Development of a Portable Circular Texture Meter for Road Texture Depth Measurement

ADRIAN MOLDOVANU

A thesis submitted to Auckland University of Technology in partial fulfilment of the requirements for the degree of Master of Engineering (ME)

2015

Faculty of Design and Creative Technologies
School of Engineering

Abstract

Chipseal is the most common road surface in New Zealand due to its low cost and high performance monitoring. One of the parameters surveyed is the road texture depth which is currently measured by a volumetric technique known as the circular sand patch method, a method that is slow to perform, requires traffic control and depends on operator skill. The objective of our research is to design and build a portable device able to estimate the texture depth and evaluate its perfomance by correlating the results with the depths obtained with the sand patch method.

The method chosen falls under the circular texture meters (CTM) category where a surface profile is acquired using a laser displacement sensor mounted on an arm that is rotated around an axis. The mechanical design follows the specifications from current practice established by a number of standards in Australia, USA and Europe. Profile processing is realized on board by a purpose built embedded system the results are available to the operator on an LCD display.

Parameters used in estimation of texture depth are the standardized mean profile depth (MPD) and three parameters borrowed from surface metrology: average roughness (Ra), average square roughness (Rq) and average maximum height (Rz) which was redefined to allow for shorter CTM profiles. Algorithms were first developed into a Matlab toolbox then ported to the embedded system. The efficiency of the parameters was determined by the extent of how parameters correlate with texture depth obtained with the sand patch method characterised by the slope α and the intercept β of the linear relation that result in the best r-square for the parameter. The intercept was derived from mean of the parameters measured over a flat surface. The slope was estimated firstly from a test area of one square meter where sand patch texture depths were obtained and a number of profiles were recorded. Secondly, the slope was estimated from tracks acquired from a variety of sites using the correlation method with the intercept being constrained to the value found from the flat surface measurements.

The results show that all parameters have an improved r-square when a second order fit is applied to the profile segments. The slopes estimated with the correlation method are in average 67% percent smaller than the slopes projected with the square meter test. Considering both the coefficient of determination of the correlations and the slope, the circular texture meter returns best estimates from the average maximum height when a second order fit is applied to the profile segments.

Table of Contents

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Date

16 OCT 2015

Adrian Moldovanu

List of Figures

List of Tables

Acknowledgments

Firstly, I would like to express my sincere gratitude to my adviser, Dr Loulin Huang for the suport of my Master study, for his patience, motivation and his guidance through my research. I could not have imagined having a better advisor and mentor for my studies.

Besides my adviser, I would like to thank Mr Kacha Vuletich from Fulton Hogan for his insights and expertise with chipsealing practices that greatly assisted me at the beginning of the research and in the final stage of data correlation.

I thank Mr Jian Huang for his technical support. I would also like to show our gratitude to Dr Andrew Hilton for bridging Fulton Hogan and AUT research programme.

Last but not the least, I would like to thank my family: my wife and son for supporting me through the ups and downs of this journey and my life in general.

1. INTRODUCTION

New Zealand has an extensive network of chip sealed roads whose maintenance necessitates regular inspections that gather information about the state of the pavement. Of particular interest is the level of embedment of the chips into the binding substrate, a parameter known as road surface texture depth (RTD). The current practice of RTD measurement is a volumetric technique known as the circular sand patch, a technique that is slow to perform, requires traffic control and depends on the operator skill.

In recent years, alternative methods to the sand patch have been proposed, taking advantage of the developments in both sensors and computing power being available and affordable. Few of those methods have been standardized and adopted in some countries, maybe the most notable being estimating the RTD from a profile acquired with a laser displacement sensor, known as mean profile depth.

1.1. Research objectives

Our research is focused on developing a portable device which can estimate the RTD quickly and is easy to manipulate. We have chosen to design and construct a circular texture meter as a data acquisition platform and to investigate how a number of surface profile parameters correlate with those obtained from the sand patch method.

The specifications of the device are as follows. It is to measure the RTD between 0.30 and 3.0 millimeters over an area defined by a square with a 400 millimeters edge and return results within 30 seconds in a form that can be read by an operator on a small digital screen; data should be also logged on a permanent storage system for later analysis. The device should be able to withstand minor shaking and damp environments inherent to outdoor activities and be powered by a battery for a day of work.

1.2. Organisation of thesis

The thesis is organized as follows. In Chapter *2. Literature review,* traditional methods of measuring RTD are reviewed and a few of the recent developments of alternative methods, especially those based on displacement sensors are examined. Furthermore, the similarity between surface profiling approaches as used by surface metrology and road profiling techniques are studied.

In Chapter 3. *Hardware*, the physical construction of the device is described. In Chapter *4. Data processing with Matlab,* the details of the custom toolbox we have constructed to investigate different road texture parameters are presented. The firmware developed for the device is described in Chapter *5. Embedded Software.*

In the first section of Chapter *6. Results*, a summary of the data collected and how we derived the parameters used to estimate the RTD are described. In the second part of the chapter the correlation between different profile parameters and the sand patch method are established.

The results of the project are presented in Chapter *7. Conclusions* where further research directions are also discussed.

Supplementary information like electrical schematics, electronic schematics and PCB layout are provided in *Annex A* and *Annex B*.

2. LITERATURE REVIEW

A chip seal consists of a layer of binder, bitumen, asphalt or tar, usually sprayed over a compacted base course and overlaid with a layer of aggregates providing protection to the binder from tire interaction, water intrusions and other factors (National cooperative highway research program, 2005). It is called a 'surface dressing' in the UK or a 'sprayed seal' in Australia. Lower production costs as well as a simpler method of manufacture makes it an attractive method of road surfacing particularly for low volume traffic or as surface treatment for both low and high volume traffic pavements. The method is widely used New Zealand, Australia, South Africa, the UK and the USA.

Chip sealing was introduced in New Zealand from about 1880 when locally sourced tar and chips were used to produce a dust-proof and water proof surfacing. The process was manual and depended on the skill of the on-site manager and his team and their experience for best practice (Transit New Zealand, 2005). At the beginning of the second quarter of 20th century increased traffic volumes and the development of the automobile required improved road surface quality. A series of test and experiments with aggregates and bitumen were initiated with the most notable result being F.M. Hanson paper *Bituminous surface treatment of rural highways* presented in 1934 at the Conference of the NZ Society of Civil Engineers (Hanson F. M., 1934).

Principles pioneered by Hanson (Figure 1) were later summarised in 1968 by National Roads Board in *Manual of sealing and paving practice* (New Zealand Roading Division; New Zealand Road Research Unit, 1968):

1. *When sufficient chips are placed on a seal binder to ultimately bed into a single layer in shoulder to shoulder contact, the percentage of voids in the initially laid loose state is approximately 50%. This is reduced to about 30% by construction rolling, and to 20% by traffic compaction.*
2. *The amount of binder required bears a definite relationship to the volume of voids in the cover stone aggregate; the quantity should be such that between 65 and 70% of the voids in the finally compacted layer of sealing chips are filled with binder.*
3. *The average depth of the layer of stone chips forming the cover coat, after construction and traffic compaction is approximately equal to the average least dimension (ALD) of the chips used (Transit New Zealand, 2005).*

Since the 1970s, with increased demand from traffic volume and heavier vehicles, the design of chip sealing systems has been further developed, making use of modified binders, multicoat seals etc. and appropriate algorithms established, concluding with the 2005 *Chipsealing in New Zealand* by Transit New Zealand, currently in use.

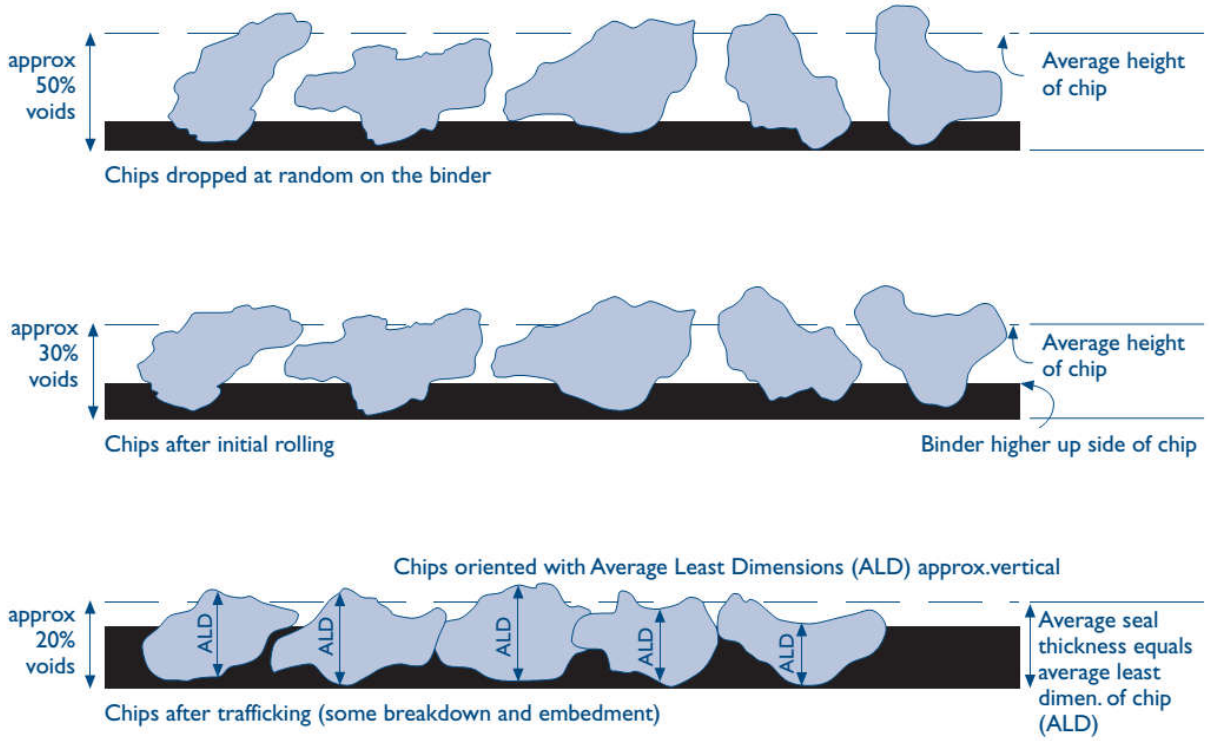


Figure 1. States of embedment of chips in binder (Transit New Zealand, 2005)

2.1. Chipseal texture

Among other road surfaces parameters used in design, the road texture is arguably one of the most used. As defined by World Road Association (WRA) the texture is defined as deviation amplitudes from the surface planes and is divided into four categories.

Microtexture is defined as having deviation amplitudes of 0.5 mm or less and is due mainly to the crystalline surface of the aggregates or the small particles in asphalt.

Macrotexture ranges in amplitudes less than 50 mm from the surface plane, down to microtexture size. It is affected by the size, shape and the spacing of coarse aggregate

particles in the surfacing or by the connection between surface and internal pores in the materials.

Megatexture and *roughness* are formations over 50 mm.

It should be noted that there are differences in the definition of macrotexture for New Zealand, where macrotexture is defined by *Chipsealing in New* Zealand as the texture of the seal that can be seen with the eye with dimensions between 0.5 mm to 10 mm (Transit New Zealand, 2005) and Australia where, the macrotexture is defined as wavelengths between 0.5 and 50 mm and includes beside the aggregates size and shape the presence of purpose made grooves or the connections between surface and internal pores in the materials (Austroads Inc, 2009). This definition is also widely adopted by other countries.

Macrotexture is further divided in two categories: positive texture, formed by aggregate protruding above the plane of the surface and negative texture when the texture of the chipseal is largely formed by voids between particles whose upper surfaces from the generally flat plane of the road surface (Aktas, Gransberg, Rienner, & Pittenger, 2011).

Road texture has an important role in chipseal design, road maintenance, skid resistance and performance indicators.

2.2. Measuring the road texture depth (RTD): traditional methods

2.2.1. Sand Patch Method

By far the most common method of measuring the RTD is a volumetric method known as the sand patch method (National cooperative highway research program, 2005). The method is described by Transit New Zealand as follows: the surface to be tested is cleared of debris and dry; if necessary the surface is brushed of any fine material. A cylinder with a calibrated internal volume of 45 ml (±0.5 ml) is filled with clean dry sand with well-rounded grains, 100% passing the 600 μm and 100% retained on the 300 μm BS410 test sieves (Transit New Zealand, 1981). The sand is then poured in a conical shape in the middle of the area measured. Using a straight edge, the sand is spread into a circular patch so that the surface depressions are filled to the level of the top of the chips. The diameter of the circle, D, is measured twice, the direction of the second measure at right angle to the first. The average Mean Texture Depth (MTD) is defined as the volume of sand divided by the area of the sand patch:

$$Average\ texture\ dept\ = \frac{Volume}{Area} \qquad Eq.\ 1$$

Or

$$MTD = \frac{4V}{\pi D^2} \qquad Eq.\ 2$$

For the known volume of 45 ml, the equation above becomes

$$MTD = \frac{57300}{D^2} mm \qquad Eq.\ 3$$

The sand circle diameter is reported to the nearest 5 mm and the average depth to the nearest 0.1 mm (Transit New Zealand, 1981). The method is suitable for texture depths greater than 0.45 mm or 350 mm sand circle diameter.

Figure 2. Sand patch method

The method is similar in other countries. In Australia the sand grain used is specified between 150 μm and 300 μm, with the volume of sand chosen from multiples of 50 ml such that the sand patch has a minimum diameter of 170 mm. The diameter of the sand patch is measured four times at equally spaced intervals and is reported to the nearest 1 mm (Mainroads Western Australi, 2012; Autroads, 2008). Another variation on the method is used in the USA where single size glass spheres are used instead of sand (ASTM International, 2015). Generally, the minimum reported depth by this method is 0.25 mm.

It should be noted that while the sand patch method is simple and has reduced equipment costs, it has some shortcomings: it is slow, requires traffic control and causes concerns

about operator safety. The measurement results are dependent on the operator skill (Austroads, 2008).

The method has a nonlinear relation between the measured diameter with a large variation in diameter required to record a 0.1 mm variance in mean texture depth. For example, for a 50 ml volume, a 460 mm diameter circle will yield a texture depth of 0.3 mm while for the next reported depth a 400 mm circle will be produced.

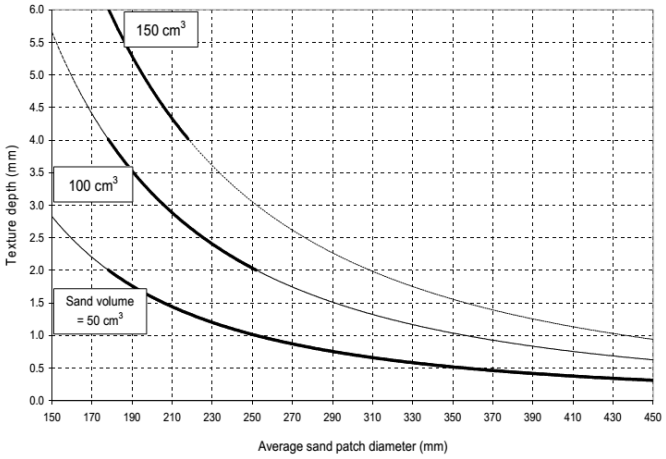


Figure 3. Relation between sand patch diameter and texture depth (Autroads, 2008)

Literature surveys (Aktas, Gransberg, Rienner, & Pittenger, 2011; Pidwerbesky, Gransberg, & Stemprok, 2006) reveals a poor reproducibility of the sand patch test, about 40%, with the method prescribed in New Zealand producing the best results due to the sand volume being about twice the volume of a coarser gradation of sand. The same sources point out that is difficult to get consistent sand patch tests in windy conditions regardless of the measures employed to break the wind from blowing a significant portion of the sampling sand.

2.2.2. Silicone Putty Method

For textures with average depths less than 0.5 mm an alternative technique was developed by Texas Transportation institute in 1970 (Department of Transport and Main Roads, 2012). The method involves placing a sphere of 12.82 or 25.64 ml silicone putty in the centre of a recessed plate and pressing it with a 25 kg weight for a determined period of time. The diameter of the flattened putty is measured to the nearest 1 mm at 4 equally spaced diagonals. The texture depth is calculated as

$$STD = \frac{4V \quad 1000}{\pi D^2} \quad 1.587 \qquad \textit{Eq. 4}$$

2.2.3. Outflow Meter Test

The method is based on drainage capability of the pavement through its surface and subsurface voids and is specified by American Society of Testing and Materials standard E2380. A purpose built device comprising of a cylinder, a plunger, a timer, two electronic float sensor and a circular rubber ring is used to measure the time required for the water to flow freely, under gravity pull, between the two sensors. The result is recorded in seconds. A faster escape time indicates a higher texture depth. However, the results must be compared by the operator with other factors such as aggregate type, consistency of texture, grade etc. The relation between texture depth and outflow time is given by (ASTM, 2015)

$$MTD = \frac{3.114}{OFT} + 0.636 \qquad \textit{Eq. 5}$$

Where MTD is mean texture depth (mm) and OFT is outflow time (seconds).

