Politecnico di Torino

M.Sc. Thesis in Mechanical Engineering

Big data management in surface topography analysis

by

Cristiana Tuttolomondo

(s276113)

Supervisors:

Prof. Gianfranco Genta

Dr. Giacomo Maculotti

Dr. Danilo Quagliotti

Prof. Hans Nørgaard Hansen

Big data management in surface topography analysis

Cristiana Tuttolomondo

(s276113)

M.Sc. Thesis in Mechanical Engineering

July, 2022

By

Cristiana Tuttolomondo

Published by: DTU, Politecnico of Turin , Denmark, Italy

Approval

This thesis has been prepared over six months at the Section of Manufacturing Engineering, Department of Civil and Mechanical Engineering, at the Technical University of Denmark—DTU, in partial fulfilment for the degree Master of Science in Engineering, M.Sc. Eng.

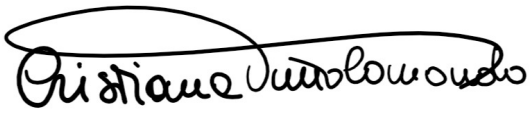


Figure 1: Signature

Cristiana Tuttolomondo - s276113

.....

Signature

Politecnico of Turin, July 13, 2022

Abstract

In last years, surface metrology has carried out numerous engineering applications (i.e., electronics, information technology, energy, optics, tribology, biology, biomimetics, etc.). The majority of surface measurements are used in industrial settings to check tolerances and characterize component surface functionality, but they are also useful for in-line quality control and to understand and govern additive manufacturing processes.

Measurements must be acquired and processed quickly, thus optical instruments are being used more and more frequently. Furthermore, a reliable acquisition should also be accompanied by an uncertainty factor, however, the available methods rarely provide a correct value due both to the complexity of the measurement and the interaction between the optical instruments and the component which is still not clear.

Additionally, computational time has increased because of both the complexity of the calculations and the size of the data; for this reason cluster and parallel computing have recently become important topics of research among academia and industry.

Hence, the purpose of this project is to handle the complexity of such measurements by exploring the applicability and limitations of automated machine-based methods to evaluate measurement uncertainty. Moreover, it also aims to analyze the application of the digital twin in this field in order to develop a model that is suitable and adaptable to any kind of sample, and ultimately develop a workframe on clusters to handle these large amounts of data.

First of all, the surfaces are processed both with two commercial software, Spip and MountainsLab, and the developed Matlab code in order to remove the form and, only in the second case, to manage possible non-measured points and/or spikes, to estimate the uncer-

tainty value and to evaluate the covariance and correlation factor. Afterwards, a Comsol random model is studied by changing its parameters and a configuration with Matlab code was established. At the end, the work is imported onto DTU HPC clusters where the power evaluation was higher and the computation faster.

Acknowledgements

This project would not have been possible without the help and support of many people.

Many thanks to my supervisors, Gianfranco Genta and Giacomo Maculotti in Turin, and Danilo Quagliotti and Hans Nørgaard Hansen in Denmark, who offered guidance, encouragement and support.

Immense gratitude as always to my FAMILY and Carlo, pillars of my life, for their patience and unconditional, unequivocal and loving support.

Thanks also to friends and my housemates who endured this long process with me, always offering support and love.

Thanks also to the DTU technical support.

”It always seems impossible
until it is done”
- Nelson Mandela -

Contents

1 Introduction

1.1 Technical background and motivations

Micro- and nano-systems, such as surfaces' features and structures, are of central importance in the modern manufacturing industry, allowing complex products' functionalities. Precision engineering and surface technologies often achieve the production of highly engineered components for a variety of applications by altering surfaces and controlling their topography: e.g., controlled drug release in biomedical field, energy harvesting, advanced optics, tribo-mechanical application, etc. Therefore, the quality assessment of the modern products requires new and more capable methods. Because of the continuous down-scaling, in fact, the products' geometric knowledge has become increasingly challenging, above all regarding the evaluation of the measurement uncertainty and establishment of the traceability.

The uncertainty is one of the most important parameters to be evaluated. The currently available methods have some limitations, such as the ISO standards in the GPS system [1]. In addition, the current framework based on the instruments' metrological characteristics provides an effective but limited description of uncertainty influence factors and measurement errors, which cannot be thoroughly evaluated, corrected or compensated [2]. Therefore, new approaches can be investigate, such as the frequentist approach, in order to deal with the uncertainty of roughness parameters [2].

Moreover, different types of engineering software and digitalized equipment are widely used throughout the product lifecycle. In this way, massive amounts of data of different types are being produced, but they are isolated from one another, which makes them inefficient

and underutilized.
To solve the problem, new-generation information and digitalization technologies has been developed to allow more data to be collected and linked; consequently, the concept of digital twin has rapidly gained traction.
The definition provides by the CIRP Encyclopedia of Production Engineering [3] is: *"A digital twin is a digital representation of an active unique product (real device, object, machine, service, or intangible asset) or unique product-service system (a system consisting of a product and a related service) that comprises its selected characteristics, properties, conditions, and behaviors by means of models, information, and data within a single or even across multiple life cycle phases."*
In other words, a digital twin is a virtual model of a physical system; in particular the term 'twin' refers to a replica which can be either an object or a process. This could further be broken down into many sub-parts in order to identify and track a particular phenomenon. Its role in the manufacturing phase involves modelling surfaces and correcting their errors, improving processing quality and reducing production costs in an efficient, dynamic, and intelligent manner that is not possible with the traditional methods, [4].
In this context, metrology can be viewed as a key discipline in enabling the quality control of the industrial production. It implements the process control of the components or of the manufacturing process itself, and allows the correct definition of the required functionalities through metrological characterizations. The interest in surface areal characterization increased with the improvement of the optical microscopes, being more suitable for relatively fast acquisitions without damaging the soft surfaces of the specimens under examination. The improvement of different measurement technologies also corresponded to a similar advance in the development of

different types of commercial software for image processing, with a multitude of directions for surface inspection.

Science and technology have made surface analysis increasingly relevant; in fact, image processing has played a significant role in metrology because it is able to transform observed measurements into desired attributes of the object being studied. Furthermore, large data sets have been an advantage in the analysis and characterization of surfaces since they provide a broad view of all their features (e.g. defects detection, evaluation of free-form structures and topographies, evaluation of measurement uncertainty, etc.).
The creation of a digital twin model can be an effective way of dealing with large data sets and simplifying calculations; in fact, as previously described, it can gather and replicate measurements for a more accurate representation.
The transition from pixel-imaging to digital twin requires a large amount of computing power and memory; for this reason, high-performance computing (HPC) is necessary and beneficial to fully exploit big data.

In this view, the main purpose of this project was to develop a software architecture for connecting the uncertainty evaluation and the digital twin generation with the aid of HPC in order to speed up the computation.
Therefore, the motivations behind the project lead the following research questions:

- *What are pros and cons of the commercial software in the analysis of measured micrographs?*
- *When analyzing complex surfaces, what are the limitations affecting the computation of parameters and measurement uncertainty?*

- *Is it possible to reduce the computational demand? What are the advantages of generating surfaces' digital twin based on measured data?*

1.1.1 Structure of the thesis

This project is divided into two main parts.
The first one examines the new critical cases linked to new manufacturing technique to understand what is required to develop them and to achieve good results. Specifically, chapter 2 deals with general information about surface metrology tasks in the project. In the second part the developed software is introduced, taking into account all the problems related to the traditional as well as the new technologies and methods. The flowchart in figure 1.1 displays its structure. Namely, the chapter 3 introduces the usage of clusters, HPC and Thinlinc platform, and a new approach to time and storage issues. Useful array structures are also examined. Chapter 4 describes the commercial softwares Spip and MountainsLab, and introduce the samples analyzed in the thesis. Chapter 5 focuses on the digital twin approach, the use of the software Comsol for a model the generation and the required configuration among COMSOL Multiphysiscs®, Matlab® and HPC. Chapter 6 provide a comparison between outputs obtained from the different types of software, and the use of a digital twin model as possible kernel model. Finally, the conclusions are drawn in chapter 7, where a brief outlook is also given.

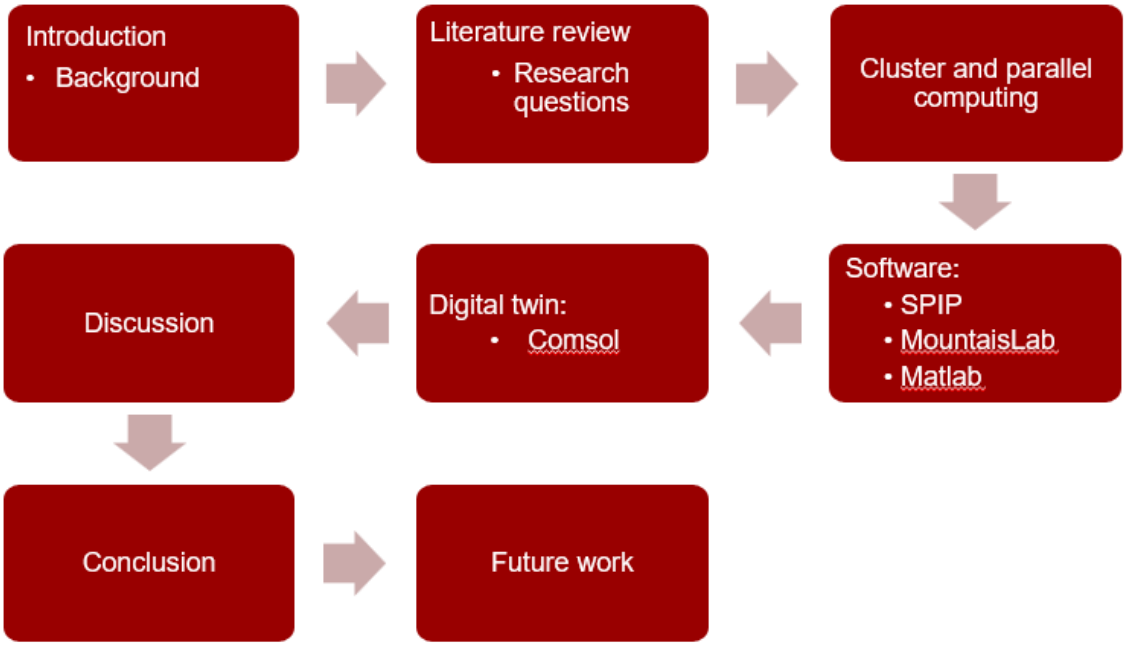


Figure 1.1: Structure of the thesis.

2 Surface metrology

2.1 Characterization of surface topography

Metrology is the science of measurement, implementing both experimental and theoretical determinations at any level of measurement uncertainty in any field of science and technology.
The application of metrology to surface analysis can improve many functions of mechanical components and manufactured items, including wear reduction, increasing component life, and optimizing efficiency. All of these functions can be achieved by designing, engineering and controlling specific surface textures.

2.1.1 Plane correction

In most cases, surface texture specifications are defined on flat surfaces, regardless of their possible geometric form. Therefore, the measured micrographs require preparatory procedures (pre-processing) for the alignment of the software reference system (plane correction) and, when required, for the extraction of the underline component's shape (form removal).
Both plane correction and form removal are usually based on the least-squares fit (LS) of the micrographs' height values (data points). LS provides a relationship among the data points, which is a reference surface to be subtracted from the original micrographs. The reference surface is a *plane* in the case of plane correction. When the reference plane is subtracted from a micrograph, the data points are centered with respect to this plane (*center plane*), and the roughness parameters can be correctly evaluated (see fig. 2.1). The same happens in the form removal, except that a generic reference surface is instead used (e.g. portion of a sphere, lateral surface of a cylinder, etc.).

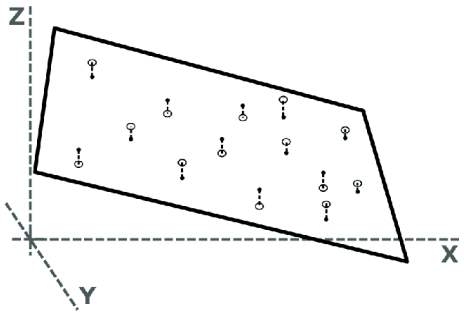


Figure 2.1: Simplified representation of the Least square method.

2.1.2 Uncertainty evaluation

The measurement uncertainty can be interpreted as the 'goodness' of a measurement result. It is defined in the Guide to the Expression of Uncertainty in Measurement (GUM), and in the International Vocabulary of Basic and General Terms in Metrology (VIM), as a "parameter, associated with the result of a measurement, that characterizes the dispersion of the values that could reasonably be attributed to the measurand" [5]-[6].

Quantifying the uncertainty is a difficult process. Both the intrinsic variability and an incomplete knowledge about a physical system are source of uncertainty. For this reason it is fundamental to take the measurement process into account, identify all possible errors, and quantify their impact.

So far, several A and/or B type methods for estimating the uncertainty have been developed such as the *frequentist approach* [5], the PUMA method in manufacturing [7], etc. In this project the uncertainty determination was not the main focus, therefore, it was evaluated as the combined standard uncertainty in eq. 2.1 multiplied by a coverage factor as shown in eq. 2.2.

$$u_c(y) = \sqrt{\sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_i}\right)^2 u^2(x_i)} \tag{2.1}$$

$$U = k * u_c(y) \tag{2.2}$$

The coverage factor k usually ranges between 2 and 3, corresponding to a particular level of confidence p, such as approximately from 95 % to 99 %. Thus, the result is then expressed by the eq. 2.3.

$$Y = y \pm U \tag{2.3}$$

Since the project focused on the software architecture, as already explained, the traceability was not addressed despite it is a fundamental property of the measurement uncertainty. The traceability can be achieved by measurements of material measures as explained in [2], or by calibrating the metrological characteristics of the optical instrument used [8]. The corresponding contributors are then to be combined with the other contributors of the uncertainty budget according to the formula 2.1.

2.1.3 The roughness parameters

Characterizing a surface topography entails the determination of parameters that can describe in a synthetic way the microgeometry of a measured surface. Measurement parameters such as amplitude and spacing contain information about the surface topography. When measuring surface parameters and comparing them with specifications, there are some factors that must be taken into consideration:

- Parameters may vary depending on the workpiece location and on the direction of the measurement.
- The traceability and uncertainty of the measured coordinates as well as uncertainties associated with the probe size, filtering, sampling, reference line or surface, etc. must be considered.

Furthermore, in the analysis, the effect due to the digitization should also be taken into account because the digitizing step cannot be infinitesimally short. For this reason, the pixel size should be sufficiently small to provide meaningful results.

Two methods are available for this evaluation: the profile method, which is based on a line (the profile) defined on a surface, and the areal method, which is based on a portion of area on the surface. Many parameters of the areal method are derived from the profile method, thus there are some similarities (see fig. 2.2).

Profile roughness parameters are usually calculated on the profile

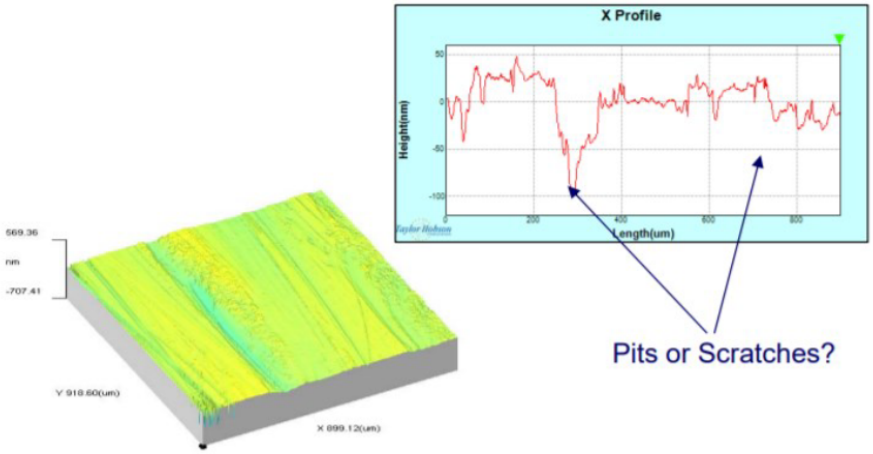


Figure 2.2: Qualitative comparison of the difference in information between the areal and the profile method.

perpendicular to the main surface texture direction, and they can be classified into four classes: amplitude, spacing, hybrid, and curves. Profile parameters are listed and categorized in the series ISO 21920 [9]–[10]. For example, P-parameters (e.g. P_a, P_{sk}, etc.) are calculated from primary profiles, R-parameters (e.g. R_a, R_{sk}, etc.) from roughness profiles, and W-parameters (e.g. W_a, W_{sk}, etc.) from waviness profiles.

Profiles alone have less significance, i.e. robustness and pertinence, than an areal evaluation from the surface functionality point of view,

which better describe the ”real surface”. A brief review on differences between the two type of parameters is in fig. 2.3.

The areal parameters are defined in the ISO 25178-2 [11] accord-

Profile		Areal	
Amplitude parameters	$Rp, Rv, Rz, Rc, Rt, Ra, Rq, Rsk, Rku$	Height parameters	$S_p, S_v, S_z, S_a, S_q, S_{sk}, S_{ku}$
Spacing parameters	Rsm	Spatial parameters	S_{al}, S_{tr}
Hybrid parameters	$R\Delta q$	Hybrid parameters	S_{dq}, S_{dr}
		Miscellaneous parameters	S_{td}
Curves and related parameters	$Rmr(c), R\Delta c, Rmr, Rk, Rpk, Rvk, M_{r1}, M_{r2}$	Functions and related parameters	$S_{mr(c)}, S_{mc(mr)}, S_{xp}, S_k, S_{pk}, S_{vk}, S_{mr1}, S_{mr2}, V_{vv}, V_{vc}, V_{mp}, V_{mc}$

Figure 2.3: Summary of profile and areal parameters.

ing to a reference area selected by the application of specific spatial filters (*scale-limited surface*). In addition, the surface is positioned on the center plane (see section 2.1), which provides the basis for calculating deviations of the height distribution and other parameters. More details are in the ISO 25178-3 [12].

The arithmetical mean height, or R_a, is the the mean of the absolute of the ordinate values of the scale-limited surface. It is evaluated according to the eq. 2.4.

$$Sa = \frac{1}{l} \iint_{\bar{A}} |z(x, y)| \, dxdy \tag{2.4}$$

The most common parameters are the root mean square height and the root mean square gradient. The root mean square height, or Sq, is the square root of the mean square of the ordinate values of the scale-limited surface, and it is equivalent to the standard deviation of height values. It is evaluated according to the eq. 2.5.

$$Sq = \sqrt{(\frac{1}{A})} \iint_{\bar{A}} z^2(x,y)dxdy \tag{2.5}$$

The root mean square gradient, or Sdq, is the square root of the mean square of the surface gradient of the scale-limited surface. Sdq is evaluated according to the eq. 2.6.

$$Sdq = \sqrt{(\frac{1}{A})} \iint_{\bar{A}} \left[\left(\frac{\delta z(x,y)}{\delta x} \right)^2 + \left(\frac{\delta z(x,y)}{\delta y} \right)^2 \right] dxdy \tag{2.6}$$

Since the acquired micrographs are discrete sets of height values, all the integrals are practically evaluated as summations.
In general, areal parameters are commonly evaluated due to the wide range of applications. For example, in quality control, the parameters allow determining if the product meets the designer's specification and to distinguish surfaces compliance with their functionality. In particular, they can also be used as a method to verify that the manufacturing process has yielded the specifies texture, since the roughness affects functions. Accordingly, they can be used in research and development in order to optimize the design and production of products according to the engineers' needs.

