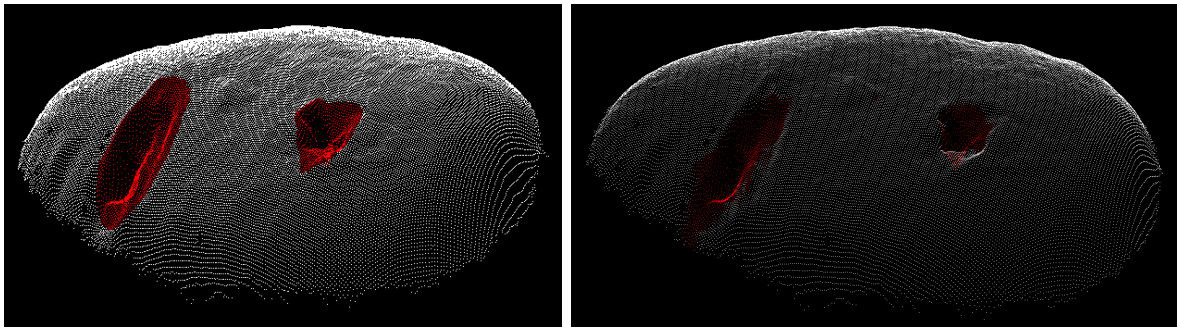
**Experiment 1**



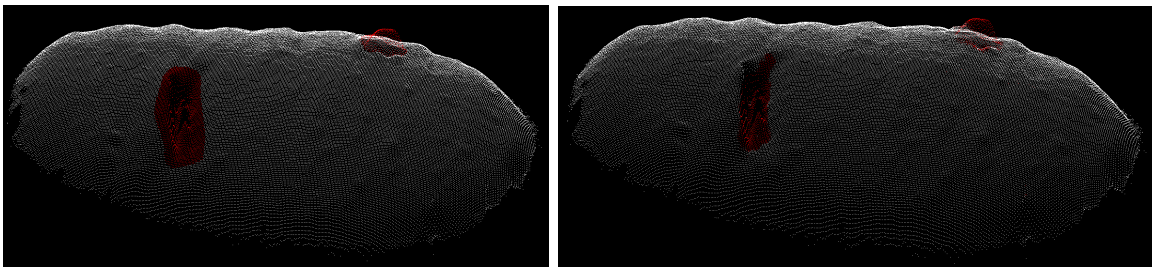
**Experiment 2**



**Experiment 3**



**Experiment 4**



**Experiment 5**

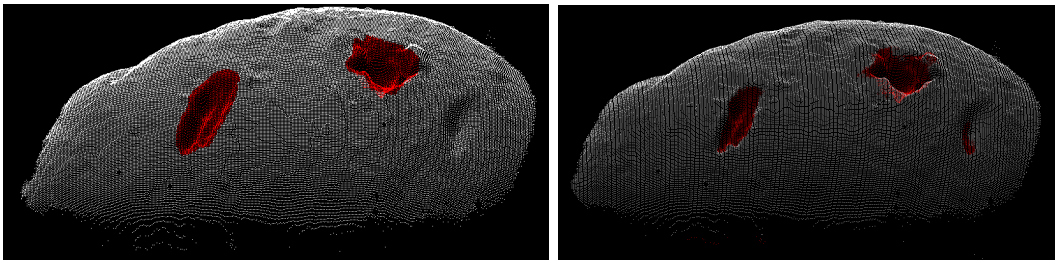


Figure 6.7: For each experiment we represent the ground truth point cloud (left) and the output point cloud (right) where the defects are colored in red.

| Experiment | Count | F1 | AUC | Acc | Prec | Rec | FNR | Time $[s]$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 46254 | 0.8212 | **0.8688** | 0.9985 | 0.9260 | 0.7378 | 0.2622 | 121 |
| 2 | 63399 | 0.6767 | **0.9865** | 0.9979 | 0.6132 | 0.7488 | 0.2512 | 168 |
| 3 | 37498 | 0.7120 | **0.9898** | 0.9975 | 0.8413 | 0.6172 | 0.3828 | 71 |
| 4 | 58985 | 0.6874 | **0.9687** | 0.9978 | 0.8272 | 0.5880 | 0.4120 | 152 |
| 5 | 53525 | 0.7764 | **0.9951** | 0.9975 | 0.8362 | 0.7246 | 0.2754 | 150 |

Table 6.5: The result of the experiments for the considered point clouds of the MVTec 3D-AD dataset. For each experiment we define the number of points that constitute each point cloud, the values of the computed metrics and the time required by the algorithm to analyze each point cloud. The AUC values that have to be compared with the benchmarks are highlighted in boldface.

