CRANFIELD UNIVERSITY

PATRICIA GIMÉNEZ BELANDO

SOFTWARE DEVELOPMENT FOR USER FRIENDLY CALIBRATION PROCEDURES

SATM
Aerospace Manufacturing

MSc
Academic Year: 2016 - 2017

Supervisor: Claudiu L Giusca
September 2017

ABSTRACT

The wide spread of areal surface topography instruments in industry requires standard calibration procedures. Current ISO calibration standardisation efforts are in an incipient stage and calibration routines are being proposed by national metrology institutions such as the National Physical Laboratory (NPL). These calibration routines estimate the magnitude of a set of limited number of parameters called metrological characteristics, which are currently defined in the ISO documents. The aim of this project is to allow industrial users to calibrate of their own instruments based on current practices.

Recently, NPL manufactured a new ISO compliant calibration artefact and developed an associated set of calibration routines. Providing a user-friendly software platform that is able to calculate the magnitude of the metrological characteristics and provide basic uncertainty estimates, NPL can take the opportunity to expand in the market of this new product. To increase the product visibility, the calibration procedures will also be integrated in MountainsMap, which is the market leader software platform in the field of surface texture analysis.

Keywords:

Uncertainties, metrological characteristics, surface topography

ACKNOWLEDGEMENTS

I would like to thank Dr Claudiu L. Giusca for his expert advice and encouragement throughout this project. I am very grateful for his support and guidance.

I would like to show my gratitude to my home university, Universitat Politècnica de València, for providing me the opportunity to undertake this masters in a double degree program.

Moreover, it is a pleasure to thank those who made my days in Cranfield, including Anna, Paola, María, Estrella, Carol and Helen. I am also grateful to the boys, Ettore, Bernardo, Jan, Marc, Juan, Quentin and Lloyd, for making my evenings more enjoyable.

I would like to thank Ivan for his patience and relief. Finally, I would like to thank my family for their economic and emotional support. I appreciate their pilgrimage to San Nicolás church despite their lack of faith, including my grandma, who did it mentally.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

3D	3-Dimensional
ACG	Areal Cross-Grating
CLSM	Confocal Laser Scanning Microscopy
CP	Complexity of Program
CU	Criticality of Use
ISO	International Organization for Standardization
IT	Information Technology
NPL	National Physical Laboratory
SIL	Software Integrity Level
UKAS	United Kingdom Accreditation System

1 INTRODUCTION

1.1 Introduction

Areal surface topography measuring instruments provide 3D information about the surface (texture and form), or data from which the surface texture parameters can be calculated. Surface texture characterisation allows to predict the interaction of the surface and its surrounding and is of great importance in the functionality of manufactured components. It is estimated that 10% of the failures on components are caused by texture effects [1].

These instruments need to be calibrated. Calibration is the comparison of the quantity values resulting from a measurement with values proceeding from a calibration standard of known accuracy. Traceability of the calibration standard is required and these are usually traceable to a national standard held by a National Metrological Institute. The traceability is a property of a measurement result where the result is related to a reference through a documented unbroken chain of calibrations, each contributing to the measurement uncertainty [2]. Measurement uncertainties represent the ambiguity in the result of a measurement introduced by the fact of performing such measurement. The estimation of measurement uncertainties can be mathematically modelled. The inputs of such model are the influence factor, each one contributing to the measurement uncertainty. However, isolating the effect of each influence factor is complicated. Instead, the uncertainties can be estimated by a simple input-output measurement model based on a limited amount of inputs, called metrological characteristics, described in ISO 25178-7, which incorporate the effect of the influence factor and which can be measured with the aid of a calibration artefact. These artefacts contain specific features over which perform the measurements required to estimate the metrological characteristics.

1.2 Aim and objectives

The aim of this project is to develop software to calculate the uncertainties of surface topography measuring instruments by using the new NPL calibration artefact. The calibration routines developed by the UK's National Measurement

Institute, associated with its new artefact and in line with the new areal ISO standards, are to be implemented. These calibration procedures present processing data complexities and knowledge on the field is required. The target here is to allow unskilled industrial users to do their own calibration routines by means of a user-friendly software. They would only need to input the required topographies and the software would process the data and present the results.

1.3 Motivation and scope

The project is part of a marketing strategy for NPL to expand its new product in the market, as the calibration artefact is needed. On their side, industrial users will gain independency, being able to calibrate their instruments on their own.

The project is divided into work packages consisting in: measurement noise, flatness deviation, step heights and grids. Every package corresponds to one calibration routine, which is implemented in MATLAB® as described in section 4.

1.4 Thesis layout

The report is divided in six sections. After the introduction, a literature review covers the areal surface topography measuring instruments and the measurement uncertainties. Next the research questions, hypothesis and methodology are exposed. The fourth section presents the software implementation of the calibration routines. After it, the results are presented and discussed. Finally, the most relevant information is summarised in conclusions.

2 LITERATURE REVIEW

2.1 Areal surface topography measuring instruments

In the past, the main methodology to measure surface topographies was based on contact stylus instruments. Contact stylus instruments have been used since 1927 and contributed considerably to the control of manufacturing process. However, optical instruments are gaining popularity in the surface topography field thanks to their non-contact and faster measurements.

Areal instruments are surface measuring instruments that perform 3D measurements of surface topographies (Figure 1). On the other hand, profile instruments make 2 dimensional measurements. To reconstruct a 3D surface, scanning of the sample is needed.

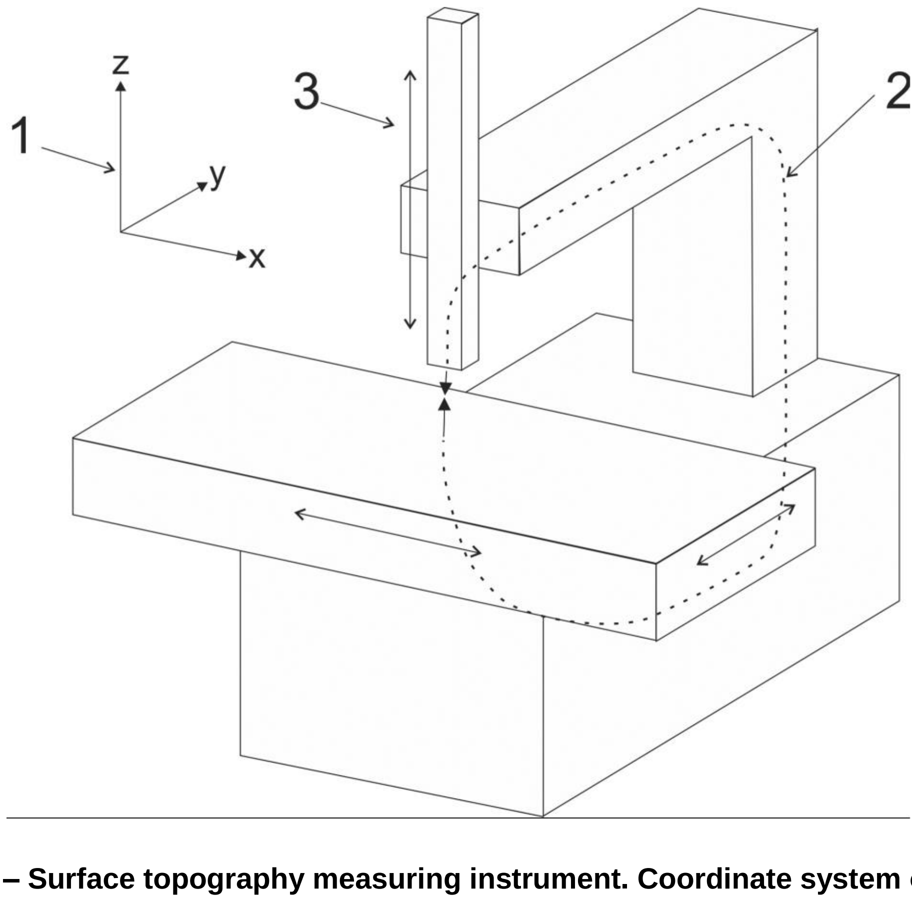

Figure 1 – Surface topography measuring instrument. Coordinate system of the instrument (1), measurement loop (2) and *z*-scan axis (3) (ISO 25178-600)

The optical methods for measuring surface topography listed in the ISO 2178-6 document are:

- Confocal Microscopy
- Focus Variation Microscopy
- Coherence Scanning Interferometry
- Phase Shifting Interferometry
- Chromatic Probe
- Structured Light and Triangulation
- SEM Stereoscopy
- Scanning Tunnelling Microscopy
- Atomic Force Microscopy
- Optical Differential Profiling
- Angle Resolved SEM

During this thesis, Coherence Scanning Interferometry has been used to test the implementation of the calibration routines. This method will be briefly introduced.

Coherence Scanning Interferometry:

Interferometers appeared in the late 19th century, invented by Albert Michelson. These instruments are based on a phenomenon called interference. This phenomenon occurs when waves with same frequency are combined, resulting in the addition of their amplitudes (superposition). The resulting wave presents its maximum when the signals are in phase.

Interferometers combine at least two beams of light coming from a single source to create an interference pattern that can be analysed. The interference pattern contains information about the object being studied. A sketch of a Mirau interferometer is shown in Figure 2. A light beam is emitted from the microscope objective. It arrives to the beam splitter, where it is divided in two. One is directed to the sample and the other is directed to a reference surface. The two beams are reflected and collected at the photodetector for analysis.

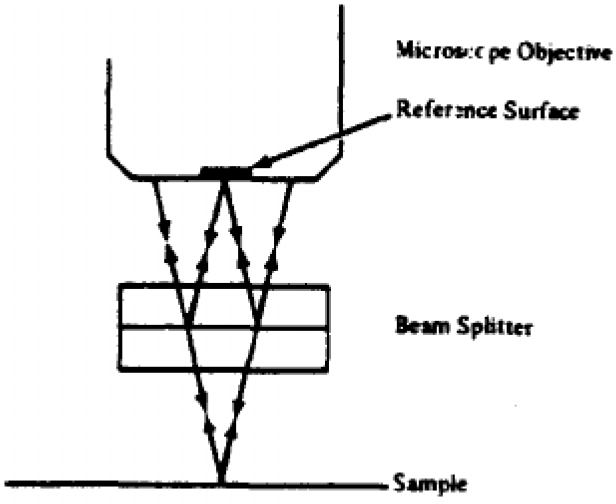

Figure 2 - Mirau interferometer configuration [3]

Figure 3 shows an interference signal over a profile. The maximum occurs when the two signals resulting after splitting the main beam are in phase. That happens when the distance to the photodetector from the reference surface and from the sample are equal. Then, the interference presents a maximum. By scanning in the vertical axes, the maximums that delineate the profile are obtained.

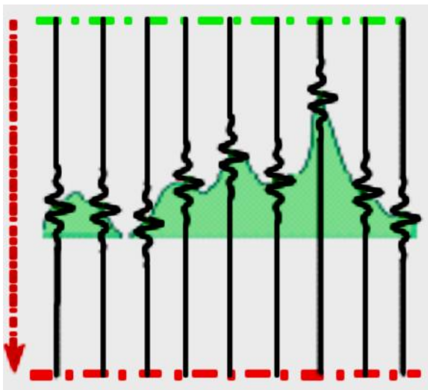

Figure 3 - Interference signal along a profile

2.2 Measurement uncertainties

Measurement uncertainty is the ambiguity that measurements present. It "is the doubt that exists about the result of any measurement" [4]. The parameters that characterise measurement uncertainties are the interval (Figure 4, in light green), the confidence level, which determines the probability of the true value relying in that interval and the coverage factor, *k*, used to re-scale the confidence level (Table 1).

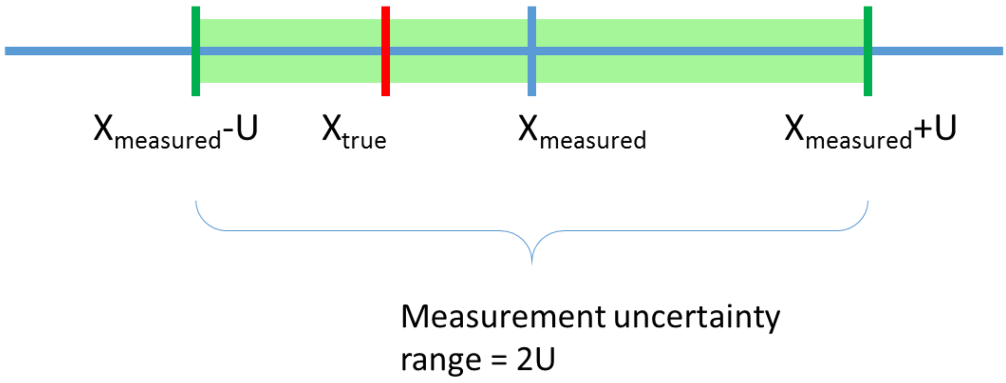

Figure 4 - Scheme of a measurement result, with the measured value on the centre of the interval defined by the uncertainty, and the true value relying on it.

Table 1 - Coverage factors and their associated confidence level

k	**Confidence level**
1	68%
2	95%
2.58	99%
3	99.7%

Thus, the way to properly express a measurement with its uncertainty so as the reader can understand and use its information may contain the measurement result with the uncertainty interval, the statement of the coverage factor, the confidence level and how the uncertainty was estimated. The "Good Practice Guide No. 11" provides de following example:

The length of the stick was 20 cm ±1 cm. The reported uncertainty is based on a standard uncertainty multiplied by a coverage factor k = 2, providing a level of confidence of approximately 95%. The uncertainty was estimated following the procedure explained in UKAS Publication M 3003.

There are many sources of uncertainties that include the measuring instrument, operator, work piece, measurement process, imported uncertainties, operator skills, sampling issues or the environment. However, in the case of surface topography, modelling each influence factor contribution is non-feasible and unnecessary for industrial applications. Measurement uncertainties can be calculated in an easier way through a set of limited number of the metrological characteristics (Table 2). A series of calibration routines which estimate their magnitude are defined in the ISO documents (ISO/CD 25178 part 600). This project does not cover resolution.

Table 2 - Metrological characteristics

Metrological characteristics	Error along
Measurement noise	x, y, z
Flatness deviation	x, y, z
Amplification	z
Linearity	z
Perpendicularity	z
Resolution	x, y, z

2.2.1 Measurement noise (N_M)

The measurement topographies are affected by the noise generated while taking the measurements. N_M is affected by different sources of noise, such as:

- Instrument internal noise (instability in the instrument electronics)
- Environmental noise (ground vibrations, ventilation, sound, temperature fluctuations or external electromagnetic disturbances)
- Drive units' noise

In order to estimate the measurement noise, generally a flat artefact with smooth surface and a maximum height of the scale limited surface (*Sz*) of 30 nm is used [5]. However, any surface could be used. No filter operations are required.