2.1.4 Spikes, voids and outliers

The surface topography parameters are calculated using the reference plane, which is established with errors and imperfections from the manufacturing process. Surface topography analysis errors can usually be divided into measurement errors, object measurement errors, and method errors, which may result in outliers and data dropouts. Furthermore, noise, as well as disturbances in general,

affect the instrument and contribute to the measurement uncertainty [13].

Moreover, the interactions between the instrument's light and the sample can lead to the following problems in the micrographs: the voids (or non-measured points) represent the absence of height values for the measured pixels; the spikes represent anomalous height values that are very different from the neighboring pixels. Both counteract the evaluation of micrographs, and need to be identified and corrected to guarantee reliable results [14]-[15]. To accomplish this, several techniques can be implemented: *thresholding*, filtering and, according to a recent study [16], Gaussian process regression modeling (GPR).

The thresholding and filtering methods are not considered in this project since they are not suitable together with the frequentistic approach [2]. The GPR method, on the other hand, predicts the model response by choosing an optimal kernel based on the surface to be modeled. Consequently, the normal probability plot (NPP) identifies and highlights the spikes, and such information on the acquisition is compared with the GPR model to correct them. Similarly, the correction of voids was also examined by comparing it with a smoothing spline algorithm.

2.1.5 Covariance and correlation factor

Covariance shows the difference between two variables, whereas correlation shows how two variables relate and how strong is the bond between them.

As shown in [2], it is suggested by the covariance that the correlation between one pixel and its neighbors fades after a certain characteristic length which mainly depends on the nature of the sensor.

According to this, the frequentist approach can only be applied under specific conditions such as when the correlation is overcome by calculating the average height values, and when the interaction be-

tween influence factors is eliminated.

2.2 Instruments

The measurements for this study were acquired by a laser scanning confocal microscope (Olympus LEXT 4100 (LSC)—figure 2.4). Using dedicated lenses, it enables both 3D scanner and laser measurement of samples at the same time.

Figure 2.4: Olympus LEXT laser scanning confocal microscope (LSC).

2.2.1 Steps

The LSC requires a warm-up time of about an hour for performing stable measurements. This precaution was always respected before the use, and also ensuring the absence of external vibration or other sources of noise which could affect the measurement. Moreover, exploiting the stage controls, a reference system on the component under measurements was always defined before starting the measurements. In this way the software controlled automatically several defined acquisition areas. This feature was essential for the identi-

fication and evaluation of several repeated measurements with the ”repositioning technique”, i.e. performing a random movement of the stage in between two acquisitions to minimize their degree of correlation.

Other available measurement parameters are: (fig. 2.5):

- The magnification: $5\times$, $10\times$, $20\times$, $50\times$ and $100\times$.
- The brightness: to control light exposure.
- The number of pixels used: up to 4096.
- The vertical range to include the total height variation of the component under measurement.

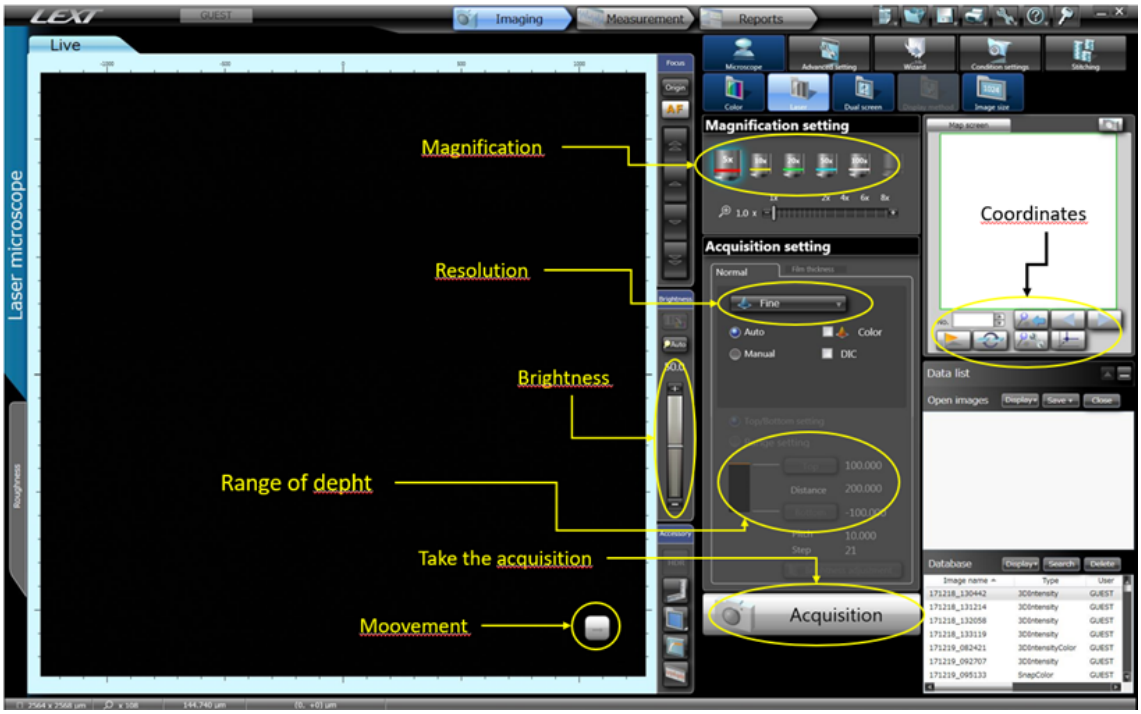


Figure 2.5: Screenshot of LSC user interface.

2.2.2 Parameters evaluation

The captured micrographs were post-processed and displayed with different color palettes. This characteristic helped to understand the components’ shape and geometry for an accurate analysis of the target object.

After the acquisition, the measurements of surface roughness, absolute and relative height, and other features were obtained by a commercial software (see chapter 4), which also provided the three-dimensional views of the surfaces and enlarged images of the micro- or nanoscale features shown in this thesis.

2.2.3 Advantages and disadvantages

Stylus instruments cannot measure narrow pits that smaller than the stylus tip radius. Moreover, the measured ”mechanical surface” [17] is affected by the interaction of the tip with the surface topography, requiring additional processing step to possibly remove the tip effect, thus increasing the measurement uncertainty. Conversely, an optical microscope, especially one using a laser source, can measure the surface roughness of micro geometries (”electromagnetic surface [11]”) at a considerably higher resolution due to a minute laser spot size.

Since a stylus instrument uses a hard needle-shape stylus, it is

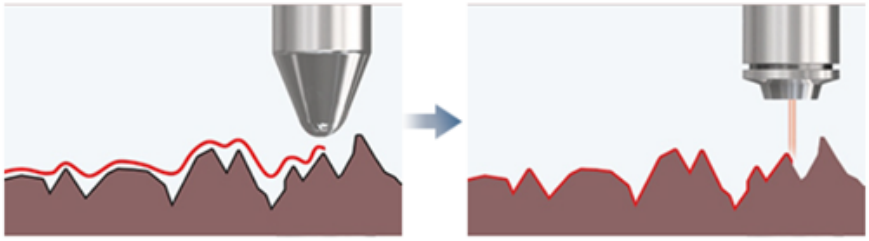
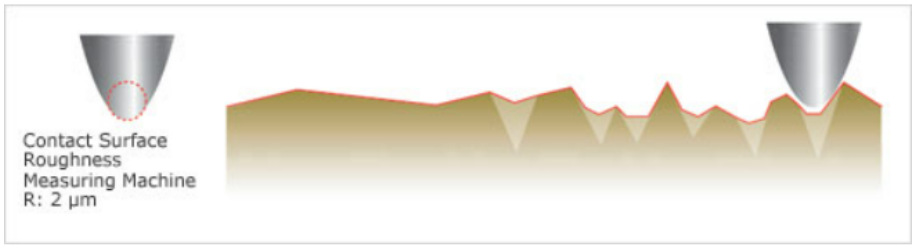


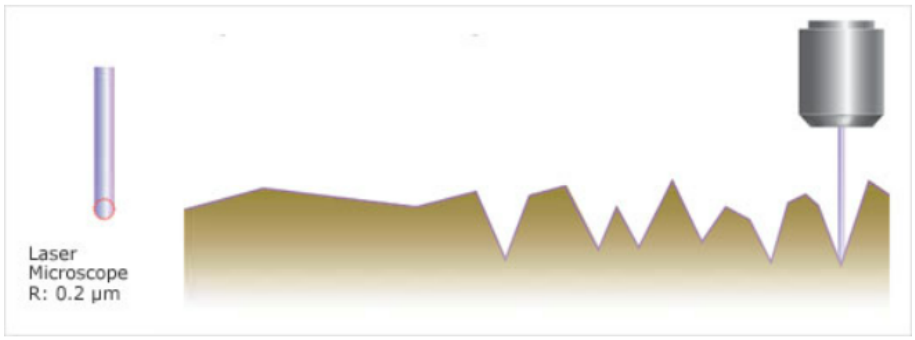
Figure 2.6: Difference between *white-light* microscope and laser microscope.

more likely to scratch the surface of a soft specimen, damaging or deforming it. In addition, with adhesive specimens, the stylus could attach to the specimen and be damaged when pulled loose, making it impossible to obtain correct results. Laser microscopes, which are non-contact, can perform accurate and non-destructive surface roughness measurement regardless of surface texture conditions. This characteristic is useful for measuring nano-structured polymer

samples without damaging them.



Stylus Roughness measuring Instruments



Laser Microscope

Figure 2.7: Difference between contact and non-contact instruments.

2.3 Specimens

Several specimens were measured, belonging to the following two main groups:

- Additive-manufactured areal standards (as defined in the ISO 25178-70 [18]):
  - Star-shape grooves (ASG) (5 component).
  - Cross grating (ACG) (9 components).
- Surfaces replicated by a silicon-based impression material.

The different configurations and manufacturing methods were analyzed in order to emphasize and gather information about those situations where a commercial software is not suitable for an effective analysis. In fact, in the cases analyzed (i.e. AM and manual replication), more flexibility would be needed locally on each component, where imperfections and defects due to the manufacturing process

may affect differently and in different portions. This work is assumed to be preliminary however necessary for an effective evaluation of the measurement uncertainty with a dedicated software.

2.3.1 Areal standard

In recent years additive manufacturing (AM) has spread rapidly, and today it is used across a wide range of applications. In fact, it achieves complex geometries and creates layer by layer high quality components. Furthermore, it reduces both wear and machine setup time and it recycles the unused material to avoid wastes.

Despite the excellent properties, the overall cost decreases from both time and material point of view making significant savings, and, therefore, it has a great potential for actual production.

A complete generic process is shown in figure 2.8.

The areal standards have been manufactured by VAT photopoly-

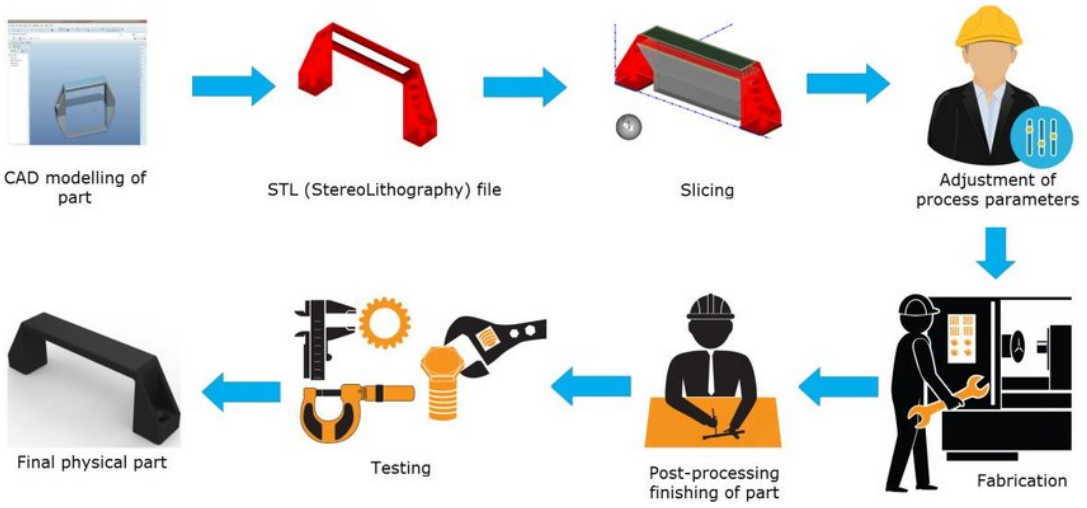


Figure 2.8: Schematic of a generic AM process.

merization [19]-[20] according to specific layout normally adopted for calibrating the metrological characteristics of surface topography measuring instruments.

Three out of fifteen features present on each component were measured to expose possible manufacturing differences. The measurement repeatability was also accounted by fifteen repeated acquisi-

tions per each feature (see figure 2.9).

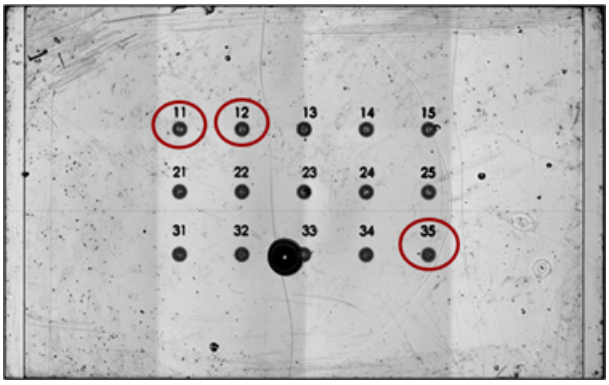


Figure 2.9: ASG features on a substrate.

ASG: star-shape grooves

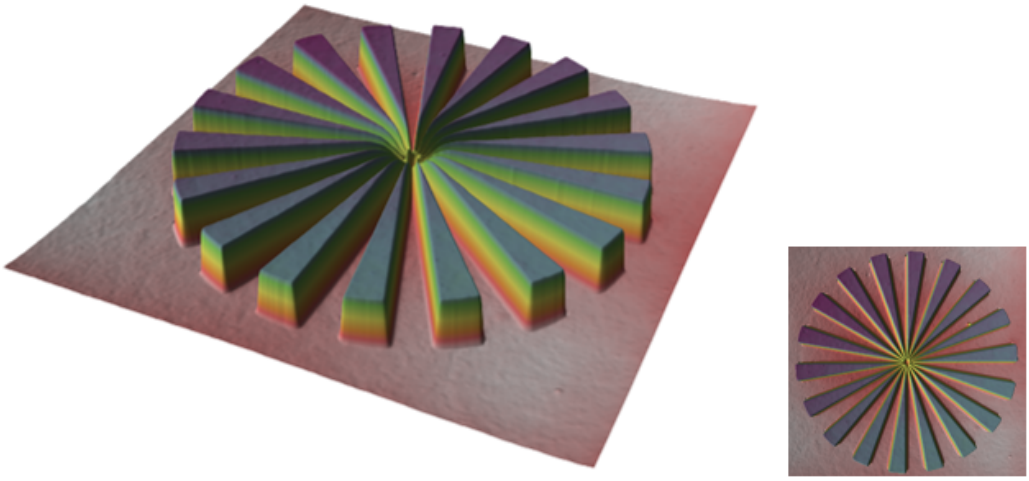


Figure 2.10: Example of ASG layout.

The component in figure 2.10 has a ”star” layout, where a two-dimensional array pattern is made up of constant-height elements and grooves oriented to form equal angles among them (see figure 2.10). A formal definition can be found in the ISO 25178-70 [18]. An example of additive-manufactured component (AM-ASG) is shown in figure 2.11.

ACG: cross grating

The component in figure 2.12 has a two-dimensional array pattern and is made up of equally spaced pillars. A formal definition can be found in the ISO 25178-70 [18].

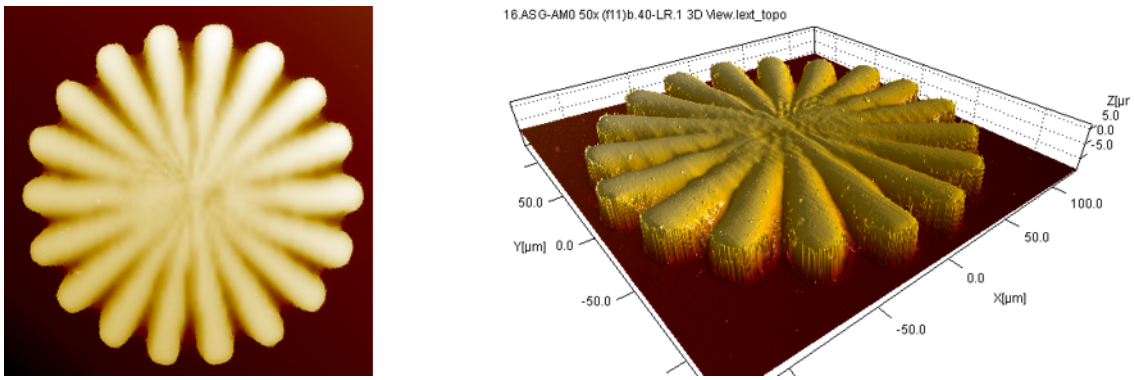


Figure 2.11: Example of measured additive-manufactured ASG feature.

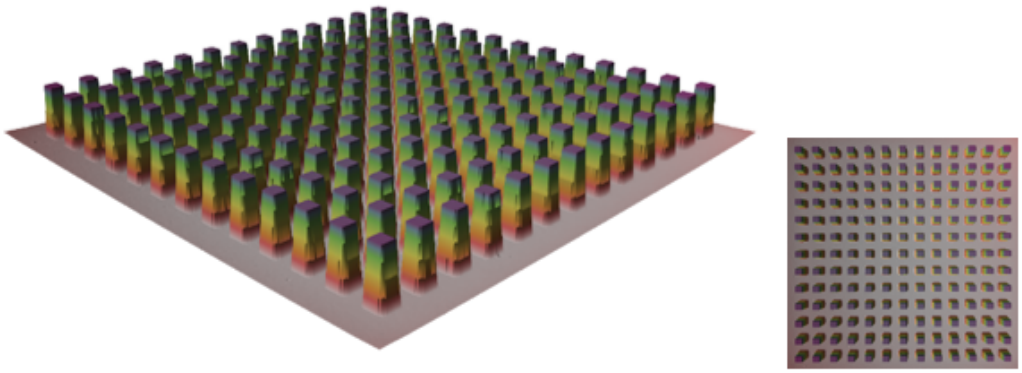


Figure 2.12: Example of ACG layout.