2.3. Laser profile techniques

2.3.1. Mean Profile Depth

Over the past two decades, laser technology has been used to evaluate macrotexture resulting in a number of systems capable of computing various surface profiles. Over the next section, a brief description regarding the methods used in these systems is provided.

The first method is *time of flight measurements* where a laser emitting diode sends a pulsed signal which is reflected by the measured surface. The time which the light pulse requires to travel to the object and back is measured and evaluated. The distance is calculated from the time and the speed of light. The second method is based on triangulation. A light beam is projected to the measured surface and the reflection through the sensor optics is detected by an array of light sensitive elements. As the distance from the sensor to the object changes, the depicted light spot on the array changes thus the distance can be determined very precisely for small displacements.

The *mean profile depth* (MPD) is obtained by using a high frequency laser sensor to produce a data profile at 1 mm intervals or less, typically over a 100 mm section

(Austroads, 2008). A zero mean profile is obtained by subtracting a linear regression of the segment. The profile is divided into two segments and the highest peak in each segment is determined. MPD is the average of these two values (ASTM, 2015).

$$MPD = \frac{1st\ peak\ level + 2nd\ peak\ level}{2} mm \qquad \text{Eq. 6}$$

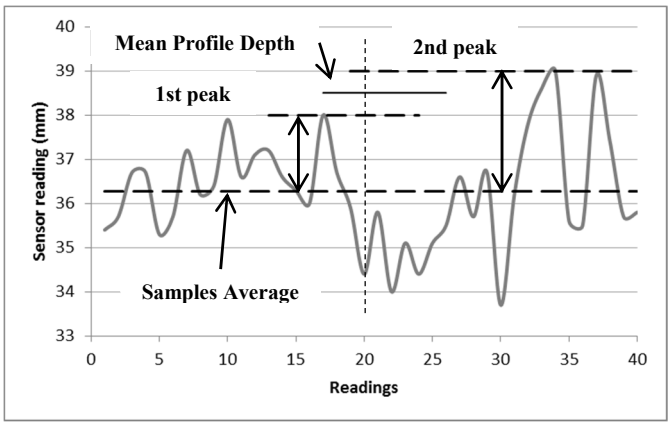


Figure 4 Obtaining MPD from a laser profile

ASTM E1845 presents the following equation for calculating the Estimated Texture Depth from MPD:

$$ETD = 0.8\,MPD + 0.2\,mm \qquad \text{Eq. 7}$$

2.3.2. Circular Texture Meter

In the USA the accepted technique to determine pavement macrotexture is prescribed in the ASTM E2157 (ASTM International, 2015). This method uses a Circular Texture Meter (CTM) where the displacement sensor is mounted on an arm measuring the profile of a circle 284 mm in diameter. The circumference of that is divided into eight segments of about 112 mm with samples spaced at about 0.87 mm. Both mean profile depth and root mean square are calculated. Distinction is made between segments aligned with the direction of travel and segments perpendicular to the direction of travel.

The estimated texture depth obtained from this method is given as (Fisco, 2009)

$$ETD = 0.947\,MPD + 0.069\,mm \qquad \text{Eq. 8}$$

A study conducted by Hanson and Prowell (Hanson & Prowell, 2004) has found consistent results with the slope of the best line fit ranging from 0.93 to 1.01. However Fisco (Fisco, 2009) has found different correlation between estimated texture depth and sand patch test for CTM:

$$ETD = 1.256\,MPD + 0.076\,mm \qquad \text{Eq. 9}$$

2.3.3. Linear Texture

The laboratory system developed by Adams and Kim (Adams & Kim, 2013) uses MPD as a method. Profile depth is acquired using a RoLine line laser produced by LMI Selcom arranged into a gantry type frame. It measures a 97 mm line on pavement surface at 1 mm intervals. The laser then moves 1 mm in transverse direction and obtains another 97 mm line. The profiles in the wheel path are retained and the final MPD value for a section is obtained averaging the individual MPD values from the line segments.

Figure 5. RoLine laser setup (Adams & Kim, 2013)

The method has been used to determine MPD values for field samples, then the samples compared with laboratory tests using a third-scale model mobile loading simulator (MMLS3). The study shows a correlation between the magnitude of MPD obtained and the aggregate loss performance due to chips being removed from the binder leaving the emulsion exposed and resulting in a low MPD value measured after traffic loading.

Aggregates that are not fully embedded in binder early in the life of the chip seal will result in high MPD values.

2.3.4. Sensor Measured Texture Depth

MPD is generally obtained from a high point density profile that requires stationary equipment to be positioned on the surface tested. This raises problems related to traffic management, operator's and road user's safety. An alternative parameter, the *sensor measured texture depth* (SMTD), is described in Austroads Test Method AG:AM/T013 (Austroads, 2007) where a high frequency laser displacement transducer is mounted on a vehicle.

The sensor measured texture depth is derived from a profile obtained from 40 sample points recorded at 7 mm intervals (273 mm). A second order polynomial (y_e) is fitted to the data using a quadratic least square regression. The SMTD is calculated as the root mean square of the residuals (y_i) for one segment. Individual SMTD results are averaged for each 100 m of road sampled.

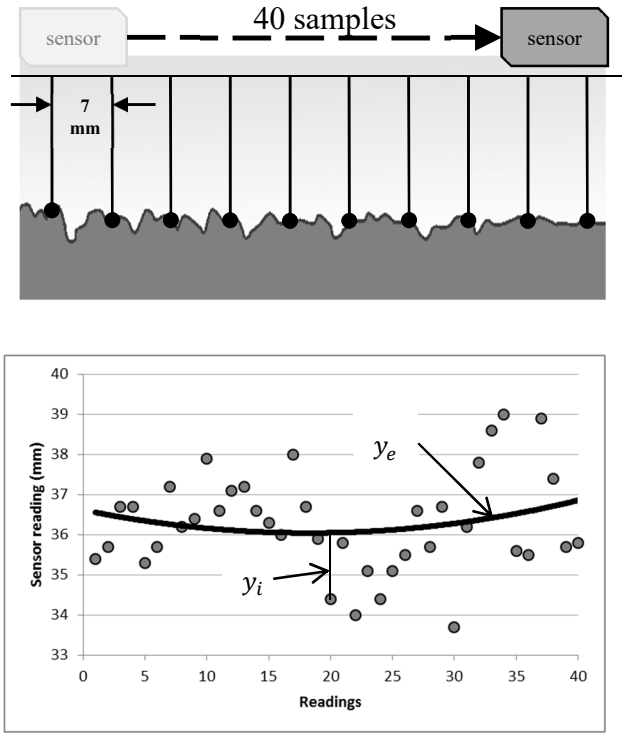


Figure 6. SMTD method

$$SMTD = \sqrt{\frac{\sum_{i=1}^{n}(y_i \quad y_e)^2}{n}} \qquad \text{Eq. 10}$$

where *n* is number of samples.

2.3.5. Correlation with sand patch method

Both MPD and SMTD have been previously shown to have a strong linear correlation with the traditional methods of texture measurement. A review of surface texture measurement methods conducted in 2008 for Austroads shows that SMTD correlates with ETD (Austroads, 2008):

$$ETD = 0.82SMTD + 0.12$$

McGhee and Flintsch (McGhee & Flintsch, 2003) shows that correlation between MPD obtained with a circular texture meter and sand patch obtained mean texture depth varies between test sites:

$ETD = 0.97MPD$	0.04	Wallops site
$ETD = 1.12MPD$	0.06	Smart Road site
$ETD = 1.08MPD$	0.07	Combined data from both sites

Elunai et al. notes that if a texture depth measurement method is developed, its efficiency can be determined by the extent of how the method correlates with the sand patch test method (Elunai, Vinod, & Edith, 2011):

$$ETD = \propto M + \beta \qquad \text{Eq. 11}$$

with values of α and β chosen for the best R^2 characterizing relation between method *M* and *ETD*.

2.4. Alternative Techniques

In addition to the mainstream laser profile methods described above, a number of different approaches to texture measurement can be found in literature. Those can be grouped in mainly 3 categories: image processing, tyre noise analysis and 3D scanning.

2.4.1. Digital image processing

Stereoscopic procedures were developed as early at 1970 by Schonfeld (Austroads, 2008) with the scope of correlating the texture with skid resistance of pavement. The method involves taking photographs of road surfaces at half-scale, permitting 'stereo model' reproduction resulting in accurate vertical measurement between 0.25 to 20 mm. The method was a manual operation, time consuming with traffic control for measurement still required.

Pidwerbesky et al. (Pidwerbesky, Gransberg, & Stemprok, 2006) developed a method employing quantification of the information contained in an image by comparing the value reflected luminance in a given pixel with pixels that surround it. Assuming that a sound and intact texture should contain more edges, thus more information, compared with flushed or stripped seals, the pavement texture can be estimated. They used the fast Fourier transformation (FFT) to automatically quantify the information contained in an image. The algorithm is as follows: the acquired image is converted to a standard 8bit grey levels image (256 levels) and the FFT of the image computed. The frequency components in the FFT are segregated into rectangular frequency band and the sum of the FFT for each band is plotted against the frequency band (Figure 7).

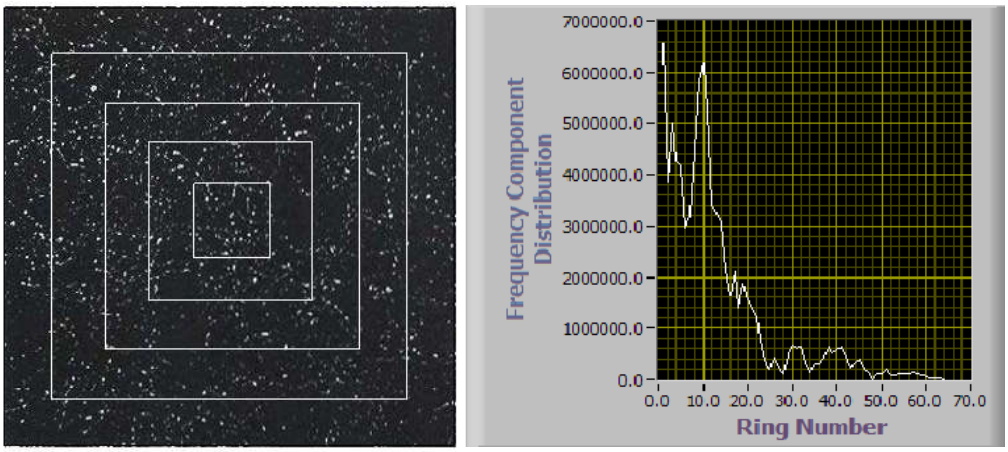


Figure 7 Rectangular frequency bands (left) and corresponding FFT sum plot (right)

The results show a consistency among samples with the same chip seal size but the method fails to estimate texture size across a range of aggregate sizes.

In recent years, development and availability of high resolution cameras made possible acquisition of automated high quality images for texture analysis. Elunai et al. (Elunai, Vinod, & Edith, 2011) investigates a granulometric method based on image processing to estimate road texture coarseness distribution from the edge profiles. Based on the assumption that the average size of texels[1] is proportional with average texture depth, they use an inverse technique to estimate the texel size from a known camera pose represented by camera angle. The average distance between two adjacent edge pixels in the horizontal and vertical direction is used to determine the texture depth. The method correlates well with SMTD however is sensitive to illuminance levels, light source position, image resolution and camera angle.

2.4.2. Tyre noise analysis

It is known that the noise produced by tyre/road contact is directly influenced by texture. Literature survey reveals numerous studies correlating the surface texture and the noise levels (Berge & Viggen, 2014). Starting from this, few studies are investigating the possibility of estimating texture from the tyre vibration and the energy transmission into vehicle passenger compartments. A study by Johnsson and Odelius (Johnsson & Odelius, 2012) shows that measurements of tyre noise and axle acceleration can be used to distinguish new asphalts from worn at speeds of 50 and 70 km/h based on octave bands levels; however at lower speeds (30 km/h) the signal to noise ratio is too low and the method is not suitable. Also the microphone is exposed to weather and wet road conditions thus altering the repeatability of results.

2.4.3. 3D scanning

It can be noted that circular texture meters and the linear method described are essentially a 2D profile parametrization. Recently, in 2012, in Italy, Bitelli et al. (Gabriele Bitelli, 2012) experiments the use of high precision laser scanners to expand the analysis of pavement from the commonly used two dimensional method to a three dimensional one with the aim to extend the range of parameters for these kind of application. The data acquired is a three dimensional profile of cylindrical specimens with a diameter of approximately 15 centimetres. After a preliminary filtering of the raw data to eliminate the outliers, multiple scans are meshed together to create a unique surface. A second

[1] A texel is the fundamental unit of texture space, textures are arrays of texels.

filtering is then applied to correct topological errors (discrepancies between meshes) to reduce redundant points and to avoid gaps and noise. The final result is known as a Digital Surface Model (DSM) structured as a mesh that is representative of that surface.

In their study, Bitelli et al. are using general texture indicators sourced from surface texture metrology to characterize texture: mean profile depth (MPD), average roughness (Ra), levelling depth (Ru), mean depth (Rm), surface roughness depth (Rp) and peak to valley height (Rt) as geometrical indicators. Statistical indicators employed are variance (VAR), average quadratic deviation (Rms), skewness (Rsk) and kurtosis (Rku). Bitelli et al. then proceed to show the use of this technique allows calculation of new surface texture parameters permitting a more complete characterization and opening possibilities of evaluating the pavement surface performances: surface average roughness (Sa) which is similar to Ra. Surface skewness (Ssk) and surface kurtosis (Sku) are obtained from a higher number count and are considered more reliable than their two dimensional counterparts.

Due to the nature of the data obtained, the DMS has been used for volumetric parametrization defined as peak material volume (Vmp), core material volume (Vmc), core void volume (Vvc) and valley void volume (Vvv).

2.5. Summary of modern road texture depth methods

In conclusion, while the potentials of different approaches have been demonstrated, the most reliable methods remain the laser displacement based approaches. The drawback for many methods is the lack of data density in essence, is an extrapolation from a 2D space to a 3D space. Another issue is a relative low number of parameters used to characterise surface texture: mean profile depth and surface measured texture depth. Correlation is made with the accepted sand patch method, which has been shown to have a poor repeatability and the correlation is dependent on device.

2.6. Surface metrology

Muralikrishnan (Muralikrishnan & Raja, 2009) defines the surface finish as “a crucial link between a component, the manufacturing process that generated it, and the functionality that is expected of it. This relationship between surface finish, part functionality, and manufacturing process parameters is the primary reason for the

measurement, characterization, and study of surface texture." The branch of science studying surface texture is surface metrology.

Surface texture, also known as surface topology, is defined by 3 characteristics: lay, roughness and waviness. It is defined by the deviations in the direction of the normal vector from an ideal plane (or form). Lay is the direction of the predominant pattern as given by the tool and method of machining. The roughness is the high frequency deviations while the waviness is made out of low frequency deviations.

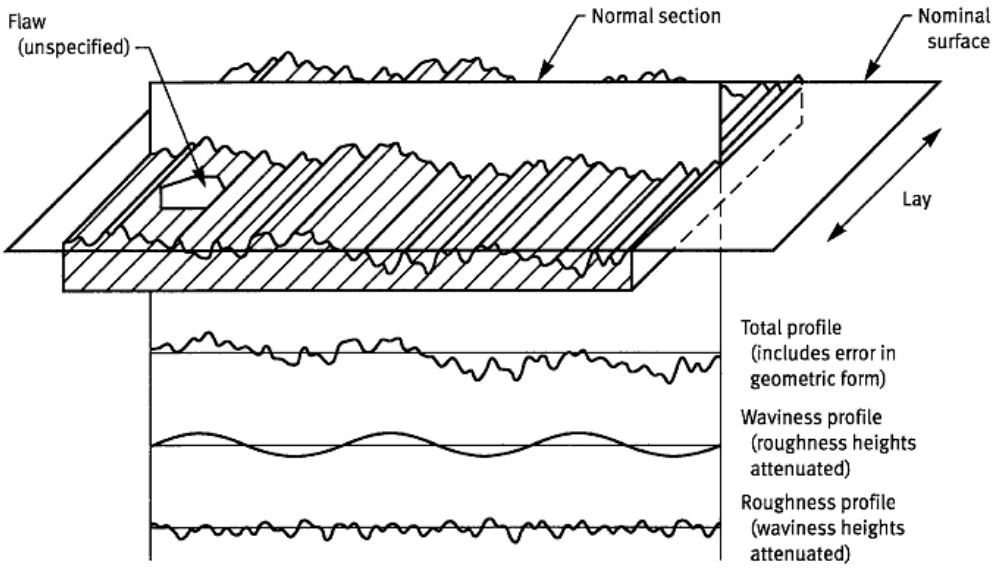


Figure 8. Lay, waviness and roughness (Muralikrishnan & Raja, 2009)

The surface texture profile is acquired by any metrology instrumentation as a discrete data set, or a finite set of spatial coordinates, and then is processed to provide surface parameters. It is recognised that the profile will contain some residual slope or part-induced errors resulting from aligning the measuring instrument with the surface. This slope is removed by a process known as fitting. The next processing step is information extraction from the resulting profile, a technique known as filtering based on the assumption that the profile is made up of wavelengths within certain bandwidths. A third computational process is parametrisation. The purpose of this step is to extract meaningful information from a profile such as amplitude and special characteristics. Last, due computational limits and algorithm used, some uncertainty is propagated from the inputs through to the parameterisation. Uncertainty is not generally analysed by commercial instrumentation but is provided by national metrology labs as a reference.