Focusing the attention on the F1-score of each experiment it is evident how the algorithm outperformed all the methods used as benchmarks. Although 5 point clouds out of the 22 available for the *potato* category have been analyzed, we can observe that the algorithm is able to clearly recognize the defects on each of them.

Generally, as we can see from all experiments, the algorithm is able to expand the DNs even on rather irregular surfaces like the ones that characterize these point clouds. However, in Experiment 2 and Experiment 5 we can see that the algorithm has recognized as defective, in addition to the areas highlighted in the ground truth, two other small areas on the surface of the point clouds. This is due to the fact that these areas, even if they do not represent defects, are clear discontinuities of the surface, thus leading the DNs to expand little in these areas.

Better results can probably be achieved if we reduce **S** such that more DNs are generated. This would obviously affects the running time of the algorithm, given the large number of points that make up the point clouds. In support of this claim, we re-run Experiment 3 using $\mathbf{S} = 3$ and the results are shown in Table 6.6

| Count | F1 | AUC | Acc | Prec | Rec | FNR | Time $[s]$ |
|---|---|---|---|---|---|---|---|
| 37498 | 0.7279 | **0.9899** | 0.9977 | 0.8707 | 0.6254 | 0.3746 | 472 |

Table 6.6: The result for the point cloud used in Experiment 5 with $S = 3$.

From what can be observed from Table 6.6 the results are generally slightly better. However the time needed to obtain them has tripled compared to the case where $S = 21$. Given the little increase in performance we can say it is useless to adopt a stepsize so small.

# 7 | Conclusions and future developments

In this thesis work, we showed how it was possible to exploit a technique mainly used in the denoising of images in order to detect defective areas within point clouds. In particular, the identification of these areas is made possible through the use of DNs obtained using the LPA and which size is variable depending on whether they are close to a sharp feature or not. The dimensions are indeed governed by the ICI rule that limits the expansion of DNs in areas with sharp geometries, characterized by a high bias of the points that compose them. These features are typical of the defective areas and this means that in these regions the DNs assume reduced dimensions, causing few points of the point cloud falling within them. This impacts the anomaly score of each point, as this is inversely proportional to the number of DNs in which each point falls into.

From the obtained results, we can say that the method presented in this work offers a valid solution in the search for defects, allowing to have a good tradeoff between the performance of the algorithm and its execution time. Moreover, the fact that this algorithm works in a completely unsupervised way offers a great advantage in the field regarding the search for defects in point clouds, currently not so sufficiently investigated. In particular, the algorithm is able to analyze point clouds representing any type of object or surface without the need to rely on a training dataset that is very difficult to obtain or create. This is a great advantage because it makes the algorithm extremely versatile, able to work in different application domains without the need for changes or training of any kind.

Thanks to the experiments conducted using the MVTec 3D-AD dataset, we can say that the algorithm is able to obtain excellent results even on real data, i.e. on point clouds obtained through real scanners. However, the tests carried out involve only a small part of the entire dataset. For this reason, we are going to test the algorithm on point clouds of all the other categories. Moreover, some of the tests that have been carried out mainly concern the civil sector as this is one of the most critical domains when it comes to research defects. The models used in this context, however, do not represent real scans of surfaces,

but simply models that are as close as possible to real civil buildings. To test the actual robustness of the algorithm it is therefore necessary to apply it on point clouds obtained through scans of these type of structures.

Another step forward that we will have to do is to find a way to automatically define which value to assign to the parameter $\Gamma$ and the threshold $T$. In particular $T$, representing the threshold to identify if a point is anomalous, should be expressed in relation to the values of the anomaly scores of all the points of the point cloud. On the other hand, if a training dataset were available, we could perform a cross-validation to tune the parameter $\Gamma$, considering AUC as a comparison metric.

# A | Performance metrics

In order to properly define the metrics used to calculate the performance of the algorithm, we must first define the following concepts:

- **True Positive (TP)**: number of points of the point cloud belonging to a defective area that have been correctly identified as anomalies by the algorithm.
- **True Negative (TN)**: number of points of the point cloud belonging to a defect-free area that have been correctly identified as not anomalous by the algorithm.
- **False Positive (FP)**: number of points of the point cloud belonging to a defect-free area that have been wrongly identified as anomalies by the algorithm.
- **False Neagtive (FN)**: number of points of the point cloud belonging to a defective area that have been wrongly identified as not anomalous by the algorithm.