The analyzed additive-manufactured ACG (AM-ACG) were of two types corresponding to different setting of manufacturing parameters. They are named *sequential* and *full* in the following. Example of additive-manufactured components (AM-ACG) are shown in figure 2.13 (sequential) and in figure 2.14 (full). Moreover, this type of material measure can be identified by the following parameters:

- l_x: pitch in the X axis
- l_y: pitch in the Y axis
- θ: angle between the X and Y axis
- d: height of the pillars or depth of the pits

The settings were chosen as specified in the following:

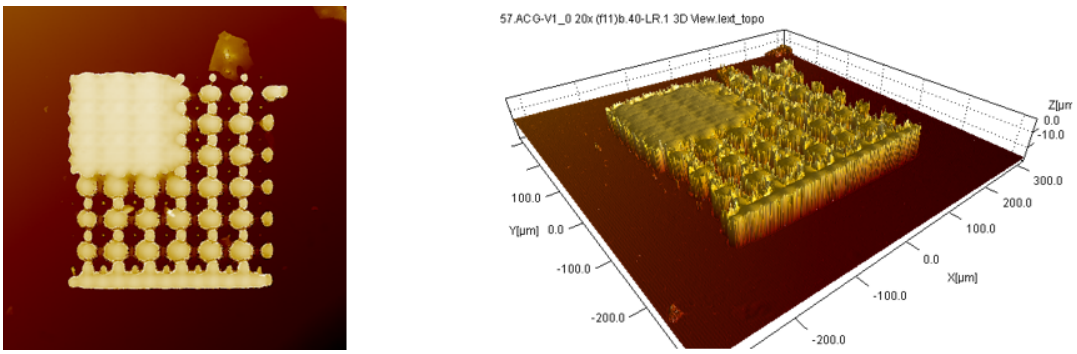


Figure 2.13: Example of measured additive-manufactured ACG feature (sequential).

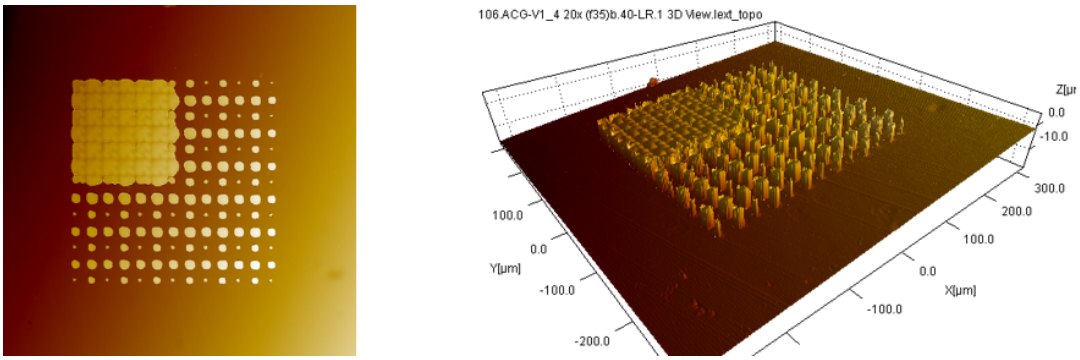


Figure 2.14: Example of measured additive-manufactured ACG feature (full).

- magnification:
  - AM-ASG: $20\times$ and $50\times$
  - AM-ACG: $20\times$
- brightness: 40
- number of pixels: 1024 $\times$ 1024
- field of view (FoV): 643 μm $\times$ 643 μm for $20\times$ lens objective; 258 μm $\times$ 258 μm for $50\times$ lens objective.

The AM measured components present several imperfection with respect to the reference geometry. The comparison between the measurements and their original layout in the figures 2.10 and 2.12 highlights that the contour is not uniform on the in-plane view, with several defects along the lateral profiles. This affects the properties of the component and represents a limiting factor against geometry

accuracy, roughness and other properties.

2.3.2 Surface replication

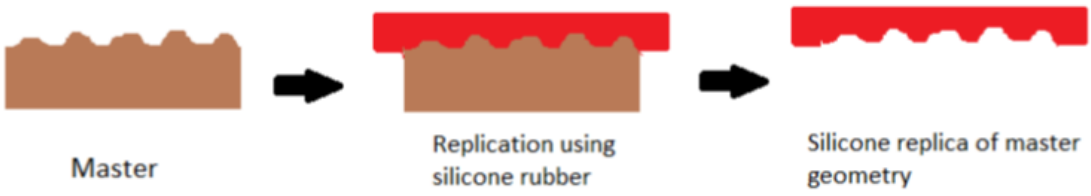


Figure 2.15: Qualitative schema of the replication process.

Figure 2.16: Silicon cartridge and gun for impression.

The other group of analyzed specimens were replicas obtained by a manual replication process. During a replication process, a master geometry must be conveyed to a substrate material (see figure 2.15). Nonetheless, due to the manual replication, unevenness of different nature can often be present on such surfaces, preventing a homogeneous analysis. The replication by impression media has several fields of application, e.g. in forensics and in all the situations where the component under investigation cannot be measured directly (pipes internal surfaces, industrial ovens, components on bridges, etc.) [21]-[22].

In the case considered, the Rubert standard 528x [23] was replicated by casting of the AccuTrans AB silicon-based impression ma-

terial (figure 2.16) [24]. The Rubert standard 528x has sinusoidal topography (type C according to ISO 25178-70 [18]) with nominal Ra of 500 nm and period of 50 μm. Both Rubert standard and an example of replica are in figure 2.17.

The replicated specimens were also measured by LSC. In fact, due

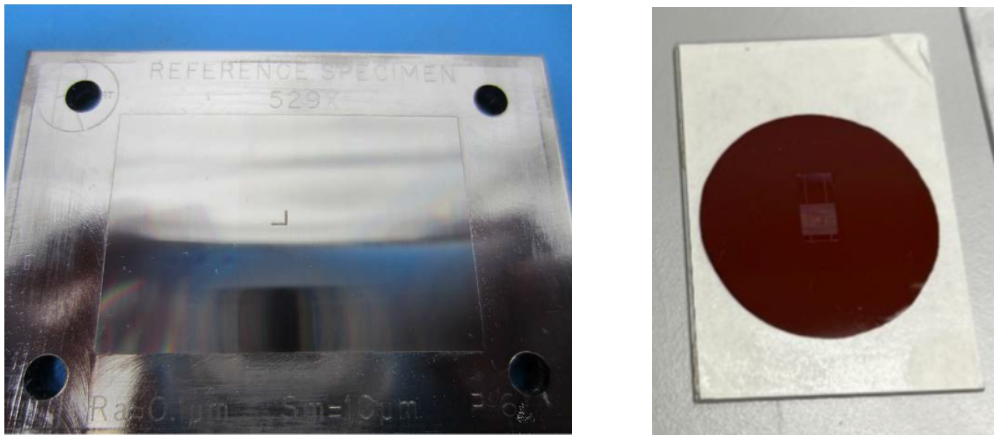


Figure 2.17: Acquired surfaces: master (left) and replicated surface (right).

to the soft nature of silicon-based replicas, a non-contact method for measuring was necessary to preserve the topography.

An L-shaped mark on the surfaces was used as local reference system (in fig. 2.18), allowing measurements of different replicated specimens at the same location.

Figure 2.18: L-shaped mark in the reference surface (master).

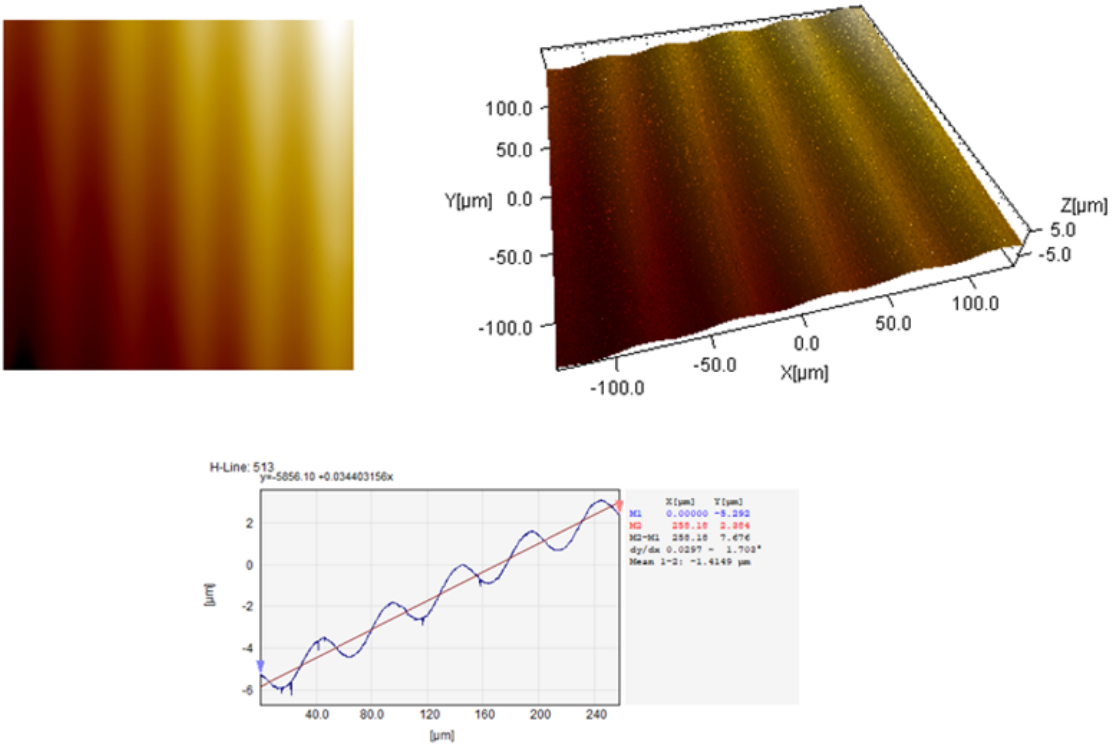


Figure 2.19: Example of replicated surface (3D view).

The figure 2.19 shows a 3D representation of the topography of the replicated surfaces.

3 Cluster

3.1 High performace computing (HPC)

The analysis of several replicated measurements, each consisting of million of points, according to the methodology described in chapter 2, and for the purpose of digital twinning, as it will be later discussed in details in chapter 5, requires massive memory and computational capability. Therefore, to make the computation feasible and having the results available in a reasonable time frame, high performance computing (HPC) was required. Thus, it was needed to change the structure of the matlab code and send it to cluster.
High-performance computing was resorted to analyze the measurements from the microscope.

A computer cluster refers to a collection of interconnected computers performing as if they were a single system. These computers are called nodes (or servers) and communicate through a local area network (LANs).
Typical components of a cluster are the same that can be found on a common laptop: CPUs (processors or cores), memory (or RAM), and disk space. The CPUs run the programs and perform calculations, the memory stores all the information about a current task and the disk is a computer's long-term storage for information it will need in the future.

3.1.1 Structure

Generally, the computing architecture is very complex. In fact, it consists of several types of nodes, each with a specific function such as disk storage management, user authentication or other infrastructure-related tasks. Regardless of the cluster sizes, which can greatly

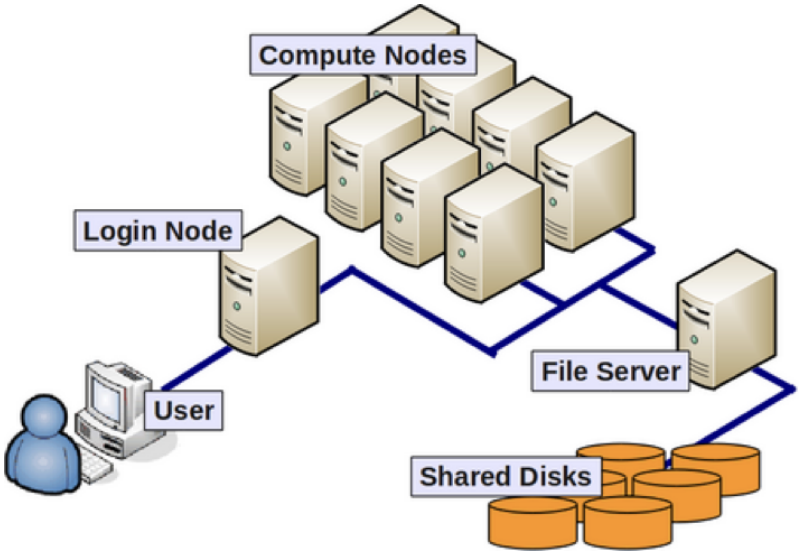


Figure 3.1: Cluster structure.

vary, clusters share the same basic framework (see figure 3.1). The main node is known as head node which provides access to computing resources. Normally, there is a relatively small number of head nodes, just one or two, and a much greater number of computing nodes. The heads nodes handle several functions including uploading and downloading files, setting up software, compiling the code, assigning and coordinating intensive tasks and jobs, running quick tests and checking traffic among all system units.

The computing (or worker) nodes are responsible for all the heavy duty operation in a cluster. While they may come in many shapes and sizes, the most important part is that they implement orders and follow the instructions as one powerful machine that executes them all simultaneously.

3.1.2 Benefits

In the process of writing a code, various combinations of factors, such as high volume of traffic, latency and downtime, can cause issues.

A load balancing strategy needs to be deployed to solve these points. The traffic is distributed among servers, allowing the system to scale

horizontally and not to overwhelm any single machine. Consequently, by using more than one server, each node can host a different set of projects resulting in better and more efficient performance of the carrying work. Moreover, all projects do not have to be run on each server, but, at the end, the results will be available for the entire cluster system.
Additional benefits of cluster are:

- Higher availability and flexibility: clusters provide fail-over support since, if one of the servers fails, the others will take on and pick up the workload and prevent time and data loss.
- Higher performance: as already mentioned, it is possible to work on more than one node simultaneously, distributing files and operations and improving the performance.
- Greater scalability: thanks to the cluster's faster evaluation, the amount of work can be increased without reducing the efficiency.
- Better data management: massive amount of data can be processed more efficiently with cluster.

3.2 ThinLinc

ThinLinc is a cross-platform remote desktop server used to transfer and upload data on DTU cluster. Using it, users can access their Linux-based desktops and applications from anywhere with an internet connection.

3.2.1 Setting and commands

Access to the ThinLinc platform is the first step: each user can log in with its own credentials as shown in figure 3.2. All files needed for the calculation can be imported via an online drive or specific programs, and gathered into a folder.
In a following step, the terminal emulator must be opened from the application or directly from the folder. Figure 3.3 shows the interac-

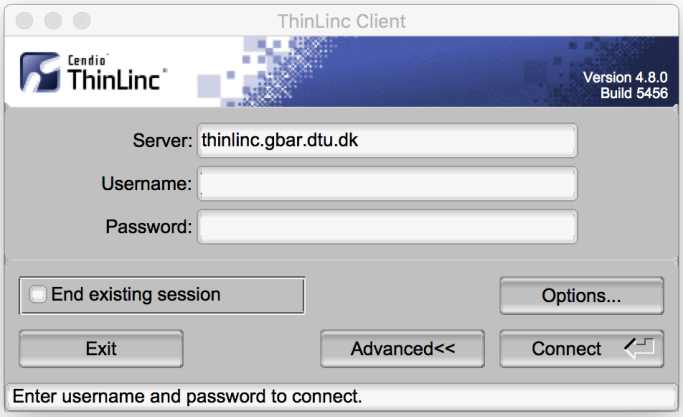


Figure 3.2: ThinLinc login window.

tive session (xterm).

Several operations can be performed on the terminal: choose the

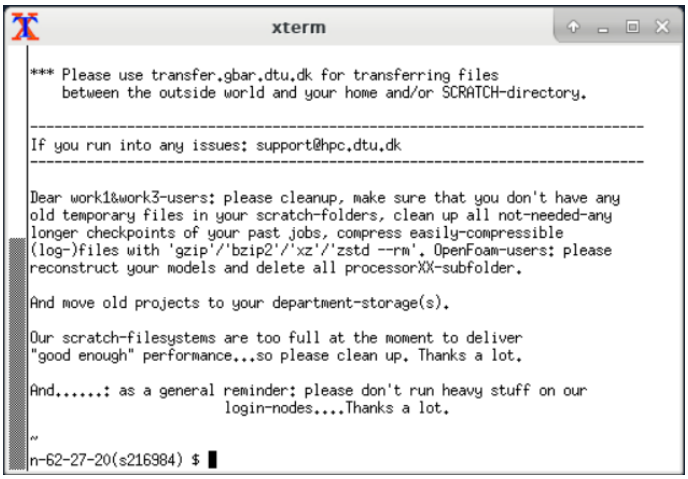


Figure 3.3: Linux prompt: xterm window.

directory, submit and check a job status, stop one or more jobs, visualize memory usage and run time, view which nodes are involved and their status. The corresponding commands are listed in table 3.1.

3.2.2 Batch file

The *Resource Manager* is a software tool that provides efficient utilization of cluster resources. In this way, the user does not have to execute the code himself, but “asks” the cluster to run it. The Re-

bsub	submit a job
bstat	check the job status (queue, job name, pend/run/done/exit job status, stanting and total time)
bstat with [-h,v,C,M, etc]	show message, version history, CPU usage, etc..
bkill	remove a job from the queue
bacct	display a summary of accounting statistic for all finished job submitted by the user
module list	show the list of loaded modules
module avail	show the list of all the available modules
module load/unload	load/unload a module
nodestat	get a list of the resources that can be selected

Table 3.1: ThinLinc commands

source Manager is in charge of parsing the code, setting up servers, scheduling work at different times, receiving and analyzing the job script containing the users' requirements and demands, and assigning cores and resources to achieve the purpose.
Consequently, the user has to provide a job script, also called batch file, containing all the information required from the cluster for handling the different requests, while monitoring and running a large number of jobs simultaneously without any interference.
An example of specification of a job script is in figure 3.4, while listing 3.1 and listing 3.2 (at the end of the section) show the specific batch files for running Matlab code and Comsol-Matlab configuration code.

Resources refer to the number of processors, cores or nodes assigned to a job, memory usage and other specific features. The main constraints are related mostly to the maximum time, but also to cores and jobs. Constraints are necessary to share the cluster among different users.

```
#!/bin/sh
# embedded options to bsub - start with #BSUB
#BSUB -J MyDistributedMatlab      # our name ---
#BSUB -q hpc                      # choose queue --
#BSUB -n 4                        # Number of cores requested
Or #BSUB -R "span[hosts=1]"       # cores must be on the same host
Or #BSUB -R "span[ptile=N]"       # reserve N cores on each machine
#BSUB -R "rusage[mem=2GB]"        # 2GB of memory needed per core/slot
#BSUB -M                          # memory limit for all process of the job
#BSUB -u your_email_address       # email address
#BSUB -B                          # Notify by email when execution begins
#BSUB -N                          # Notify by email when execution ends
#BSUB -o Output_%J.txt            # Output File
#BSUB -e Error_%J.txt             # Error File
#BSUB -W 10:00                    # estimated time: dd:hh:mm:ss

# -- end of LSF options --
# -- commands you want to execute --
# module load matlab/R2021a       # specific matlab module remember to load it


myapplication.x < input.in > output.out
example:
matlab -nodisplay -batch my_distributed_matlab -logfile MyDistributedMatlabOut
```

Figure 3.4: Example of batch file.