N_M needs to be isolated from the intrinsic roughness and flatness deviation of the sample. The recommended technique is subtraction, where *Sq* (2-1) is a 3D parameter expressing the Root Mean Square Roughness of the resulting topography. Because all the values are squared, it does not differentiate between peaks and valleys.

$$Sq = \sqrt{\frac{1}{n}(z_1^2 + z_2^2 + \cdots + z_n^2)} \qquad \textbf{(2-1)}$$

The subtraction technique requires two different measurements at the same position on the work piece taken in quick succession. The resulting two measurements will include the same region of the sample (ε_{sample}) and the flatness deviation of the instrument (ε_{FLT}), and the contribution of the measurement noise (δ_{NM}). By subtracting one measurement from another, the result will contain only the effect of the measurement noise, because the sample topography and flatness deviation are cancelled (2-2).

$$\begin{array}{rl} & z_1 = \delta_{NM} + \varepsilon_{sample} + \varepsilon_{FLT} \\ - & \\ & z_2 = \delta_{NM} + \varepsilon_{sample} + \varepsilon_{FLT} \\ \hline & z_{avg} = 2\delta_{NM} \end{array} \qquad \textbf{(2-2)}$$

Assuming that the measurement noise follows a normal distribution centred in the origin, the *Sq* calculated from the subtraction results will be equal to the standard deviation of two combined normal distributions. The rule of variance (squared standard deviation) combination for normal distributions is shown in equation (2-3), where σ_x and σ_y correspond to two different variances, and σ_{x+y} is the variance resultant from the combination of the other two. As by subtraction two measurement noises are combined, the previous equation can be simplified to (2-4) and (2-5) can be obtained.

$$\sigma_{x+y}^2 = \sigma_x^2 + \sigma_y^2 \qquad \textbf{(2-3)}$$

$$\sigma_{x+x}^2 = 2\sigma_x^2 \qquad \textbf{(2-4)}$$

$$N_M = \frac{Sq}{\sqrt{2}} \qquad \textbf{(2-5)}$$

The operation should be repeated at least three times to ensure no large variability in the results [6]. For every subtraction, N_{Mi} is calculated, and a final value of measurement noise is obtained by calculating root mean square (2-6). The result should be the same as the standard deviation of all the set of values obtained in every subtraction. Measurement noise uncertainty is thus propagated as a normal distribution centred in the origin and with a standard deviation equal to N_M

$$N_M = \sqrt{\frac{1}{n}(N_{M1}^2 + N_{M2}^2 + \cdots + N_{Mn}^2)} \tag{2-6}$$

There is another technique that averages many measured topographies so that the influence of the measurement noise is decreased by the square root of the number of measurements. See reference [7] for more detail.

2.2.2 Flatness deviation (Z_{flt})

Flatness deviation characterises the quality of the areal reference of the instrument, against which the surface topographies will be measured. The surface texture parameter used to quantify it is *Sz*, which is the maximum height of the scale limited surface, i.e., the difference between the highest peak and the deeper valley (Figure 5).

The calibration artefact employed to estimate the value of this metrological characteristic is a flat like the one described in the previous section. The analysis is performed over an averaged topography. Averaging decreases the influence of spurious data caused by undesirable factors and reduces de influence of measurement noise and sample topography. These measurements are to be performed by only moving the sample in *x*-and/or *y*-direction.

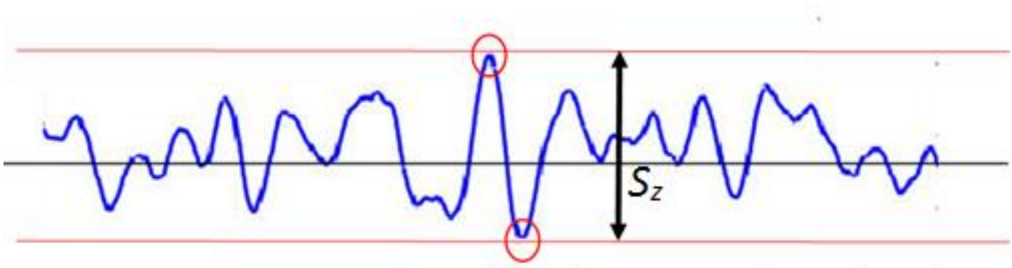

Figure 5 - Sketch of an areal reference surface profile and the parameter S_z

The number of measurements required to define Z_{flt} is not clearly stablished. In general terms, it can be affirmed that the necessary number is this such that the value of S_z stabilises, not varying more than 5%. However, spurious data can suddenly increase the average values of peaks and valleys and hence, a big amount of data may be required. To mitigate this problem, a threshold operation can be applied.

Spurious data are generally high spatial frequency components. The errors introduced by the instruments, as these caused by the lenses, are normally low spatial frequencies. To analyse the parameter Z_{flt}, these low frequencies should be neglected. The spurious data can be eliminated by fitting a polynomial and subtracting it, or by using a high pass Gaussian filter. The implementation of this method is described in detail in section 4.2.

The flatness contribution to measurement uncertainties is propagated as a rectangular distribution, R($-Z_{flt}/2$, $Z_{flt}/2$), with a variance equal to $Z_{flt}^2/12$

2.2.3 Amplification, linearity and perpendicularity

Amplification and linearity establishes how well the instrument response curve fits the ideal response curve for the three of the scales, *x*, *y* and *z*. The linearity is defined by the maximum distance between the measured curve of the instrument and its linear regression curve, whose slope is the amplification coefficient (Figure 6). This coefficient can be numerically calculated as equation (2-7):

$$\alpha = \frac{\sum_i^n C_i I_i}{\sum_i^n C_i^2}, \quad \textbf{(2-7)}$$

where C_i are the calibrated values, I_i are the indicated values, n indicates the number of step height artefacts used.

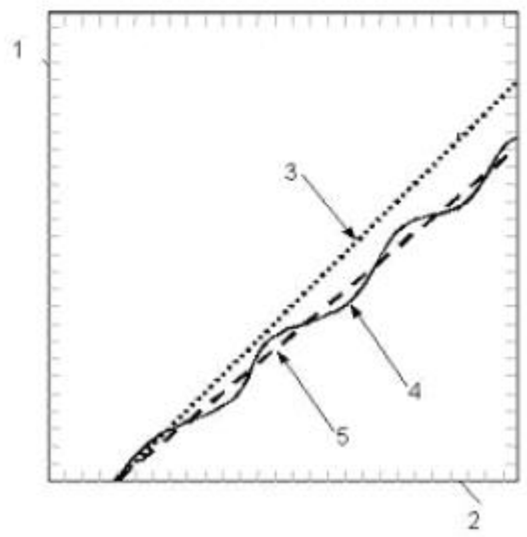

Figure 6 Instrument response curve: 1 - measured quantities; 2 - input quantities; 3 - ideal response curve; 4 - instrument response curve; 5 - linear curve (from ISO 25178-601 2010)

Repeatability contribution to the measurement uncertainty is defined as the standard deviation of a set of measurements taken under the same conditions and in a short period of time. Reproducibility contribution is the standard deviation of a set of measurements taken in different conditions, as for example measuring at different positions in the vertical scale. According to International Vocabulary of Metrology (IVM), measurement traceability is a “property of a measurement result whereby the result can be related to a reference through a documented unbroken chain of calibrations, each contributing to the measurement uncertainty.” Error contribution to the uncertainties can be estimated as the difference between the measured value and the calibrated value.

The overall scale contribution to the measurement uncertainty can be calculated as a combination of the repeatability and/or reproducibility, measurement errors and traceability. If it is assumed that these contributions are not correlated, a linear addition model can be used to obtain the overall uncertainty associated with the point coordinate measurement for each of the three axes (2-8).

$$u = \sqrt{u_{repeat}^2 + u_{reprod}^2 + u_{err}^2 + u_{trac}^2} \quad \textbf{(2-8)}$$

The error contribution is propagated as a rectangular distribution centred in zero, with $-\delta_{err}$ as a minimum value and δ_{err} as a maximum and a variance equal to δ_{err}^2 /3, R($-\delta_{err}$, δ_{err}). On the other hand, both repeatability and reproducibility present a normal distribution centred in the origin with a variance equal to the square of their value, N(0, δ^2_{repeat}) and N(0, δ^2_{reprod}).

Calibration of *z*-axis scale

Although there is no standardised procedure for step height areal measurement, the standardised profile procedure can be applied by extracting profiles from the surface. The ISO 5436-1 suggests an average of 5 measurements. However, a much higher amount of profiles can be extracted and analysed from an areal surface topography.

The above mentioned standardised method corresponds to Type A1 grooves: "These measurement standards have a wide calibrated groove with a flat bottom, a ridge with a flat top, or a number of such separated features of equal or increasing depth or height. Each feature is wide enough to be insensitive to the shape or condition of the stylus tip" (ISO 5436-1, section 5.2.1). Such grooves are characterised by their width, *W*, and their depth, *d*. Over a profile of width 3*W*, regions A, B and C are determined as shown in Figure 7.

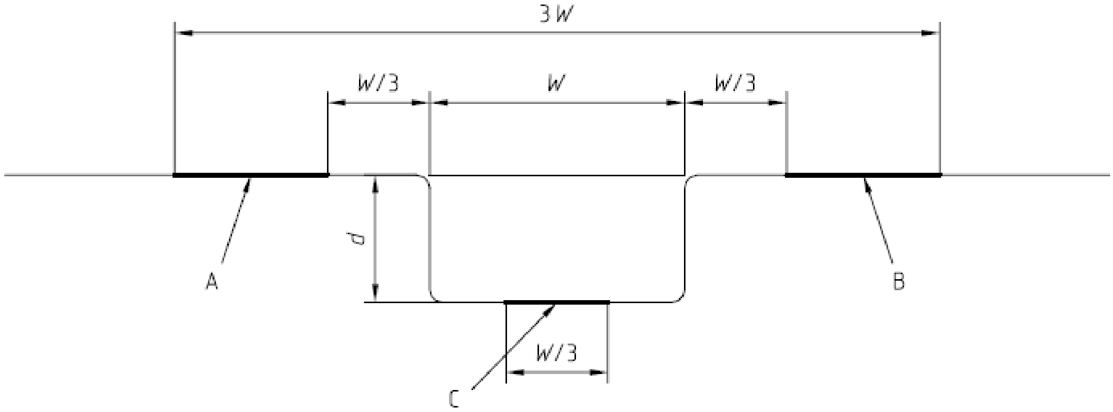

Figure 7 - Type A1 groove profile for depth calibration

The procedure consists in fitting equation (2-9) by the method of the last squares to the regions of the average profile, where α, β and δ are unknown coefficients and where δ takes the value +1 in regions A and B and the value -1 in region C. The depth of the step height, *d*, is twice the value of the estimated *h*. In simple words, the depth is being determined by calculating the distance between two parallel lines, one passing through regions A and B, and the other passing through region C.

$$Z = \alpha \cdot X + \beta + h \cdot \delta \quad \textbf{(2-9)}$$

Calibration of the *x*-and-*y*-axes scales

To calibrate the horizontal scales of the instrument, a calibrated cross-grafting artefact (type ACG) can be used (Figure 8). Amplification, linearity and perpendicularity are assessed by measuring the centre of gravity of the squares of the ACG. There are two approaches for that. The first consists in calculating the centre of gravity of the pore, thus considering its depth. The second is based on feature identification. The squareness between axes *x* and *y* can be determined by measuring the angle between two nominally orthogonal rows of square holes.

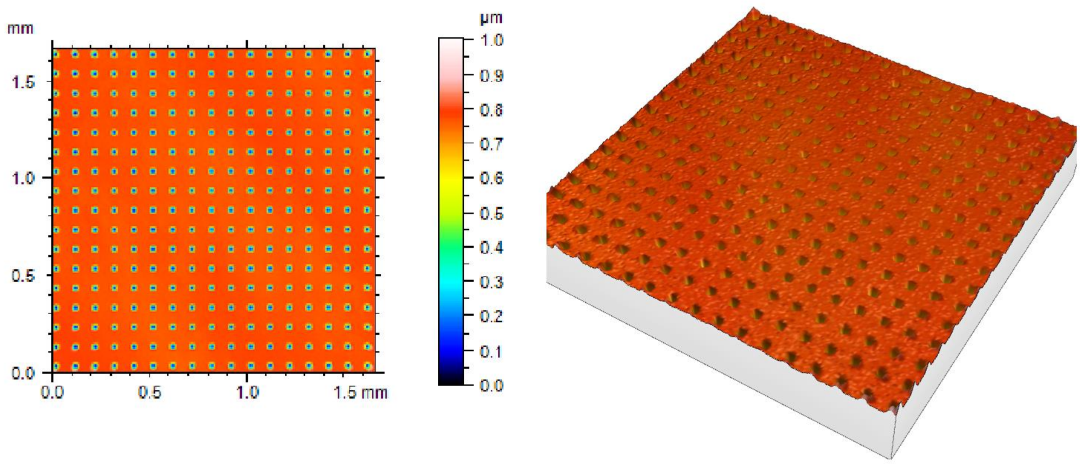

Figure 8 – 2D and 3D surface topography representation of a type ACG artefact plotted by MountainsMap®

Figure 9 - Pore identification by fitting a binary mask (blue) and their centroids (red)

2.3 Knowledge gaps

The knowledge gap in the field of calibration routines for topography measuring instruments is how to implement the procedures in software. The definitions present in the standard documents (ISO 25178-600 and ISO 5436-1) need to be shaped and implemented in code. Own contributions need to be developed to follow the standards while treating the data in an automatic way.

Moreover, the calibration routines are to be implemented in MATLAB® and then called by MountainsMap®. The reason why the routines cannot be directly implemented in MountainsMap® is its lack of automation capability. The aim of the project is to procure automatic calibration routines. Hence, the link with another platform that supports automation is needed. MATLAB® is able to read the files generated by the instrument and MountainsMap®, process them, generate results and send them back to MountainsMap®.

3 RESEARCH QUESTIONS, HYPOTHESIS AND METHODOLOGY

3.1 Research questions

The main question of this project concerns the possibility of creating an automated calibration system that allows unskilled users to perform the calibration of their instruments and obtain measurement uncertainties.

3.2 Research hypothesis

The main hypothesis is that it is possible to implement such automation and that MountainsMap® is able work with the MATLAB® extension to successfully obtain the measurement uncertainties.

For every metrological characteristic package, particular hypotheses have been assumed regarding the data that is going to be used to estimate their magnitude:

Measurement noise

- It is stationary in a statistical sense
- It presents a normal distribution centred in the origin
- There are not non-measured points

Flatness deviation

- It behaves as a systematic error
- There are not non-measured points
- The influence of noise decreases with the number of averaged topographies
- The effect of the sample contributions decreases with the number of averaged topographies

Amplification, linearity and perpendicularity

- There are no interpolation errors
- There are not non-measured points

However, the measured data does not always accomplish these assumptions. Hence, some methods to test the data and solvent these issues are developed in the software.

3.3 Research methodology

Figure 10 illustrates the methodology of this project. It starts with an initial research and understanding effort of the surface topography measuring instruments field. It continues with a review of the metrological characteristics and calibration procedures. The next step enters in an iterative process where the routines are implemented in MATLAB and the code is tested. Finally, the procedures are implemented in MountainsMap.

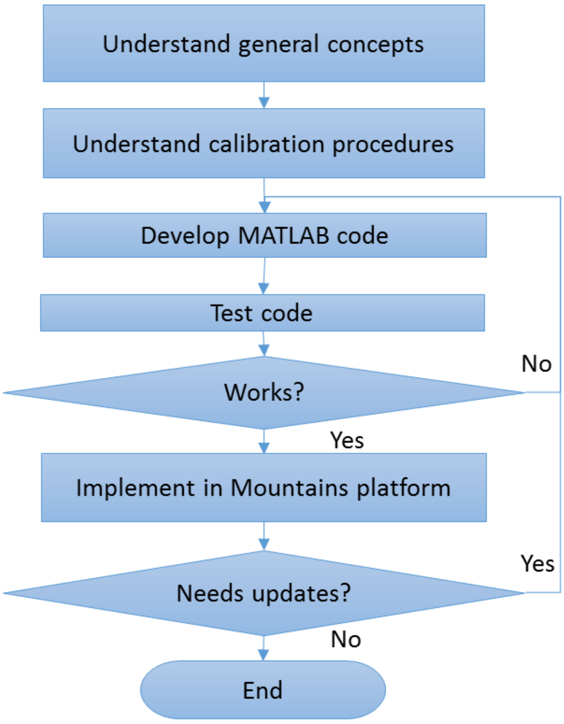

Figure 10 - Flowchart of the research methodology

4 SOFTWARE DEVELOPMENT

Every software developed or maintained at or for NPL requires a quality plan. This software is to be developed using approved procedures. The necessary documents are subjected to the Software Integrity Level (SIL), which is determined by using a risk based approach.