2.6.1. Fitting

While the surface metrology deals with different shapes, most common being circle, cylinder and sphere, to which profile data needs to be fitted, a pavement profile will be a line or a plane. One of the most common methods to remove the inherent slope is least-squares best fit lines.

Considering x and y two vectors of equal length n, where x_i is holding the coordinate of ith point and $y_i$is the measured height at that point, the best fit line can be defined as $y = a + bx$ where a is the intercept and b is the slope. A function E is defined as the sum of deviations of each ordinate from the line:

$$E = \sum_{i=1}^{n} d_i^2 \qquad \textit{Eq. 12}$$

Where d_i is the deviation function given by $d_i = y_i \quad a \quad bx_i$.

The parameters of the line can be determined by partially differentiating E with respect to a and b.

$$0 = \frac{\partial E}{\partial a} = \quad 2\sum_{i=1}^{n}(y_i \quad a \quad bx_i)$$

$$0 = \frac{\partial E}{\partial b} = 2\sum_{i=1}^{n} \quad x_i(y_i \quad a \quad bx_i) \qquad \textit{Eq. 13}$$

Separating knowns from unknowns

$$na + b\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i$$

$$a\sum_{i=1}^{n} x_i + b\sum_{i=1}^{n} {x_i}^2 = \sum_{i=1}^{n} x_i\, y_i \qquad \textit{Eq. 14}$$

Solving for a and b we get

$$a = \frac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i \quad n\sum_{i=1}^{n} x_i y_i}{(\sum_{i=1}^{n} x_i)^2 \quad n\sum_{i=1}^{n} {x_i}^2}$$

$$b = \frac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i\, y_i \quad \sum_{i=1}^{n} {x_i}^2 \sum_{i=1}^{n} y_i}{(\sum_{i=1}^{n} x_i)^2 \quad n\sum_{i=1}^{n} {x_i}^2} \qquad \textit{Eq. 15}$$

In Matlab, solution to the above can be obtained using the `polyfit` command:

```
[b, a] = polyfit(x, y, 1)
```

Where x and y are the vectors mentioned above.

2.6.2. Filtering

Filtering of discrete sets of data is covered by digital signal processing theory and there are many approaches and methodologies that cover a wide variety of applications in either frequency domain or on time domain. We will mention only filters used later by CTM firmware.

Filtering can be viewed as an averaging process designed to obtain a smoother line from a noisy set of data, or a texture profile in our particular case. One of the simplest, yet effective, is a moving average filter.

Considering a vector y holding a set of data taken at equal intervals we can define a filter vector S containing the weighting function. The filtered profile is obtained by positioning the filter vector at the beginning of the profile such that the first element of the profile overlaps the last element of the filter. A weighted average is then computed to obtain the first term of the filtered profile; then the filter moved to the left and the process is repeated until the filter vector is at the end of the profile vector.

The resulting filtered profile will have a length one less than the sum of the number of elements in y and S. This can be trimmed by discarding half of length of S from the filtered profile extremities. The length of S, chosen as an odd number, is an important parameter in the filtering process and typically depends on a variable called cut-off; the larger the cut-off, the higher the number of elements in S are required. Another aspect of the filter is the distribution of weights in the filter. A filter with all weights equal to the inverse of the filter length $(1/n)$ is called a rectangular filter. A triangular filter will have the highest weight in the centre of S with values decreasing linearly towards the edges. A

Gaussian filter will have the values decreasing gradually. It should be noted that applying a moving average filter to a profile will always result in a smoother profile.

Matlab provides an inbuilt function `conv` that will return a filtered profile C computed from vectors y and filter S provided that S weights are symmetric distributed.

```
plot(y)

% rectangular filter
S = [ 1/4, 1/4, 1/4, 1/4 ];
C = conv(y, S);

% extract relevant portion only
q = numel(S);   % size of S
C = C(q+1:numel(C)-q);

% show resulted filter
hold on
plot(C, 'r')
```

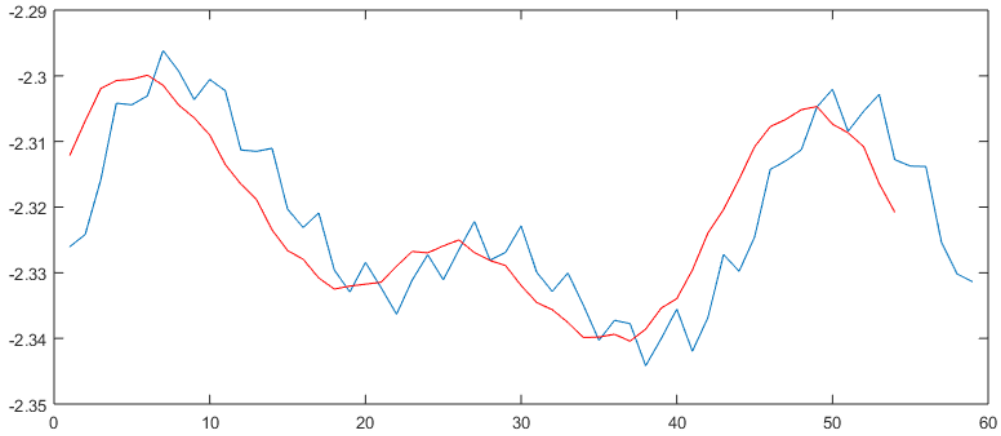


Figure 9. Moving average filter

Inherent to data acquisition is the existence of abnormal values, normally due to flaws in the surface. Those values are known as outliers. Unfortunately, the filter described above produces poor results in the presence of outliers as it is illustrated in Figure 10 where an outlier was introduced in the data used in the example above. Later, in 4.1 Outliers removal chapter we describe the method we used to detect and remove the outliers.

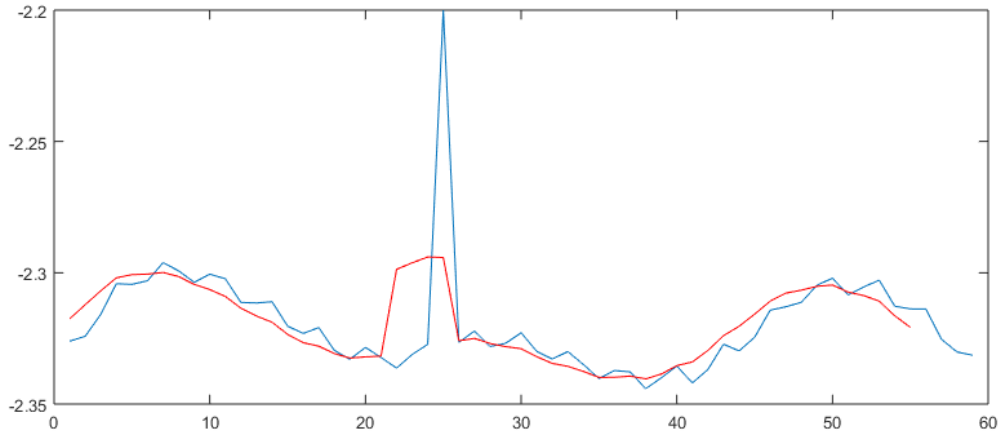


Figure 10 Moving average filter in the presence of an outlier

2.6.3. Parameterization

Surface parameters is information about a surface profile that can be used to predict the functional performance of the surface. A number of parameters used in surface metrology can be related with pavement structure; those are described below (Muralikrishnan & Raja, 2009).

2.6.3.1. Amplitude parameters

Average roughness, Ra, is defined as the mean of the absolute values of a profile y measured from the mean line:

$$Ra = \frac{1}{n}\sum_{i=1}^{n}|y_i| \qquad Eq.\ 16$$

Root mean square roughness, Rq is defined as the root mean square average of profile y measured from the mean line:

$$Rq = \sqrt{\frac{1}{n}\sum_{i=1}^{n}{y_i}^2} \qquad Eq.\ 17$$

Maximum profile peak height, Rp, is defined as the highest point of the profile measured from the mean line.

Maximum profile valley height,Rv, is defined as the lowest point of the profile measured from the mean line.

Maximum height of the profile, Rt, is defined as the sum of maximum profile peak height and maximum profile valley height.

Average maximum height of the profile, Rz is defined as the average of successive Rt_i calculated over the length of the profile, where Rt_i is the maximum profile height within ith sample length. The sampling length is generally equal to a cut-off.

2.6.3.2. Spacing parameters

Spacing parameters are used to describe the distance between peaks and valleys. There are multiple spacing parameters, the most common are

RSm is the mean peak spacing defined as

$$RSm = \frac{1}{q}\sum_{i=1}^{q} Sm_i \qquad \textit{Eq. 18}$$

where q is the number of zero crossings meeting two conditions: Rt between two zero crossings is greater than a threshold (usually 10% of Rz) and the spacing between two zero crossings is greater than a threshold (usually 1% of the sampling length).

High spot count is the number of peaks that cross a set threshold. To be counted the profile must raise above the threshold and fall back below the same threshold.

Peak count is similar to high spot count but requires the profile to fall below a lower threshold to be considered a peak.

2.6.3.3. Shape parameters

Slope is the ratio of the vertical to the horizontal displacement of two consecutive points. ISO 4287 defines the slope over 6 contiguous points:

$$\delta = \tan^{-1}\sqrt{\frac{\sum_{j=4}^{n-3} \quad _j^{\,2}}{n \quad 6}} \qquad \textit{Eq. 19}$$

converted to degrees. $_j$ is defined as

$$_j = \frac{y_{j-3} \quad 9y_{j-2} + 45y_{j-1} \quad 45y_{j+1} + 9y_{j+2} \quad y_{j+3}}{60 \quad x} \qquad \textit{Eq. 20}$$

The asymmetry of the profile, or skew, is defined as

$$Rsk = \frac{1}{nRq^3}\sum_{i=1}^{n} {y_i}^3 \qquad \textit{Eq. 21}$$

While the spikiness of the profile, or kurtosis, is defined as

$$Rku = \frac{1}{nRq^4}\sum_{i=1}^{n} {y_i}^4 \qquad \textit{Eq. 22}$$

2.7. Analogy with road texture measurement

Ignoring the scale for a moment, similarities between surface finish and pavement surface can be observed. Apart from lay, which is not applicable in pavement surface, the transversal road profile can be identified as waviness and the longitudinal profile (in the traffic direction) present in high speed devices using SMTD can be also identified as waviness. The circular texture meter, by nature, will have a periodical waviness due to difficulty in aligning the measuring plane and surface plane. After removing the waviness from pavement profile, the resulting roughness profile is in fact surface texture.

It can be observed that the methodologies prescribed for obtaining the mean profile depth is similar to that of computing the average maximum height of a profile. Both methods prescribe a linear regression over the sample length, with MPD being equal to Rz where the sample interval cut-off is half of the length of the profile. However, the ASTM method does not prescribe any filtering of data to remove the microtexture component or outliers in the sampled profile. The sensor measured texture depth is obtained in a similar fashion with that of root mean square roughness, Rq, with the difference that the profile is sampled at larger intervals compared to the height of the profile and a second order polynomial fit is subtracted from the sampled data.

The amount of data contained in a pavement profile is much less than that of a typical surface profile about 100 points per 100 mm for MPD and 40 points per ~300 mm for SMTD (ASTM International, 2015). This is somehow compensated in circular texture meter method by averaging 8 profiles per track or from a surface methodology point of view, using a sampling interval cut-off of ~50 mm. SMTD results are also reported as averages over 100 m of road sampled. Due to this, the microtexture is poorly represented in a MPD profile and is completely missing in SMTD profiles.

Considering the scale and the size of the samples, we can conclude that the waviness of pavement surface, while it is present in both MPD profiles and SMTD profiles, can be removed by simple least-square polynomial fit as in current practice. As the microtexture information is attenuated by the relatively large sample interval for the MPD profiles it can also be filtered by a low-pass filter. It is thus feasible to apply surface metrology techniques and parameters to measure the road texture depth.

3. HARDWARE

An important part of CTM is the displacement sensor. The AGAMT013 (Austroads, 2007) requires the distance measuring transducer to have an accuracy of ± 0.1%, the sample interval no greater than 1 mm for MPD and the spot size of the laser must be less than or equal to the specified sampling interval. We have identified a short range distance sensor produced by Sick, part number OD1-B100C50I14, that has all characteristics required. The output of transducer is industry standard 4 to 20mA current source allowing for reduced noise in the analog line. Another important factor in selection was the low supply voltage accepted by sensor, 12V to 24V, permitting the use of a 12V power tool battery. Last, its small enclosure with an ingress protection (IP) 67 makes it suitable for outdoor applications. Sensor specifications can be summarised as follows (Sick, 2015):

- Measuring range 50 to 150 mm
- Resolution 20 µm
- Repeatability 30 µm at constant ambient conditions
- ± 100 µm linearity
- 2 ms response time
- 2 kHz measuring frequency
- 700 µm x 600 µm light spot size at 100 mm

3.1. Mechanical Part

The sensor is mounted on a rotating arm permitting continuous rotation. At the other arm end two reed switch sensors are mounted with the magnet counterpart mounted on the plate in a manner that sensor 1 will register four positions per revolution and sensor 2 will register one position per revolution. The height of the sensor is approximately 100 mm from the plane formed by the footrests of the frame placing it in the middle of the measuring range.

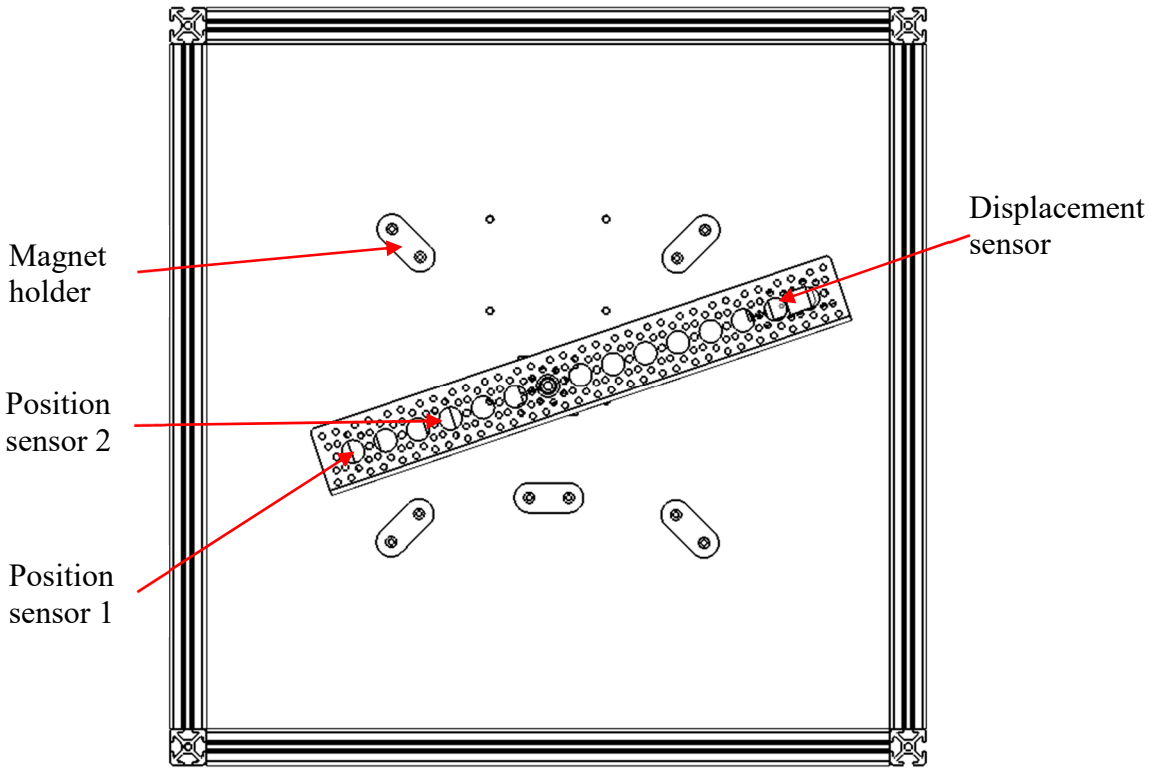


Figure 11 CTM bottom view

The arm is rotated by a 168 RPM geared motor through a power gearbox. The pinion/spur gear ratio is 20/84, giving a 40 RPM rotational speed of the arm, or one cycle every 1.5 seconds. The motor is a brushed type with an operating voltage range 3 to 12 V. It has a no load current of 45 mA with a stall current of 500 mA at 12V, making it suitable for a portable application. The measured operating current is 60 mA.