3.2.3 Script execution

Once the batch job is ready, it can be executed on the terminal emulator using the command: *bsub batch_file_name*. Afterwards, it is also possible checking the status of the job just sent with the command: *bstat*.

The display will appear as shown in the figure 3.5.

The execution generates and stores three additional files in the folder where user's scripts (see figure 3.6):

- Error file
- Output file

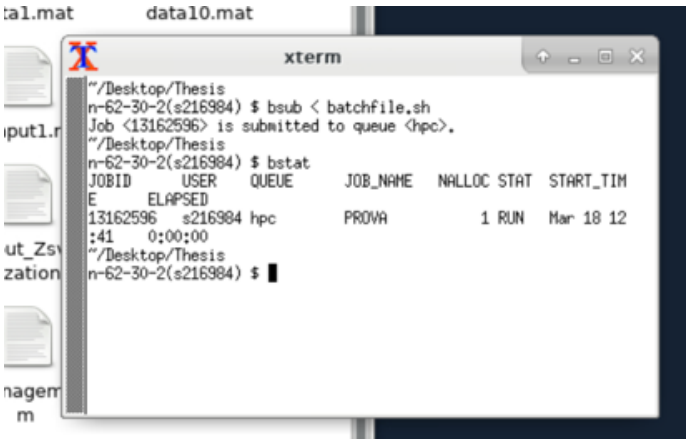


Figure 3.5: Command for running a script in xterm window.

- General review (which includes output and error)

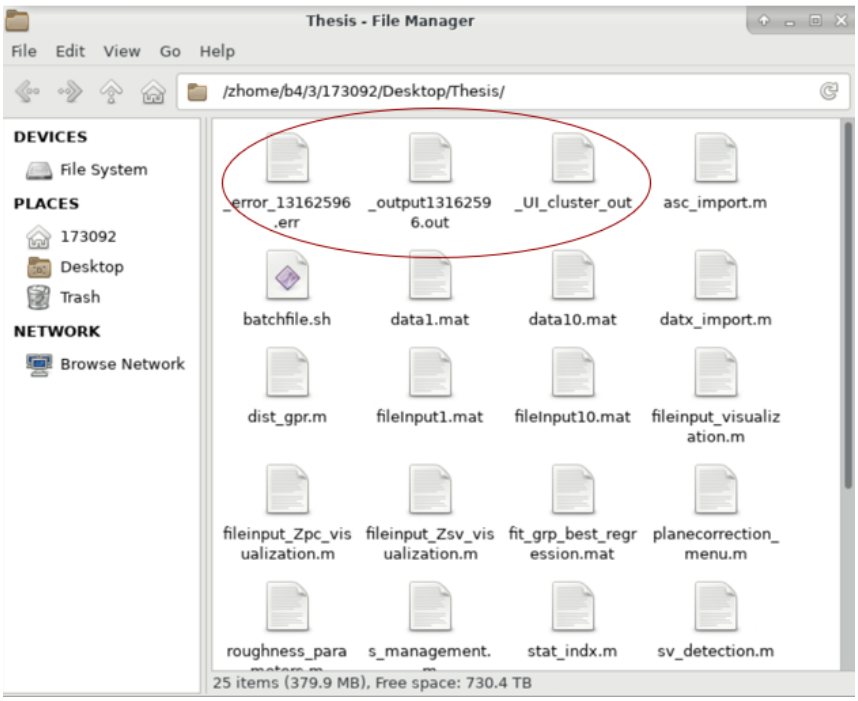


Figure 3.6: Output files.

Additionally, the use of graphics is not possible, i.e. the desktop GUI, plots, etc. on cluster evaluation. However, these files can be saved, as well as other outputs (structures, variables, etc..) by including specific commands in the main script.

3.3 Parallel computing and Arrays

MATLAB provides useful tools for parallel processing from the Parallel Computing Toolbox. To optimize and speed up the work proper data structure was required. In fact, by making use of *parfor-loops* and *spmd* statement, and *tall*, *distributed* or *codistributed* arrays, loop interactions can occur simultaneously on both personal computers as well as on cluster.

The usage of these structures is shown in chapter 4. It requires first to open the ”parallel pool” on matlab or on the cluster. In other words, the client needs to open the communication line to get data to the matlab workers (see figure 3.7).

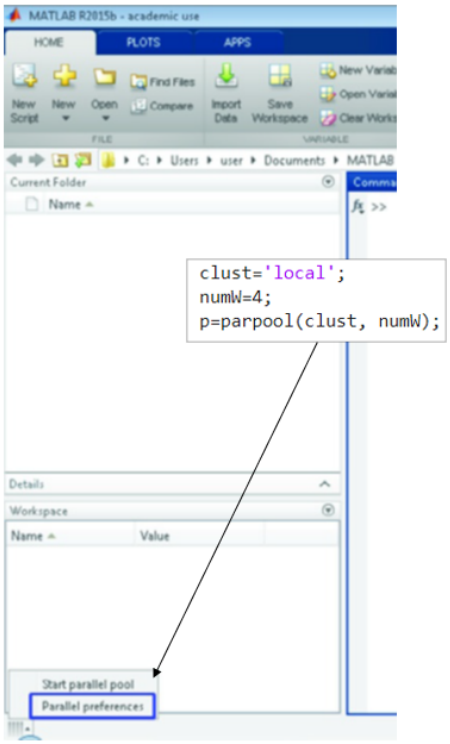


Figure 3.7: Matlab parallel pool.

The parallel pool can be shut down once the computation is completed.

3.3.1 Parfor-loop

Parfor-loop consists of a series of statements run over a range of values and it can be useful when there is a slow for-loop.
Since each execution of a parfor-loop is an independent iteration (see figure 3.8), there is no guarantee that they will be synchronized with each other. Therefore, they elaborate at the same time certain amount of jobs, and merge all the results at the end.

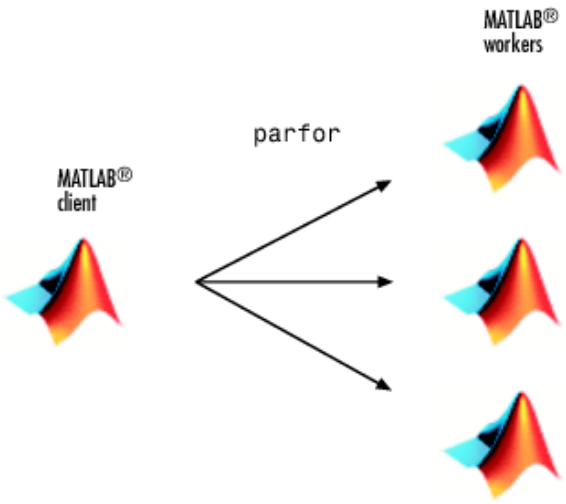


Figure 3.8: Schematic of a parfor-loop execution.

3.3.2 Tall arrays

Tall arrays are used to elaborate large data sets with one dimension much greater than the others allocated in a datastore, as shown in figure 3.9. Instead of loading all of the data into memory at once, these arrays let you work with big sets of data in smaller blocks that fit in memory. Moreover, the number of passes through the data is minimized during calculations, allowing them to be processed with common functions.
Essentially, it is the same procedure described previously; several operations and functions work similarly, but the results are only evaluated when they are explicitly requested using the “gather” function.
The produced code did not utilize tall arrays because the data to

be analyzed were micrograhs with a 1024 $\times$ 1024 number of pixels, which were not ideal for the type of structure that tall arrays require.

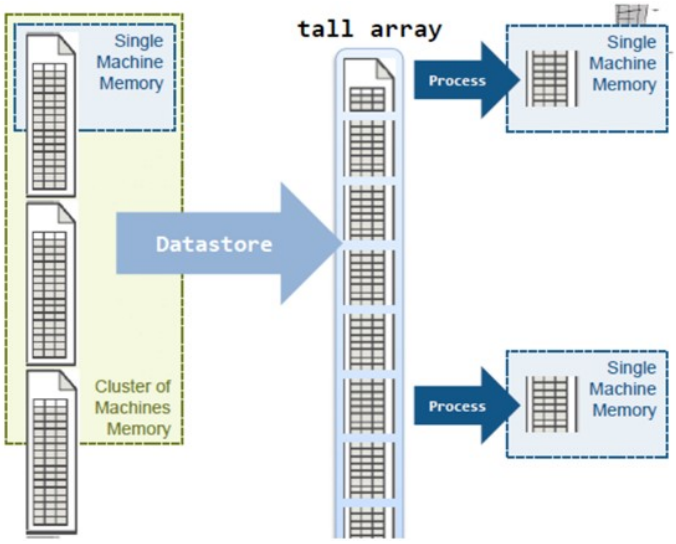


Figure 3.9: Tall array structure.

3.3.3 Distirbuted and codistributed arrays

Several ways are available in Matlab for analysing a N $\times$ N matrix (same dimension in both sides). As developed in the thesis Matlab code, one can use distributed and codistributed arrays, figure 3.10. The main difference between distributed and codistributed array is that the former is distributed among workers in the parallel pool, whereas the latter is distributed among the workers where the code is executed. Codistributed can also be modified and manipulated by the same main code. Moreover, it is possible to switch from one type to the other through using the *spmd* (single program multiple data) statement. In fact, a distributed array created on a client can be accessed in spmd as a codistributed, and vice versa.

In the produced code, it is possible to see how the structure was created and converted into distributed variables. Afterwards, according to the needs, one could access them as codistributed with spmd.

Table 3.12 and in table 3.13 provide a general guide from Matlab

documentation [25] to choose the arrays structure properly.

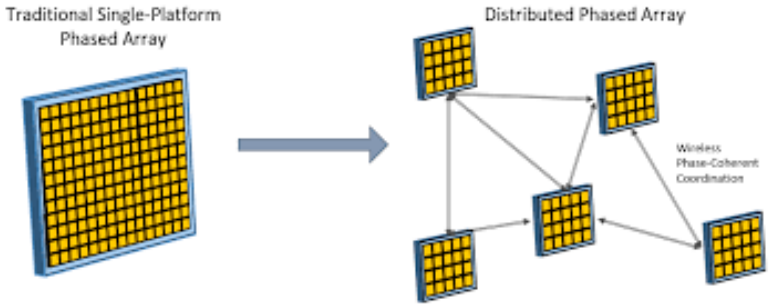


Figure 3.10: Distributed array

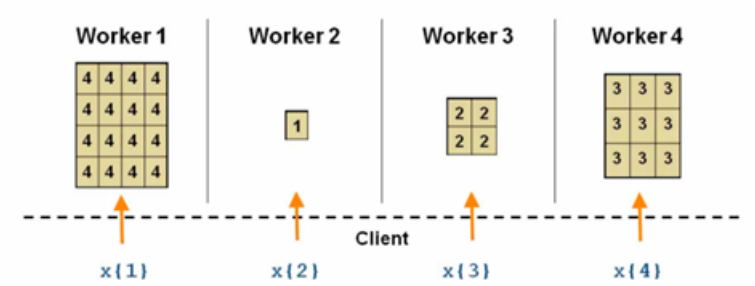


Figure 3.11: Codistributed array structure.

Problem	Solution	Structure
Is your data too big?	To work with out-of-memory data with any number of rows, use tall arrays. This workflow is well suited to data analytics and machine learning.	Tall Arrays for Out-of-Memory Data
	Use tall arrays in parallel on your local machine.	Use Tall Arrays on a Parallel Pool
	Use tall arrays in parallel on your cluster.	To use tall arrays on a Hadoop cluster
	If your data is large in multiple dimensions, use distributed instead.	Distributing Arrays to Parallel Workers

Figure 3.12: Guidelines for choosing arrays - 1° table.

Problem		Solution	More information
How data fit in memory?	In local memory	Distributed function	Distribute an existing array from the client workspace to the workers of parallel pool. Optimal for testing.
	In memory of the cluster (not in local memory)	Datastore with distributed function	Parallel pool
	Nor in local memory neither in cluster	Datastore with tall arrays	Partition and process data in chunks

Figure 3.13: Guidelines for choosing arrays - 2° table.

```
#!/bin/sh
#BSUB -J ctJOB1
#BSUB -q hpc
#BSUB -W 24:00
#BSUB -n 4
### -- reserve N cores on each machine, up to the total number of cores
    requested with the-n flag --
#BSUB -R "span[hosts=1]"
#BSUB -R "rusage[mem=2GB]"
#BSUB -u s216984@student.dtu.dk
#BSUB -B
#BSUB -N
#BSUB -o _output%J.out
#BSUB -e _error_%J.err

module load matlab/R2021b

matlab -nodisplay -batch UI_codistr_new -logfile _UI_codistr_out
```

Listing 3.1: Batch file for running Matlab code.

```
#!/bin/sh
#BSUB -J comsolmatlab
#BSUB -q hpc
#BSUB -W 05:00
#BSUB -n 4
### -- reserve N cores on each machine, up to the total number of cores
    requested with the-n flag --
#BSUB -R "span[hosts=1]"
#BSUB -R "rusage[mem=2GB]"
#BSUB -u s216984@student.dtu.dk
#BSUB -B
#BSUB -N
#BSUB -o _outputcomsol%J.out
#BSUB -e _errorcomsol_%J.err

module load comsol
unset JAVA_TOOL_OPTIONS

comsol mphserver matlab

CSCMD=random_surf
MLSCR=random_surf_matlab




##############################################
### Don't monkey with the stuff below!!! ###
...
```

Listing 3.2: Batch file running a Comsol model.

4 Software

Three different types of software were used to analyze the micrographs obtained by the LSC:

- Spip
- MountainLabs
- Matlab

In all cases, the micrographs were used as input and, after applying the plane correction, it was possible to perform statistical analysis, estimate the level of uncertainty, and provide graphical representations of the samples.
As anticipated, the number of repeated measurements and the subsequent memory requirements necessitated the use of clusters to process the data. Thus, some of the original Matlab code structure and parameters had to be adapted and modified to work in parallel. Moreover for optimum results, the three types of software provided different plane correction settings.

(a) (b) (c)

Figure 4.1: Three types of software used in the project. a) SPIP [26]; b) MountainsLab [27]; c) Matlab [28].

4.1 Spip

Scanning probe image processor, also called SPIP, is a user interface useful to process and analyze images at nano- and microscale. It reads the measured micrographs, extracting data in different formats, and includes several toolkit for plane correction and for reporting the analysis of results.

4.1.1 Steps

According to the previous section, the measured data were imported and opened in Spip as input files and a 3D visualization of each acquisition was built (see figure 4.2). The graphs provided a general overview of the sample's shape and characteristics, and were helpful in determining the type of plane correction to apply.
Spip allows applying both automatic and manual plane correction. The manual one are managed by the user, instead automatic one is divided into four further methods, shown in fig. 4.3 and in fig. 4.4. They are:

- global leveling
- global bow removal
- linewise leveling
- linewise bow removal

They represent a quick access to 1^{st} order plane correction methods.

It is possible to observe from the figures (4.3, 4.4, 4.6 and 4.7) that sometimes the automatic plane correction does not work properly. So, it is necessary to apply the manual leveling.
In that case, the cross section tool defines two main profiles on the sample. Once entering in the plane correction set up and setting the mean value of the height equal to zero, the plane can be twisted and

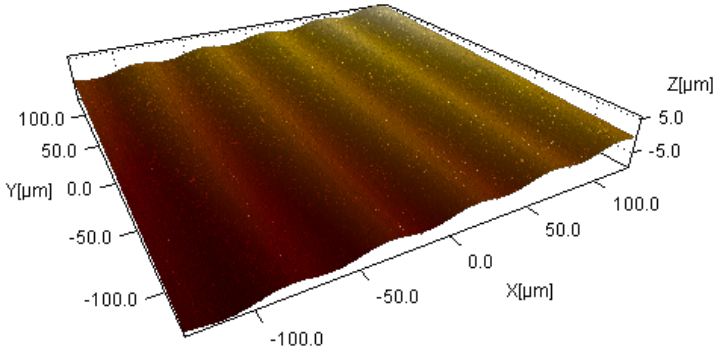


Figure 4.2: 3D visualization in Spip of raw replicated surface measurement.

(a) (b)

Figure 4.3: Global plane correction of raw replicated surface measurement: a) Levelling; b) Bow removal.

(a) (b)

Figure 4.4: Linewise plane correction of raw replicated surface measurement: a) Levelling; b) Bow removal.

the profile is adjusts until satisfied results are achieved.
In figure 4.8, it was possible to notice that the surface of the sample appeared inclined from both axes views, so it was turned to the left-/right or up/down to look flat from both sides as in fig. 4.9.

Once leveling has been performed, the surface topography parameters evaluation has been carried out, another 3D graph of the

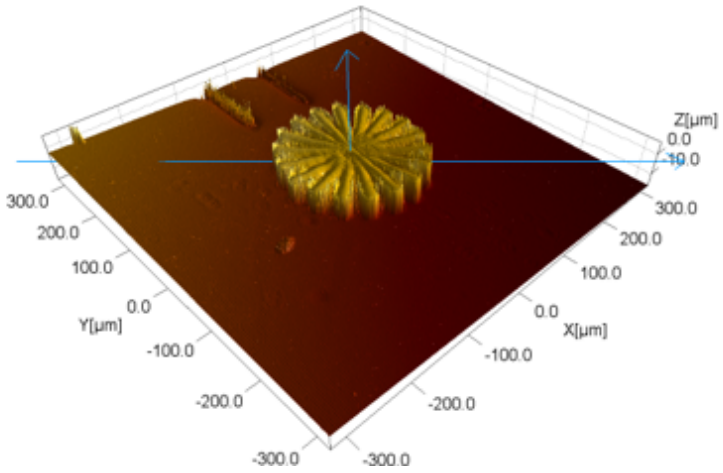


Figure 4.5: 3D visualization of ASG raw measurement in Spip.

(a) (b)

Figure 4.6: Global plane correction of ASG raw measurement: a) Levelling b) Bow removal.

(a) (b)

Figure 4.7: Linewise plane correction of ASG raw measurement: a) Levelling b) Bow removal.

modified surfaces after the plane correction was generated, and, at last, the corrected surfaces were saved in new files to be used or modified in the future.

(a) (b)

Figure 4.8: Profiles on the measurement surface before plane correction: a) x axis b) y axis.

(a) (b)

Figure 4.9: Profiles on the measurement surface after plane correction: a) x axis b) y axis.