The primary model for software development at NPL, as well as the corresponding to this project, is the iterative development process. It starts with risk assessment and continues with an iterative sequence consisting of user requirements, functional specifications, design and code, verification, validation and review (Figure 11).

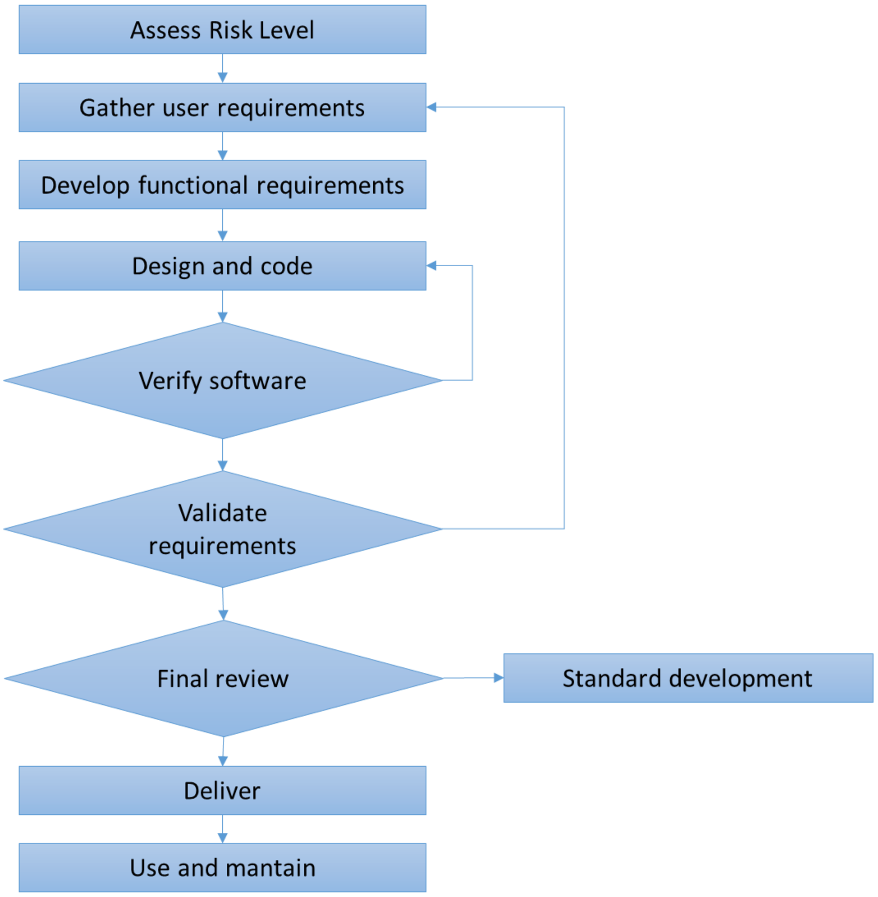

Figure 11 - Flowchart of the iterative software development process [8]

The calibration routines developed by NPL to calculate the measurement uncertainties of the areal surface topography measurement instruments have been implemented in MATLAB®. The routines to estimate the metrological characteristics have been grouped in three work packages:

- Measurement noise
- Flatness deviation
- Amplification, linearity and perpendicularity

where the last one is subdivided in *z*-axis scale and *x*-and-*y*-axis scales. The implementation of the routines is described below.

4.1 Measurement noise

There are two approaches for the subtraction technique. To do *n* subtractions, it is possible to take *n+1* measurements in a same position, or *n* pairs of measurements, each one in a different position by moving in the horizontal plane. The last option enables the use of the same measurements to calculate another metrological characteristic, flatness deviation, which will be addressed in the next subsection. The developed code is suitable for both options.

The methodology employed to estimate measurement noise is outlined in Figure 12. The 3D topographies are transferred to MATLAB. Then, in order to eliminate spurious data, a threshold is applied. If the amount of thresholded pixels and/or non-measured points is above 5% (see ISO 25178-700 27/02/2017), the measurement is considered invalid and thus discarded. Subtraction is performed at every pair of valid measurements and N_{Mi} is calculated. After all the subtractions have been performed, the overall N_M is estimated.

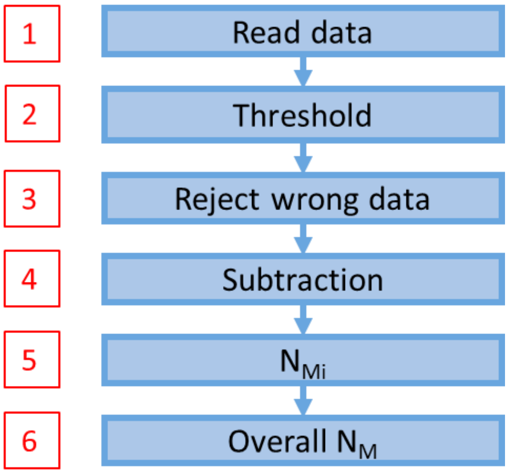

Figure 12 - Flowchart of measurement noise process

- The first task (step 1 in Figure 12) is to read the data generated by MountainsMap® by using a modified version of the module "SUR_read2" provided by NPL. This module transfers the data to MATLAB® as a matrix of height values. It also determines the units of the measurements. The modifications include the suppression of the outputs that are not used in this application, such as *x* and *y* data, and the addition of an output coefficient to transform the data into micrometres. A pair of measurements is stored in two matrices at each iteration. The matrices are overwritten every time .
- Because non-measured points from MountainsMap® do not appear as NaN after transferring the data to MATLAB®, non-measured points need to be identified. For every topography, the NPL module assigns all non-measured points an equal value, being the minimum. Hence, data having a value equal to the minimum is diagnosed and set as NaN. After that, pixels exceeding a ±3σ threshold are rejected and thus set as NaN (step 2).
- If the total amount of NaN exceeds 5%, the measurement is not considered (step 3).
- At every iteration, two surfaces are subtracted (step 4) and stored in a matrix, and N_M is obtained as defined in (2-5), (step 5), whose values are stored in another matrix. This process is repeated for every pair of measurements and the overall N_M is calculated as detailed in (2-6), (step 6).

4.2 Flatness deviation

In this implementation, it has been applied a methodology in which the long wavelength terms were removed to threshold the surface. After that, the terms were added back to estimate Z_{flt}. The main steps of the procedure are described in Figure 13. The routine consists in averaging surfaces until Z_{flt} does not vary more than 5%. The form removal operation is applied by fitting a polynomial to the topography and subtracting it. The threshold limits are calculated so as the error data produced by the instrument are not eliminated. After thresholding and adding back the form, the topography is levelled and added to the average. Z_{flt} is estimated over the average surface.

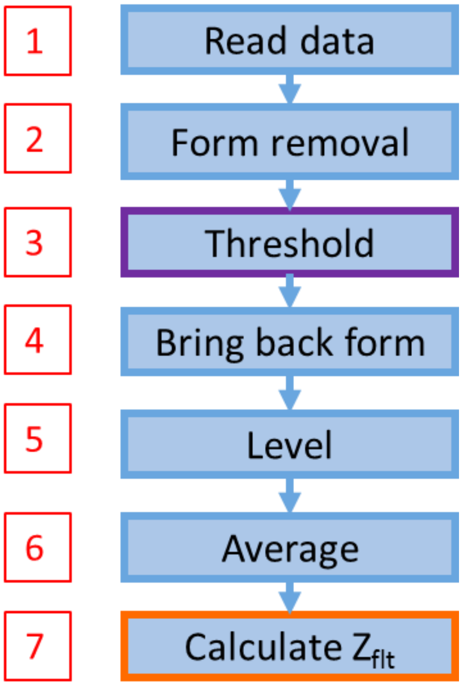

Figure 13 - Flatness deviation flowchart

- The data is transferred to MATLAB® by using the NPL module and stored in a matrix, as explained before (step 1, Figure 13).
- Form removal is applied to remove the long wavelength components (step 2, Figure 13). A fifth order polynomial (4-1) is applied by using the MATLAB® function “fit” and the coefficients are obtained with the function “coeffvalues”. The polynomial is substracted from the original data (Figure 14).

$$p = c_{10}x + c_{01}y + c_{20}x^2 + c_{11}xy + c_{02}y^2 + c_{30}x^3 + c_{21}x^2y + c_{12}xy^2 + c_{03}y^3 + \cdots + c_{05}y^5 \quad \textbf{(4-1)}$$

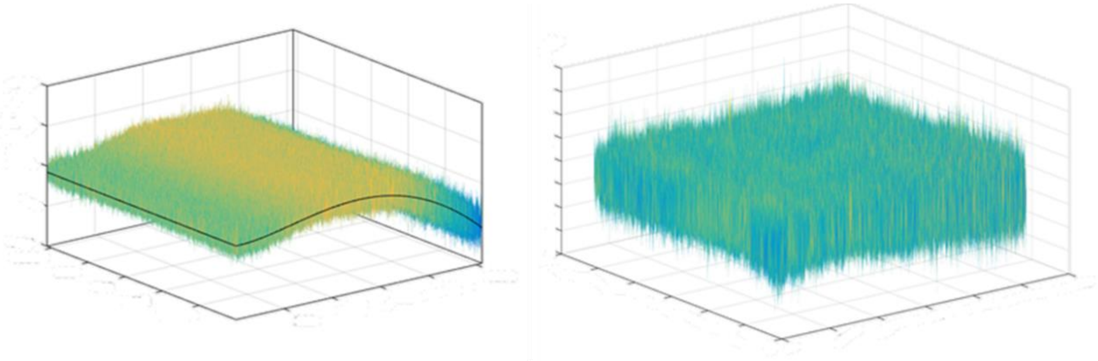

Figure 14 - Polynomial fitting on a surface affected by long wavelength components (left) and same surface after form removal (right)

- The next step is thresholding. However, the threshold limits need to be defined. The procedure to do so is sketched in Figure 15. The data originated by the instruments will be repeated in every measurement. Hence, it is not necessary repeating the procedure for every topography. This means that the definition of the limits needs to be performed only once.
- Hence, only two topographies are considered (step 1, Figure 15).
- Mean and standard deviation of this set of measurements are calculated (step 2, Figure 15).
- An initial pair of threshold limits are defined as Th1=$\mu \pm 3\sigma$ (step 3, Figure 15).
- The data outside the threshold is diagnosed and the coordinates of the pixels are determined (step 4, Figure 15).
- The two topographies are averaged (step 6, Figure 15).
- To determine if the error data was produce by the instrument or it is spurious data, the value of the diagnosed data is compared before and after averaging. In the case of belonging to the instrument, the error pixels will be present in both topographies and thus their values will not be decreased after averaging. Th2 is initialised at 0 and defined as the maximum value belonging to the interval [$Z_{i,j,1}$ - Sq < $Z_{i,j,avg}$ < $Z_{i,j,1}$ + Sq] (step 6, Figure 15).
- The final threshold is defined as the maximum value of Th1 and Th2 (step 7, Figure 15).

- Once the limits are defined, the maps are thresholded (step 3, Figure 13).
- The polynomial removed in step 2, Figure 13, is added back (step 4, Figure 13).
- The topography is levelled by using the module “level” provided by NPL (step 5, Figure 13).
- The topography is added to the average (step 6, Figure 13) and Z_{flt} is calculated (step 7, Figure 13).
- As mentioned before, topographies are added and the process (excluding the threshold definition) is repeated until Z_{flt} acquires a stable value, not varying more than 5%.
- Whenever a surface presents more than 5% of NaN, corresponding to non-measured points and/or spurious data, the topography is removed.

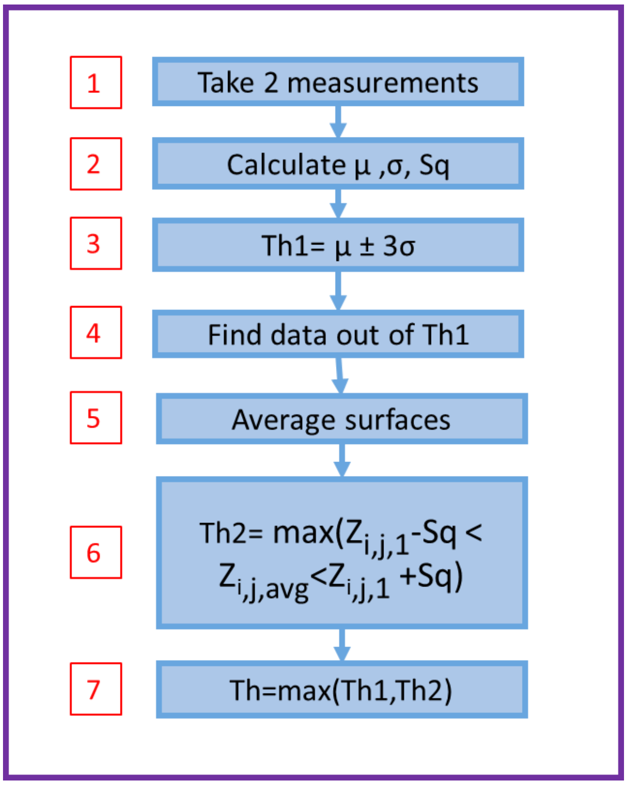

Figure 15 - Flowchart of threshold limits selection process

4.3 Amplification, linearity and perpendicularity

Two independent modules have been programmed for the calibration of the vertical scale and the horizontal scales.

4.3.1 *z*-axis calibration

Figure 16 summarises the most important steps of the *z*-scale calibration routine. A binary mask is created over the step topography transferred to MATLAB® to apply a level operation over this mask. Then, another binary mask is applied to identify the edges of the groove. The angle of the groove is determined and if it is higher than 1º, the measurement is considered invalid. Many profiles are extracted from the identified groove and an average profile is calculated. The Type A1 step height analysis described in ISO 5436-1 is applied over the average groove and its depth is estimated. Every groove is measured *n* times, and so the process is repeated *n* times. An average value of the *n* depths is calculated. The same process is repeated for every groove. The amplification coefficient is calculated as the slope of the linear fitting of the estimated depths against the calibrated values, crossing zero. The linearity is estimated as the maximum difference between the linear regression and the ideal curve. After repeating the whole process for different positions of the instrument in the vertical range, the reproducibility, repeatability, error and traceability contribution to the measurement uncertainty are determined.