Power transmission between gearbox and arm is made though a custom made hollow shaft and held by a 3 bearing system to minimise inherent vibration arising from use. The arm, made of a 12 inch aluminium channel is clamped off centre with two clamping hubs. The reason for off centre mounting is to permit the sensor mounting at a distance of approximately 284 mm recommended by ASTM E2157. The arm, gearbox, geared motor and clamping hubs were sourced from Servocity, an online store supplying a variety of mechanical components for industrial mechatronics.

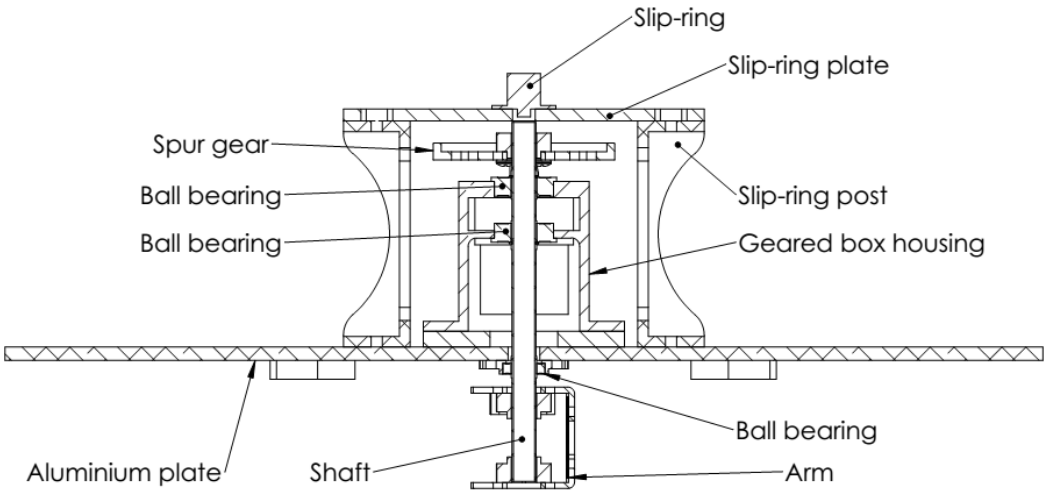


Figure 12. Section view through gearbox assembly

Electrical connections required by the sensor are made via a slip-ring and through the hollow shaft. The slip-ring is supported by two posts and a plate assembly purpose designed and 3D printed.

The arm/gears assembly is sitting on a 5 mm thick aluminium plate with dimensions of 400 mm square. The plate is enclosed by frame built from aluminium extrusion profile commonly found in industrial applications, which offers excellent rigidity. The top front edge has been retracted 150 mm and is used to support an acrylic plate used for fixing the main circuit board, battery assembly and the fuse holder.

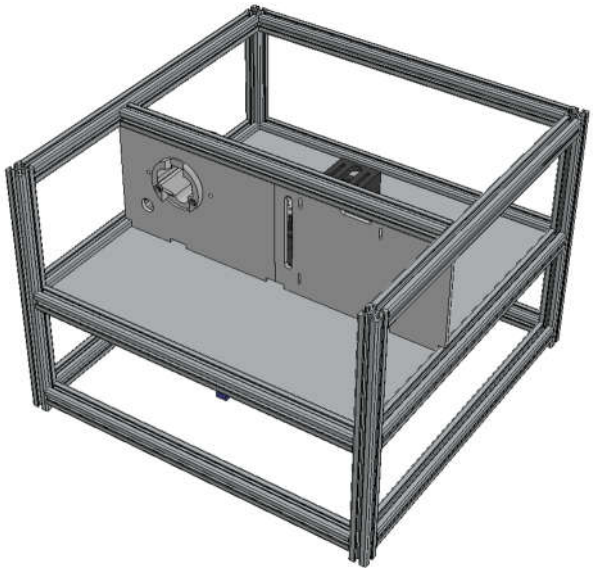


Figure 13. Circular Texture Meter frame and plate assembly

The frame has aluminium composite panels on all sides excepts the bottom to protect the assembly from dust or weather and to provide a barrier between operator and moving gear parts. The top frame hosts the operator screen, power and start push buttons. Access to the battery, fuse and the SD card on the main PCB is facilitated though a hinged door provided with a magnetic catch.

Figure 14. Circular Texture Meter

3.2. Electrical Part

The CTM is powered from a 12V, 4A Lithium Ion battery, produced by AEG for a range of power tools. The battery was chosen for the adequate voltage and power available, versatility of charging and changing the battery. A spare battery assures uninterrupted operation. A purpose built assembly has been designed and 3D printed for securely housing the battery inside the frame.

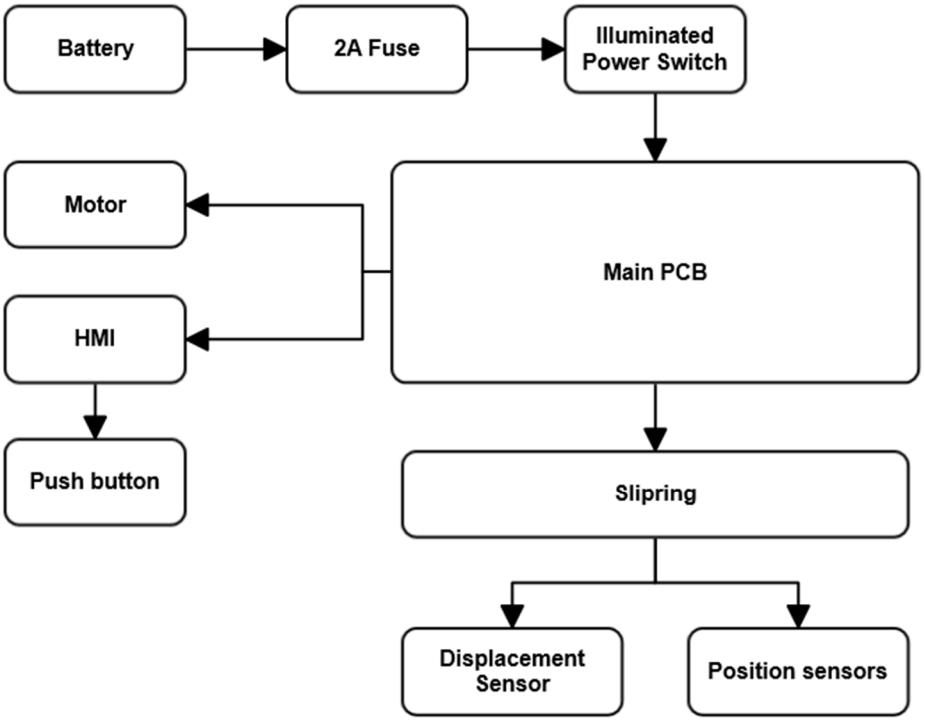


Figure 15 Electrical schematic block diagram

The voltage is then fed to the main PCB through a protection fuse and an illuminated maintained power switch. The connections to the switch indicator are made to indicate the integrity of the fuse. Circuitry present on texture meter PCB distributes power to serial-LCD converter (HMI), controls the polarity and frequency of the motor supply. The displacement sensor and position sensors are connected though a slip-ring to allow for continuous arm rotation. The slip-ring can pass 6 connections at up to 2A per circuit, with a range of speeds 0 to 300 RPM. The connections are common ground, permanent Vdd for reed switches, reed switches output, displacement sensor supply and analog output from the sensor.

The sensor has an industrial 4-pin M8 connector, connections diagram shown in Figure 16 (L+ is voltage supply, Q_A is the returned current pin). The Sick OD1-B series have a teaching feature; however, the factory default values are kept unaltered.

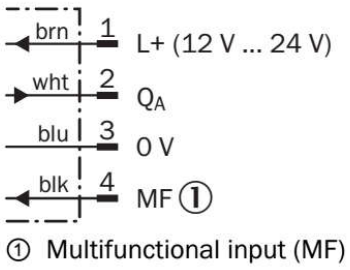


Figure 16 OD1BxxxxxI14 connection diagram

An electrical schematic is provided in *Annex A*.

3.3. Embedded system

The embedded system consists in a single printed circuit board housing power regulators, analog converters, motor control circuitry and the microcontroller. A block diagram is presented in Figure 17.

3.3.1. Power block

The CTM is powered from a 12V battery housed separately, the circuit being protected by a 2A replaceable thermal fuse. The voltage is applied at terminal blocks 1 (positive) and 2 (ground). A diode, 1N4007 or equivalent, is used for reverse polarity protection before the voltage is fed into 5V regulator. The regulation is obtained with a low drop voltage regulator, AP1117 from Diodes Incorporated, with a range of capacitors either side of the IC permitting good noise rejection. The VR can supply up to 1A which is well in excess of board requirements. The 3V regulated voltage is fed from the 5V supply and is used mainly for the CPU supply. It has a similar setup as the 5V stage.

A voltage divider formed from a pair of 10.0/1.0 kΩ is used to monitor the battery voltage by the CPU to avoid depleting the battery.

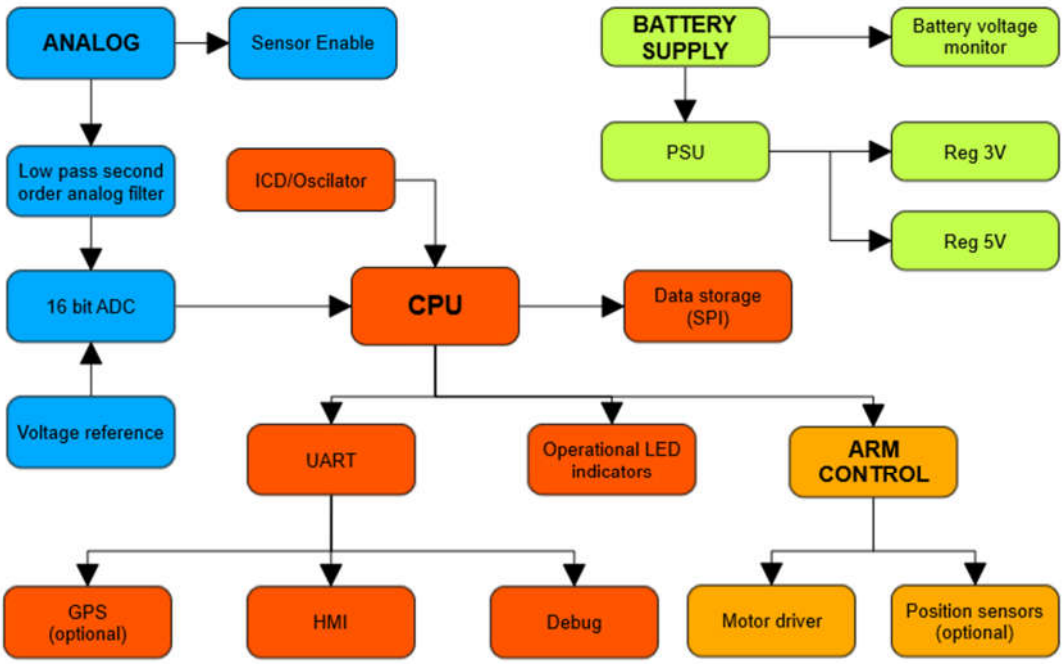


Figure 17. Block diagram of the embedded system

3.3.2. Analog block

Analog block has provisions to use either of 0 to 10V or 4 to 20 mA as input generally provided as industrial standard by laser displacement sensors. The maximum voltage accepted by the converter is 4.096V therefore the voltage signal is conditioned through a voltage divider 5.6/2.4 kΩ outputting 3.0 V for the maximum 10V that can be presented. The voltage analog signal is then passed to a Sallen-Key low pass filter with a cut-off frequency of 4468 Hz.

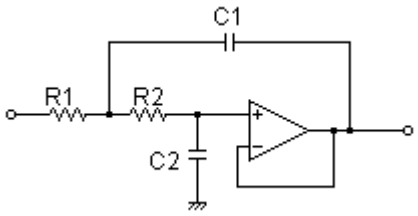


Figure 18. Sallen-Key second order filter

Designing the filter had two objectives: fast response and small overshoot. This is achieved by passive components choice:

R1	4.7 kΩ	C1	100 nF
R2	270R	C2	10 nF

The filter has a damping ratio $\zeta = 0.698$ and the first overshoot peak for a step input $g_{pk} = 1.05$ at time 0.00016 seconds.

The 4 to 20 mA current input side of the analog block requires to convert the current to voltage. The maximum current that can be presented is 24 mA therefore is passed through a 120 Ω resistor resulting in a maximum 2.88 V output. The voltage obtained is then filtered by a second Sallen-Key active filter with the same characteristics as described for the voltage section.

The IC used is a single supply dual operational amplifier MCP602 produced by Microchip. The op-amp utilise CMOS technology that provides low bias current, high-speed operation and rail to rail output swing (Microchip, 2007). It has as main features:

- Single supply: 2.7V to 6.0V;
- Rail to rail output;
- Input range includes ground;
- Gain bandwidth: 2.8 MHz;
- Low quiescent current: 230 µA.

As the microcontroller used in this project has an on chip ADC unsuitable for this application (10bit), an external 16bit analog to digital converter is used to convert the voltage output of the filters, passed by means of a selector jumper mounted on the printed circuit board. The ADC is provided by Analog Devices, part number AD7694, operating from a single power supply with up to 250 k samples per second throughput. Its input range is matched with that of operational amplifiers used in previous stage. The conversion is achieved with successive approximation with data made available through SPI interface port. The sampling takes places on CNV raising edge, with data available on falling edge of CNV (Analog Devices, 2005). It has as main features:

- Single supply 2.7V to 5V;
- Input voltage range: 0 to Vref with Vref up to Vdd;
- Signal to noise ratio: 92dB @ 20 kHz;
- Current consumption: 800 µA @ 5V/100 kSPS;
- Serial interface SPI.

The converter requires reference voltage which is provided by a MCP1541, 4.096V precision voltage reference. This type of devices is using an advanced CMOS circuit and EPROM trimming to provide an initial accuracy of (max ±1%) and a temperature stability of ±50 ppm/°C (Microchip, 2012).

For safety reasons the laser displacement sensor is powered only when data acquisition is in progress. This is achieved through a single pole single throw relay controlled from the CPU.

3.3.3. Motor control

Sensor arm is actuated by a brushed DC motor controlled with a L293D quadruple half bridge driver that incorporates internal clamp diodes. For thermal reasons the two bridges are linked together. The device accepts standard TTL logic levels and is designed for driving inductive loads.

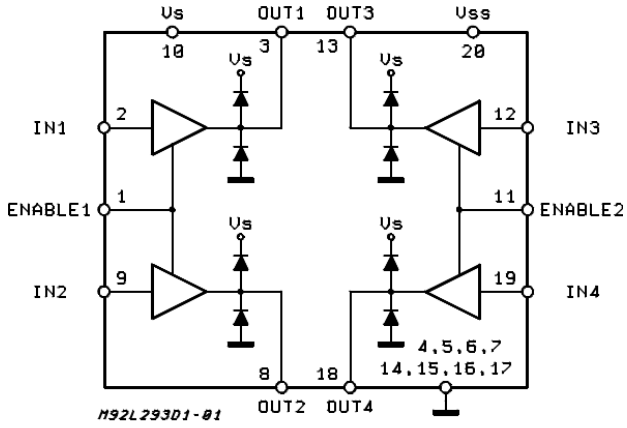


Figure 19 Block diagram of L293D (ST Electronics, 2003)

The TTL inputs Enable1, Enable 2, IN1, IN2, IN3, IN4 are controlled from CPU. When required for speed control, the enable inputs are pulsed using pulse width modulation. Main features of L293 are

- 600 mA continuous output current per channel;
- Over-temperature protection;
- High noise immunity, logical '0' input voltage up to 1.5V;
- Internal clamp diodes.

For arm positioning, two optional inputs are provided conditioned with 3.3/1.0 kΩ voltage dividers permitting the use of standard industrial reed switches.

3.3.4. CPU

At the core of the embedded system lies the central processor unit. At hardware level, it performs functions such as acquiring of signal data from the analog block, perform logic for the motor control, controls the universal asynchronous receiver/transmitter (UART)

ports for debug and human-machine interface, writes data to an external SD card and controls the on board LED operational indicators.

The minimum requirements for the CPU are 3 UART ports, 2 SPI (serial peripheral interface), one analog channel for battery monitoring, a number of digital inputs and outputs for other peripheral control. For this application a versatile Microchip PIC24FJ256GA006 has been chosen. It has as main features (Microchip, 2010):

- Operating voltage range of 2.0V to 3.6V
- 5.5V tolerant inputs
- High current sink/source on all IO
- Peripheral pin select allows independent mapping of peripherals at run time
- 3x SPI modules
- 4x UART modules
- 10bit, up to 16 channels analog to digital converter at 500 kSPS
- 256 kb flash memory
- 16 kb random access memory (RAM).