4.2 MountainsLab

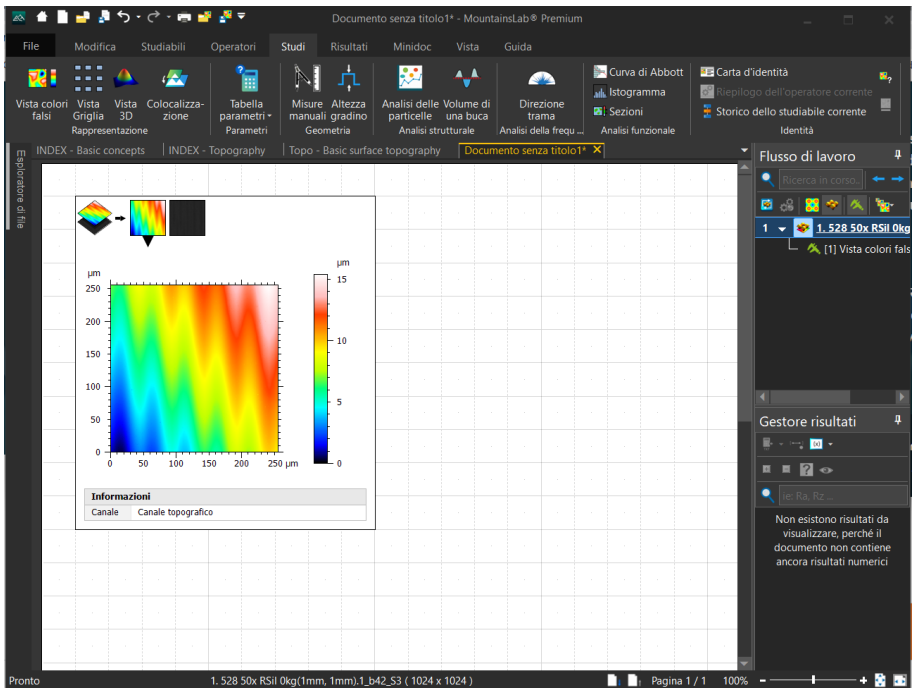


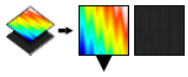
Figure 4.10: Main window of MountainsLab software.

Data and parameters of the specimens were collected, analyzed and extracted also from MountainsLab, another commercial software that can perform advanced processing and evaluation. This software is able to performed 2D and 3D model generation from imaging data, measure any structure, execute particle analysis and manipulate datasets with several layers. Moreover, each information is gathered on a series of pages and it is possible to review the steps done thanks to the workflow window.

4.2.1 Steps

As with Spip, the acquisitions were opened within the software and visualized in the 3D view, fig 4.11.

A variety of plane correction options are available in the software



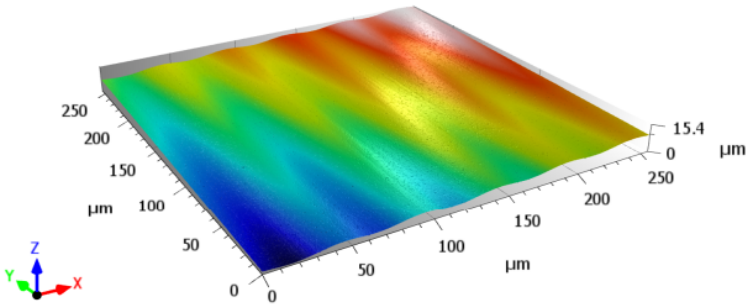


Figure 4.11: 3D visualization of raw measurement in MountainsLab.

(see figure 4.12):

- Polynomial of n-th degree evaluated by least square method: the polynomial function is automatically created by the software and it is possible to choose from the first to the thirteenth degree.
- Sphere or cylinder evaluated by least square method.

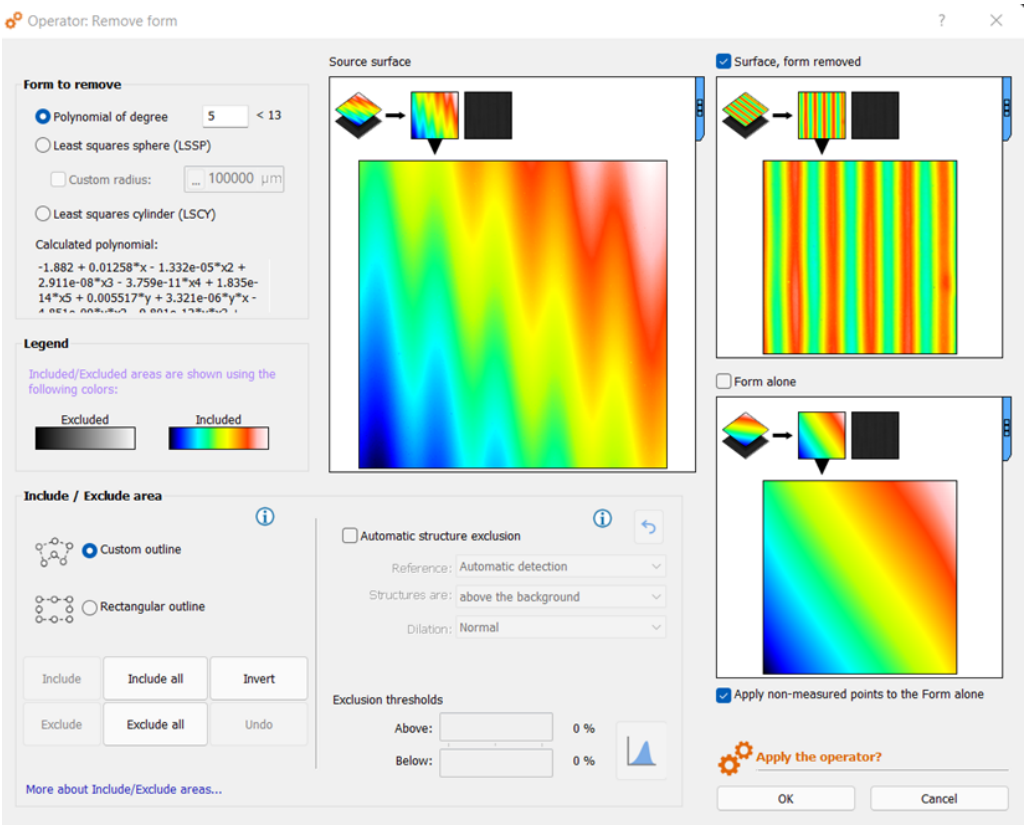


Figure 4.12: Plane correction set up in MountainsLab.

- Application of the following by fitting the models on user defined of automatically identified, i.e. by structure exclusion, portion of the surface.

In the considered cases, simple plane leveling was applied to the samples. By the comparison of the results obtained by the different methods, this choice is the most suitable.
After the plane correction, a 3D visualization was provide and it was also possible to perform the parameters evaluation of surface with dedicated tools. To conclude, the new files were saved.

4.3 Matlab

In addition to commercial software used to perform post-processing and parameter evaluation, a Matlab code created for a previous MSc project was used [29]. In particular, the code was optimized for cluster computing and augmented in its characterization capa-

bilities. Thanks to the upgrades, this software package was able to control acquisitions of big dimension samples (1024$\times$1024 pixels) and offered more extensive post-acquisition image processing, including 3-D rendering. In order to handle these type of samples, changing the whole structure was necessary.

4.3.1 Code

In the first part of the code, it was possible to choose or not the plane correction, handle spikes and voids, and print the output images.
The acquisition of the files (see listing 6.4) involved four different formats: .lext, .daxt, .asc, .txt, (adopted by some microscope manufacturers), and the code worked also with ASCII files. The data were imported and stored as distributed arrays in a dedicated structure.
Once all data have been gathered and are ready to be elaborated, the next step is the detection and the management of spikes and voids (see listing 4.2). This is achieved by a Gaussian process regression performed by a subroutine, i.e. the ”gpr function”. It seeks an ideal surface fitting with the samples, also known Kernel, which has to be compared to the acquired original surfaces. Once the kernel is estimated, the residuals of the model are studied, and each residual point is considered as an extreme value, i.e. an outlier, can be regarded as a spike, hence, detected and managed accordingly.
Subsequently, the plane correction is performed, script in listing 4.3. Based on the least square method, a surface fitting is created and processed and the resulting model is subtracted from the original surface. At current state, the code implements two types of polynomial plane correction distinguished only by the way in which they are constructed:

- surface leveling: the form is modeled as a polynomial surface with first degree both in x and y axis, i.e. a plane; based on the library model poly11.

```
%% Input
data=struct('XIn',[],'YIn',[],'ZIn',[],'Zpc',[],'Zv',[],'Z_thr',[],'Z_gpr
    ',[],'Z',[],'parameters',[],'Ix',[],'Iy',[],'time',[],'C',[],'Cf',[]); %
    Xin,Yin,Zin: initial coordinates (before any correcion)
load('saveVar.mat'); %simplify names and directory's files

for q=1:lCount
    [data.XIn,data.YIn,data.ZIn(:,:,q),...
        data.Zv(:,:,q),data.parameters,...
        data.Ix,data.Iy,data.time(q,1)] = lext_import(strcat('L',num2str(
            lCount),'.lext')); %define parameters
        data.Zv=logical(data.Zv);
end
for q=lCount+1:lCount+dCount
    i=1;
    [data.XIn,data.YIn,data.ZIn(:,:,q),...
        data.Zv(:,:,q),data.parameters,...
        data.Ix,data.Iy,data.time(dCount,1)] = datx_import(strcat('D',num2str(
            i),'.datx')); %define parameters
        data.Zv=logical(data.Zv);
    i=i+1;
end
for q=lCount+dCount+1:lCount+dCount+aCount
     i=1;
     [data.XIn,data.YIn,data.ZIn(:,:,q),...
        data.Zv(:,:,q),data.parameters,...
        data.Ix,data.Iy,data.time(dCount,1)] = asc_import(strcat('A',num2str(i
            ),'.asc'));
     i=i+1;
end
for q=lCount+dCount+aCount+1:lCount+dCount+aCount+tCount
    i=1;
    [data.XIn,data.YIn,data.ZIn(:,:,q),...
        data.Zv(:,:,q),data.parameters,...
        data.Ix,data.Iy] = txt_import(strcat('T',num2str(i),'.txt'));
        data.Zv=logical(data.Zv);
    i=i+1;
end
    %data.Zv=logical(dt.asc.Zv);
data.XIn=distributed(data.XIn);
data.YIn=distributed(data.YIn);
data.ZIn=distributed(data.ZIn);
data.Zv=distributed(data.Zv);
clear i
fprintf('Acquisition OK \n');
```

Listing 4.1: Matlab code for the acquisition of files.

```
% GPR method
[data,Sa,Sq,fit_grp] = dist_gpr(data,tot,m);
fprintf('dist_gpr OK \n');
%
fit_grp=fitrgp(SPos,Sresp,'BasisFunction','constant','
    OptimizeHyperparameters',{'KernelFunction','KernelScale','Sigma'},...
        'HyperparameterOptimizationOptions',struct('UseParallel',1));%,'
            ActiveSet',(ones(size(Sresp),'logical')));
```

Listing 4.2: Matlab code for spike and voids correction.

- surface curvature removal: the form is modeled as a polynomial surface with second degree both in x and y axis; based on the library model poly22.

Once the form correction has been applied, the acquisitions should consistently appear uniform and without systematic form errors. Therefore, the surface parameters and statistical indexes can be evaluated with the previous formulas, as shown in 4.4. Furthermore, uncertainty (script 4.5) and covariance and correlation factor (script 6.9) evaluation are computed. This is an additional feature that only the Matlab code is able to provide compared to the other commercial software.
As a last step, the 3D representation of samples will be created for each micrograph using almost the same function (4.7), with the only difference being the input variable.

4.3.2 Memory management

In accordance with expectations, due to the large amount of time and memory required, the entire code had to be uploaded and run on a cluster with appropriate architecture. In fact, the whole code presents references to the parallel computation, such as parfor, distributed or codistributed, parpool, etc.
When converting data in distributed arrays, a different type of structure was required in order to run certain functions, such as ”prepareSurfaceData” and ”fit”; to execute these functions, the com-

```
%% Plane Correction
[data] = planecorrection_menu(tot,data,indx_pc,tf_pc); %choose the type and
    apply the correction

% cases
    ...
    switch indx
        case 1
            [XOut,YOut,ZOut]=prepareSurfaceData(XIn,YIn,ZIn);   %transforms
                data for surface fitting with the fit function
            sf=fit([XOut,YOut],ZOut,'poly11');                  %create a
                surface model
            Zsf=sf(XIn,YIn);
            Zpc=ZIn-Zsf;                                        %difference
                between the surfaces
            ...
        case 2
            [XOut,YOut,ZOut]=prepareSurfaceData(XIn,YIn,ZIn);
            sf=fit([XOut,YOut],ZOut,'poly22');
            Zsf=sf(XIn,YIn);
            Zpc=ZIn-Zsf;
            ...
        ...

fprintf('Plane correction OK \n');
```

Listing 4.3: Matlab code for plane correction.

```
%% Statistical Indexes and Surface Parameters
%mean, median, variance, standard deviation
%Sq and Sa
statIndx=struct('M',[],'Me',[],'V',[],'S',[]);
% Statistical Indexes for files .asc
[statIndx.M,statIndx.Me,statIndx.V,statIndx.S] = stat_indx(data.Zpc);
fprintf('Statistical indexes OK \n');

% statistical indexes
    M = mean(Zpc,3);
    Me = median(Zpc,3);
    V = var(Zpc,0,3);
    S = sqrt(V);
%  surface parameters
    coord=[X Y Z(:,q)];
    A=max(coord(:,1))*max(coord(:,2));
    px=abs((max(coord(:,1))-min(coord(:,1))))/m;
    py=abs((max(coord(:,2)))-min(coord(:,2)))/m;
    Sq_pc(q)=(sqrt(sum((coord(:,3).^2).*(px*py),'omitnan')/A))*10^3; %nm
    Sa_pc(q)=(sum(abs(coord(:,3)).*(px*py),'omitnan')/A)*10^3; %nm
```

Listing 4.4: Matlab code for statistical indexed and surface parameters evaluation.

```
%% Uncertainty evaluation
uncertainty=struct('q',[],'U',[]);
% Uncertainty evaluation for files .asc
[uncertainty.q,uncertainty.U] = uncertainty_evaluation(data.Zpc);

    % Expected value
    q = mean(Z,3);

    % Experimental standard deviation of the mean
    s_q = std(Z,0,3)/sqrt(numbObs);

    % Expanded uncertainty
    k = 2;                      % Confidence level
    U = k*s_q;

fprintf('Uncertainty evaluation OK \n');
```

Listing 4.5: Matlab code for uncertainty evaluation.

```
%% Covariance and Correlation factor
    data.C(:,:,q)=cov(Zpc(:,:,q));
    data.Cf(:,:,q)=corrcoef(Zpc(:,:,q));
```

Listing 4.6: Matlab code for covariance and correlation factor estimation.

```
%% 3D Visualization
v = '3D visualization of the imported surface';
ind=1;                                  %original data
file_visualization(tot,data,v,ind);
clear v
    ...
        ind=2                           %plane correction
        ind=3                           %spikes and voids
        ind=4                           %covariance
        ind=5                           %correlation factor
    ...
        U_visualization(uncertainty);   %uncertainty
```

Listing 4.7: Matlab code for 3D visualization of raw measurements.

```
%% Plane correction
...
    XIn=gather(XIn);
    YIn=gather(YIn);
    ZIn=gather(ZIn);
    m=length(XIn);
    [XOut,YOut,ZOut]=prepareSurfaceData(XIn,YIn,ZIn);
    sf=fit([XOut,YOut],ZOut,'poly11');
```

Listing 4.8: Function ”gather”.

mand ”gather” was used to turn the arrays into double and to allow the code to work properly, script 4.8.

To speed up the computation, the commands ”parfor” or ”spmd” were used, script in listing 4.9; both allowed to reduce the working time, as it is possible to see in listing 6. However, the first one allocated the work evenly across all the workers, while spmd enabled access to the distributed (or double) arrays as codistributed, and allocated work unevenly.

```
%% Extraction of image parameters from the file
...
    spmd
        codist=codistributor1d(2,codistributor1d.unsetPartition,[parameters(2)
            ,parameters(3)]);
        Z=zeros(parameters(2),parameters(3),numlabs,codist);
        Z=Z+array(1:1:parameters(3),1:1:parameters(2)).*scala;
                                                % Definition of the matrix
            with all the measurement data.
        Zin=getLocalPart(Z);
    end
...
    Zin=[Zin{:}];
    ZIn=Zin(:,:,1);
...
data.ZIn=distributed(data.ZIn);
```

Listing 4.9: Example of codistributed structure and spmd command.

5 Digital Twin

The concept of digital twin was introduced the first time in 2002 by Michael Grieves at the University of Michigan. Since then, the concept has evolved and today is much more widespread.

Digital twin is a digital representation, or abstraction, of real physical objects, processes or systems. A computer program uses real-world data as input to generate predictions or simulations about how objects will behave and be affected before they are actually produced and deployed.

Until some years ago, the process of building a digital twin has been difficult because most of the steps are non-linear. However, machine learning, big data, and artificial intelligence have made it more achievable, even if still elaborate. This is explained in details in figure 5.1.

In comparison to the traditional simulation, in fact, the digital twin is based on real equipment; it is constantly in contact with real-world, in real-time connection, providing stable measurement conditions for a more accurate representation and trying to replicate uniform surface characteristics. A conventional simulation, instead, tends to be a static representation of a system or one of its part; it is modeled using CAD software and it cannot meet the requirements of real-time because it only represents real objects, without a direct connection and update to the real world.

In short, the benefit of creating a digital twin model is that it can facilitate analysis, enhance strategic technology trends, prevent failures, reduce the cost of system verification, perform tests on processes and plan future updates or new developments.

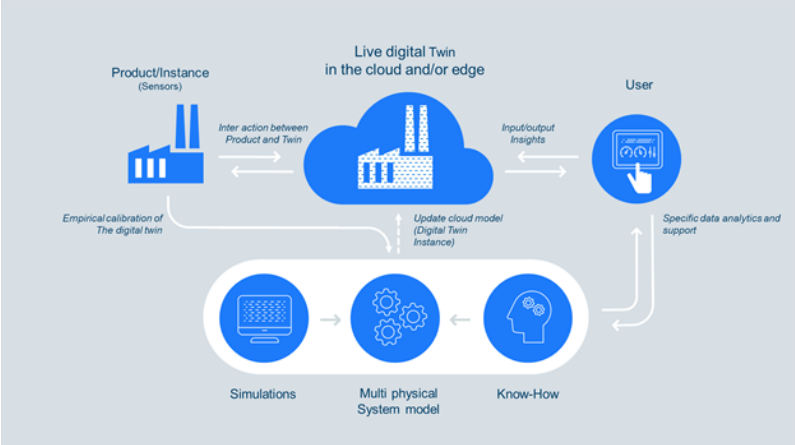


Figure 5.1: Overall scheme of digital twin.

5.1 Digital twin generation

The creation of a digital twin requires several considerations. As previously stated, this model requires bidirectional exchange of data from the real world to the digital software and vice versa, which is not always easy. In fact, everything has to be replicated in the software, so it is necessary to transfer those data over the cloud, where they can be more easily stored, accessed and manipulated by several users for the purposes of managing information and running simulations.