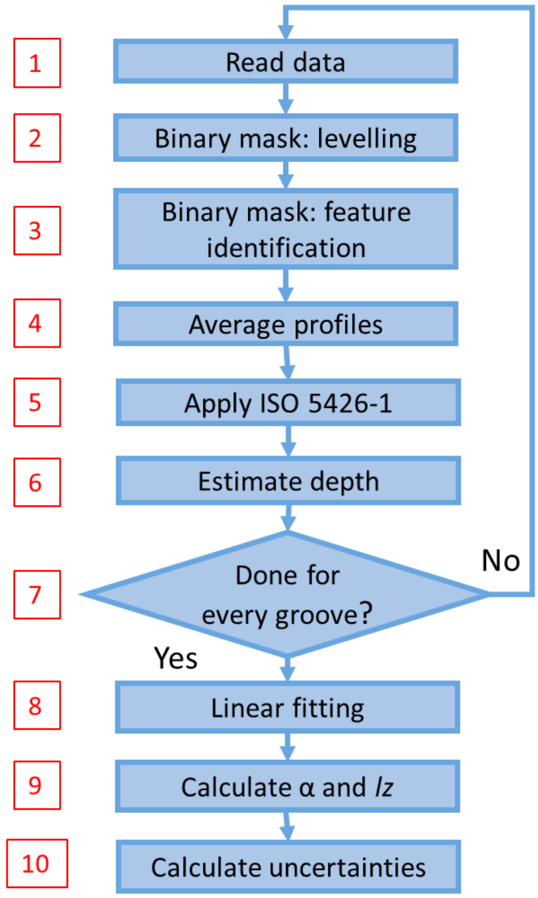

Figure 16 - Flow chart of the calibration of the *z*-scale

- The groove topography (Figure 17) is transferred to MATLAB® (step 1, Figure 16).

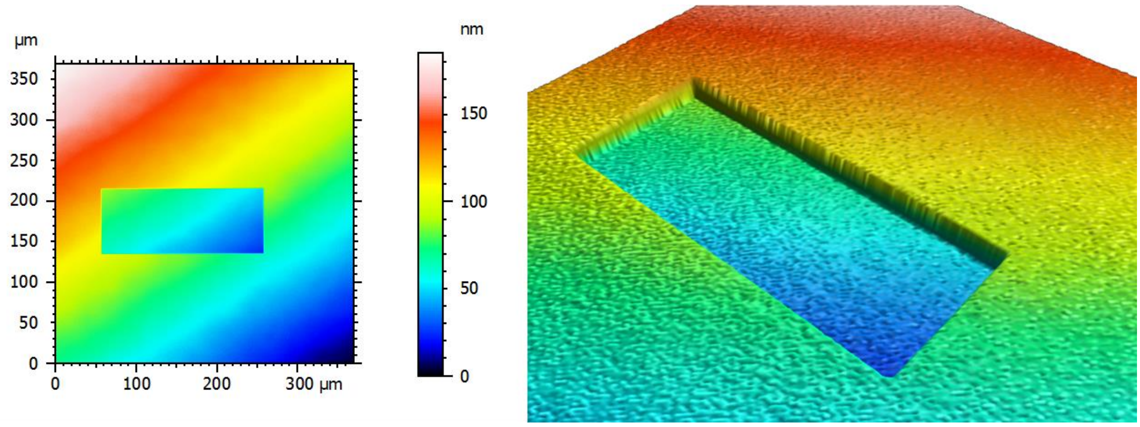

Figure 17 - Topography map of a step height (left) and its 3D view from MountainsMap®

- The MATLAB function “imgradient”, which provides the gradient of the *z* data, is used to create binary masks. A threshold gradient is defined as the mean value of the maximum and minimum gradient. An initial mask is created, setting all the values at “0”. Then, the pixels on the mask corresponding to the gradient image pixels with values higher than the threshold are set as “1”. The module “level” provided by NPL levels the surface using the mask for plane fitting. The points with “1”, corresponding to the transition zone of the step, are excluded from the plane fitting (step 2, Figure 16).
- After levelling, a second binary mask is created following the same procedure. The binary mask is used to identify the groove edges (Figure 19). A minimal bounding rectangle is fitted to the cloud of binary data by using the function “minboundrect” (by John D'Errico, 2007), (step 3, Figure 16). Nonetheless, it may introduce interpolation errors.
- By using the positions of the corners, the angle of the groove respect to the x-and-y axes is determined. If it is higher than 1º, the measurement is discarded. The orientation (vertical or horizontal) of the longest direction of the groove is also identify.
- A set of profiles perpendicular to the longest direction of the groove, ranging from 30% to 70% of its length, are extracted. The module “Depth_a1_pmh” provided by NPL averages the profiles (step 4, Figure 16) and applies the procedure described in ISO 5426-1 (step 5, Figure 16) to estimate the depth (step 6, Figure 16). See Figure 18

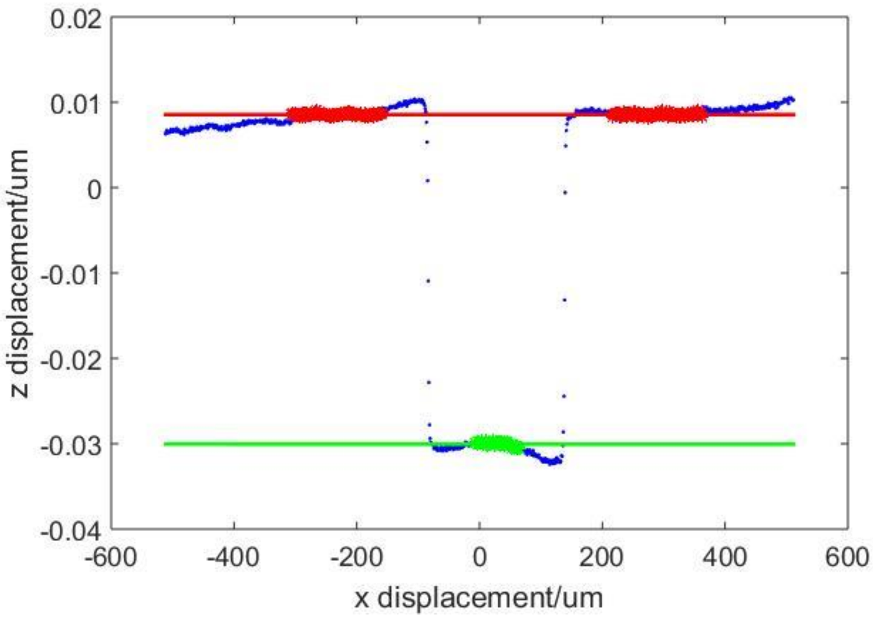

Figure 18 - Type A1 step height analysis (ISO 5436-1)

- This process is repeated for the *n* measurements of every groove. Then, an average depth is estimated. The routine does the same procedure for the six steps present in the calibration artefact. After the completion of the analysis of each groove, the routine checks if all the grooves have been analysed (step 7, Figure 16). If it is not so, the routine goes back to step 1, Figure 16.
- When all the grooves have been analysed, the function “polifix” (by Are Mjaavatten, Telemark University College, 2015) finds the coefficients of the polynomial that does a linear regression of estimated depth data against the nominal values, crossing zero (step 8, Figure 16).

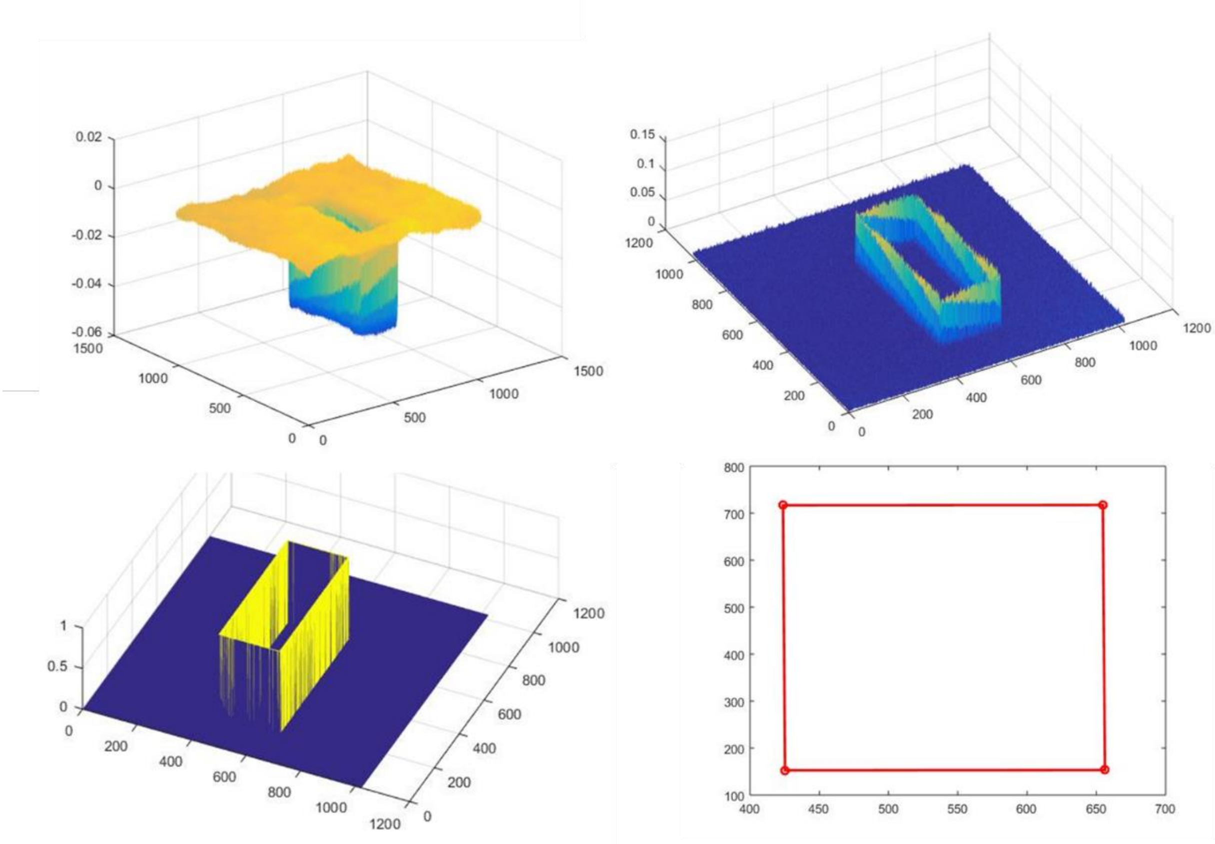

Figure 19 - Raw surface in MATLAB® (top left), gradient image (top right), binary mask (bottom left) and identified edges of the groove (bottom right)

- Amplification is defined as the first coefficient provided by "polifix", which corresponds to the slope of the curve. Linearity is obtained as the maximum absolute value of the difference between the linear fitting and the nominal values (step 9, Figure 16).
- Finally, the contribution to the measurement uncertainties is calculated (step 10, Figure 16). Repeatability is estimated as the maximum standard deviation of depth values at each groove. Reproducibility is calculated as the maximum standard deviation of linearity and amplification. The error contribution is defined as the maximum absolute error plus its repeatability. Finally, the traceability contribution is read from.

4.3.2 *x*-and-*y*-axes calibration

The procedure is similar to the previous routine. The data is transferred to MATLAB® and two binary masks are applied. The first mask is used for levelling, and the second to identify the pores of the grid. Then, the centroids are found and their locations are compared to the calibrated locations (Figure 20).

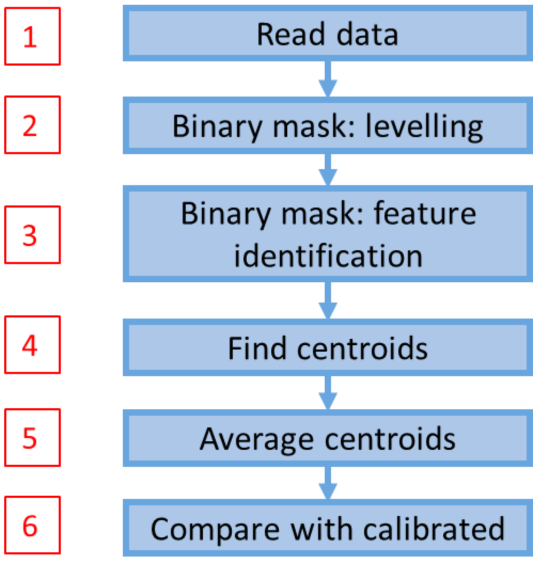

Figure 20 - Flowchart of the *x*-and-*y* scales calibration routine

- Data is transferred to MATLAB® (step 1, Figure 20).
- A binary mask is applied to level the topography (step 2, Figure 20).
- A binary mask is applied to identify the pores of the grid (Figure 21) (step 3, Figure 20).

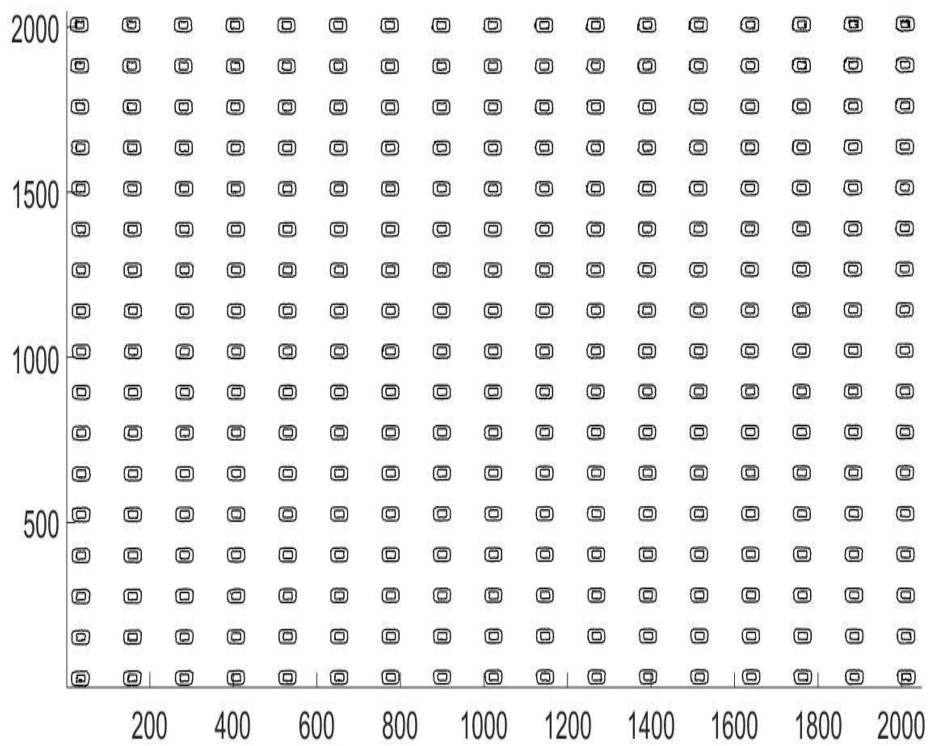

Figure 21 - Pores identification by the use of a binary mask

- The function “regionprops” returns measurements for the specified properties. The function is used with the property ‘Centroid’. This module finds the centroids as the centre of mass of the regions and returns a vector with their coordinates (step 4, Figure 20).
- The described steps are repeated for the set of measurements and the centroids are stored in a matrix. Then, the module “ls2dptm” provided by NPL is applied to align the grids. The module provides translation and rotation matrices to switch the centroids of each grid to a same coordinates system.
- Every centroid needs to be averaged with its analogue in the rest of grids. Thus, the matrix storing the centroids coordinates needs to be ordered. The matrix is ordered by ascendant *y* coordinate and grouped in sets of the number of points per row. By this, every group of pairs of coordinates contains a grid row. Then, for every row, the data is ordered by ascendant *x* coordinate.
- Then, the centroids are averaged (step 5, Figure 20).
- The “ls2dptm” module is applied to align the averaged grids and the calibrated values.