Programming and reset circuit is made in accordance with Microchip recommendations ((Microchip, 2010) chapter 2.3). The oscillator is an external 8 MHz resonator for precise timing required by data acquisition specifications.

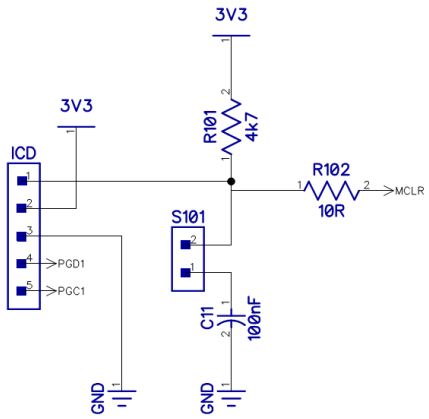


Figure 20 Programming and reset circuit

Algorithm building and system validation requires data acquired to be made available. During the acquisition process, the slow nature of serial communications provided by the UART port is inadequate for real time data relaying. In addition, a real time system would require a host attached to CTM which would make the process difficult to manipulate. The solution is to write data after it has been processed on an SD card. The CPU can

easily handle this task using serial peripheral interface with which most secure digital cards are compatible. The necessary connections are made through a SD card reader assembly manufactured by FCi.

The UART modules are provided as direct connections to the external modules at TTL levels. One of those modules is assigned to the HMI, one for debug port and one for an optional GPS.

Last, the various states of blocks are indicated by LEDs. Correct operational state is indicated by a green LED marked “OK” on PCB; if for any reason the CPU has encountered a fault a red LED is illuminated, marked “Fault”. The rest of the LEDs have orange colour and are used for indicating UART activity from HMI and GPS; motor control: enabled, forward and reverse motion; position sensors feedback; state of displacement sensor power and digital to analog converting in progress.

3.3.5. Human-machine interface

Interaction with the operator is essential for CTM functionality. The results, states, operation and faults are shown by an alphanumeric dot matrix liquid crystal display (LCD). This is interfaced with a serial-LCD converter designed by us in 2011 with the objective to facilitate a simple interface with HD44780 LCD controller developed by Hitachi. The board is attached directly to the LCD and requires 5V, 100 mA supply.

The converter will present a character received through its UART module to the standard Hitachi 4-bit operation mode. At a baud rate of 4800 bps the operation is in real time, generally the LCD executes an instruction before the next character is received.

The serial-LCD converter has the option to capture 4 digital inputs, normally push-button switches, to enable two-way human-machine interaction. The state of the buttons is periodically updated though serial port to the host as a one byte transmission.

For this application a 20 characters, 4 lines display has been chosen and the operator input requires only one button for start. The assembly is mounted separately from the main board on the top face of CTM.

4. DATA PROCESSING

The first step taken to process the data was to assess the suitability of the circular texture meter as a platform to acquire a pavement profile. For this the CTM embedded software was setup to acquire a profile of 2500 points per revolution at an interval of approximately 0.345 mm. The analog signal from displacement sensor was converted to a distance value, in millimetres, and the profile was saved on the SD card for post processing.

Data processing was performed in two stages. First, a Matlab toolbox was developed with the intention to examine the profiles obtained from CTM and identify an algorithm that would produce optimal results. In the second stage, we implemented the algorithm in C for embedded systems into the CTM firmware. The Matlab toolbox is described in this chapter.

As the position of displacement sensor is above the profile, we have inverted the profile saved by CTM to reflect the true orientation of the profile.

4.1. Outliers removal

First profile acquired was that of a flat laminated surface with relative small roughness. We will refer to this profile as Profile 1 later.

A plot of profile is shown below.

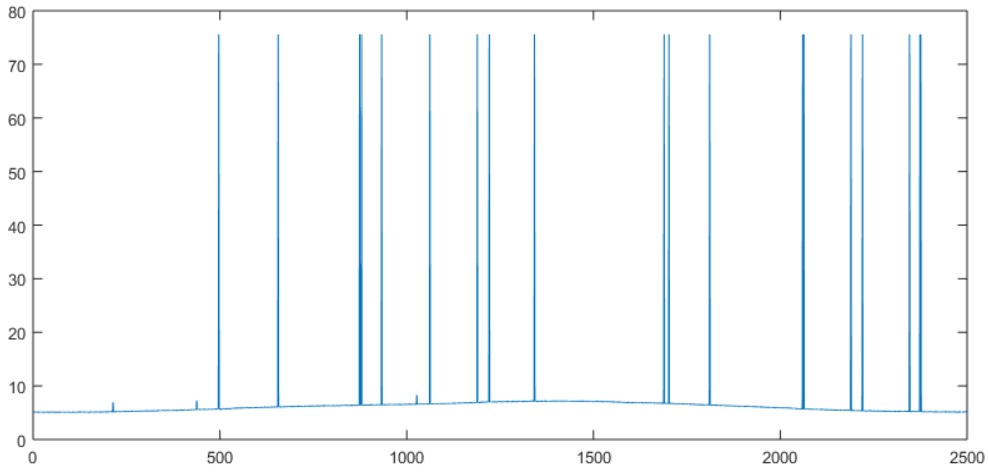


Figure 21. Raw profile of a flat laminated surface

Firstly, it can be observed that the majority of the outliers have a very large value. In fact, the value is outside of the range of the displacement sensor used (±50 mm) which makes the identification trivial. However, for smaller value outliers the method above is not suitable. We have chosen a parametric approach for removing the outliers, where a threshold is used to identify large changes (d) in the values between two neighbour points.

$$d = \frac{|y_i \quad y_{i-1}|}{x} \qquad \textit{Eq. 23}$$

The outlier value is replaced with the average obtained from points immediately before and after the outlier.

$$y_o = \frac{|y_{o-1} \quad y_{o+1}|}{2} \qquad \textit{Eq. 24}$$

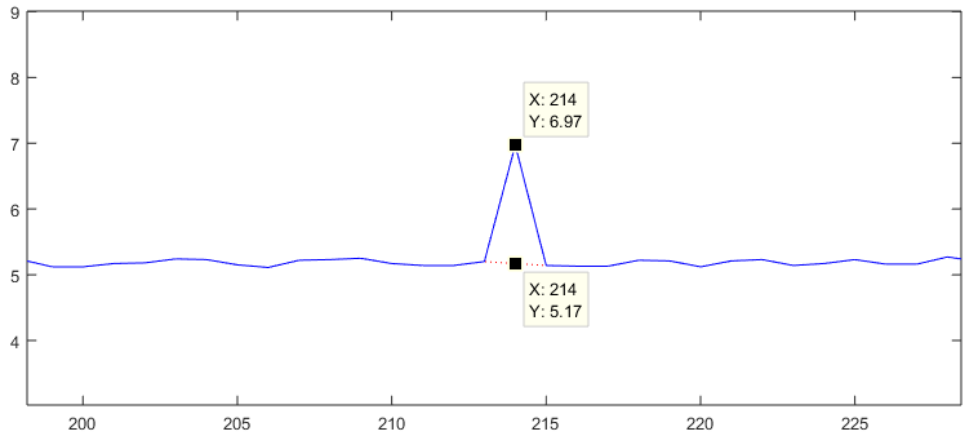


Figure 22 Outlier removal

A shortcoming of this method is that in the presence of two or more consecutive outliers Eq. 24 fails to deliver a desired outcome due to the fact the point after the first outlier is an outlier itself. Therefore, a segment of outliers is identified by using the same threshold but we search for the next index that will produce a value of d less than the threshold.

$$d = \frac{|y_i \quad y_{i-j-1}|}{(j+1) \quad x} \qquad \textit{Eq. 25}$$

Where j is the size of the segment identified. The values of outliers are replaced with linearly interpolated values obtained from points before and after the outlier segment.

Exceptions to the rule described in Eq. 25 are outliers present at profile extremities where the replacement value is obtained from the slope of the segment neighbouring the outlier.

The Matlab toolbox function developed is

```
[Y, e] = rtm_remout( Y, t, dx )
```

Where arguments passed are **Y**, the vector containing the profile data, **t** is the rate of change threshold and **dx** is the sampling interval. The function returns the filtered vector in **Y** and the percentage of outliers to the size of **Y** in **e**. The function will return an error if the number of consecutive outliers exceeds a maximum value defined inside the function.

The results of the Profile 1 after the outliers were remove is shown in Figure 23.

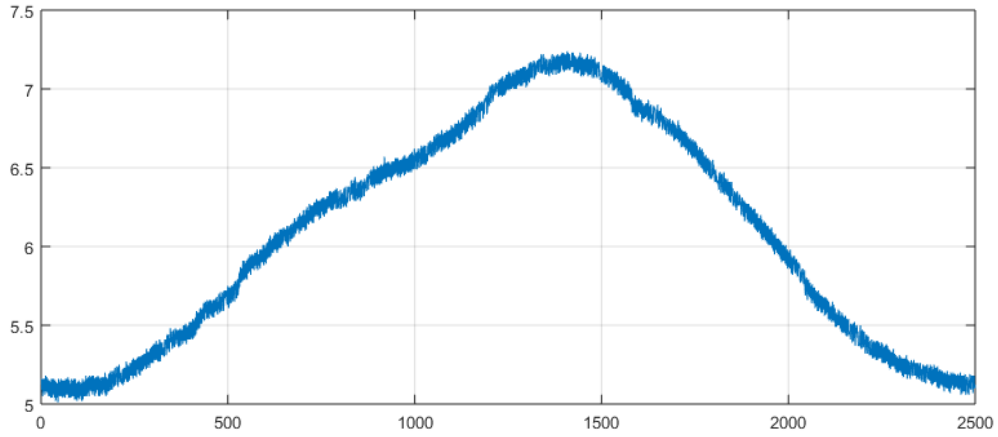


Figure 23. Profile of the flat laminated surface with outliers removed. e = 0.0088

4.2. Filtering

As we are interested only in macrotexture features, filtering is required to minimise the effects of microtexture in measurements and filter out unwanted noise from sensor. The moving average filter with the method described in *2.6.2. Filtering* results in two undesirable outcomes: (1) the filtered profile length is longer and (2) due to the nature of the filter the edges of the filtered profile are distorted. An example is shown in Figure 24 where a rectangular filter with the length of 20 was applied to the profile shown in Figure 23.

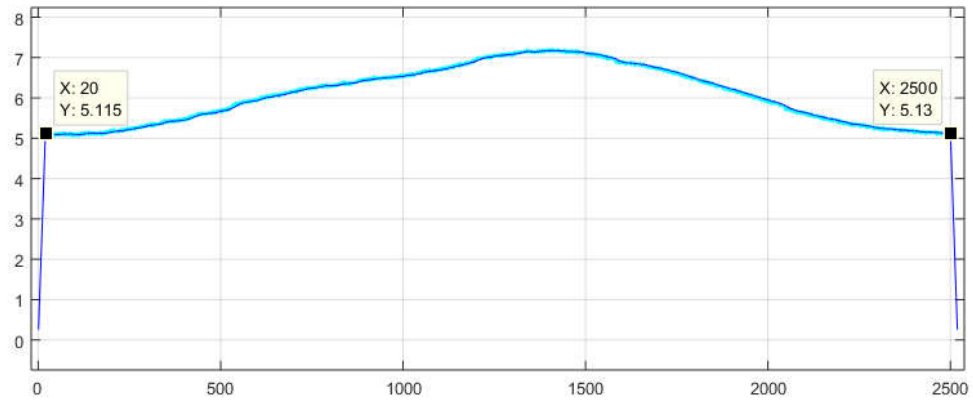


Figure 24. A rectangular moving average filter applied to the flat laminated surface profile. Filter order: 20.

A common technique to avoid the distortion is padding the edges of the profile with a copy of the profile extremities with a length equal to the filter order. To obtain a filtered profile of the same length and with the same phase we use only the centre of the resulted profile. In Figure 25 the result of the approach is illustrated where the same rectangular filter is applied. The original profile is coloured cyan, magenta profile represents the padding addition and the resulting filtered profile is blue; the x-axis has been scaled to represent the true profile length in millimetres.

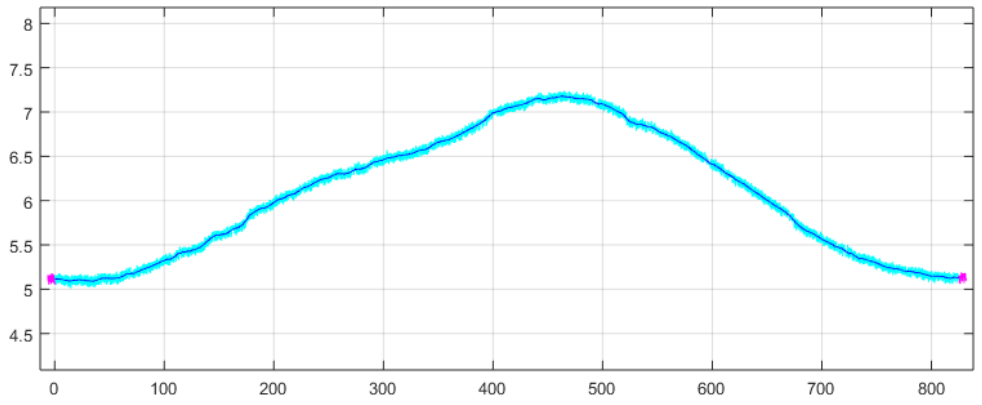


Figure 25. Filtered profile using the toolbox function; x-axis is scaled to profile length in mm.

The Matlab toolbox function developed is

```
Yf = rtm_filter(Y, f)
```

Where the arguments passed are **Y**, the vector containing the profile data and **f** a vector containing the filter weights applied to it. If **f** is a scalar, then the function will build a rectangular filter with the order specified by **f**:

```
if isscalar(f)
    f = ones(1, f)/f;
end
```

Probably the most used filter for surface profile analysis is the Gaussian filter (Tomov, Kuzinovski, & Trajcevski, 2010), defined by ASME B46.1:2009 and by ISO 11562:1996 with weighting function defined as

$$F_x = \frac{1}{\alpha\lambda_c} e^{-\pi\left(\frac{x}{\alpha\lambda_c}\right)^2} \qquad \textit{Eq. 26}$$

where $\alpha = \sqrt{\ln 2 / \pi} = 0.4697$, x is the index of the weight filter and λ_c is the long wavelength roughness cut-off. The filter transmits 50% of the cut-off amplitude of a sinusoid whose wavelength is equal to λ_c.

The Matlab toolbox function that implements *Eq. 26* is

```
f = rtm_gauss(dx, lc)
```

where **dx** is the sampling interval (mm) of the filter and **lc** is λ_c (mm) in the *Eq. 26*. The function returns vector **f** with sum of weights normalised to 1.

An example of usage is presented below:

```
f = rtm_gauss(0.33, 10);
x = -10:0.33:10;
plot(x,f, 'o')
grid on
xlabel('mm');
ylabel('Weight function');
```

returns a filter with 61 elements shown in Figure 26.

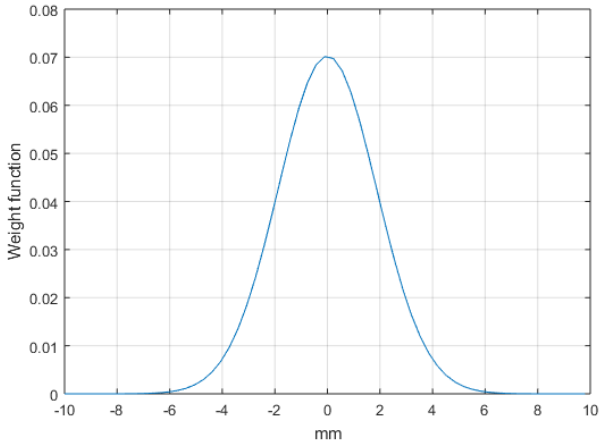


Figure 26. Gaussian filter

Tomov et al. (Tomov, Kuzinovski, & Trajcevski, 2010) have shown that Gaussian filtering is better suited for non-periodic profiles, returning the mean of the profile as it can be compared in Figure 27 where the Gaussian filter (bottom) obtained above is paralleled with a rectangular filter of the same order (top) applied to the first 120 mm of the profile 1.