The process to create a digital twin can be broken down into several stages, figure 5.1.

To begin with, selecting the right technology and device is crucial to successfully managing the flow of information; they must be able to authenticate, provision, configure, monitor, and manage each device and information. At the same time, it is important to understand the type of data required, where it is stored, if it is correctly structured and how it can be accessed and used quickly.

The function of a digital twin must also be determined, because the final aim, such as controlling, analyzing, or simulating, affects the previous decisions, as well as the preparation of the data and management requirements [30].

As soon as the acquisition mode and the types of data have been established, it is possible to move forward with the implementation of the digital twin. In this phase more sophisticated capabilities, functionality and boundary conditions are added so that high-quality results can be achieved.
Now that the model is constructed and optimized, it is time to run the simulation and compare the solutions until a satisfactory digital twin is created.

5.2 Digital Twin for Surface topography

According to the previous studies, a further goal of this project was to develop a digital twin model of the measured surfaces, which represents and predicts the characteristics and parameters of the samples, in order to estimate their measurement uncertainty and achieve the other objectives described above. Comsol software and Comsol-Matlab interface were used for the model creation and for the connection with the developed code.
Comsol Multiphysics is a platform for modelling and simulating scientific and engineering problems using advanced numerical methods. It offers an interactive environment with a broad range of tools that can be used to create models and simulate application in electrical, mechanical, fluid flow, and chemical field (see figure 5.2).

5.2.1 Model

It is possible to create randomized geometric surfaces with detailed control of the spatial frequency components that affect the surface's roughness. Technological surfaces show self-affinity because they seem smooth at large observation scales and rougher at lower observation scales. In particular, roughness can be interpreted as a geometrical irregularity that can be modelled by both fractals or har-

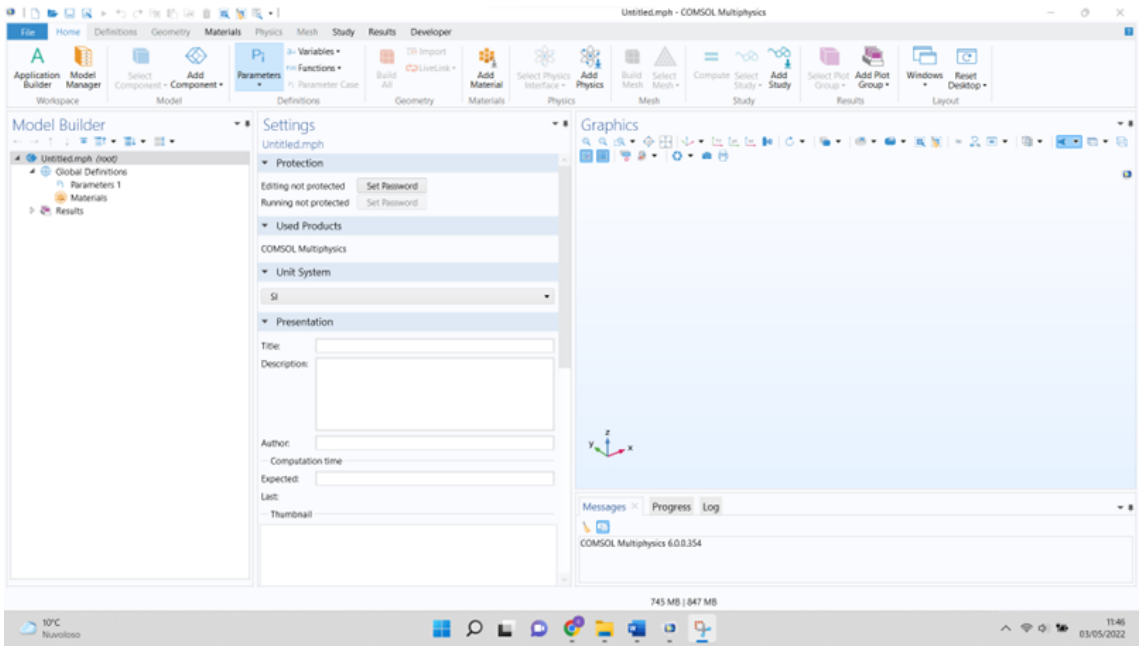


Figure 5.2: Windows of Comsol Multiphysics.

monic functions [31]. Changing the parameters related to the generation scale, different degree of approximation can be obtained.

A few parameters must be defined before the model can be determined.
Firstly, the spatial frequency (eq. 5.1) replaced the frequency over time and corresponds to the oscillation frequency through space; it is usually expressed by wave numbers (eq. 5.2) and it is related to wavelengths (eq. 5.3).

$$cos(2\pi ft) \rightarrow cos(2\pi vx) \tag{5.1}$$

$$k = 2\pi v \tag{5.2}$$

$$\lambda = \frac{1}{v} \tag{5.3}$$

There are a number of elementary waves that form the surface, each represented by the eq. 5.4 where ϕ is the phase angle; in random surfaces, such as this one, latter oscillates between 0 and π to reach

the maximum and the minimum peak of the wave.

$$cos(kx + \phi) \tag{5.4}$$

Furthermore, the amplitude for each elementary wave is represented by A in the eq. 5.5; this is typically a uniform or Gaussian distribution, usually the simplest choice, and it is connected to the spectral exponent β, eq. 5.6, which indicates how quickly high frequencies decay.

$$f(x) = \sum Acos(kx + \phi) \tag{5.5}$$

$$\begin{aligned} A &= \frac{1}{|m^2 + n^2|^\beta} \\ m &= v_x \\ n &= v_y \end{aligned} \tag{5.6}$$

Finally, is necessary to choose the geometry of the model and apply the parameters; all these information enable the creation of the random surface as shown is figure 5.3.

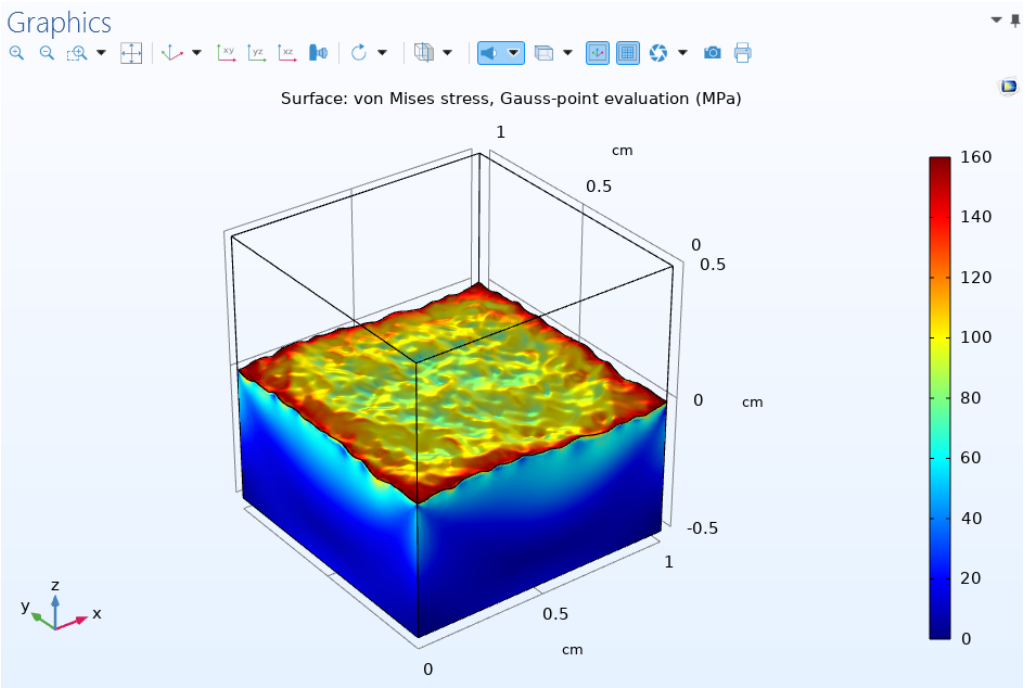
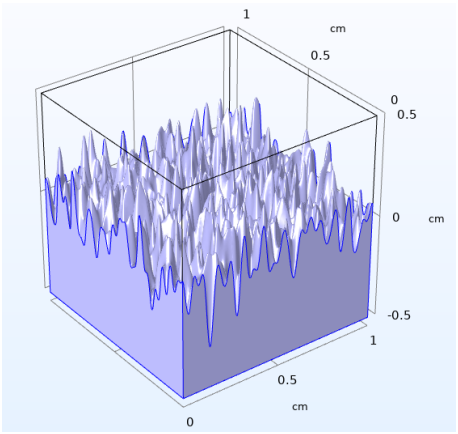


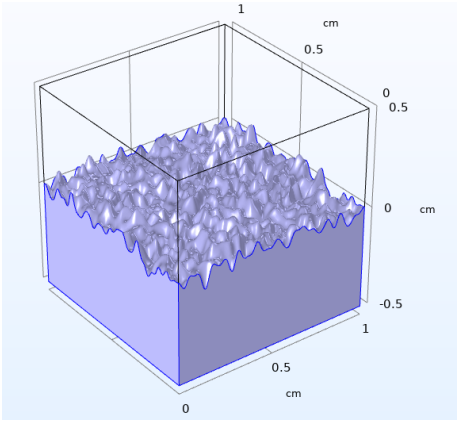
Figure 5.3: Random surface exported by Comsol. [32]

Essentially, the model is built around the function defined by the previous parameters; changing the parameters will change the structure of the function and therefore the information related to the surface. This entails that simply adjusting some parameters value of the model, it can result in different random surfaces generation and representation.

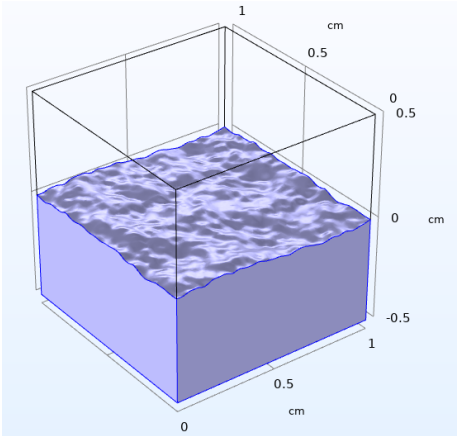
A practical example is provided in figure 5.4 where surfaces have generated on the square [0,1] × [0,1] by superimposing 20 frequency components with amplitude spectral exponents β = 0.5, β = 1.0, and β = 1.8.



(a)



(b)



(c) β=1.8

Figure 5.4: Random surfaces with different β values: a) β=0.5 b) β=1 c) β=1.8.

5.2.2 Configuration Comsol-Matlab

LiveLink is a tool that integrates Comsol Multiphysics with MATLAB to extend its modeling with scripting programming in the Matlab environment. Through it, it is possible to exploit most of the functionality of MATLAB and its toolboxes for pre-processing, modeling, as well as post-processing.

As shown in fig. 5.5, the Comsol server communicates with both

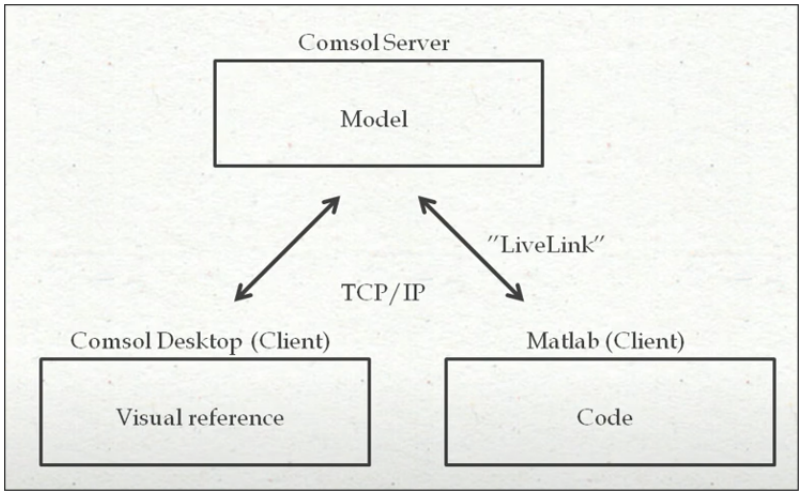


Figure 5.5: LiveLink connection scheme.

software simultaneously and, once the parameters have been defined and the model has been created, the server is called from Matlab using the script 5.1. This code allows establishing the connection and opening the protocol to use (in this case the 2036),fig.5.6, while retaining the original directory where the model script was stored.

The function with the structure is in the main .m file, as in 5.2, with the following sections:

- Parameters description
- Geometry shape
- Features
- Boundary conditions
- Material

```
%% Launch comsol server
currentdir=pwd; %keep the directory of the function
cd('C:\Program Files\COMSOL\COMSOL60\Multiphysics\bin\win64'); %open Comsol
    Multiphysics server with Matlab
system('comsolmphserver.exe &');
cd(currentdir); %return on the directory

%% Establish connection to the server
currentdir=pwd;
cd('C:\Program Files\COMSOL\COMSOL60\Multiphysics\mli');
mphstart(2036);
cd(currentdir);
```

Listing 5.1: Code for configuration between Comsol-Matlab.

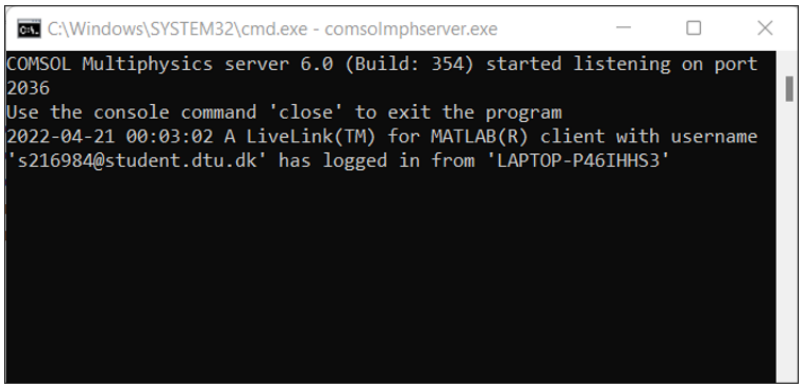


Figure 5.6: Comsol server window.

- Datasets

At this point, it was possible run the program and get the result about the model, fig. 5.7. Similar to the Matlab software developed for

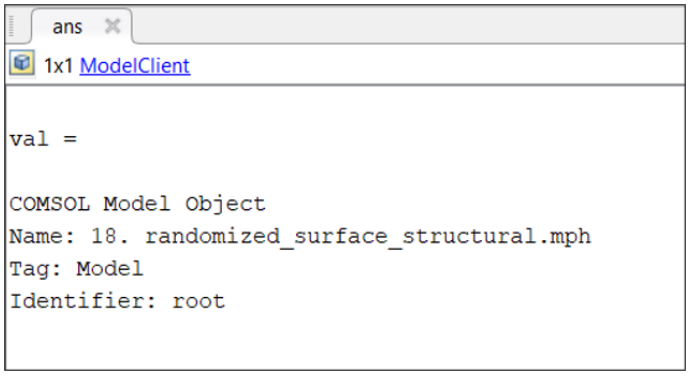


Figure 5.7: Matlab output of Comsol model.

the analysis of samples, in this case, the main limitation was the

```
function out = model
% randomized_surface_structural.m
% Model exported on Apr 20 2022, 20:07 by COMSOL 6.0.0.354.

import com.comsol.model.*           %library
import com.comsol.model.util.*
model = ModelUtil.create('Model');  %name of the model
model.modelPath('C:\CRI\MATLAB-COMSOL');    %the folder
model.comments(['Untitled\n\n']);
model.component.create('comp1', true);
model.component('comp1').geom.create('geom1', 3);   %geometry
model.component('comp1').mesh.create('mesh1');  %mesh
model.param.set('N', '20', 'Spatial frequency resolution');     %physics
...
model.func.create('rn1', 'Random');
...
model.component('comp1').geom('geom1').create('ps1', 'ParametricSurface');
...
model.component('comp1').mesh('mesh1').run;
model.label('randomized_surface.mph');
...
model.component('comp1').material.create('mat1', 'Common');
model.component('comp1').material('mat1').label('Aluminum');
...
model.component('comp1').physics.create('solid', 'SolidMechanics', 'geom1');
...
model.study.create('std1');
model.study('std1').create('stat', 'Stationary');
...
model.sol.create('sol1');
model.sol('sol1').study('std1');
...
model.result.create('pg1', 'PlotGroup3D');
...
out = model;
```

Listing 5.2: Matlab script of imported Comsol model.

computation time. In fact, although LiveLink optimized the process, it still consumed a substantial amount of computational time when compared to a non-LiveLink solution; therefore, the entire running process needed to be moved onto clusters.
As before, the steps were the same: to summarize, it was necessary to create the batch file, run the code on cluster and extrapolate the results, or, in this case, the digital twin model.

6 Results

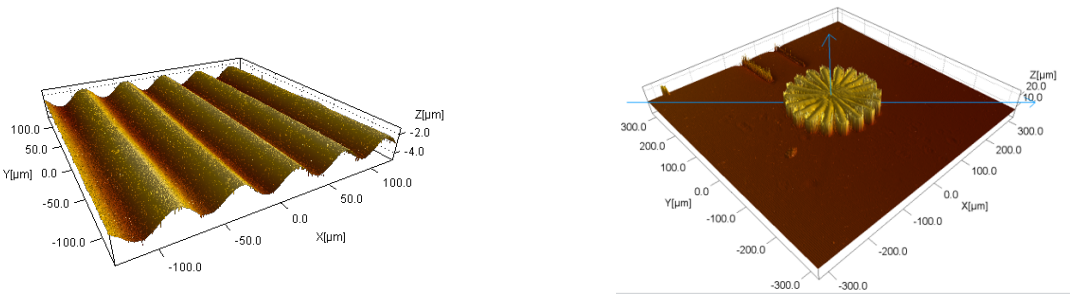
In this chapter, the outputs generated by the two commercial software, Spip and MountainsLab, and Matlab code are analyzed and compared.

Ten acquisitions of 1024 $\times$ 1024 pixels were examined, but some steps and evaluations analyzed only a small portion of them (100 $\times$ 100 pixels) due to time and memory limitation.

6.1 Sofware outputs

According to the discussion in chapter 2, the several types of plane and shape correction methods were considered, and the output from each software can be seen in in fig. 6.1 for Spip, in fig. 6.2 for MountainsLab and in fig. 6.3 for Matlab. The differences between methods are not always substantial, like in these graphs, but to achieve optimum results, more than one plane correction needs to be tried.

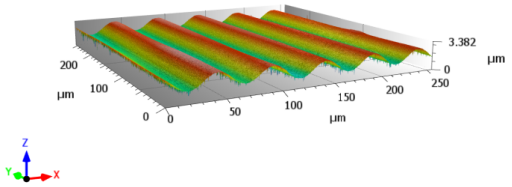
On the contrary, the evidence comes in when there is a slight dif-



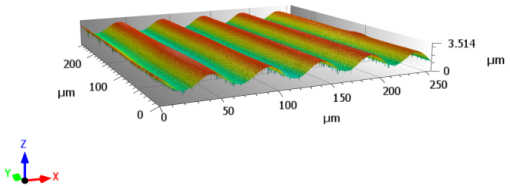
(a) (b)

Figure 6.1: 3D visualization after plane correction in Spip:a) Replicated surface b) ACG sample.