From these, the various metrics can be calculated. Specifically:

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP}$$

**Accuracy** is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations.

$$Precision = \frac{TP}{TP + FP}$$

**Precision** is the ratio of correctly predicted positive observations to the total predicted positive observations.

$$Recall = \frac{TP}{TP + FN}$$

**Recall** (also called **True positive Rate (TPR)**) is the ratio of correctly predicted positive observations to the all observations in the positive class.

$$F1 - score = \frac{2 * Recall * Precision}{Recall + Precision}$$

**F1-score** combines the Precision and Recall into a single metric by taking their harmonic mean.

$$FNR = \frac{FN}{FN + TP}$$

**False Negative Rate (FNR)** is the probability that a true positive will be missed by the test.

$$FPR = \frac{FP}{FP + TN}$$

**False positive Rate (FPR)** is the probability that a false alarm will be raised: that a positive result will be given when the true value is negative.

Another important metric that has been exploited is the **AUC** (Area Under the ROC Curve). The **ROC** (Receiver Operating Characteristic) curve plots TPR vs. FPR at different classification thresholds and the AUC is the area underneath such curve. It provides an aggregate measure of performance across all possible classification thresholds.

# List of Figures

# List of Tables

# Bibliography


[1] Pingbo Tang et al. "Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques". In: *Automation in Construction* 19.7 (2010), pp. 829–843.

[2] H. S. Park et al. "A New Approach for Health Monitoring of Structures: Terrestrial Laser Scanning". In: *Computer-Aided Civil and Infrastructure Engineering* 22.1 (2007), pp. 19–30.

[3] Anand Vetrivel et al. "Disaster damage detection through synergistic use of deep learning and 3D point cloud features derived from very high resolution oblique aerial images, and multiple-kernel-learning". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 140 (2018). Geospatial Computer Vision, pp. 45–59.

[4] Mehdi M. Akhlaghi et al. "Post-earthquake damage identification of an RC school building in Nepal using ambient vibration and point cloud data". In: *Engineering Structures* 227 (2021), pp. 111–413.

[5] Mohammad Nahangi and Carl T. Haas. "Automated 3D compliance checking in pipe spool fabrication". In: *Advanced Engineering Informatics* 28.4 (2014), pp. 360–369.

[6] Robin H. Helle and Hirpa G. Lemu. "A case study on use of 3D scanning for reverse engineering and quality control". In: *Materials Today: Proceedings* 45 (2021). Second International Conference on Aspects of Materials Science and Engineering (ICAMSE 2021), pp. 5255–5262.

[7] Seokbae Son, Hyunpung Park, and Kwan H. Lee. "Automated laser scanning system for reverse engineering and inspection". In: *International Journal of Machine Tools and Manufacture* 42.8 (2002), pp. 889–897. ISSN: 0890-6955.

[8] Oliver Holzmond and Xiaodong Li. "In situ real time defect detection of 3D printed parts". In: *Additive Manufacturing* 17 (2017), pp. 135–142. ISSN: 2214-8604.

[9] Abd Almamou et al. "Quality Control of constructed Models using 3D point cloud". In: July 2015.

[10] Engin Burak Anil et al. "Deviation analysis method for the assessment of the quality of the as-is Building Information Models generated from point cloud data". In: *Automation in Construction* 35 (2013), pp. 507–516. ISSN: 0926-5805.

[11] Yadong Li and Peihua Gu. "Free-form surface inspection techniques state of the art review". In: *Computer-Aided Design* 36.13 (2004), pp. 1395–1417. ISSN: 0010-4485.

[12] Giordano Teza, Antonio Galgaro, and Francesca Moro. "Contactless recognition of concrete surface damage from laser scanning and curvature computation". In: *NDT E International* 42.4 (2009), pp. 240–249. ISSN: 0963-8695.

[13] Burcu Guldur Erkal and Jerome F. Hajjar. "Laser-based surface damage detection and quantification using predicted surface properties". In: *Automation in Construction* 83 (2017), pp. 285–302. ISSN: 0926-5805.

[14] Igor Jovančević et al. "3D point cloud analysis for detection and characterization of defects on airplane exterior surface". In: *Journal of Nondestructive Evaluation* 36.4 (2017), pp. 1–17.

[15] Matthew M. Torok, Mani Golparvar-Fard, and Kevin B. Kochersberger. "Image-Based Automated 3D Crack Detection for Post-disaster Building Assessment". In: *Journal of Computing in Civil Engineering* 28.5 (2014), A4014004.

[16] Min-Koo Kim, Hoon Sohn, and Chih-Chen Chang. "Localization and Quantification of Concrete Spalling Defects Using Terrestrial Laser Scanning". In: *Journal of Computing in Civil Engineering* 29.6 (2015), p. 04014086.

[17] J. Valença et al. "Assessment of cracks on concrete bridges using image processing supported by laser scanning survey". In: *Construction and Building Materials* 146 (2017), pp. 668–678. ISSN: 0950-0618.

[18] Mohammad Ebrahim Mohammadi, Richard L. Wood, and Christine E. Wittich. "Non-Temporal Point Cloud Analysis for Surface Damage in Civil Structures". In: *ISPRS International Journal of Geo-Information* 8.12 (2019). ISSN: 2220-9964.