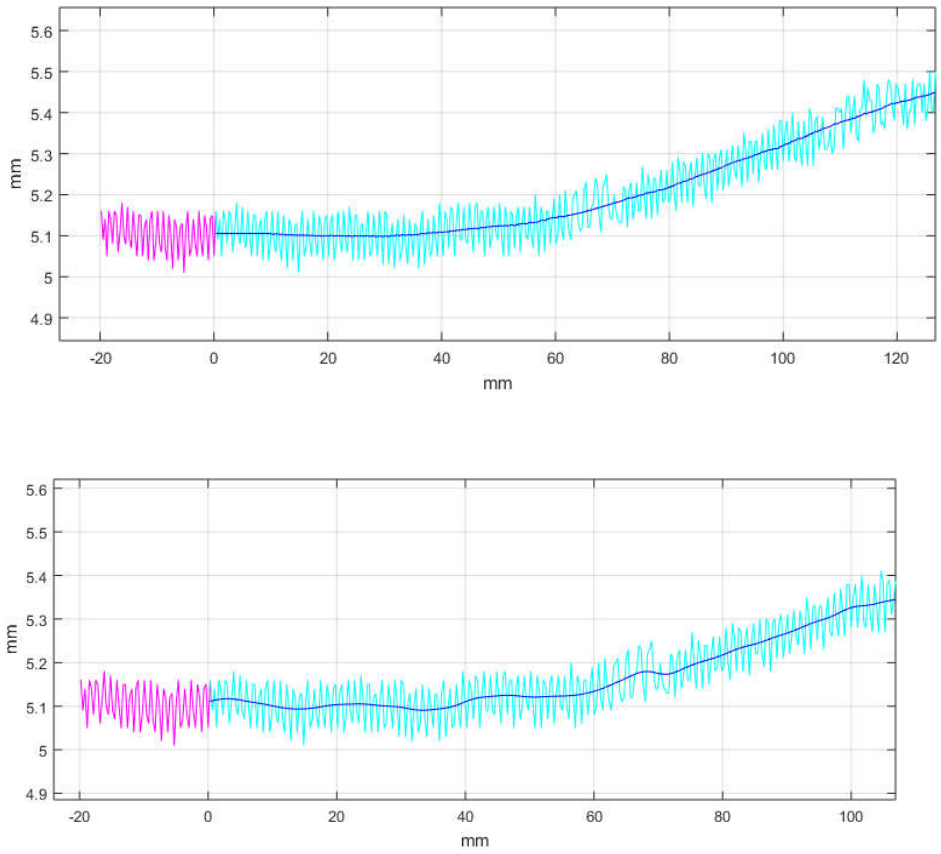


Figure 27. Comparison between rectangular filter (top) and Gaussian filter (bottom).

4.3. Fitting

In contrast with surface metrology practices, where the waviness of the profile is obtained through filtering, calculation of mean profile depth as recommended by ASTM E965 (ASTM International, 2015) requires a linear regression applied to a profile of about 100 mm length while SMTD necessitates a second order polynomial fit over approximately 300 mm profile length. We have written a simple Matlab function that will return the mean profile.

```
[Yi, r2, rmse] = rtm_fit(Y, dx, o)
```

The arguments passed to the function are **Y** as the profile vector, **dx** is the sampling interval and **o** is the order of the fitting. The function returns a vector **Yi** which is the zero mean profile obtained by subtracting the regression from the profile, the coefficient of determination, **r2**, and root mean squared error of the fit, **rmse**.

4.4. Amplitude parameters

Calculating amplitude parameters is a trivial task in Matlab. Average roughness Ra is calculated as the mean of absolute values of the profile:

```
Ra = sum (abs(Y))/Nt;
```

Where **Y** is the profile vector and **Nt** is number of elements in **Y**.

Root mean square roughness, Rq, is calculated as the root mean square average of the profile Y:

```
Rq = sqrt(sum(Y.*Y)/Nt);
```

Maximum profile peak height, Rp, and maximum profile valley height, Rv, are calculated using Matlab built in max and min functions.

```
Rp = max(Y);
Rv = abs(min(Y));
```

The last parameter, maximum height of the profile, Rt, is simply the sum of the last two parameters:

```
Rt = Rp + Rv;
```

We have written a toolbox function that returns all parameters in a single call:

```
[Ra, Rq, Rp, Rv, Rt] = rtm_ap(Y)
```

4.5. Spacing parameters

We have implemented only the high spot count from the spacing parameters; for completeness, a low spot count is also returned by our implementation. In addition, we are interested in the value of the peak or valley identified. Generally, detecting peaks and valleys in a function requires the identification of minima and maxima, also known as extrema. The Rp and Rv parameters discussed in the previous section are returning the value of the global maxima and global minima respectively, however, we need to know the number of the peaks and valleys and their values which is a problem of finding local extrema.

The method used for finding the local extrema examines the sign of the derivative of the segments before and after the index examined. Local maximum is identified when

$$(y_{i-1} \quad y_i) < 0 \wedge (y_i \quad y_{i+1}) > 0 \qquad \textit{Eq. 27}$$

evaluates to true; while local minimum is identified when

$$(y_{i-1} \quad y_i) > 0 \wedge (y_i \quad y_{i+1}) < 0 \qquad \textit{Eq. 28}$$

evaluates to true.

The toolbox function is:

```
[vmax, imax, vmin, imin] = rtm_extrema( Y, Tp, Tv )
```

Where the arguments passed in are **Y**, the vector containing the profile data, **Tp** and **Tv** are the peak and valley thresholds. The function returns the maxima values in **vmax** and the indexes of those in **imax** and the minima in **vmin** and **imin** respectively. The high peak count is equal to the number of elements in vmax. If Tp and Tv are not specified, no threshold is applied.

It should be noted that the function does not deal with edges of the profile as minimum or maximum local values, to be considered an extremum the expressions in *Eq. 27* and *Eq. 28* must be true. Also, the thresholding values are playing an important role in correctly identifying the peaks and valleys of interest. Considering the example below where no threshold is specified.

```
dx = 0.345;
plot(x, Yf3, 'b');
grid on
hold on
[vmax, imax, vmin, imin] = rtm_extrema(Yf3);
plot((imin-1)*dx, vmin, 'o', 'color', 'g');
plot((imax-1)*dx, vmax, 'o', 'color', 'r');
axis([-5, 115, min(Yf3)-1, max(Yf3)+1])
xlabel('Profile length (mm)')
ylabel('Profile height (mm)')
```

will yield the following values for vmax and vmin:

```
vmax =     0.2914   -0.1484    0.0128    0.5611    1.1603    0.2287
vmin =     0.1660   -0.1641   -1.0153   -0.5599   -0.1085   -0.0310
```

The profile and extrema are plotted in Figure 28.

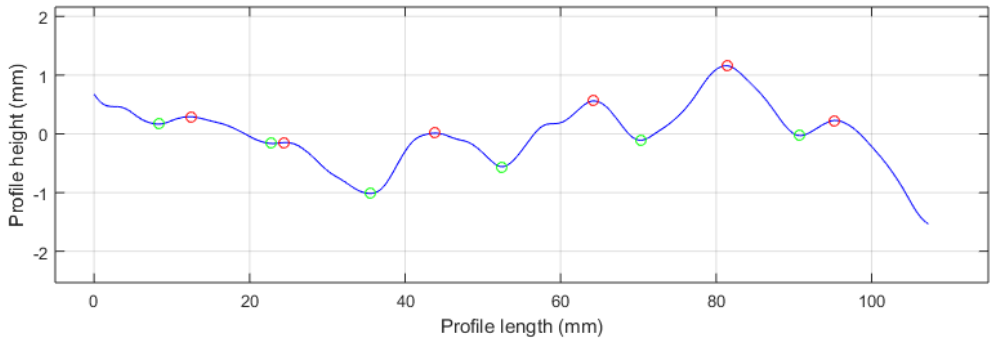


Figure 28. Peak and valleys detection, no threshold applied.

It can be noted that there are local maxima with values smaller than minima are identified. Surface metrology practice recommends a threshold of 10% of maximum profile peak height. Introducing this threshold for both peaks and valleys produces results where only the highest and lowest local extrema are returned.

```
[vmax, imax, vmin, imin] = rtm_extrema(Yf3, Rp*0.1, -1*Rv*0.1);
```

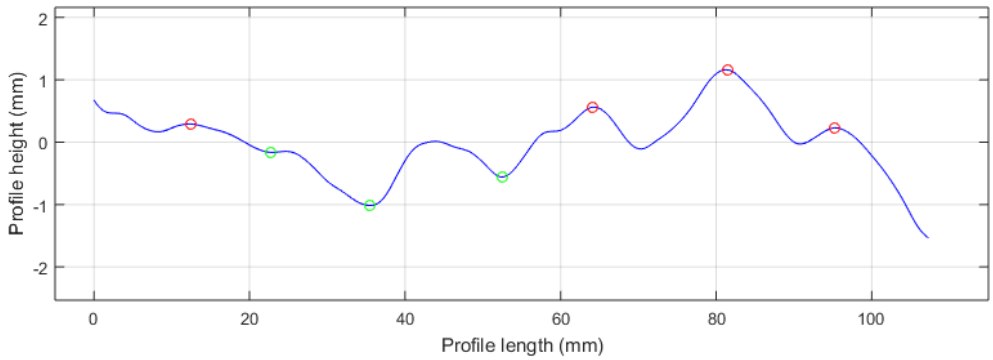


Figure 29. Peak and valleys detection, 10% threshold applied.

As a final observation, the performance of the detection depends on the smoothness of the profile investigated.

4.6. Importing data

Circular Texture Meter saves data as a comma separated values file on an SD card in the format [index], [value]. For ease of operation, an importing function has been written to automate the process:

```
Y = rtm_read(filename, startRow, endRow)
```

If the `startRow` and `endRow` are omitted, the whole file is imported.

As the displacement sensor is mounted above the surface, the raw values obtained needs to be inverted in order to represent correctly the profile orientation. This is dealt within this function.

4.7. Calculating the Mean Profile Depth for a track

ASTM E965 requires the track obtained from a full revolution to be divided in 8 segments, then the mean profile depth is reported as the average of MPDs for each individual segment. The first operation is to read the data from the file:

```
Y = rtm_read('data1002r.txt');
```

followed by outliers removal, then the filtering of the noise and dividing the track into segments:

```
Yc = rtm_remout(Y, 0.5, dx);
f = rtm_gauss(dx, 5.5);
Yf = rtm_filter(Yc, f);
```

```
for index = 1:8
    v(index,:) = Yf( 1+(index-1)*(312) : index*(312) );
end
v = v';
```

For each segment a linear regression is applied (first segment shown)

```
Yi = rtm_fit(v(:,1), dx, 1);
```

Then the peaks of each half are identified and the averages returned:

```
P1 = Yi(1:floor(Nt/2));
P2 = Yi(floor(Nt/2)+1:Nt);
[vmax1, imax1, vmin1, imin1] = rtm_extrema(P1);
[vmax2, imax2, vmin2, imin2] = rtm_extrema(P2);
P1max = max(vmax1);
P2max = max(vmax2);
mpd = (P1max + P2max)/2;
```

The algorithm presented is packed into a single function that returns the mean profile depth from a segment with a single call:

```
mpd = rtm_mpd(Y, dx, 'plot');
```

The arguments passed to the function are **Y** as the segment profile, **dx** as the sampling interval. The last argument is optional, when included it will plot the profile.

The track algorithm is managed through a purpose built script

```
[mpdtrack, mpd] = rtm_mpdtrack(filename, dx, lc, 'plot');
```

The arguments passed are the name of the comma separated values file containing the raw profile as acquired by the Circular Texture Meter, **dx** is the sampling interval and **lc** is the cut-off of the Gaussian filter used. The last argument is optional and will display all eight segments of the track into subplots. The function returns the track MPDs average into **mpdtrack** and the segments MPDs in **mpd** as a row vector. An example is given below:

```
[mpdt, mpd] = rtm_mpdtrack('data1021r.txt', 0.345, 5.5, 'plot')
mpdt =
    1.0499
mpd =
    0.8385    1.5075    0.6535    1.4914    1.4085    0.8636    0.7511    0.8854
```

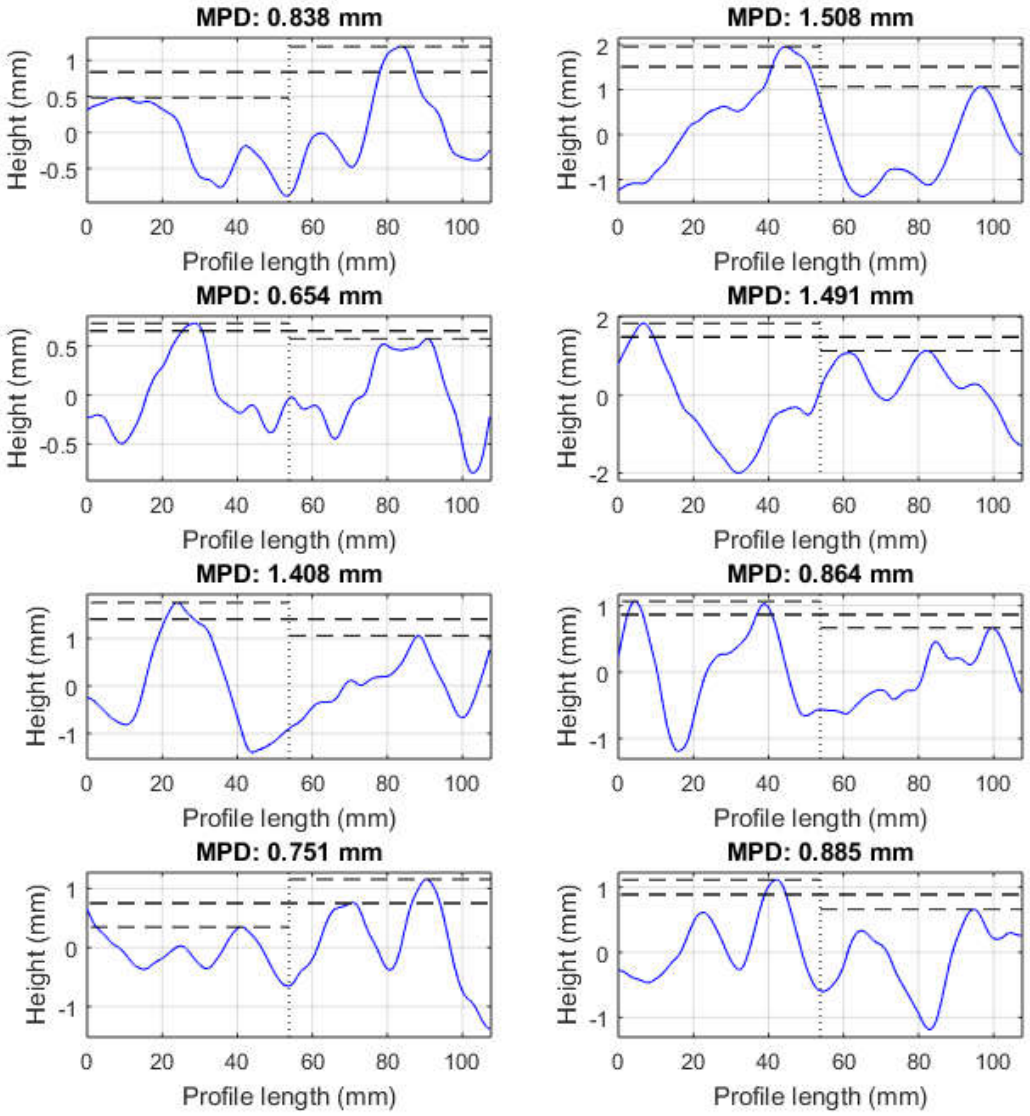


Figure 30. Mean profile depths for a track obtained with Circular Texture Meter.

In the case of a flat surface mentioned at the beginning of the chapter, the mean profile depths obtained with the prescribed MPD algorithm for the first segment are shown in Figure 31. It can be noted that the linear regression applied does not produce a very good fit, the coefficient of determination of the fit is 0.7890. This is due to the circular track that is used to acquire the profile where any misalignments between measured plane and the reference plane of the sensor will produce a curved form track rather than a line form.

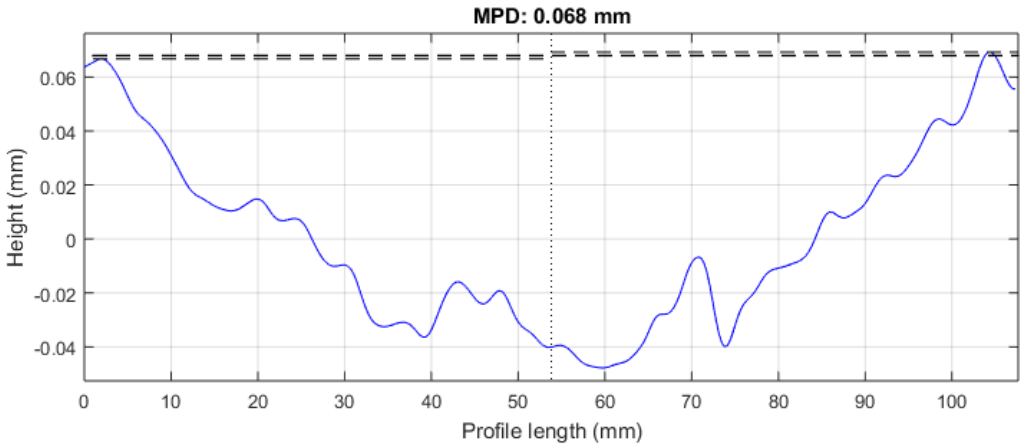


Figure 31. Mean profile depth for a flat surface, segment 1 of a circular track.