- Their values are compared and errors and standard deviations are extracted (step 6, Figure 20).

5 RESULTS AND DISCUSSIONS

At the completion of this project, the calibration routines to estimate measurement uncertainties associated with measurement noise, flatness deviation, amplification, linearity and perpendicularity have been successfully implemented in MATLAB. Some examples are provided below. The measurements were performed with a CSI instrument.

Measurement noise

N_M is isolated by subtraction. The program outputs a histogram with the set of values resulting from all the subtractions, together with its associated normal distribution (Figure 22). Whenever a topography contains more than 5% of non-measured and/or spurious data, a message box is displayed warning about it (Figure 23). At the end of the analysis, a message box is displayed showing the number of subtractions considered (Figure 24). Another box shows the overall N_M and the standard deviation of the set of values resulting from the subtractions. Ideally, they should have the same value (Figure 25).

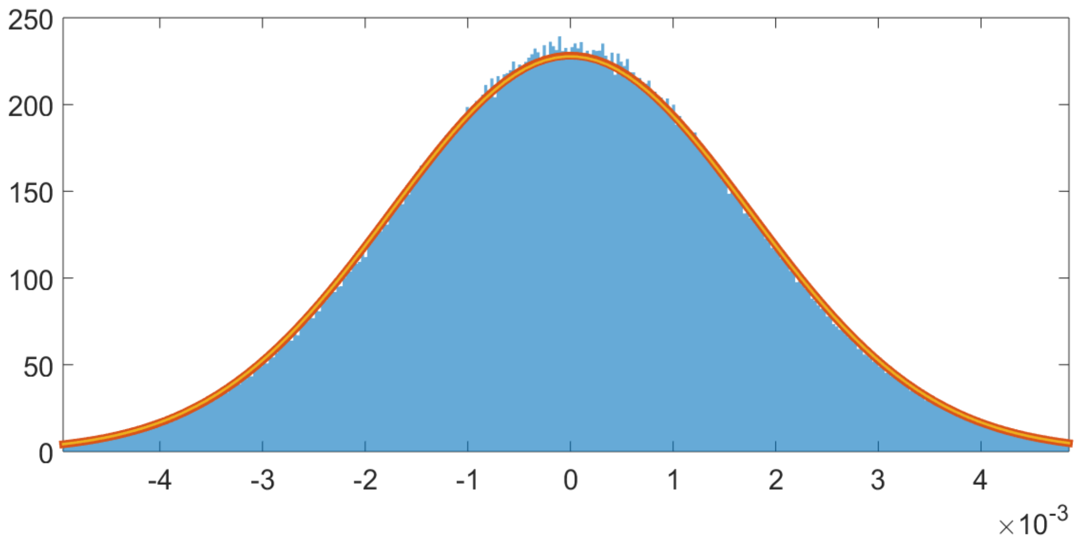

Figure 22 - Histogram of all the values corresponding to the subtractions in blue. Normal distribution curve N(μ,σ) with μ and σ corresponding to the mean and standard deviation of all the set of values, in red. Over this curve, normal distribution N(μ,σ) with μ and σ corresponding to 0 and $\sqrt{2}N_M^2$, in yellow

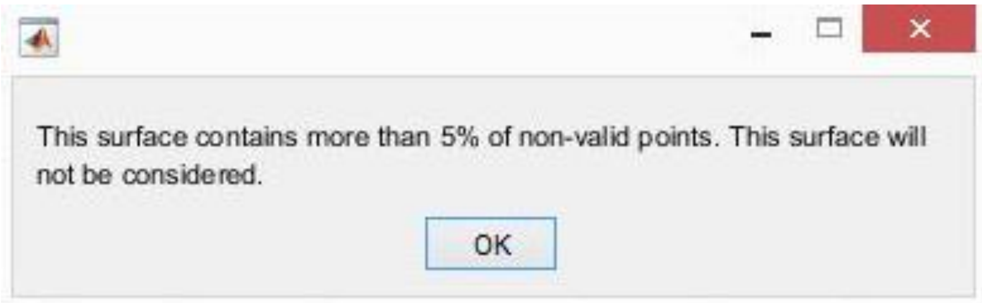

Figure 23 - Message box warning that a surface has been rejected

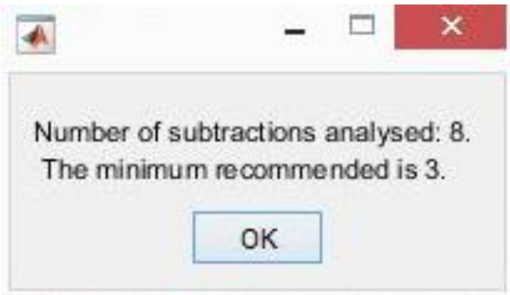

Figure 24 - Message box indicating the number of subtractions considered

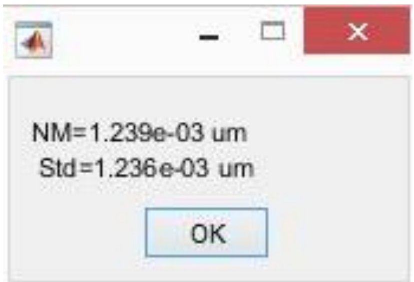

Figure 25 - Message box showing the overall N_M and its standard deviation

Flatness deviation

Flatness deviation routine outputs a graph showing the values of Z_{flt} for each iteration. An example is provided in (Figure 26). A message box displays the Z_{flt} value (Figure 27). For this case, Z_{flt} did not acquire a stable value not changing more than 5% after averaging 24 topographies and refusing 8, being 32 the total number of topographies in the set of measurements. The flatness deviations estimated for this case was Z_{flt}=3.426 nm. However, although changing less than

4 nm, the graphs shows a decreasing trend which indicates a potentially lower value. More measurements would be required for this case.

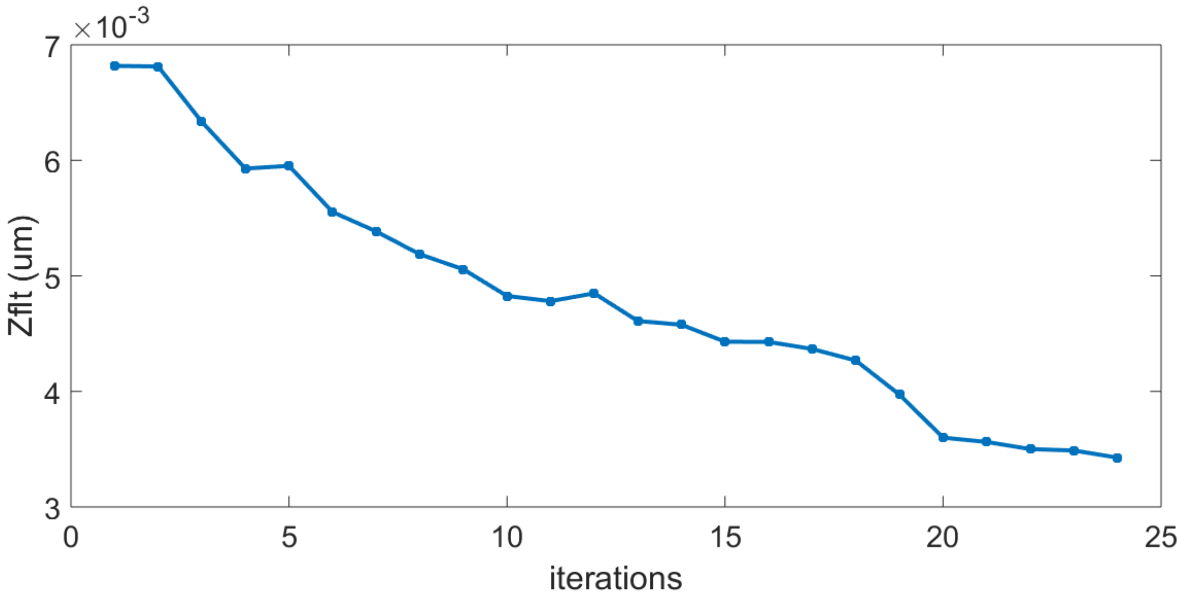

Figure 26 - Flatness deviation for each iteration

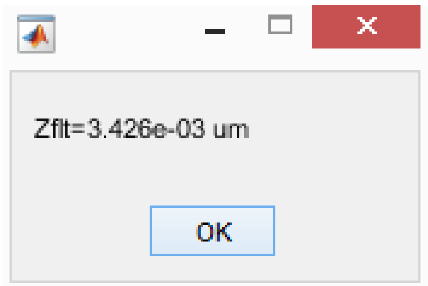

Figure 27 - Message box displaying the flatness deviation

Not considering any operation on the surface to reduce the spurious data can lead to the necessity of a large amount of topographies to stabilise Zflt. It was wanted to be analysed the improvements of this averaging technique, in which the topographies are thresholded, against the simple method, were the maps are averaged without any operation. It was made a comparison of both methods to analyse the improvements (Figure 30). The stabilisation criterion has been changed to 0% to analyse the behaviour of Z_{flt} along the whole set of measurements. For the first 12 iterations, both results converge to a close value. However, the simple averaging method (in blue) presents instabilities due to spurious data, which causes huge slopes followed by a decreasing trend that tries to reach the stable value. Thus, this method is very dependent on the quality of the measurements.

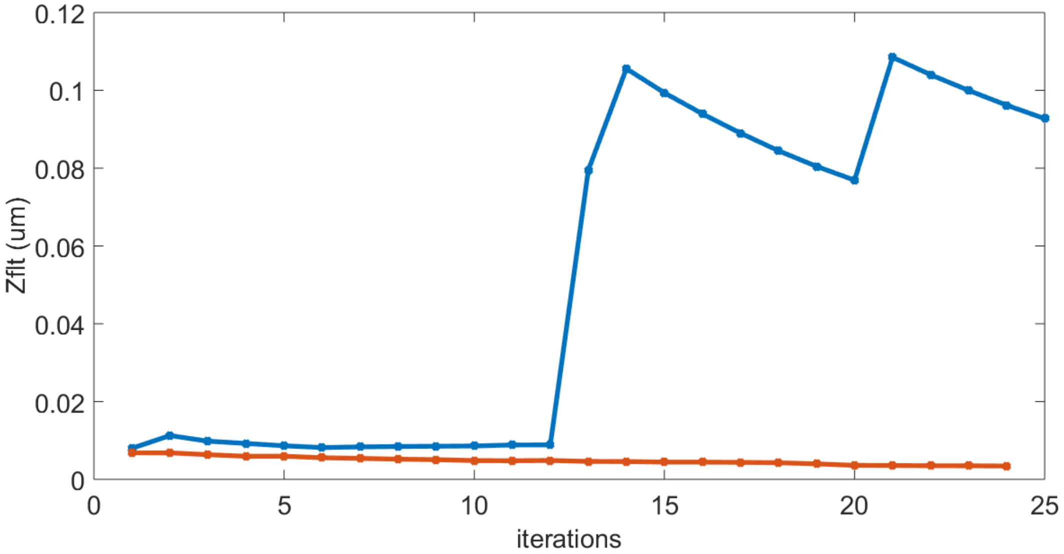

Figure 28 - Comparison between simple averaging method (blue) and thresholding-averaging method (orange)

Amplification, linearity and perpendicularity – Vertical scales

The routine to determine the amplification and linearity of the vertical scale outputs a graph with the instrument curve, its linear fitting and the ideal instrument curve (Figure 30). It also provides a message box showing a table with the calibrated depth (CT), measured depth (M), error (e), traceability (Tr), repeatability (R) and the measurement uncertainty for each groove (u) (Figure 29). The traceability data was not available at the moment of the analysis and so the values were set as 0.

	CT(um)	M (um)	e (um)	Tr (um)	R (nm)	u (um)
1	0.030	0.039	0.009	0.000	0.926	0.009
2	0.100	0.101	0.001	0.000	2.014	0.001
3	0.200	0.224	0.024	0.000	2.662	0.024
4	0.500	0.426	-0.074	0.000	2.092	0.074
5	1.000	0.894	-0.106	0.000	11.371	0.106
6	2.000	1.953	-0.047	0.000	3.325	0.047

OK

Figure 29 - Message box showing calibrated depth (CT), measured depth (M), error (e), traceability (Tr), repeatability (R) and the measurement uncertainty for each groove (u)

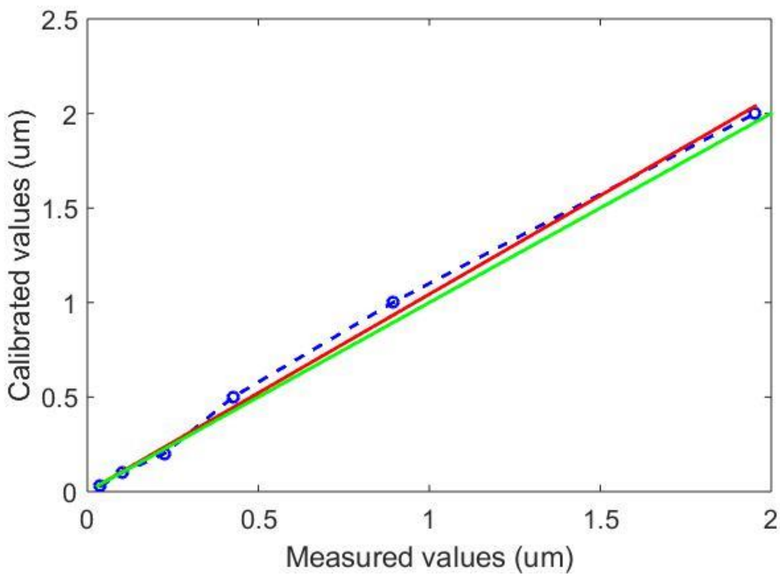

Figure 30 - Curve response of the instrument (blue), linear fitting passing per zero (red) and ideal curve (green)

Amplification, linearity and perpendicularity – Horizontal scales

Finally, the routine for the estimation of the uncertainties in the *x*-and-*y*-scales provides a plot with of the calibrated and measured centroids of the pores, showing the distortion of the measurements (Figure 31). The error of the measured centroids is magnified 50 times for better comprehension. These errors are presented in a histogram (Figure 32). The standard deviation for the coordinates of each centroid is also provided (Figure 33).