[19] Czesław Suchocki et al. "Detection of defects in building walls using modified OptD method for down-sampling of point clouds". In: *Building Research and Information* (Mar. 2020), pp. 1–19.

[20] Nguyen LD Khoa et al. "Robust dimensionality reduction and damage detection approaches in structural health monitoring". In: *Structural Health Monitoring* 13.4 (2014), pp. 406–417.

[21] Adam Santos et al. "Machine learning algorithms for damage detection: Kernel-based approaches". In: *Journal of Sound and Vibration* 363 (2016), pp. 584–599. ISSN: 0022-460X.

[22] Guoqing Gui et al. “Data-driven support vector machine with optimization techniques for structural health monitoring and damage detection”. In: *KSCE Journal of Civil Engineering* 21.2 (Feb. 2017), pp. 523–534. ISSN: 1976-3808.

[23] Ramin Ghiasi, Peyman Torkzadeh, and Mohammad Noori. “A machine-learning approach for structural damage detection using least square support vector machine based on a new combinational kernel function”. In: *Structural Health Monitoring* 15.3 (2016), pp. 302–316.

[24] C. Qi et al. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 77–85.

[25] Majid Nasrollahi, Neshat Bolourian, and Amin Hammad. “Concrete Surface Defect Detection Using Deep Neural Network Based on LiDAR Scanning”. In: June 2019.

[26] Thorsten Wuest et al. “Machine learning in manufacturing: advantages, challenges, and applications”. In: *Production & Manufacturing Research* 4.1 (2016), pp. 23–45.

[27] Yingjie Zhang et al. “Extraction and evaluation of melt pool, plume and spatter information for powder-bed fusion AM process monitoring”. In: *Materials  Design* 156 (2018), pp. 458–469. ISSN: 0264-1275.

[28] Barath Narayanan Narayanan et al. “Support vector machine and convolutional neural network based approaches for defect detection in fused filament fabrication”. In: 11139 (2019). Ed. by Michael E. Zelinski et al., pp. 283–291.

[29] Alessandra Caggiano et al. “Machine learning-based image processing for on-line defect recognition in additive manufacturing”. In: *CIRP Annals* 68.1 (2019), pp. 451–454. ISSN: 0007-8506.

[30] Luke Scime and Jack Beuth. “Anomaly detection and classification in a laser powder bed additive manufacturing process using a trained computer vision algorithm”. In: *Additive Manufacturing* 19 (2018), pp. 114–126. ISSN: 2214-8604.

[31] Rui Li, Mingzhou Jin, and Vincent C. Paquit. “Geometrical defect detection for additive manufacturing with machine learning models”. In: *Materials  Design* 206 (2021), p. 109726. ISSN: 0264-1275.

[32] Paweł Trybała et al. “Damage Detection Based on 3D Point Cloud Data Processing from Laser Scanning of Conveyor Belt Surface”. In: *Remote Sensing* 13.1 (2021). ISSN: 2072-4292.

[33] Alessandro Foi. “Anisotropic nonparametric image processing: theory, algorithms and applications”. In: *Dip. di Matematica, Politecnico di Milano* (2005).

[34] *Local Approximations in Signal and Image Processing*. `https://webpages.tuni.fi/lasip//`.

[35] Giacomo Boracchi. *Image Processing case study:LPA-ICI Denoising*. https://boracchi.faculty.polimi.it/docs/2009_10_27_Cuda.pdf.

[36] Zhongwei Xu and Alessandro Foi. "Anisotropic Denoising of 3D Point Clouds by Aggregation of Multiple Surface-Adaptive Estimates". In: *IEEE Transactions on Visualization and Computer Graphics* 27.6 (2021), pp. 2851–2868. DOI: 10.1109/TVCG.2019.2959761.

[37] *Sketchfab*. https://sketchfab.com/tags/blender.

[38] Paul Bergmann et al. "The MVTec 3D-AD Dataset for Unsupervised 3D Anomaly Detection and Localization". In: *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5* (2022).

[39] Jaime Simarro Viana et al. "Unsupervised 3D Brain Anomaly Detection". In: *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*. Springer International Publishing, 2021, pp. 133–142. DOI: 10.1007/978-3-030-72084-1_13. URL: https://doi.org/10.1007%2F978-3-030-72084-1_13.

[40] Marcel Bengs et al. *3-Dimensional Deep Learning with Spatial Erasing for Unsupervised Anomaly Segmentation in Brain MRI*. 2021. DOI: 10.48550/ARXIV.2109.06540. URL: https://arxiv.org/abs/2109.06540.

[41] Steger C, Ulrich M, and Wideman C. *Machine Vision Algorithms and Applications*. 2nd ed. Wiley-VCH, 2018.