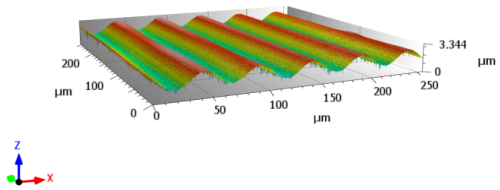
ference among acquisitions; in fig. 6.4 acquisitions 1, 2, 5 and 10 are compared. In fact, when the spikes and voids of the surface are removed as well as when the plane correction is applied, fig. 6.6,



(a)

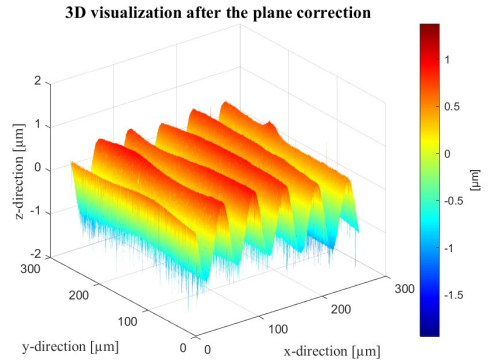


(b) Polynomial degree=5

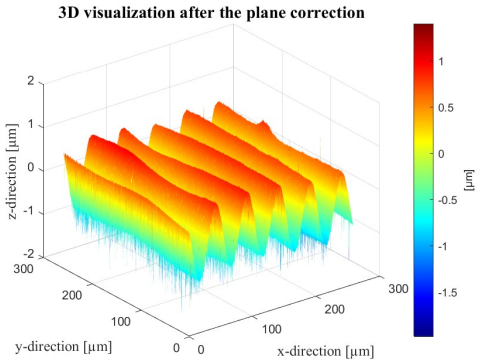


(c) Least square sphere

Figure 6.2: 3D visualization after plane correction in MountainsLab: a) Polynomial degree = 2 b) Polynomial degree = 5 c) Least square sphere.



(a)



(b) surface curvature removal

Figure 6.3: 3D visualization after plane correction in Matlab: a) Surface levelling b) Surface curvature removal

the contrast is more visible, affecting consistently parameter values and results.

In Matlab software, the GPR function detects and correct spikes and voids in different position from one acquisition to another, as in figure 6.5. After that, the plane correction was applied with several results; in fig. 6.6 are shown reduced acquisitions after plane correction, and therefore the full acquisition were as in fig. 6.7.

Due to time and memory limitations, reduced acquisitions were

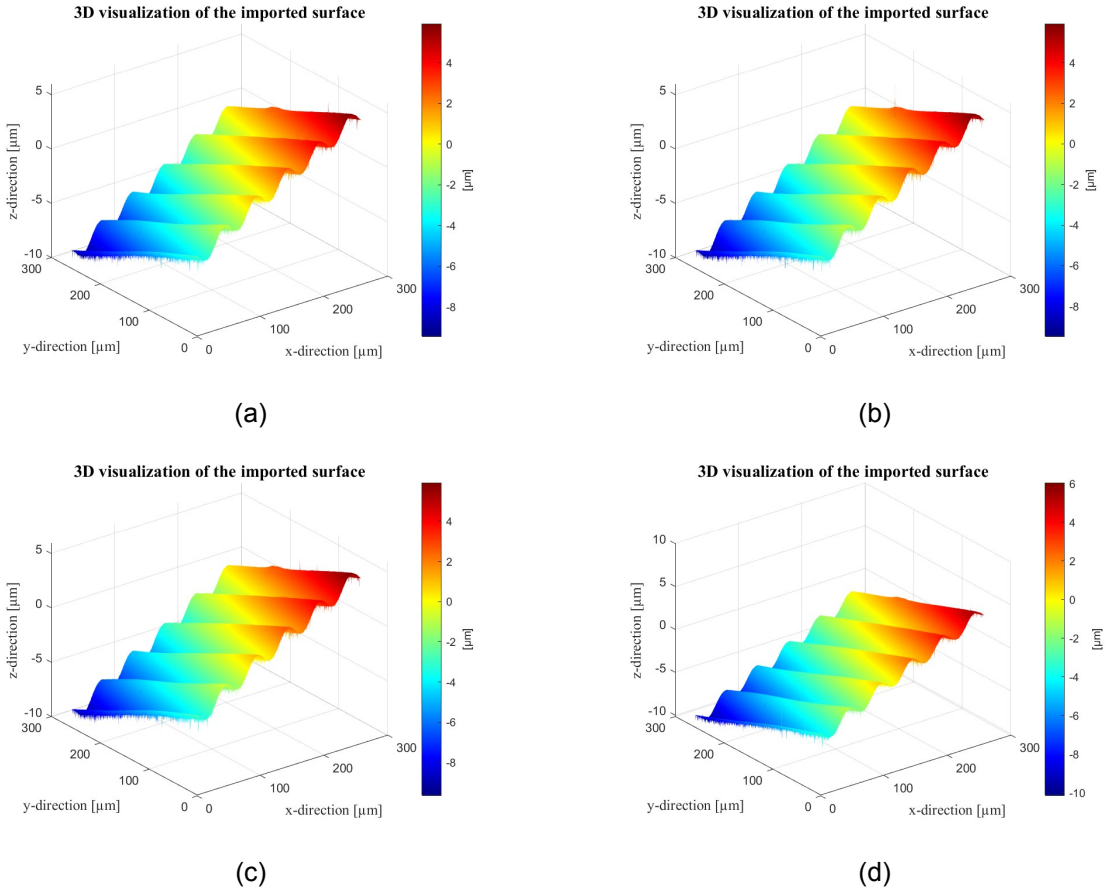


Figure 6.4: 3D visualization raw measurement. Acquisition number: a) 1 b) 2 c) 5 d) 10.

used for testing the code with the GPR correction and, thereby, for detecting and correcting spikes and voids. The plane correction, instead, was shown both on reduced and full acquisitions.

The gathered data were used to evaluate the parameters introduced previously by the application of topographical characterization as introduced in chapter 2. The gathered data regarding the Sq parameter, refer to the sample in fig. 4.2, are collected in tables 6.1.

The results allow a comparison of different plane and shape correction method and characterization software. Hypothesis test based on t-Student distribution [16] was performed to pairwise compare the obtained average results at a confidence level of 95%. In particular, a systematic difference could be highlighted between the parameter evaluated after a global leveling and a global bow removal and all

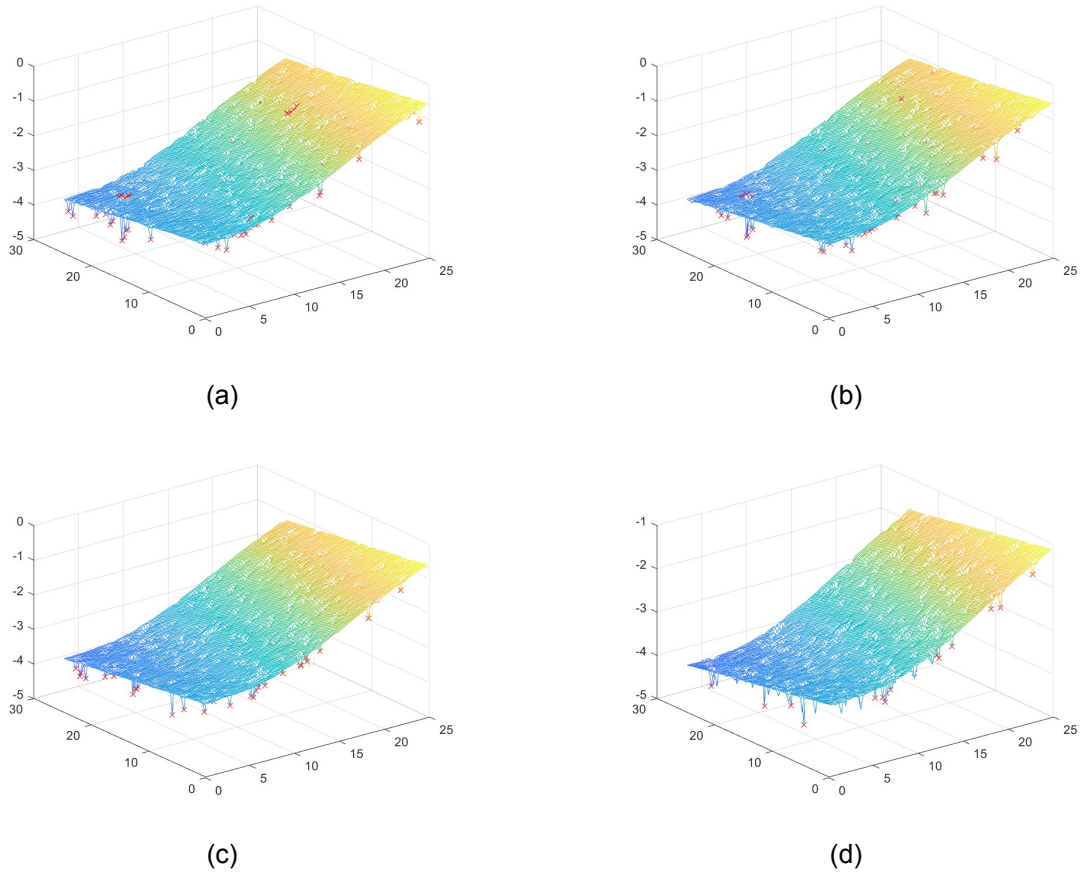


Figure 6.5: 3D visualization with spike and voids detection. Acquisition number: a) 1 b) 2 c) 5 d) 10.

the other alternatives. Moreover, no systematic differences could be highlighted between Mountains Lab, SPIP and MATLAB software. In average, it is possible to see how the values do not differ one from the other; by this logic, the obtained values can be considered reliable.

6.1.1 Uncertainty evaluation

Furthermore, unlike the commercial software, the Matlab code provided the uncertainty evaluation, in figure 6.8, and the covariance and correlation factor graphs in figure 6.9.

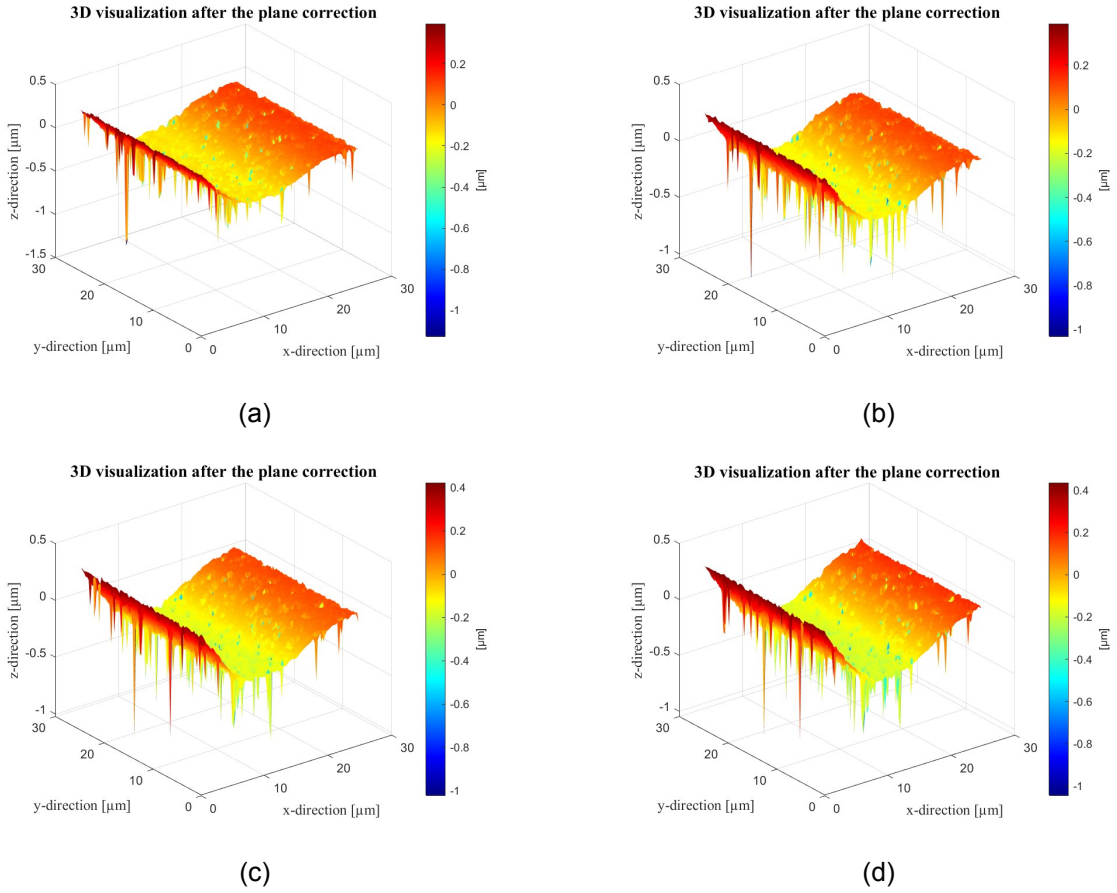


Figure 6.6: 3D visualization after plane correction of reduced acquisitions. Acquisition number: a) 1 b) 2 c) 5 d) 10.

6.2 Clusters

The limitations related to the computing power made it necessary to transfer all the work to the DTU cluster; the tables 6.4 and 6.5 summarise of all the findings related to the time evaluation on both the personal computer and the cluster, and memory requirement (average and maximum values) on cluster. The specification are:

- Personal computer: 6 cores, 2.70GHz, 8GB RAM, 500GB SSD
- DTU clusters: more than 120 computer, 48-912 cores, 2.20-2.90GHz, more than 500GB RAM each, more than 500GB SSD

The study considered simulation both with and without GPR evaluation; in the latter case it was not much convenient to use the clusters

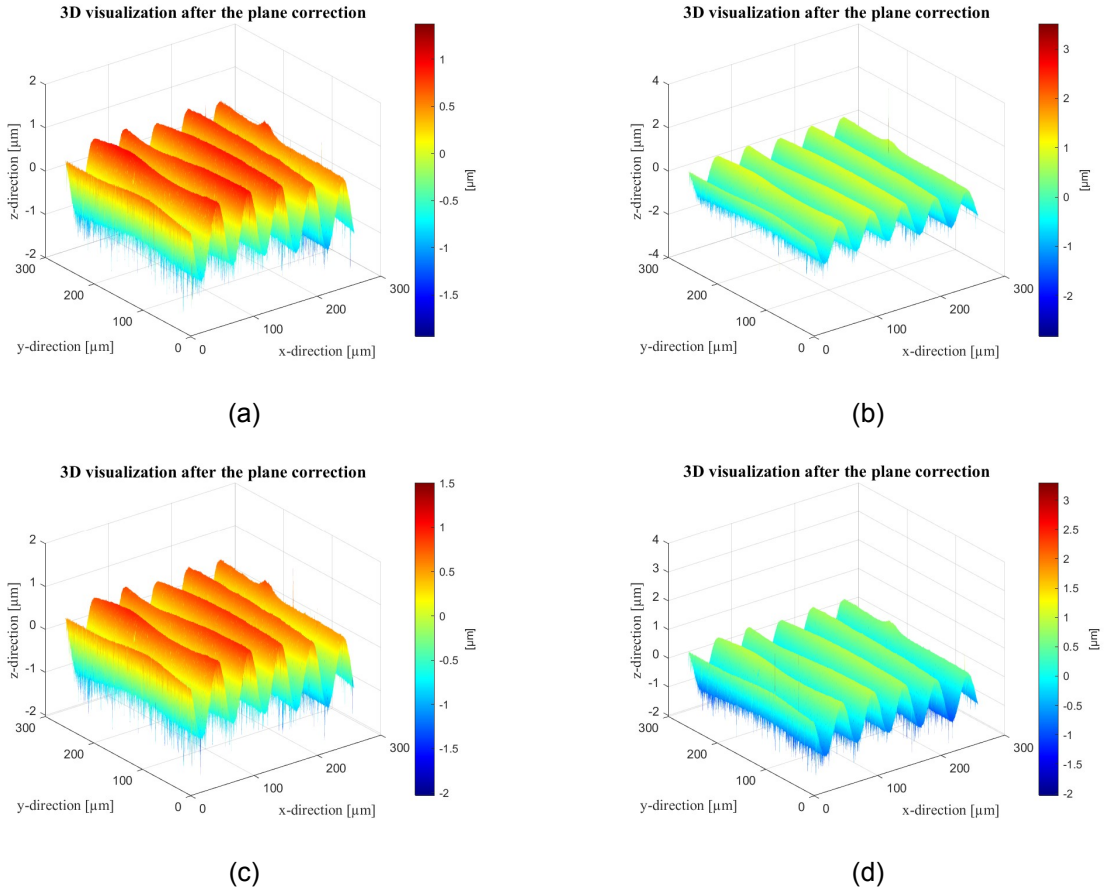


Figure 6.7: 3D visualization after plane correction of full acquisitions. Acquisition number: a) 1 b) 2 c) 5 d) 10.

because the number of data were relatively small and so the computation was faster; however, when GPR was applied, to reduced sample on cluster, the computation time was almost 66% less. On the other hand, when full acquisitions were processed, neither the pc nor the cluster was able to optimize the GPR function and provide results. In fact, the local evaluation was ineffective due to limited memory resource, whilst the cluster could not complete the execution and hence uptput results because the time limit resource for the DTU cluster is set at 24 hours.