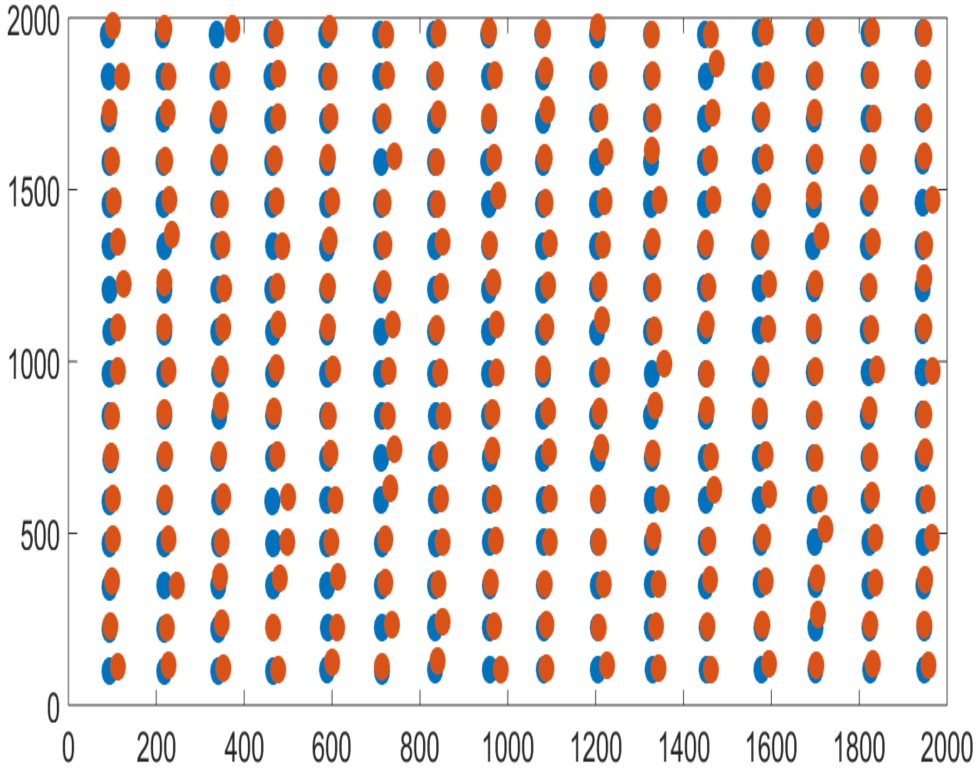

Figure 31 - Calibrated centroids (blue) and measured centroids with errors magnified x50 (orange)

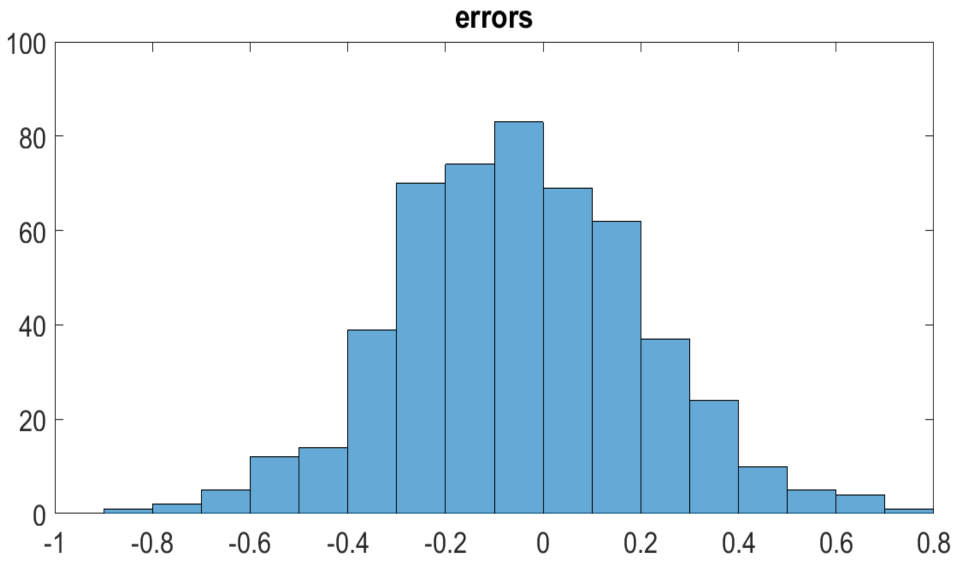

Figure 32 - Histogram of centroid errors

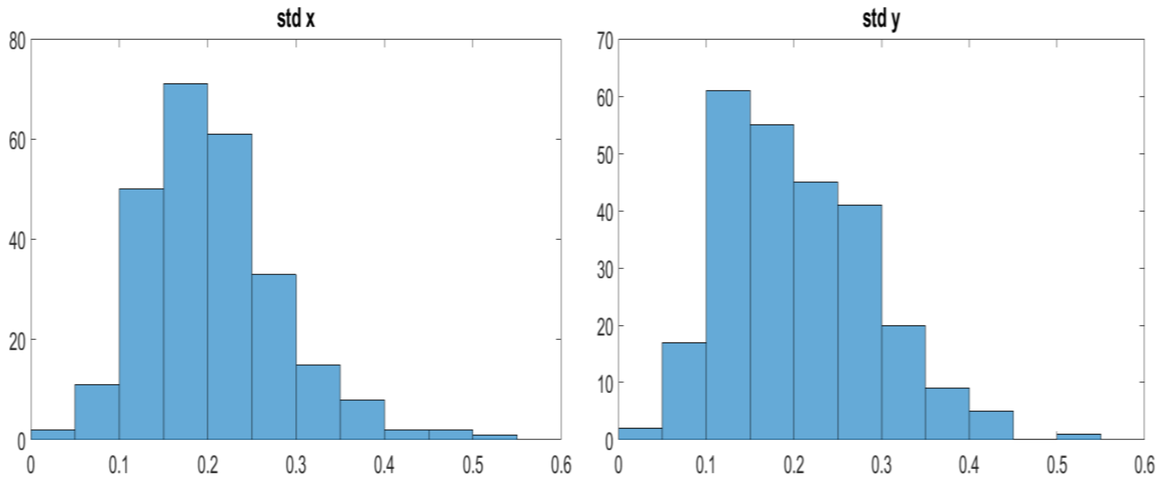

Figure 33 - Histograms of the standard deviation of the coordinates of the centroids

5.1 Key issues and challenges

The MATLAB extension in MountainsMap does not allow to run scripts which call external functions. Moreover, it only allows to work on one loaded studiable, applying the operations coded on a single script. The calibration routines developed in this project use external functions and work with many topography files at a same time. Hence, it has not been possible to include the routines in the metrology software. However, the routines are still useful as they are available to run directly on MATLAB and could potentially be integrated on MountainsMap after some modifications on the topography software.

The light source of the CSI instrument that was being used for performing the measurements required to test the routines did not work. Thus, it was necessary to wait for measurements taken at NPL, delaying the process.

Finally, it has been challenging to work within the metrology field, which has exposed myself to numerous new concepts, in whose comprehension is based the implementation of the routines developed in this thesis.

6 CONCLUSIONS

At the completion of this project, the calibration routines to estimate the measurement uncertainties related to measurement noise, flatness deviations, and amplification, linearity and perpendicularity of the scales have been successfully implemented in MATLAB ®. However, for reasons out of the scope of this project, it has not been possible to integrate them in MountainsMap®. Nonetheless, the routines are ready to be included in the metrology software and it will potentially be possible in the future.

It has been made a comparison between the flatness deviation routine implemented by following the minimum requirements specified in ISO 25178 and an optimization of it, thresholding the topographies. The simple technique presents instabilities when spurious data is present, leading to the requirement of a big set of data. However, by thresholding the topographies before averaging, the convergence of the flatness deviation presents a stable trend.

Amplification and linearity of the vertical scale has been faced with a new method based groove identification by using binary masks, followed by a minimal bounding rectangle fitting. Based on this fitting, the profiles are extracted and analysed.

Amplification, linearity and perpendicularity of the horizontal scales have been implemented following a similar approach based on feature identification. A binary mask is applied to identify the pores of the grids. Then, the centroids are estimated as the centre of mass of the regions formed by the binary mask. The results are to be compared with the current NPL approach, which estimates the centroids by finding the centre of gravity of the pores and thus considering their depth.

Although the routines are successfully implemented, further development, modifications and comparisons are suggested as future work:

- The routines for the calibration of the vertical and horizontal scales have not been tested in reproducibility conditions and it is something suggested to be done.

- In addition, the results obtained with the methodology developed in this project to determine the centroids in the analysis of the vertical scales needs to be compared with the NPL method, so as to contrast their performance in the uncertainty estimation.
- The developed routines for *x*-and-*y*-axes calibration considered that the measured grids contain the same number of pores than the calibrated sample. However, the field of view of some instruments may not enclose the whole grid. Thus, the measurement would not match in numbers of pores and the analysis would not be possible. To face it, it needs to be implemented some routine that identifies the pores in the measured topography, select the analogue pores in the calibration sample and aligns them for comparison.
- The results of the flatness deviation routine were compared with the simple averaging method. The implementation and comparison of another approach to threshold the topographies before averaging is suggested. It consists in subtracting pairs of topographies. The high values belonging to the instrument would be eliminated together with the flatness, and the remaining spurious data could be identified.
- The integration of the routines in the market leader software platform, MountainsMap®, is still pending and recommended to be carried out.
- Finally, it is encouraged to record and edit a video showing the how software works and presents the results, for marketing purposes.

REFERENCES

1. Leach R. Introduction to Surface Texture Measurement. Optical Measurement of Surface Topography. 2011. pp. 1–14.
2. Joint Committee For Guides In Metrology. Vocabulaire international de métrologie. VIM3: International Vocabulary of Metrology. 2012; 3(third): 104. Available at: DOI:10.1016/0263-2241(85)90006-5
3. Bhushan B., Wyant JC., Koliopoulos CL. Measurement of surface topography of magnetic tapes by Mirau interferometry. Applied Optics. 1985; 24(10): 1489–1497. Available at: DOI:10.1364/AO.24.001489
4. Bell S. Good Practice Guide No . 11 The Beginner ' s Guide to Uncertainty of Measurement Measurement Good Practice Guide No . 11 (Issue 2). (11).
5. Giusca CL., Leach RK. Measurement Good Practice Guide No. 127. Calibration of the metrological characteristics of Coherence Scanning Interferometers (CSI) and Phase Shifting Interferometers (PSI). (127).
6. Giusca CL., Claverley JD., Sun W., Leach RK., Helmli F., Chavigner MPJ. Practical estimation of measurement noise and flatness deviation on focus variation microscopes. CIRP Annals - Manufacturing Technology. CIRP; 2014; 63(1): 545–548. Available at: DOI:10.1016/j.cirp.2014.03.086
7. Giusca CL., Leach RK., Helary F., Gutauskas T., Nimishakavi L. Calibration of the scales of areal surface topography-measuring instruments: part 1. Measurement noise and residual flatness. Measurement Science and Technology. 2012; 23(3): 35008. Available at: DOI:10.1088/0957-0233/23/3/035008
8. Documents R. QPNPL / M / 013 NPL-Written Software. 2017; : 1–16.

APPENDICES

Appendix A MATLAB® scripts

A.1 Measurement noise

```
%% *Calculus of measurement noise (Sqnoise)*

%Sqnoise: standard deviation of noise distribution

%This script loads data from *.sur files and process it to calculate
%Sqnoise

%All the *.sur files must be gather together in a folder. No other
*.sur
%files than the ones object of analysis should be in that folder.

%Units: um

%Patricia Giménez Belando
%Last updated: 03/09/17
%%
close all
clear all
clc

%%
%READ DATA FROM
FOLDER%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% directory='C:\Users\Patricia
Giménez\Documents\CRANFIELD\THESIS\Coding\Noise data 05062017';
prompt = {'Enter the directory of the files:'};
dlg_title = 'Directory';
num_lines = 1;
directory = strjoin(inputdlg(prompt,dlg_title,num_lines));
direc=[directory '/*.sur'];
names=dir(fullfile(direc));
list_names={names.name};

%DEFINE/INITIALISE VARIABLES%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
noise=[];
measurements=length(list_names);

% figure
Sqnoisei=[];
noise_all=[];
for h=2:2:24
    file_name_cell_1=list_names(h-1);
    file_name_cell_2=list_names(h);
    file_name_1=strjoin(file_name_cell_1);
    file_name_2=strjoin(file_name_cell_2);
    [z,a]=SUR_read2_z(file_name_1);
    z_data_1=z*a;
```

```
        p=length(z_data_1);
        sum=zeros(p);
    [z_data_2]=SUR_read2_z(file_name_2);
    z_data_2=a*z_data_2;


  %%
    % ELIMINATE SPURIOUS DATA%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Identify NA from Mountains and make them NaN
    NA1=min(z_data_1(:)); % because NA from mountains are established
as a same min value
    find1=find(z==NA1);
    if length(find1>1)
        for j=1:length(find1)
            l=ceil(find1(j)/p);
            k=find1(j)-p*(l-1);
            z_data_1(k,l)=NaN; %change to NaN
        end
    end
    NA2=min(z_data_2(:));
    find2=find(z==NA2);
    if length(find2>1)
        for j=1:length(find2)
            l=ceil(find2(j)/p);
            k=find2(j)-p*(l-1);
            z_data_2(k,l)=NaN; %change to NaN
        end
    end
    z_data=[z_data_1,z_data_2];

    %Threslhold
    z_data_test=z_data;
    z_data=[];
    for i=1:2
        flag=0;
        m=z_data_test(:,p*i-(p-1):p*i);
        mu=nanmean(m(:));
        vari=nanvar(m(:));
        for i=1:p
            for j=1:p
                if m(i,j)>mu+3*sqrt(vari)||m(i,j)<mu-3*sqrt(vari)
                    m(i,j)=NaN;
                end
            end
        end
        length(find(isnan(m(:))));
        if length(find(isnan(m(:))))>0.05*p^2 %if NaN > 5% of data,
don not consider it (ISO 25178-700 27/02/2017)
            msgbox('This surface contains more than 5% of non-valid
points. This surface will not be considered.');
            flag=1;
            break;
        else
            z_data=[z_data m];
        end


    end
```

```
%%
    %SUBSTRACT AREAL TOPOGRAPHIES%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    if flag==0 && nanvar(z_data(:))~=0
        i=2;
        sub=z_data(:,p*i-(p-1):p*i)-z_data(:,p*(i-1)-(p-1):p*(i-1));
%subtraction of each pair of measurements
        mu_sub=nanmean(sub(:));
        sub=sub-mu_sub;
        noise=sub;
        noise_all=[noise_all noise];
        %borrar
%         figure
%         surf(noise,'EdgeColor','none')
 %%
        %     noise=[noise sub];



 %%
        %CALCULATE
SQNOISE%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        Sqi=sqrt(nanmean(noise(:).^2));%RMS excluding NaN
        Sqnoisei=[Sqnoisei Sqi/sqrt(2)];
 %%
       %PLOT
%         subplot(4,4,h/2)
%         hold on
%         histogram(noise,'Normalization','pdf','EdgeColor','None')
% %         hold on
%         mu=nanmean(noise(:));
% %         str=sprintf('mean=%f\nvar=%f\n',mu,nanvar(noise(:))/2);
% %         legend(str); %%%%%%%%%%%%%%%%%%How to determine a fixed
place?%%%%%%%%%%%%%%%%%
%         hold on
%         x=[mu-
4*sqrt(nanvar(noise(:))):0.0001:mu+4*sqrt(nanvar(noise(:)))];
%         norm=normpdf(x,mu,Sqnoisei(end)*sqrt(2));
%         plot(x,norm,'LineWidth',4)
%         xlim([x(1), x(end)])
    end
end
%%OBTAIN FINAL SQNOISE
Sqnoise=sqrt(nanmean(Sqnoisei(:).^2)) %ojo, no RMS
Sqnoise2=Sqnoise^2
%Calculus of normal distribution of global measurement noise: mu and s
variance_all=nanvar(noise_all(:))/2 %divided by 2 because noise(i) is
a combination of 2 distributions
mu_all=nanmean(noise_all(:));
msgbox( sprintf('NM=%1.3d um\n Std=%1.3d um',Sqnoise,
sqrt(variance_all)));
msgbox( sprintf('Number of subtractions analysed: %d.\n The minimum
recommended is 3.',length(Sqnoisei)));

%%
%
%PLOTS%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure
```

```
histogram(noise_all,'Normalization','pdf','EdgeColor','None')
hold on
x=[mu_all-4*sqrt(variance_all):0.0001:mu_all+4*sqrt(variance_all)];
norm=normpdf(x,mu_all,sqrt(nanvar(noise_all(:))));
plot(x,norm,'LineWidth',6)
xlim([x(1), x(end)]);
hold on
norm2=normpdf(x,0,Sqnoise*sqrt(2));
plot(x,norm2,'LineWidth',2)
% str=sprintf('mean=%f\nvar=%f\n',mu_all,variance_all);
% legend(str);
```