Fitting a second order polynomial to the segment produces a much better fit as it allows for the form introduced in the measurement. This can be observed in Figure 32 where the fit produces a coefficient of determination is 0.9885; also it can be noted that the MPD value has dropped significantly.

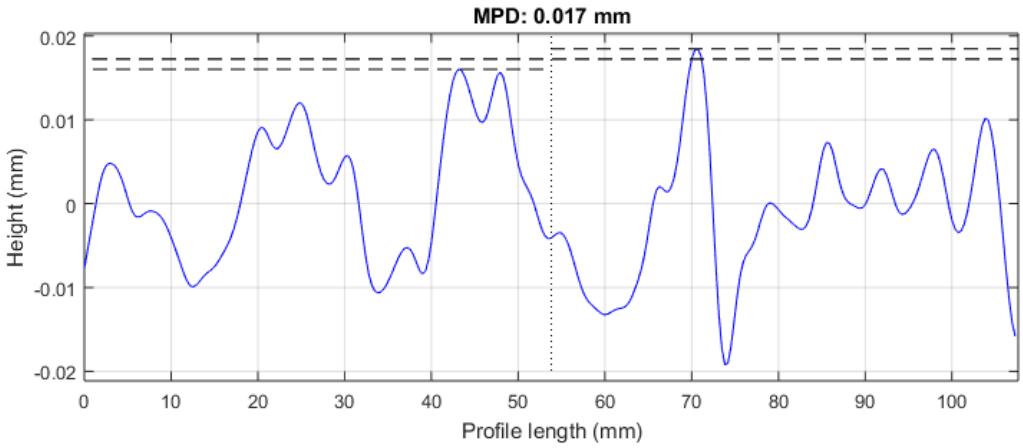


Figure 32. Mean profile depth obtained using a second order polynomial fit.

Two functions, rtm_mpd2 and rtm_mpdtrack2 will return mean profile depths where a second order polynomial fit has been used.

4.8. Calculating Average Maximum Height of the profile

Both ISO and ASME standards offer a method of calculating the average maximum height of the profile, Rz, with the main difference being the choice of the sampling length over the evaluation profile. ASME B46.1 computes Ra from the whole profile length, while ISO 4287 defines the roughness average as computed from the sampling length. Rz

has also multiple definitions in different standards (Tabenkin, 2007), with the most adopted being 10 point average distance between five highest peaks and five deepest valleys within a sampling length from a roughness profile, known as *Rz (JIS B0601)*:

$$Rz = \frac{\sum_{i=1}^{5}|y_{pi}| + \sum_{i=1}^{5}|y_{vi}|}{5} \qquad \textit{Eq. 29}$$

Due to the nature of road surface profile, where the macrotexture wavelengths have generally large values, the method described by Japanese Industrial Standard B0601 is not suitable as the profile will not always have 5 peaks and valleys, especially when thresholding is applied. As such we are determining the average maximum height of the profile by averaging all available peaks and valleys over a threshold:

$$Rz = \frac{1}{n}\sum_{i=1}^{n}|y_{pi}| + \frac{1}{m}\sum_{i=1}^{m}|y_{vi}| \qquad \textit{Eq. 30}$$

Where n is the number of peaks in y over a threshold and m is number of valleys in y over a threshold.

Choosing the thresholds is important for the performance of the method proposed. Surface metrology practice recommends a 10 percent of the maximum peak and/or valley height to be applied. This approach might introduce faults due to the way the extrema are calculated over the sample length, Rp is defined as the highest point of the profile while Rv is defined as the lowest point of the profile; however, the assumption that Rp or Rv are actually lying on a peak or valley is not always true as it can be observed in Figure 33 were $Rv = 1.4735$ and is the very last point in the profile. Therefore, we defined the peak threshold to be a percentage of the maximum peak detected and the valley threshold as a percentage of the maximum valley detected.

Average mean profile can be calculated using the toolbox functions previously presented. For consistency with the mean profile depth algorithm, a first order linear regression is applied to the profile:

```
dx = 0.345;  % sampling interval
t = 0.1;     % threshold percentage
Yi = rtm_fit(Y, dx, 1);
```

We need to know the values of thresholds applied,

```
[vmax, ~, vmin, ~] = rtm_extrema(Yi);
```

```
Tp = t*max(vmax);
Tv = t*min(vmin);
```

followed by peaks and valleys detection:

```
[vmax, imax, vmin, imin] = rtm_extrema(Yi, Tp, Tv);
```

Average maximum height is given by:

```
amh = mean(vmax) + abs(mean(vmin));
```

We have packed the algorithm in a single function

```
[amh, ap] = rtm_amh(Y, dx, t, 'plot')
```

The arguments passed to the function are **Y** as the segment profile, **dx** as the sampling interval and **t** as the threshold percentage. The last argument is optional, when included it will plot the profile. The function will output **amh** as the average maximum height and the amplitude parameters as a row vector, **ap**. An example of the output plot is shown in Figure 33 where the peaks are marked with a red circle, the valleys with a green circle and the mean values are represented with a dotted line.

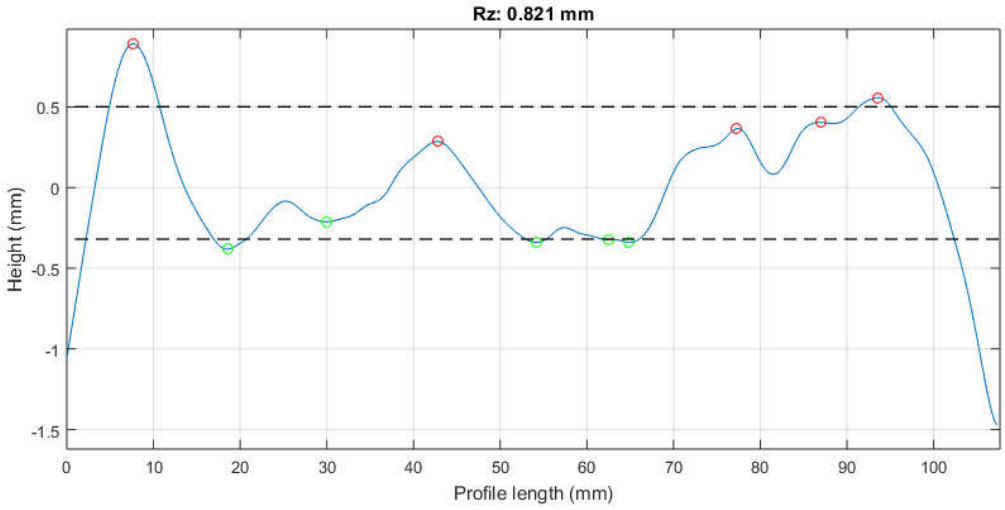


Figure 33. Average maximum height plot

Similar to the MPD algorithm, we have implemented a function that proceses a track obtained from the Circular Texture Meter. The profile track with outliers removed and filtered is divided in 8 segments; each segment has a linear regression applied.

```
[amhtrack, ap] = rtm_amhtrack (filename, dx, lc, t, 'plot')
```

The function returns the average Rz for the whole track in **amhtrack**, and each segment amplitude parameters in an 8 by 6 matrix, **ap**, in the following order: Rz, Ra, Rq, Rp, Rv, Rt. The last argument is optional and will display all segments of the track into subplots.

The second order polynomial fit mentioned in the MPD calculation is implemented for the amplitude parameters, rtm_amh2 and rtm_amhtrack2 respectively.

4.9. Complete track processing

In practice, the circular texture meter acquires two profiles, one in the forward direction and one in reverse. A small time delay in arm positioning assures the tracks are not fully mirrored. The profile measurements are saved in two files, named 'dataxxxxf.txt' and 'dataxxxxr.txt' for a single site, where 'xxxx' is the site number generated by the CTM firmware. We have written a function that will return all parameters for a site in a 16 by 7 matrix in the following order: MPD, Rz, Ra, Rq, Rp, Rv, Rt.

```
param = rtm_track (site, dx, lc, t, order, 'remout')
```

The arguments passed to the function are **site**, as the site name string, **dx**, **lc**, and **t** as described in mean profile depth and average maximum height algorithms; the order of the fit to be used is passed by **order** argument. The last argument instructs the function to replace the outliers from results with NaN values.

To be considered an outlier, a measurement will need to be larger than $Q3 + 1.5 \times IQR$ or smaller than $Q1 \quad 1.5 \times IQR$; where Q1 and Q3 are the 25th and 75th percentile and IQR is the interquartile range This method is generally accepted as a means of identifying outliers in statistics. As a note, the outliers are considered for each column of the matrix returned, therefore, it is not necessary to have the NaN values in the same row as each parameter has a different method of calculation. An example is shown in Table 1 were maximum valley height has an outlier in row 1 and maximum peak height has an outlier in row 14. Finally, the outlier removal algorithm is not applied recursively; that is, it will run only once, therefore, reapplying the rule with the resulting matrix might identify further outliers.

Table 1. Results from a site track processing

#	MPD	Rz	Ra	Rq	Rp	Rv	Rt
1	0.788	1.688	0.555	0.728	0.859	**NaN**	2.793
2	0.503	0.878	0.283	0.353	0.557	0.979	1.536
3	0.798	1.455	0.463	0.537	0.820	1.140	1.960
4	0.857	1.304	0.475	0.588	1.145	0.946	2.091
5	0.470	1.027	0.399	0.458	0.728	0.909	1.636
6	0.275	0.717	0.212	0.311	0.414	1.222	1.636
7	0.756	1.231	0.487	0.563	0.929	1.008	1.937
8	1.030	1.351	0.415	0.558	1.348	1.264	2.612
9	0.601	0.784	0.303	0.357	0.877	0.505	1.382
10	0.641	0.921	0.420	0.497	0.661	1.068	1.729
11	0.639	0.932	0.368	0.460	1.072	0.904	1.975
12	0.582	0.827	0.300	0.352	0.625	0.628	1.253
13	0.600	1.041	0.370	0.430	0.704	0.823	1.527
14	0.619	0.940	0.368	0.453	**NaN**	0.865	2.424
15	0.816	1.252	0.460	0.568	0.868	1.274	2.142
16	0.460	0.657	0.280	0.328	0.547	0.822	1.369

5. EMBEDDED SOFTWARE

At the heart of circular texture meter lies the CPU. The microcontroller powering the system is a general purpose, low power, 16-bit processor produced by Microchip. The part number is PIC24FJ256GA106 and it features large flash programming memory (256 kB) and the largest amount of RAM available for the range, 16 kB.

The embedded firmware is divided into a [1] *hardware layer* where physical control takes place, including pins switching as well as peripheral modules interaction such as timers, universal asynchronous receiver transmitter (UART) or serial peripheral interface (SPI); [2] *machine state layer* where the actions of the circular meter are performed; and [3] *application layer* where operations related to the data and parameter calculation are executed. Various functions or methods related to an action are generally grouped into a single C file, sometimes being associated with a header file.

Microchip provides C language compilers for their microcontrollers with a free licence, the trade-off being the level of optimisation of the assembly output. The firmware uses MPLAB XC16, version 1.24. The development was done in MPLABx IDE version 3.10, also a free resource from Microchip.

5.1. Hardware layer

We have used a combination of peripheral libraries available with the compiler for function initialisation and peripheral modules. The documentation for those is available in the complier installation folder, under peripheral libraries; generally, the libraries provide clearer information about the actions performed. When libraries were not available or the use of them would not benefit the code clarity, we have written directly to the registers.

5.1.1. Configuration

Setting up the microcontroller requires writing the configuration bytes with a logical 0 to select various features. PIC24FJ family devices have the configuration bytes implemented in the volatile memory, starting at the program memory F80000h. This is achieved by ‘configuration_bits.c’ using special compiler macros to write to the required memory address (this is out of user memory space).

For this application we have disabled the watchdog timer and configured the microcontroller to use an external oscillator as a clock source with PLL enabled; this achieves the fastest operation speed, 16 million instructions per second. We have also prevented writing to the program memory and enabled code protection. The oscillator requires configuration at run time, this is achieved writing to the OSCCON register in `ConfigureOscillator` routine.

The central point for header inclusion is 'system.h', also, here are declared the system frequency used by complier at compiling time to generate the assembly code for `__delay_ms` macro. Friendly names are defined here for peripherals IOs under 'hardware configuration' heading. Any other C file includes this system.h file to have a direct reference to the libraries used in project.

5.1.2. Application initialization

Any C program has an entry point defined by the `main()` routine, which we have placed in the 'main.c' file. In addition to the entry point, global variables are declared here. On power up, the first routine called by `main()` is oscillator configuration.

Next step requires configuration for the pins direction and start up state. This is coded into `InitApp()` routine which is called immediately after oscillator configuration. Some of the modules are using a special feature of the microcontroller, peripheral pin select, that enables mapping of an input and/or output to a module; a special compiler macro is used for this operation.

Finally, different modules used in application are initialized here, those are described in the following sections.

5.1.3. Universal asynchronous receiver transmitter objects

In general, the scope of the UART serial port is to handle ASCII character strings, data being used for human-machine interface and debug purposes; a third port has been reserved for a future addition of a GPS if required. Those transfers have been implemented similar to a PC serial port where the data stream received by module is buffered into a char array until a terminator character is received. All serial ports share a common structure, defined in 'serialobject.h':

```
struct UART_OBJECT{
    char         Terminator;
    char         Starter;
    unsigned int TxBusy;
    unsigned int BytesAvailable;
    unsigned int BytesCount;
    unsigned int BufferSize;
    COM_STEP     Step;
    char*        RxBuf;
    char*        TxBuf;
    void       (*FcnBytesAvailable)(void);
    COM_STATE    State;
    COM_FAULT    Fault;
};
```

Each module implementation is placed in a 'COMx_RS232.c' file with a corresponding header file present hosting serial port parameters and function prototypes. Where the application permits, the receiving part is handled by an interrupt service routine that examines the character received and decides to raise the `BytesAvailable` flag and run `FcnBytesAvailable` when the terminator character is received. Every time serial port activity is present, a corresponding orange LED is blinked.

Each serial port requires initialization on power up, this is handled within `InitCOMx()` routine, where x represents the serial port number. Two buffers, one for received string and one for transmitting string, are allocated at runtime from heap using `malloc` instruction; the size of each is defined by `COMx_BUFFER_SIZE`; failure to allocate memory results in permanent fault indicated by `LED_FAULT` on PCB. The pointer to the `FcnBytesAvailable` is also set here.

Serial port 1, declared as COM1 is dedicated to the HMI and is setup for 4800 bauds, 8N1. The serial LCD board requires a small protocol, where a command to the LCD is preceded by a null character; this is handled by the `writeLCD` routine. The arguments passed to the function are the memory address of the LCD and the string to be written to the LCD. The routine writes the null character, the address and then initializes an asynchronous transmission by enabling the UART1 transmit interrupt which transmits each character until it has seen the end of the string to be transmitted. The method uses less CPU cycles to handle this type of communications which is inherently slow.

The HMI board also captures the state of the operator button and transmits it continuously through UART as a single byte, therefore no terminator is used, instead each character received is mapped directly into the variable `hmiButtons`.

Serial port 2, declared as COM2, is used for debug purposes and is intended to only output strings to a PC. It is setup at 115200 baud, 8N1. Its `FcnBytesAvailable` is therefore summary implemented to return an idle state. Transmission through the port is achieved in a similar manner as described for the serial port 1, using UART2 transmission interrupts.

The last port, declared as COM3, is reserved for a future GPS implementation as is not yet implemented, however, the port is initialized and the resources allocated for RAM management purposes.

5.1.4. Timer objects

While short delays required by various processes can be coded with the built-in compiler macro `__delay_ms()`, longer delays that do not stop the execution of code are required. We have implemented a timer object that deals with this requirement using linked lists, placed in 'timers.c'. The structure of the object is defined in corresponding header file:

```
typedef struct timer_object {
    unsigned long PRE;
    unsigned long ACC;
    unsigned long LastTime;
    unsigned      EN    :  1;
    unsigned      TT    :  1;
    unsigned      DN    :  1;
    unsigned  unused    : 12;
    struct timer_object *Next;
} TIMER_OBJECT;
```

Timers operation requires initialization, this is handled by `InitTimers()` routine, where the timers are also initialized with the default values. The preset timer value, in milliseconds is loaded into PRE, the processing increases the ACC value when the enabled flag EN is true. When the accumulated value is greater or equal to the preset value, the DN flag is set. A time base of 1 ms is maintained through hardware timer 4 interrupt, which also updates the accumulated time for each timer that is enabled and set timer timing, TT, flag.

The circular texture meter uses 3 timers, a heart beating pulse, declared as `t1s` and is used to blink the green `LED_OK` to indicate continuous function, a startup timer, declared as `tStartUp`, used to display the splash screen on LCD while the circular texture meter is initializing and a sensor enable, declared as `tSensorEn`, used to allow the displacement sensor to start up before a profile is acquired.