To summarize, the matlab software was demonstrated to work properly, to compute surface parameters compatible with state-of-the-art commercial software, and to source 3D visualizations, but no opti-

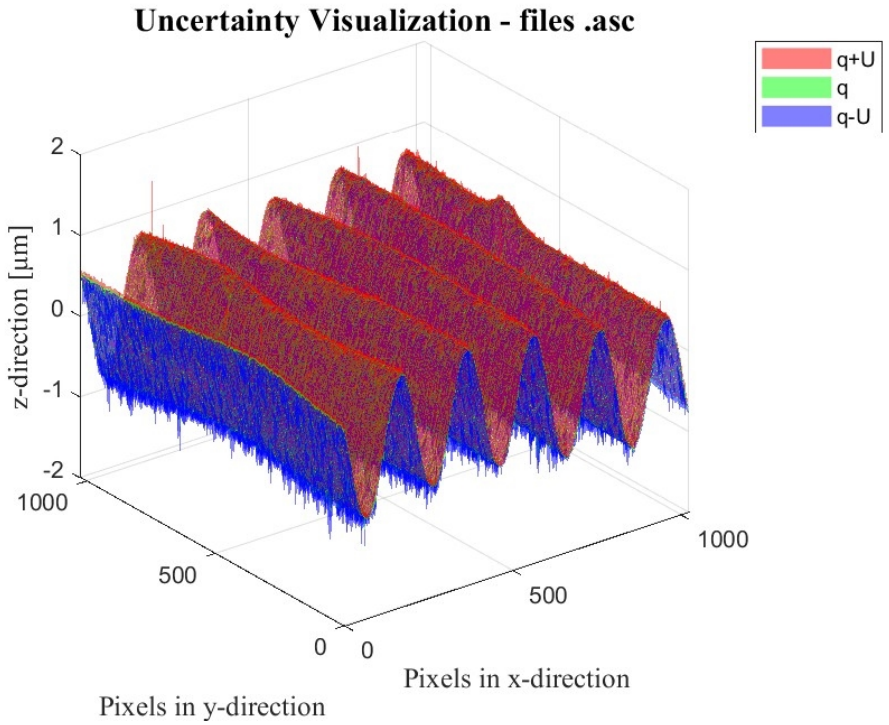
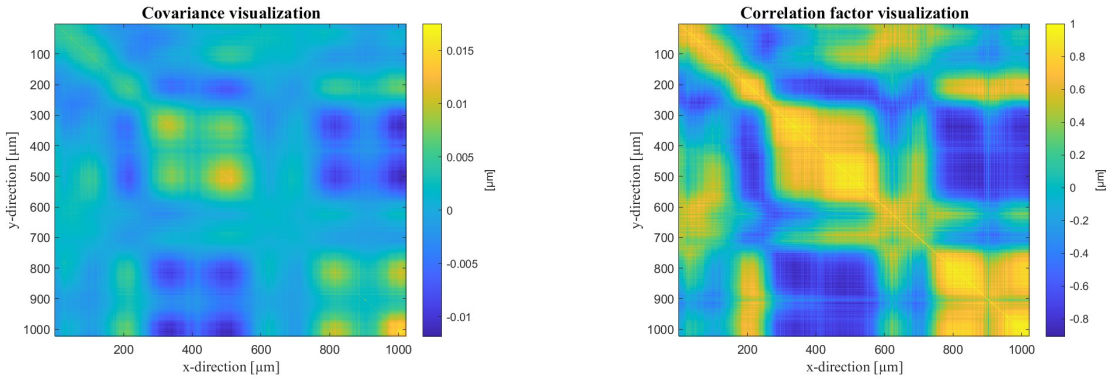


Figure 6.8: 3D visualization of uncertainty evaluation.



(a) (b)

Figure 6.9: Visualization of covariance (a) and correlation factor (b).

mization could be developed for the fitting GPR function despite use of clusters.

A possible solution exploited fitting the GPR model on the first acquisition and exploit it to correct measurement disturbances on all the replicated measured micrographs. However, it could be noticed that the correction of spikes and voids gradually lost their benefits from the first to the last sample, thus suggesting a systematic drift

superimposed to measurement disturbances, i.e. spikes and voids, which cannot be modeled by GPR.

6.3 Digital twin

According to the obtained results, it was crucial to attempt to develop a digital twin model of the measured surface on Comsol.
A random model was imported from the Comsol website [32] which allowed to modify some parameters to better understand their impact on the visualization. A cluster was also necessary in this case due to the large memory and time requirements. In fact, the framework was created on the cluster after the Matlab and Comsol configurations.
The digital twin model was supposed to correspond to a function that should be used after a surface acquisition to model the deterministic surface, and that could be exploited later on for measurement error identification and corrections.

	Sq
1	547,11
2	547,75
3	547,43
4	547,99
5	548,00
6	547,67
7	548,09
8	547,99
9	547,92
10	548,13
Average	547,81
St.dev.	0,33

(a) Global leveling

	Sq
1	545,32
2	545,83
3	545,62
4	546,29
5	546,38
6	546,92
7	546,19
8	546,56
9	546,59
10	546,92
Average	546,26
St.dev.	0,53

(b) Global bow removal

	Sq
1	546,17
2	546,73
3	546,37
4	546,95
5	546,95
6	546,69
7	547,01
8	547,14
9	546,92
10	547,19
Average	546,81
St.dev.	0,33

(c) Linewise leveling

	Sq
1	543,94
2	544,56
3	544,40
4	545,04
5	545,20
6	545,61
7	544,87
8	545,25
9	545,34
10	545,62
Average	544,98
St.dev.	0,54

(d) Linewise bow removal Spip

	Sq
1	547,11
2	547,75
3	547,43
4	547,99
5	548,00
6	548,09
7	547,67
8	547,99
9	547,92
10	548,13
Average	547,80
St.dev.	0,33

(e) Manual plane correction

	Sq
1	547,30
2	547,80
3	547,40
4	548,00
5	548,00
6	547,70
7	548,00
8	547,90
9	547,10
10	547,10
Average	547,81
St.dev.	0,33

Table 6.2: Parameters evaluation in MountainsLab

	Sq
1	546,84
2	547,49
3	547,16
4	547,72
5	547,74
6	547,40
7	547,72
8	547,65
9	547,86
10	547,82
Average	547,54
St.dev.	0,33

Table 6.3: Parameters evaluation in Matlab

		Time computation [sec]	
		without GPR	with GPR
Reduced acquisitions	PC	41,72	1974,56
	DTU cluster	172	672
Full acquisitions	PC	115,92	not supported
	DTU cluster	3041	more than 24h

Table 6.4: Time computation

		Memory [GB]	
		without GPR	with GPR
Reduced acquisitions	Average	15,49	33,72
	Maximum	12,58	22,39
Full acquisitions 5	Average	18,03	not supported
	Maximum	17,31	not supported

Table 6.5: Memory computation

7 Conclusion

Surface topography characterisation is relevant because it provides a better representation of the surface based on qualitative as well as quantitative information, such as shape, size, volume, etc.
The current application requires big data because of time and memory requirements.
This thesis addressed a comparison of current alteranitve software for characteisation showing strenght and weaknesses in regards of current characterization needs. Additionally, this thesis developed and demonstrated a framework for metrological characterisation of big data surface topography measurements based on DTU.

This thesis shows that the commercial software are suitable for the acquisition, elaboration and analysis of data, and parameters evaluation. However, some limitations are evident regarding the spikes and voids identification and correction, plane correction, whilst uncertainty, covariance and correlation factor evaluation cannot be performed at all.
An in-house developed MATLAB code was validated against state-of-the art software. The code allows importing a variety of formats. The code features effective plane correction and enhanced and machine learning-based spikes and voids correction methodology. In fact, commercial software rarely provide spikes and voids detection and correction, or other specific functions, such as uncertainty evaluation or covariance and correlation factor estimation.
Consequently, the software has been developed to address some of these issues and to meet the requirements.
This thesis shows that thorough metrological management and characterization of surface topography measurements is resource inten-

sive and requires critical computational resources to process the related big data. The critical point to consider is the time and memory requirement; in fact, the large number of pixels in the samples require a big amount of memory for analysis and elaboration, thereby forcing a long computation time. Because personal computers have limited memory, to satisfy these requirements it is necessary to use clusters whose processing is much faster and whose computation power is higher.

Despite the fact that the code speeds up the process, some tasks require alternative approaches. In particular spikes and voids detection and correction by GPR model provides poor results because it still requires unavailable time and memory and the kernel modeled on the first acquisition is not general to multiple replicated acquisitions.

This results, which is not trivial, indicates the presence of a systematic error in measurement and different surface modeling approach is required.

Hence, digital twinning of the measured surface is address to create a mathematical representation of the deterministic measured surface which could be updated to describe the measurement error. However, the creation of a surface topography digital twin is unprecedented, and requires first the development of a methodological and computational framework. This thesis work has demonstrated the methodology to create a parametric model describing the measured surface by Comsol. Essentially, the function represents the digital twin model and it is capable of describing and representing several classes of surface by analyzing different samples and changing parameters in the new function.

The integration of the parametric model in MATLAB was implemented and the successful demonstration of the evaluation of the model based on empirical data was achieved by importing the Comsol model

into Matlab to enable configuration with developed code. Because of model and data size, this computation requires the use of a cluster.

The developed framework is a powerful result of the present work that will be exploited in future research to create the digital twin of a measured surface. In conclusion, this study has focused on the creation of an HPC framework that can be expanded in the future with machine learning and digital twin for the metrological management and characterization of the measurement of surface topography by thoroughly exploiting big data.

Bibliography

[1] Hans Nørgaard Hansen et al. “Dimensional micro and nano metrology”. eng. In: *C I R P Annals* 55.2 (2006), pp. 721–743. ISSN: 17260604, 00078506. DOI: 10.1016/j.cirp.2006.10.005.

[2] Danilo Quagliotti. “Modeling the systematic behavior at the micro and nano length scales”. eng. In: *Surface Topography: Metrology and Properties* 10.1 (2022), p. 015011. ISSN: 2051672x. DOI: 10.1088/2051-672X/ac4ba7.

[3] The International Academy for Production Engineering. *CIRP Encyclopedia of Production Engineering*. Ed. by Gunter Reinhart Sami Chatti Luc Laperrièr and Tullio Tolio. Paris, France, 2019. DOI: https://doi.org/10.1007/978-3-662-53120-4.

[4] Mengnan Liu et al. “Review of digital twin about concepts, technologies, and industrial applications”. eng. In: *Journal of Manufacturing Systems* 58 (2021), pp. 346–361. ISSN: 18786642, 02786125. DOI: 10.1016/j.jmsy.2020.06.017.

[5] JCGM 100:2008 (Joint Committee for Guides in Metrology). *Evaluation of measurement data – Guide to the expression of uncertainty in measurement (GUM)*. Ed. by Bureau International des Poids et Mesures (BIPM). Sèvres, France, 2008.

[6] JCGM 200:2012 (Joint Committee for Guides in Metrology). *International vocabulary of metrology – Basic and general concepts and associated terms (VIM)*. Ed. by Bureau International des Poids et Mesures (BIPM). Sèvres, France, 2012.

[7] ISO 14253-2:2011. *Geometrical product specifications (GPS)—Inspection by measurement of workpieces and measuring equipment – Part 2: Guidance for the estimation of uncertainty in GPS measurement, in calibration of measuring equipment and in product verification*. Ed. by International Organization for Standardization (ISO). Geneva, 2011.

[8] ISO 25178-600:2019. *Geometrical product specifications (GPS)—Surface texture: Areal – Part 600: Metrological characteristics for areal topography measuring methods*. Ed. by International Organization for Standardization (ISO). Geneva, 2019.

[9] ISO 21920-1:2021. *Geometrical product specifications (GPS)—Surface texture: Profile – Part 1: Indication of surface texture*. Ed. by International Organization for Standardization (ISO). Geneva, 2021.

[10] ISO 21920-3:2021. *Geometrical product specifications (GPS)—Surface texture: Profile – Part 3: Specification operators*. Ed. by International Organization for Standardization (ISO). Geneva, 2021.

[11] ISO 25178-2:2012. *Geometrical product specification (GPS)—Surface texture: Areal – Part 2: Terms, definitions and surface texture parameters*. Ed. by International Organization for Standardization (ISO). Geneva, 2012.

[12] ISO 25178-3:2012. *Geometrical product specification (GPS)—Surface texture: Areal — Part 3: Specification operators*. Ed. by International Organization for Standardization (ISO). Geneva, 2012.

[13] F. Gao et al. “Surface measurement errors using commercial scanning white light interferometers”. eng. In: *Measurement Science and Technology* 19.1 (2008), p. 015303. ISSN: 13616501, 09570233. DOI: 10.1088/0957-0233/19/1/015303.

[14] Danilo Quagliotti. “Multi Scale Micro and Nano Metrology for Advanced Precision Moulding Technologies”. eng. 2017. ISBN: 8774754807, 9788774754800.

[15] D. Quagliotti, M. Calaon, and G Tosello. “Metrological Quality Assurance in Micro Injection Molding”. In: *Micro Injection Molding*. Ed. by G. Tosello. Munich, Germany: Carl Hanser Verlag GmbH & Co., 2018, pp. 241–288.

[16] Giacomo Maculotti et al. “Gaussian process regression-based detection and correction of disturbances in surface topography measurements”. eng. In: *Quality and Reliability Engineering International* 38.3 (2022), pp. 1501–1518. ISSN: 10991638, 07488017. DOI: 10.1002/qre.2980.

[17] ISO 14406:2010. *Geometrical product specifications (GPS)—Extraction*. Ed. by International Organization for Standardization (ISO). Geneva, 2010.

[18] ISO 25178-70:2014. *Geometrical product specification (GPS)—Surface texture: Areal – Part 70: Material measures*. Ed. by International Organization for Standardization (ISO). Geneva, 2014.

[19] Macarena Mendez Ribo. “Vat photopolymerization process chain”. PhD thesis. PhD Thesis—Technical University of Denmark, 2020.

[20] ISO/ASTM 52900:2021. *Additive manufacturing—General principles – Fundamentals and vocabulary*. Ed. by International Organization for Standardization (ISO). Geneva, 2021.

[21] Hans Nørgaard Hansen, R.J. Hocken, and Guido Tosello. “Replication of micro and nano surface geometries”. eng. In: *C I R P Annals* 60.2 (2011), pp. 695–714. ISSN: 17260604, 00078506. DOI: 10.1016/j.cirp.2011.05.008.

[22] Robert H. Goodall, Laurent P. Darras, and Mark A. Purnell. “Accuracy and precision of silicon based impression media for quantitative areal texture analysis”. eng. In: *Scientific Reports* 5.1 (2015), p. 10800. ISSN: 20452322. DOI: 10.1038/srep10800.

[23] Rubert & Co Ltd. URL: www.rubert.co.uk/reference-specimens.

[24] AccuTrans. URL: www.accutransusa.com.

[25] Matlab® documentation. URL: https://se.mathworks.com/help/matlab.

[26] MathWork®. URL: https://se.mathworks.com/.

[27] SPIP. URL: https://www.imagemet.com/products_/spip/.

[28] MountainsLab®. URL: https://www.digitalsurf.com/software-solutions/multi-instrument/.

[29] Matteo Gilardi. “Statistical processing and analysis of surface topography data for a machine-based evaluation of the measurement uncertainty, Statistisk beregninger

og analyse af overfladetopografi data til en maskinbaseret evaluering af måleusikkerheden". und. 2020.

[30] David Jones et al. "Characterising the Digital Twin: A systematic literature review". eng. In: *Cirp Journal of Manufacturing Science and Technology* 29 (2020), pp. 36–52. ISSN: 18780016, 17555817. DOI: 10.1016/j.cirpj.2020.02.002.

[31] Christopher A. Brown et al. "Multiscale analyses and characterizations of surface topographies". In: *CIRP Annals* 67.2 (2018), pp. 839–862. ISSN: 0007-8506. DOI: https://doi.org/10.1016/j.cirp.2018.06.001. URL: https://www.sciencedirect.com/science/article/pii/S0007850618301586.

[32] Comsol®. URL: www.comsol.com.

[33] Richard Leach. *Optical measurement of surface topography*. eng. Ed. by Richard Leach. Springer, 2011, XIII, 323 S. (unknown). ISBN: 3642120113, 3642120121, 3642426840, 9783642120114, 9783642120121, 9783642426841.

[34] Surjya Kanta Pal et al. "Digital Twin Application". eng. In: *Digital Twin – Fundamental Concepts To Applications in Advanced Manufacturing* (2022), pp. 413–465. DOI: 10.1007/978-3-030-81815-9_7.

[35] Nikolaos Ploskas and Nikolaos Samaras. *GPU Programming in MATLAB*. eng. Elsevier Inc., 2016, pp. 1–302. ISBN: 0128051329, 0128051337, 9780128051320, 9780128051337.

[36] DTU Computing Center. *DTU Computing Center resources*. 2021. DOI: 10.48714/DTU.HPC.0001. URL: https://doi.org/10.48714/DTU.HPC.0001.

[37] David J. Whitehouse. *Handbook of surface and nanometrology*. eng. CRC Press, 2011, XXIII, 975 S. (unknown). ISBN: 042914069x, 1322614709, 1420082019, 1420082027, 9780429140693, 9781322614700, 9781420082012, 9781420082029.

[38] *Statistical methods in process management – Capability and performance – General principles and concepts*. eng. 2014.

[39] 22514-4:2016. *Statistical methods in process management – Capability and performance – Part 4: Process capability estimates and performance measures*. eng. 2016.

[40] Jinlong Li, Yue Yu, and Xiaorong Gao. "Reference plane correction of online grating projection-based 3D measurement". eng. In: *Optik* 127.2 (2016), pp. 608–612. ISSN: 16181336, 00304026. DOI: 10.1016/j.ijleo.2015.10.011.

[41] Daniel Teixidor, Guillem Quintana, and Joaquim de Ciurana. "Experimental introduction to surface roughness parameters measurement". eng. In: *Materials Science Forum* 759 (2013), pp. 63–71. ISSN: 16629752, 02555476. DOI: 10.4028/www.scientific.net/MSF.759.63.

[42] Azad M. Madni, Carla C. Madni, and Scott D. Lucero. "Leveraging digital twin technology in model-based systems engineering". eng. In: *Systems* 7.1 (2019), p. 7. ISSN: 20798954. DOI: 10.3390/systems7010007.

[43] Lilan Liu et al. “Digital twin-driven surface roughness prediction and process parameter adaptive optimization”. eng. In: *Advanced Engineering Informatics* 51 (2022), p. 101470. ISSN: 18735320, 14740346. DOI: 10.1016/j.aei.2021.101470.

[44] Flavia Pires et al. “Digital twin in industry 4.0: Technologies, applications and challenges”. eng. In: *Ieee International Conference on Industrial Informatics (indin)* 2019-(2019), pp. 721–726. ISSN: 19354576, 2378363x. DOI: 10.1109/INDIN41052.2019.8972134.

[45] Jingxiao Wang et al. “Complex 3D geological modeling based on digital twin”. eng. In: *Iop Conference Series: Earth and Environmental Science* 861.7 (2021), p. 072046. ISSN: 17551315, 17551307. DOI: 10.1088/1755-1315/861/7/072046.

[46] Robin Willink and Rod White. “- 1-Disentangling Classical and Bayesian Approaches to Uncertainty Analysis”. In: 2012.

[47] ISO 21920-2:2021. *Geometrical product specifications (GPS)—Surface texture: Profile – Part 2: Terms, definitions and surface texture parameters*. Ed. by International Organization for Standardization (ISO). Geneva, 2021.

[48] ISO 16269-4:2010. *Statistical interpretation of data – Part 4: Detection and treatment of outliers*. Ed. by International Organization for Standardization (ISO). Geneva, 2010.

[49] Henry T. Lancashire. “A simulated comparison between profile and areal surface parameters: R_a as an estimate of S_a”. und. In: (2017), p. 9.

[50] Baofeng He, Siyuan Ding, and Zhaoyao Shi. “A comparison between profile and areal surface roughness parameters”. eng. In: *Metrology and Measurement Systems* 28.3 (2021), pp. 413–438. ISSN: 23001941, 20809050, 08608229. DOI: 10.24425/mms.2021.137133.

List of Figures

List of Tables