A.2 Flatness deviation

```
%% *Calculus of flatness deviation (Sz)*

%Sz: maximum height of the scale limited surface

%This script loads data from *.sur files and process it to calculate
Sz

%Units: um;

%Patricia Giménez Belando
%Last updated: 31/08/17


%%
close all
clear all
clc
%%
%DEFINE/INITIALISE VARIABLES
noise=[];
meas=1;
Sz0=Inf;
counter=0;
z_d=[];
sum1=0;
Szs=[];
hh=0; % counter that indicates the surface treated
Th2=0;
%%
%READ DATA FROM FILES
directory='C:\Users\Patricia
Giménez\Documents\CRANFIELD\THESIS\Coding\Noise data 05062017';
% prompt = {'Enter the directory of the files:'};
% dlg_title = 'Directory';
% num_lines = 1;
% directory = strjoin(inputdlg(prompt,dlg_title,num_lines));
direc=[directory '/*.sur'];
names=dir(fullfile(direc));
list_names={names.name};
measurements=length(list_names); %number of stored measurements
for h=1:1:measurements %because are pairs of measurements, only one of
each pair is taken
    file_name_cell_1=list_names(h);
```

```
    file_name_1=strjoin(file_name_cell_1);
    [z_data_1,a]=SUR_read2_z(file_name_1);
    z_data=a*z_data_1;

    if h==1 %to initialise the values only the first time
        p=length(z_data);
        sum=zeros(p);
        d=2*ones(p);%number of measurements at each pixel
    end
    %Identify NA from Mountains
    NA=min(z_data(:)); % because NA from mountains are established as
a same min value
    NAs=length(find(z_data(:)==NA));
    if NAs/p^2>0.05 % if more than 5% non measured points
        flag_nan=1;
        msgbox('This surface contains more than 5% of non-valid
points. This surface will not be considered.');

    else
        flag_nan=0;
        hh=hh+1;
    end
    if h==5
        borrar=1;
    end
    if flag_nan==0


        %%
        %To threshold the surface, a form removal operation is
applied. By
        %removing the long wavelength coponents it's possible to
identify
        %te spurious data and eliminate them. Once thresholded, the
form is
        %added back.
        %The surfaces are levelled and the mean is subtracted so as to
sum
        %at the same level when averaging

        %FORM REMOVAL by polynomial fitting
        %Create x and y vectors
        x=[]; y=[];z=[];
        y1=1:p;
        x1=ones(1,p);
        for i=1:p
            y=[y y1];
            x2=i*x1;
            x=[x x2];
        end
        x=x';
        y=y';

        %put z_data in a vector
        z=reshape(z_data,p^2,[]);
        f=fit([x,y],z,'poly55'); %For surfaces, degree of x and y can
be up to 5.
```

```
        coeff=coeffvalues(f);    %The degree of the polynomial is the
maximum of x and y degrees.
        for i=1:p^2
            pol(i) = coeff(1) + coeff(2)*x(i) + coeff(3)*y(i) +
coeff(4)*x(i)^2 + coeff(5)*x(i)*y(i) + coeff(6)*y(i)^2 +
coeff(7)*x(i)^3 + coeff(8)*x(i)^2*y(i) + coeff(9)*x(i)*y(i)^2 +
coeff(10)*y(i)^3 + coeff(11)*x(i)^4 + coeff(12)*x(i)^3*y(i) +
coeff(13)*x(i)^2*y(i)^2 + coeff(14)*x(i)*y(i)^3 + coeff(15)*y(i)^4 +
coeff(16)*x(i)^5 + coeff(17)*x(i)^4*y(i) + coeff(18)*x(i)^3*y(i)^2 +
coeff(19)*x(i)^2*y(i)^3 + coeff(20)*x(i)*y(i)^4 + coeff(21)*y(i)^5;
        end
        %   put pol (polynomial height data) in a matrix shape
        pol_mat=[];
        for i=1:p
            col=pol((i-1)*p+1:p*i);
            %pol_mat=[pol_mat col'];
        end
        pol_mat=reshape(pol,[p,p]); %polynomial in matrix shape
        z_data=z_data-pol_mat; %form removal
        if hh==1
            p1=pol_mat;
        elseif hh==2
            p2=pol_mat;
        end

        %ESTABLISH THRESHOLD LIMITS
        counter=counter+1;
        sum1=sum1+z_data;
        if counter <= 2
            z_d=[z_d z_data]; %hold the first 2 measurements
            if counter==2
                mean_t=mean(z_d(:)); %mean of the 2 surfaces
                sigma_t=std(z_d(:)); %standard deviation of the 2
surfaces
                Sq=sigma_t;
                y=[find(z_d(:)>mean_t+3*sigma_t); find(z_d(:)<mean_t-
3*sigma_t)];
                m1=z_d(:,1:p);
                m2=z_d(:,p+1:end);
                th_avg=sum1/2; %th_avg (threshold average) is the
average of the first 2 surfaces, to determine the threshold
                th_up=mean_t+3*sigma_t;
                th_down=mean_t-3*sigma_t;

                for j=1:length(y)
                    l=ceil(y(j)/p);
                    k=y(j)-p*(l-1);
                    if l>p
                        l=l-p;
                    end
                    if th_avg(k,l)>mean_t && abs(th_avg(k,l))>(1-
Sq)*abs(m1(k,l))&& abs(th_avg(k,l))<(1+Sq)*abs(m1(k,l)) %means that it
comes from the instrument
                        th_1=th_avg(k,l);
                        if th_1<th_up
                            th_up=th_1;
                        end
```

```
                    elseif th_avg(k,l)<mean_t && abs(th_avg(k,l))>(1-
Sq)*abs(m1(k,l))&& abs(th_avg(k,l))<(1+Sq)*abs(m1(k,l)) %means that it
comes from the instrument
                        th_2=th_avg(k,l);
                        if th_2>th_down
                            th_down=th_2;
                        end
                    end

                end

                %% FOR THE FIRST 2 SURFACES
                %eliminate spurious data  of the first two
measurements
                m1(m1>th_up|m1<th_down)=NaN;
                m2(m2>th_up|m2<th_down)=NaN;
                %Identify NA from Mountains and make them NaN
                NA1=min(m1(:)); % because NA from mountains are
established as a same min value
                m1(m1==NA1)=NaN;
                NA2=min(m2(:)); % because NA from mountains are
established as a same min value
                m2(m2==NA2)=NaN;
                if NA1>0.05
                    msgbox('The 1st topography contains more than 5%
NaN and may lead to wrong results.');
                end
                if NA2>0.05
                    msgbox('The 2nd topography contains more than 5%
NaN and may lead to wrong results.');
                end

                %add polynomial
                m1=(m1+p1);%-mean(nanmean(m1+p1)); %bring back form &
remove mean
                m2=m2+p2;%-mean(nanmean(m2+p2)); %bring back form &
remove mean

                %average the first 2 surfaces
                ms=[m1,m2];
                z_data=[];
                for s=1:2
                    m=ms(:,p*(s-1)+1:p*s);
                    mask=zeros(size(m));
                    m=level(m,mask); %level surface
                    m=m-nanmean(m(:));%remove mean
                    x=find(isnan(m));
                    m(isnan(m))=0;%change NaN to 0
                    d0=d;
                    for j=1:length(x)
                        l=ceil(x(j)/p);
                        k=x(j)-p*(l-1);
                        %m(k,l)=0;
                        d(k,l)=d(k,l)-1; %recalculate number of
measurements at this pixel
                    end
                    z_data=[z_data m];
```

```
            end
            sum=z_data(:,1:p)+z_data(:,p+1:end);%m1+m2
            %sum=m1+m2;
            average=sum./d;
            Szs=max(average(isfinite(average(:))))-
min(average(isfinite(average(:)))) %calculate Sz

        end


    else %after the first 2 measurements calculation
        %I'm putting the thresholded values as mean instead of NaN
        %because level.m can't work with NaN. Because it is a
levelled
        %(after form removal) flat, it should affect too much.
        z_data(z_data>th_up|z_data<th_down)=NaN;%mean(z_data(:));
        %Check NaN
        %NaNs=length(find(isnan(z_data(:))));
        NaNs=length(z_data>th_up|z_data<th_down);

        %ADD POLYNOMIAL
        z_data=(z_data+pol_mat);%-mean(nanmean(z_data+pol_mat));
%bring back form & remove mean
        %Identify NA from Mountains and make them NaN
        NA=min(z_data(:)); % because NA from mountains are
established as a same min value
        z_data(z_data(:)==NA)=NaN; %not sure how mean affects here
        x=find(isnan(z_data)); %identify NaN
        z_data(isnan(z_data(:)))=mean(z_data(:)); %level can't
work with NaNs
        %LEVELLING

        mask=zeros(size(z_data));
        z_data=level(z_data,mask); %level surface
        z_data=z_data-nanmean(z_data(:)); %subtract mean
        %AVERAGING
        z_data(z_data(:)==mean(z_data(:)))=0;
        %z_data(isnan(z_data))=0; %change NaN to 0
        d0=d;
        d=d+1;
        for j=1:length(x)
            l=ceil(x(j)/p);
            k=x(j)-p*(l-1);
            d(k,l)=d(k,l)-1; %recalculate number of measurements
at this pixel
        end
        sum=sum+z_data;
        %     end

        average=sum./d; %calculate average surface
        average(average==0)=NaN;
        Sz=max(average(isfinite(average(:))))-
min(average(isfinite(average(:)))) %calculate Sz
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        Szs=[Szs Sz];

        if length(Szs)>5
```

```
                max1=max([Szs(end-4),Szs(end-3),Szs(end-2),Szs(end-
1),Szs(end)]);
                min1=min([Szs(end-4),Szs(end-3),Szs(end-2),Szs(end-
1),Szs(end)]);
                max2=max(Szs);
                min2=min(Szs);
                fraction=(max1-min1)/(max2-min2);

                if NaNs<0.05*p^2  %less than 5% of NaN (ISO 25178-700
27/02/2017)
                    if fraction<0.05 %decreases and stable -> finish
                        flag=1;
                        break
                    else %not stable-> add measurement
                        %                   Sz0=Sz;
                        meas=meas+1;
                        %d=d+1; think I have to delete
                    end
                else % too many NaN -> remove
                    sum=sum-z_data;
                    d=d0;
                    Szs=Szs(1:end-1);
                    %Sz=Sz0; %is it used for something? wrong? remove?
in case of break, to update Sz.
                    msgbox('This surface contains more than 5% of non-
valid points. This surface will not be considered.');
                end
            end
            if flag==1
                break
            end
        end
    end
end
plot(1:length(Szs),Szs,'-*')
msgbox( sprintf('Zflt=%1.3d um\n',Sz));
```

A.3 Amplification and linearity of the vertical scale

```
%%
%This script calculates the amplification and linearity of the z-
scale.
%Measurement uncertainties are calculated, considering the
contributions
% of the traceability, errors, repeatiility and reproducibility

%The calibrated values of the grooves are to be ordered from less to
more
%depth.

%To check the reproducibility, different studies can be performed at
%different positions in the vertical scale of the instruments ->
levels
%To check repeatibility, the measurement of a groove is repeated
several
%times -> cicles
```

```
%The measurements should be ordered as follows:
%Level1
%  -Groove 1
%    - cycle 1
%    - cycle 2
%    -...
%  -...
%...

%Units: um.

%Two binary masks are applied: one for levelling, one for feature
%identification.

%%
clear all, close all, clc
cycles=3; % number of repeated measurements of each groove
grooves=6; %number of grooves (different height)
linearities=[];
amplifs=[];
repeatibilities=[];
dep_reprod=[];

%%
%READ DATA FROM FILES
directory='C:\Users\Patricia
Giménez\Documents\CRANFIELD\THESIS\Coding\Noise data 05062017';
% prompt = {'Enter the directory of the files:'};
% dlg_title = 'Directory';
% num_lines = 1;
% directory = strjoin(inputdlg(prompt,dlg_title,num_lines));
direc=[directory '/*.sur'];
names=dir(fullfile(direc));
list_names={names.name};
levels=1; %vertical range of the machine

depths_g=[];
sd_g=[];
for k=1:levels %reproducibility
    for i=1:grooves
        depths=[];
        sum_z=0;
        counter=0;
        for j=cycles*i-(cycles-1):cycles*i %repeatibility
            file_name_cell_1=list_names(j);
            file_name_1=strjoin(file_name_cell_1);
            [z,a]=SUR_read2_z(file_name_1);
            z=a*z;
            p=length(z);
            filex=1:p;

            %%
            %LEVEL AND FIND EDGES

            gimg=imgradient(z); %gradients matrix
            mask=zeros(size(z)); %create base of 0s
            thresh=mean([max(gimg),min(gimg)]); %define mask threshold
```

```
            mask(gimg>thresh)=1; %make 1s
            mask = bwareaopen(mask,500); %delete spurious points out
of edges
            z=level(z,mask); %level

            gimp=imgradient(z); %recalculate for levelled image
            mask=zeros(size(z));
            thresh=mean([max(gimg),min(gimg)]);
            mask(gimp>thresh)=1;
            mask = bwareaopen(mask,500);
            %         mask=imclearborder(mask);

            edg=edge(mask); %find edges
            list_edges=find(edg==1);
            xr=[]; yr=[];
            if isempty(length(list_edges))==0
                for j=1:length(list_edges)
                    l=ceil(list_edges(j)/p);
                    k=list_edges(j)-p*(l-1);
                    xr=[xr l];
                    yr=[yr k];
                end
            end


            clear rectx recty area perimeter
            [rectx,recty,area,perimeter] = minboundrect(xr,yr);
            if (rectx(2)-rectx(1))>(recty(4)-recty(1)) %check
orientation
                orientation=0; %the longest side of the groove is
horizontal
            else
                orientation=1; %the longest side of the groove is
vertical
            end
            angle=atan(abs(recty(1)-recty(2))/(abs(rectx(1)-
rectx(2))))*180/pi; %groove angle

            %MAIN CODE
            %%
            if orientation==1
                length_groove=floor(recty(3)-recty(2));
                width_groove=ceil(mean([rectx(2)-rectx(1), rectx(3)-
rectx(4)]));

            else
                length_groove=floor(rectx(3)-rectx(4));
                width_groove=ceil(mean([recty(4)-recty(1), recty(3)-
recty(2)]));
            end

            clear filez
            if orientation==1
                for i=ceil(recty(3)-0.3*length_groove):-
20:floor(recty(3)-0.7*length_groove)
                    filez=z(i,:);
                    filez = filez - mean(filez);
```

```
                sum_z=sum_z+filez;
                counter=counter+1;
            end
            filez=sum_z/counter; %averaged profile
        else
            for i=ceil(rectx(2)-0.3*length_groove):-
20:floor(rectx(2)-0.7*length_groove)
                filez=z(:,i);
                filez = filez - mean(filez);
                sum_z=sum_z+filez;
                counter=counter+1;
            end
            %              filez=filez'; %CHECK THIS
            filez=sum_z/counter;
        end

        %%
        % Mean-centre data.
        %
        filex = filex - mean(filex);

        %     end %%%%%%%%%%%%%%%%

        % Find the start and end point of the data.
        %
        % hold off; %modified from: hold('off');
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             figure
%             plot(filex,filez,'b.','MarkerSize',20);
%             xlabel('x displacement/um');
%             ylabel('z displacement/um');
% %             [x,y] = ginput(2); %commented by PGB
%             hold on; %modified from: hold('on');
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        %%
        %Added by PGB on 30/06/2017
        %FIND EACH SIDE OF THE GROOVE

        pos1=0;
        for i=1:1024
            if filez(i)<0 && pos1==0
                pos1=i;
            elseif filez(i)>0 && pos1~=0
                pos2=i;
                break
            end

        end
        x=[filex(pos1); filex(pos2)];
        %End of added by PGB on 30/06/2017
        %%

        % Scale x-data, including start and end-points.
        %
        scale = max(abs(filex));
        filex = filex/scale;
```

```
            x     = x/scale;
            %
            % Calculate the areas A, B and C stipulated by ISO 6394.
            %It's ISO 5436-1 by PGB on 17/07/17
            %
            W  = x(2) - x(1);
            L  = length(filex);
            xc = (x(1) + x(2))/2;
            %
            % Limits for area C ...
            %
            xLL  = xc - W/6;
            xLR  = xc + W/6;
            %
            % Limits for area A ...
            %
            xUL1 = xc - (3*W/2);
            xUR1 = xc - (5*W/6);
            %
            % Limits for area B ...
            %
            xUL2 = xc + (5*W/6);
            xUR2 = xc + (3*W/2);
            %
            % Data in area C ...
            %
            pointer = 1;
            clear xL zL
            for i = 1:L;
                if filex(i)>xLL && filex(i)<xLR %modified from: if
and((filex(i)>xLL),(filex(i)<xLR));
                    xL(pointer) = filex(i);
                    zL(pointer) = filez(i);
                    pointer = pointer + 1;
                end;
            end;
            xL = xL';
            zL = zL';
            %
            %Remove flatness
            %Added by PGB on 01/07/2017 -- Aparently don't need to in
C
            % polC=polyfit(xL,zL,12);
            % polyC=polyval(polC,xL);
            % zL=zL-polyC+mean(zL);
            %End of added by PGB on 01/07/2017

            %
            % Data in areas A and B ...
            %
            pointer = 1;
            clear  xU zU
            for i = 1:L;
                if (filex(i)>xUL1 && filex(i)<xUR1)||(filex(i)>xUL2 &&
filex(i)<xUR2)
                    %modified from: if
or((and((filex(i)>xUL1),(filex(i)<xUR1))),(and((filex(i)>xUL2),(filex(
i)<xUR2))));
                    xU(pointer) = filex(i);
```

```
                zU(pointer) = filez(i);
                pointer = pointer + 1;
            end;
        end;
        xU = xU';
        zU = zU';

        %Remove flatness
        %Added by PGB on 01/07/2017
        polAB=polyfit(xU,zU,12);
        polyAB=polyval(polAB,xU);
        zU=zU-polyAB+mean(zU);
        %End of added by PGB on 01/07/2017
        %
        % Plot data.
        %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             plot(scale*xL,zL,'g*','MarkerSize',20);
%             hold on %added
%             plot(scale*xU,zU,'r*','MarkerSize',20);
%             hold on %added
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %
        % Calculate least-squares best-fit parallel lines.
        %
        A = [ xL, ones(size(xL)), -ones(size(xL)); xU,
ones(size(xU)), +ones(size(xU)) ]; %%%%Check if that's working
normal%%%
        b = [ zL; zU ]; %modified
        fit = A\b;
        %
        % Plot lines.
        %
        low1 = filex(1)*fit(1) + fit(2) - fit(3);
        low2 = filex(L)*fit(1) + fit(2) - fit(3);
        lowx = [filex(1),filex(L)];
        lowz = [low1,low2];
        up1  = filex(1)*fit(1) + fit(2) + fit(3);
        up2  = filex(L)*fit(1) + fit(2) + fit(3);
        upx  = [filex(1),filex(L)];
        upz  = [up1,up2];
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             plot(scale*lowx,lowz,'g','LineWidth',3);
%             hold on %added
%             plot(scale*upx,upz,'r','LineWidth',3);
%             hold on %added
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %
        % Plot residuals associated with fitted model.
        %
        % figure
        % plot(b - A*fit, 'k.')
        % ylabel('Residual errors/nm')
        %
        % Calculate depth.
        %
        depth = 2*fit(3); %depth of the profile
        depths=[depths depth]; %depths of the profiles of a same
groove
```

```
        end
        depth=mean(depths); %averaged depth
        repeat=std(depths); %repeatibility: std of the depth at each
groove
        depths_g=[depths_g depth];
        repeatibilities=[repeatibilities repeat];
    end
dep_reprod=[dep_reprod mean(depths_g)];
%Calibrated values may need to be read from some file
calibrated=[30 100 200 500 1000 2000]*1e-3; %calibrated values in um

figure, plot(depths_g, calibrated, 'b--o','LineWidth',4);
p=polyfix(depths_g,calibrated,1,0,0);
lin_fit=p(1)*depths_g+p(2);
hold on, plot(depths_g,lin_fit,'r','LineWidth',4), hold on,
plot(calibrated, calibrated,'g','LineWidth',4)
amplification=p(1)

dif=abs(lin_fit-calibrated);
linearity=max(dif)

%Plot errors
err1=depths_g-calibrated;
figure, errorbar(depths_g,err1,repeatibilities,'o')
err2=lin_fit-calibrated;
figure, errorbar(depths_g,err2,repeatibilities,'o')

linearities=[linearities linearity];
amplifs=[amplifs amplification];
end



% %Calculate uncertainties
% utrac=0; %data --> UPDATE!
% urepeat= max(repeatibilities);
% ureprod=std(dep_reprod);
% %ureprod=max([std(linearities), std(amplifs)]);
% uerr=max(abs(err1)+repeatibilities); %just invented it, CHECK if
that's it
% %uerr=sumsqr(err1)/3;
% uz=sqrt(utrac^2+urepeat^2+ureprod^2+uerr^2)

figure, plot(calibrated,err1,'-*')

% msgbox( sprintf('amplif=%1.3d um\nlz=%1.3d um\nu_trac=%1.3d
um\nu_repeat=%1.3d um\nu_reprod=%1.3d um\nu_err=%1.3d um\n\nuz=%1.3d
um\n',amplification,linearity,utrac,urepeat,ureprod,uerr,uz));

C{1} = '     CT(um)    M (um)       e (um)     Tr (um)      R (nm)    u
(um)' ;
C{2} = sprintf('1    %1.3f        %1.3f          %1.3f        %1.3f
%1.3f     %1.3f', calibrated(1), depths_g(1), err1(1),utrac,
repeatibilities(1)*10e3, sqrt( err1(1)^2+utrac^2+
repeatibilities(1)^2)) ;
```

```
C{3} = sprintf('2    %1.3f        %1.3f          %1.3f        %1.3f
%1.3f     %1.3f', calibrated(2), depths_g(2), err1(2),utrac,
repeatibilities(2)*10e3,sqrt( err1(2)^2+utrac^2+
repeatibilities(2)^2)) ;
C{4} = sprintf('3    %1.3f        %1.3f          %1.3f        %1.3f
%1.3f     %1.3f', calibrated(3), depths_g(3), err1(3),utrac,
repeatibilities(3)*10e3,sqrt(err1(3)^2+utrac^2+ repeatibilities(3)^2))
;
C{5} = sprintf('4    %1.3f        %1.3f          %1.3f        %1.3f
%1.3f     %1.3f', calibrated(4), depths_g(4), err1(4),utrac,
repeatibilities(4)*10e3,sqrt( err1(4)^2+utrac^2+
repeatibilities(4)^2)) ;
C{6} = sprintf('5    %1.3f        %1.3f          %1.3f        %1.3f
%1.3f     %1.3f', calibrated(5), depths_g(5), err1(5),utrac,
repeatibilities(5)*10e3,sqrt(err1(5)^2+utrac^2+ repeatibilities(5)^2))
;
C{7} = sprintf('6    %1.3f        %1.3f          %1.3f        %1.3f
%1.3f     %1.3f', calibrated(6), depths_g(6), err1(6),utrac,
repeatibilities(6)*10e3,sqrt( err1(6)^2+utrac^2+
repeatibilities(6)^2)) ;
msgbox(C)
```

A.4 Amplification, linearity and perpendicularity of the horizontal scale

```
%%
%This script calculates the amplification, linearity and
perpendicularity of the x-and-y-scales.

%Patricia Giménez Belando
%Last update: 31/08/2017
%Units: um.

%Two binary masks are applied: one for levelling, one for feature
%identification.
%%
clear all, close all, clc
cycles=1; % number of repeated measurements of each grid
% p=1024;
% filex=1:1024;
grids=5;
CX=[];
Ix=[];
wX=[];
sum=0;

%%
%READ DATA FROM FILES
directory='C:\Users\Patricia
Giménez\Documents\CRANFIELD\THESIS\Coding\Noise data 05062017';
% prompt = {'Enter the directory of the files:'};
% dlg_title = 'Directory';
% num_lines = 1;
% directory = strjoin(inputdlg(prompt,dlg_title,num_lines));
```

```
direc=[directory '/*.sur'];
names=dir(fullfile(direc));
list_names={names.name};
grids=length(list_names);

for i=1:grids
for j=cycles:cycles%*i-2:cycles*i %check this. why -2?
        centroids=[];
        file_name_cell_1=list_names(i)
        file_name_1=strjoin(file_name_cell_1);
        [z,a]=SUR_read2_z(file_name_1);
        z=a*z;
        z=double(z);

        %%
        %LEVEL AND FIND EDGES

        gimg=imgradient(z); %gradients matrix
        mask=zeros(size(z)); %create base of 0s
        thresh=mean([max(gimg),min(gimg)]); %define mask threshold
        mask(gimg>thresh)=1; %make 1s
        mask = bwareaopen(mask,500); %delete spurious points out of
edges
        z=level(z,mask); %level
%         figure, surf(z,'EdgeColor','none')

        gimg=imgradient(z); %recalculate for levelled image
        mask=zeros(size(z));
        thresh=mean([max(gimg),min(gimg)]);
        mask(gimg>thresh)=1;
        mask = bwareaopen(mask,500);

cent=regionprops(mask,'Centroid'); %find centroids
centroids = cat(1, cent.Centroid); %put centroids in a matrix
CX=[CX;centroids];
Ix = [Ix;[1:length(centroids)]' (i*ones(1,length(centroids)))' ];
wX = [wX;ones(1,length(centroids))'];
% wX =
[wX;ones(1,length(centroids))',ones(1,length(centroids))',ones(1,lengt
h(centroids))'];

% figure, plot(centroids(:,1),centroids(:,2),'r.','MarkerSize',20)
% hold on, contour(mask,'k')
% hold on, plot(x,y,'c.','MarkerSize',20)

end
end
wX=[wX wX];

% CX=[CX, ones(length(CX),1)];
[Y,UY,X0,R0,f,F,sigmah,SS,SB,S0]=ls2dptm(CX,Ix,wX); %Least squares
point matching in 2d. Alineate the measurements.
nP=length(centroids); %number of pores in the grid

%To average, it is necessary to match every pore with his
corresponging
```

```
%pore in the other grids. Below, all data is ordered in ascendent y.
Then
%it's grouped on sets of the pores in a row. For every row, the pores
are
%ordered in ascendent x.
CX_final=[];CX_final_all=[]; sum=0;
for i=1:grids %grids? cycles?
    X01=repmat(X0(i,:),nP,1);
    CX((i-1)*nP+1:i*nP,:)=(R0((i-1)*2+1:2*i,:)'*CX((i-
1)*nP+1:i*nP,:)'+X01')'; %traslate & rotate
    CX_g=CX((i-1)*nP+1:i*nP,:);
    CX_y_o=sortrows(CX_g,2) %order CX_g in ascendent y
    CX_final=[];
    for j=1:sqrt(nP)
        CX_row=CX_y_o((j-1)*sqrt(nP)+1:sqrt(nP)*j,:)
        CX_row_ordered=sortrows(CX_row,1);
        CX_final=[CX_final; CX_row_ordered]; %ordered CX for one grid
    end
        CX_final_all=[CX_final_all; CX_final];
        sum=sum+CX_final;
end

x_coordinates=reshape(CX_final_all(:,1),nP,grids);
y_coordinates=reshape(CX_final_all(:,2),nP,grids);
figure, histogram(std(x_coordinates')), title('std x')
figure, histogram(std(y_coordinates')), title('std y')

CX_avg=sum/grids; %average of the position of the centroids of the
measurements
%figure, plot(CX_avg(:,1),CX_avg(:,2),'.','MarkerSize',20),hold on,
plot(CX(1:256,1),CX(1:256,2),'.','MarkerSize',15)


% %%%%%%%%%%%%%%%%%%%%%
%INVENTED
% x=32:124:2048;
x=[94:124:2048];
%x=[x x x x x x x x x x x x x x x x x];
x=[x x x x x x x x x x x x x x x x ];
y=[];
for i=94:124:2048%32:124:2048
    y1=i*ones(1,16);%(1,17);
    y=[y, y1];
end
CX_cal=[x' y'];
% %%%%%%%%%%%%%%%%%%%%%
%
% CX_cal=CX(1025:end,:); %TRIAL. DELETE
CX2=[CX_cal;CX_avg];
Ix21 = [[1:nP]' (ones(1,nP))'];
Ix22= [[1:nP]' (2*ones(1,nP))' ];
Ix2=[Ix21;Ix22];
wX2=ones(2*nP,2);
[Y,UY,X02,R02,f,F,sigmah,SS,SB,S0]=ls2dptm(CX2,Ix2,wX2); %Least
squares point matching in 2d. Alineate the measurements.
X02=repmat(X02(2,:),nP,1);
%CX_comp=CX2(nP+1:2*nP,:)+X02; %traslate
CX_comp=R02(3:4,:)'*CX2(nP+1:2*nP,:)'+X02'; %rotate
```

```
CX_final2=[];CX_final_all2=[];
for i=1:2 %grids? cycles?
    X012=repmat(X02(i,:),nP,1);
    CX2((i-1)*nP+1:i*nP,:)=(R02((i-1)*2+1:2*i,:)'*CX2((i-
1)*nP+1:i*nP,:)'+X012')'; %traslate & rotate
    CX_g=CX((i-1)*nP+1:i*nP,:);
    CX_y_o=sortrows(CX_g,2) %order CX_g in ascendent y
    CX_final2=[];
    for j=1:sqrt(nP)
        CX_row=CX_y_o((j-1)*sqrt(nP)+1:sqrt(nP)*j,:)
        CX_row_ordered=sortrows(CX_row,1);
        CX_final2=[CX_final2; CX_row_ordered]; %ordered CX for one
grid
    end
        CX_final_all2=[CX_final_all2; CX_final2];
end

CX_cal=CX_final_all2(1:nP,:);
CX_comp=CX_final_all2(nP+1:end,:);


errors=CX_comp-CX_cal;
figure, histogram(errors), title('errors');
CX_err=CX_comp+errors*50;
figure, plot(CX_cal(:,1),CX_cal(:,2),'.','MarkerSize',30),hold on,
plot(CX_err(:,1),CX_err(:,2),'.','MarkerSize',30)

figure, plot(CX_cal(:,1),CX_cal(:,2),'r.','MarkerSize',30)
hold on, plot(CX_comp(:,1),CX_comp(:,2),'c.','MarkerSize',20)
```