5.1.5. Power supply monitoring with on board analog to digital converter

The system is monitoring the power supply voltage through the on board analog to digital converter module. Initialization of the module is performed in `InitADC()` routine placed in 'application.c' using the compiler peripheral libraries. The module converts the voltage presented at its pin mapped at port B, pin 2, into a 10-bit unsigned integer value; the result is converted into a voltage by `vbat()` routine. The parameters for conversion are hardcoded into this routine.

5.1.6. Analog displacement sensor input through SPI2

The displacement sensor output, either 0 to 10 volts or 4 to 20 mA, is converted to a voltage through on board analog circuitry. As the analog to digital capabilities of the microcontroller are not sufficient for this application, a high speed, 16-bit ADC external device (AD7694) is used to convert the output to a digital value The device outputs the converted value through serial peripheral interface and sampling/converting operation is controlled by its CNV pin.

The SPI2 module of the microcontroller is dedicated to the operation and is initialized in `InitSPI2()` routine placed in 'application.c' using the peripheral libraries. Converting the analog value to distance is coded in two stages into `displacement()` and `miliamps()` functions. The sampling interval is managed through hardware Timer 2, this is initialized by the machine state startup routine and is setup to interrupt the CPU every 2 ms, giving about 1200 samples per full rotation of the sensor arm. Enabling the timer interrupt triggers automatic conversion procedure. An orange LED is used to indicate module operation.

5.1.7. Motor control with pulse width modulation

Movement of the arm is achieved through an external half bridge driver and requires control of digital outputs pins for enabling and directions; those pins are configured in `InitApp()` routine. For speed control we have reserved the microcontroller PWM module to pulse the enable pin; this module is configured by `InitPWM()` routine and is using direct register writing.

5.1.8. SD card management

Implementation of profile data output from the circular texture meter is done with Microchip Libraries for Applications that provides the software, drivers and libraries to

manage a range of standard operational protocols such as USB, graphics, TCP/IP or File IO into a convenient C language interface. The Memory Disk Drive stack which deals with file operation on SD cards has two instances, supporting or not long file name functionality; we have chosen to support long file names.

The physical layer of the SD driver is using SPI 1; the initialization of which is managed by the driver; however, the library requires four user routines to map the physical state of IO pins: `USER_SdSpiConfigurePins_1()`, the pin directions and power up state are set here; `USER_SdSpiSetCs_1()`, sets the chip select pin of the SPI 1 module; the card detect and write protect pins state are managed through `USER_SdSpiGetCd_1()` and `USER_SdSpiGetWp_1()` routines. All user routines are placed in 'system.c' file. The initialization of the library is completed in `main()` after the rest of the modules are initialized. At run time the state of the SD card operation is monitored by polling the state of the card detect pin. The operational functions of mounting/unmounting the drive and handling of the faults are coded in `PollSDCard()` routine placed in 'main.c' file.

The library has features that can be user configured to suit the application. The configuration macros are placed in 'fileio_config.h' file. For the CTM we have disabled the formatting and the use of multiple drives options as these are not used. Time-stamping of written files is using a pseudo real time generator as we did not implement RTC for this system.

Operation of SD card is indicated with an orange LED connected to the chip select pin of the SPI 1 module.

*

As a general note, the hardware layer is initialized only once at the start of the run time, the operation is controlled by a forever loop placed inside the `main()` routine where the rest of operations are performed.

5.2. Machine state layer

The circular texture meter is programmed to perform its actions as a finite state machine. The machine state is initialized in `InitMachineState()` routine placed in the 'machinestate.c' file; this routine does not initialize any physical modules as those should

be completed first, with the exception of the hardware Timer 2 that is used for sampling operations. The state and fault strings descriptions are stored in flash memory as constant strings. To access those we have defined an array of pointers holding the address at which the strings are held:

```
char *rtmState[RTM_STATES];
char *rtmFault[RTM_FAULTS];
```

The initialization of those pointers is made in `InitMachineState()`, the strings are used in conjunction with state and faults enumerations to display the correct state (or fault) message on LCD screen. The structure of the machine control is

```
typedef struct {
    unsigned int Samples;   /* number of samples in a data set */
    unsigned int Sample;    /* samples counter */
    unsigned Run                :1;
    unsigned ProcessComplete    :1;
    unsigned WriteComplete      :1;
    unsigned Direction          :1;
    RTM_STATE   State;
    RTM_FAULT   Fault;
}RTM_OBJECT;
```

where the `Samples` is the number of samples per profile, `Sample` is the current sample being acquired. `State` and `Fault` are enumerations typedefs. `Run`, `ProcessComplete`, `WriteComplete` and `Direction` are flags used in operation.

5.2.1. State descriptions and transitions

At run time, the machine state operations and transitions are managed by polling the trigger and events raised by hardware layer or circular texture meter operations. The polling is coded in `MachineState()` routine.

Normal operation requires 10 states, those are shown in Figure 34. During the initialization stage, the machine is placed in *Power On* state; this state is used only once. The transition to the next step is made in the first call to `MachineState()` when the CTM is placed in an *Idle* state.

Transition from *Idle* state to *Sensor Enable* is made when the operator button is pressed. This action resets the sensor enable timer, clears the current sample counter and flags the HMI to update.

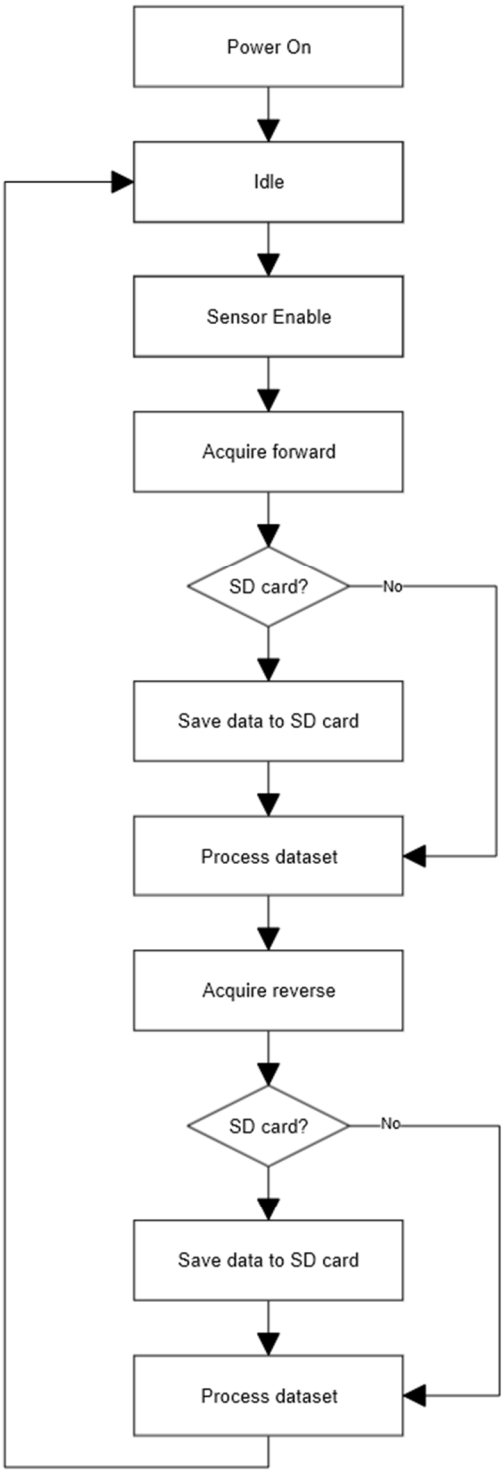


Figure 34. Machine states

Sensor enable state introduces a delay in the profile acquisition process to allow for the displacement sensor to start. The delay is completed when the `tSensorEn.DN` flag is set. Before transitioning to the next step, the `Run` flag is set.

In *Acquire forward* state the forward profile points are acquired and placed in `data[]` array. The transition to the next step is done when the `Sample` counter is equal or greater than number of samples in the profile. If a SD card is present and mounted then the next operation is to write to SD card, otherwise the flow skips to processing data. Before transitioning the `WriteComplete` and `ProcessComplete` flags are cleared.

Write data forward state is transitioned to the processing stage when the writing of the data to the SD card is completed.

In *Process data forward* state, the CPU is performing the numerical calculations of the profile parameters, the transition to the next step is made when the `ProcessComplete` flag is set.

The last three states are then repeated for the reverse profile. On completion of the *Process data reverse* state the HMI is updated with the results and the CTM waits for the operator to acknowledge them by pressing the operator button.

If, for any reason, during the operation a fault is raised, the machine state is transitioned to the *Faulted* state. In this situation, the HMI will update the state description and the fault reason and wait for operator acknowledgment through the button. Transition from this state is always to the *Idle* state.

5.2.2. Modules outputs control

At the end of machine state polling, the actions required by a particular state are performed, those are coded in `MachineStateOutput()` routine.

The motor is enabled in all states except the power up; doing this allows for the arm to be braked when stopping. The forward and reverse inputs in motor driver are set when the machine is in *Acquire forward* or *Acquire reverse* state. The interrupt for Timer 2 is enabled in those states.

The relay allowing the power supply is enabled in steps between *Sensor Enable* and *Processing data reverse*. Profile parameters calculations are also initiated from this routine when the state is either process forward data or process reverse data.

Due to the nature of Microchip Libraries for Applications, file operations related with the machine state are coded within `main()`; also, the `WriteComplete` flag is set here once the operation are completed.

5.2.3. HMI messaging

A protocol needs to be followed for a successful HMI messaging: the backpack will transfer the received character as soon it is received by its serial port. The HD44780 LCD controller will increment the address for the next character received, therefore little intervention from end user is required. However, when a new address is required a command to set the CGRAM address is required. To achieve this, the backpack makes use of the fact that none of the commands accepted by HD44780 controller is null, therefore, when the backpack receives a null character it will leave the register select pin low when the next character is transferred to the LCD pins. Typically, printing a character and most of the commands takes less than 40 microseconds therefore the screen is updated before the next character is received. Exception to this are the clear display instruction (01h) and cursor home (02h) that needs more than 1.5 ms to execute.

The mechanism is handled by a set of routines extending serial port COM1 implementation. `writeLCD()` routine accepts two arguments, the CGRAM address and a pointer to the string that will be passed to the HMI; the start of the line address is defined in RS232h. The `cls()` routine instructs the backpack to clear the display.

Due to the slow rate of update of the LCD screen, the communications with the serial backpack are taking place at 4800 bps, to update a full screen every scan would take most of the processor time. To accommodate for this, the HMI is updated only when an event has taken place, signaled by `UpdateHMI` flag which is polled inside the main loop.

The information is organized in display pages those being managed by `DisplayPage()` routine placed in ‘hmi.c’ file. The routine takes one argument in the form of an enumerated variable, `HMI_PAGE` declared in ‘hmi.h’.

Page zero is the splash screen shown on powering up the system; doing so it provides feedback on the successful hardware initialization and confirms the LCD backpack is working.

Page one is the most used by the normal CTM operation. In the first line it displays the current state using a string array saved in the flash memory:

```
sprintf(buf, "State: %s", rtmState[pRTM.State]);
writeLCD(LcdLine1, buf);
```

The second line is reserved for fault messages, if no fault is present then the line is left blank. The forth line is used for system status display: the presence of SD card is indicated by 'SD' letter in the first two addresses or '--' is displayed. The voltage of the battery is shown in the next field. The last field indicates the file number that is will be written if the SD card is present. Line three is not used by this screen and is left blank.

Page two shows the profile parameters results in the third row and estimated texture depth in the fourth row as shown in the example below:

```
RESULTS (val/est)
MPD  Rz   Ra   Rq
0.23 0.23 0.23 0.23
1.23 1.23 1.23 1.23
```

In the next section we will address the numerical processing stage.

5.3. Application specific libraries

Digital signal processing and some of the Matlab toolbox libraries we constructed have been ported for embedded C for profile processing. The fitting part of the algorithm makes use of least square polynomial data fit, a simple operation in Matlab, but not so in C which lacks such libraries. To overcome this, we have adapted a number of linear algebra problems for Microchip embedded C compilers using as guideline principles laid in *Numerical recipes C. The Art of Scientific Computing* (Press, Tuekolksy, Vetterling, & Flannery, 2002).

The routines dealing with numerical processing are placed in four files: [1] 'nr_utils.c' which contains the common techniques used by *Numerical recipes* to allocate vectors and matrices from heap memory and free the resources. For embedded systems we have added messaging to the debug port for faults that might be raised when insufficient memory is

available for operations. [2] 'linsolve.c' holds routines used in solving linear equations; [3] in 'polyfit.c' we have coded routines performing functions similar to the Matlab polyfit command. [4] Fourth file, 'prtm.c' contains the methods specific to the road texture processing.

The profile points are stored in a 32-bit double array, `data[]`; the size of the array is equal to the number of samples plus the size of the filter used. Due to memory restrictions placed by the limited amount of RAM available for the application, the profile data array memory is reused, thus altering the original profile; the saving of raw data on SD card has to take place before any processing. During the profile acquisition the points obtained are placed at an index offset by a number equal to the filter size, the final profile is padded later in the filtering phase.

Two tracks are processed for a site, one in forward direction and in reverse; following the current practices, each profile is divided in 8 segments and profile parameters are calculated for each segments. The results are placed in 32-bit double arrays corresponding to each parameter: MPD, Rz, Ra, Rq, Rp, Rv, and Rt.

In a similar manner that our Matlab toolbox operates, one profile is processed within a method, `track()`. This method takes as argument the number of the track processed, 1 or 2, and returns null if processing has finished without errors or -1 if errors has been raised. The argument is used to instruct the method where to place the results in theirs arrays. Within this routine a variable x[], is declared and initialized representing the independent variable used with the polynomial fitting algorithms.

5.3.1. Raw profile outlier removal

Identification of sensor readings that are outliers follows closely the algorithm we have written for Matlab toolbox. The routine, `remout()` returns the number of outliers found, and updates the data[] array with a version that has the outliers removed.

```
int remout(double *Y, int n, double T, double dx)
```

The routine takes as arguments a pointer to profile array, n as the size of the array; T is the threshold over the change in signal is considered an outlier, and dx is the sample interval. The method returns -1 if the maximum permitted number of contiguous outliers has been reached; this is defined in `MAX_OUTLIERS` macro in 'prtm.h' as 15. In this case,

the track processing is aborted and the machine state is updated to *Faulted* with 'Too many outliers' fault shown on the HMI.

If outlier removal is successful, `track()` checks if the maximum percentage of outliers has been reached and sets 'Too many errors' fault if this is the case. This threshold has been defined in `MAX_ERRORS` macro as 10% of the sample size.

It should be noted that when the method is called, the pointer to the Y is offset by the filter size for memory management reasons.

5.3.2. Profile low-pass filtering

The moving average filtering is a common digital signal processing operation and is implemented by `wfilter()` function:

```
void wfilter(double *Y, const double *W, int n, int o)
```

The function takes as arguments pointers to the data profile array, Y; the pointer to the window filter, W; the sizes of the arrays are passed in n and o respectively.

Prior to filtering, the routine is padding the left hand side of the array with a copy of the first o-1 elements from the profile in order to avoid the output distortion. As the application uses the same filter, this has been hardcoded in CPU flash memory.

The resulting profile is shifted to the left so the first element in data[] array contains the first element of the filtered profile. When the method is called, the pointer to Y points to data[0].

5.3.3. Fitting

Next stage of processing involves dividing the profile into segments and calculating the residuals which is placed inside a for loop in the `track()` routine. Each segment has a polynomial fitted to the data and the residuals form the new profile. The algorithm employed for fitting is solving for a in the polynomial regression $y = aV + e$ where V is the Vandermonde matrix obtained from the regression model $y_i = a_0 + a_1 x_i + a_2 x_i^2 + + a_m x_i^m + e_i$, a is vector of coefficients from the model and e is the errors vector.

The algorithm is implemented in two steps, first we build the Vandermonde matrix and the vector y, then we solve the resulting linear system. The routine implementing the above algorithm is

```
inline void pfit(double *p, double *x, double *y, int nv, int k)
```

The arguments passed are: p, pointer to the array of doubles where the coefficients of the polynomial are returned; x, pointer to the independent variable, y, pointer to the dependent variable (the profile); nv, the number of elements in x and y and k is the order of the fit. It should be noted that in line with *Numerical recipes* approach, the first element in p is 1 as 0 is not used.

The routine first calculates the sums of the Vandermonde matrix and places them in the vector `svm`. The function that returns this is

```
inline void svander(double *svm, int nsvm, double *v, int nv)
```

where svm is the pointer to the array of doubles where the sums are returned, nsvm is the number of elements in svm; v is the pointer to the array from which the sums are computed and nv is the number of elements in v.

`svander()` employs another function, `double sump(double *v, int n, int p)` that returns the sum of elements in v raised at the power p.

The matrix V in the algorithm is constructed with

```
inline void lsqra(double **a, double *svm, int n)
```

where the arguments passed to the routine are: a, pointer to the array of pointers representing V; svm is the pointer to the vector calculated in the previous step and n is the number of elements in a.

The vector y is returned by

```
inline void lsqrb(double *b, int nb, double *x, double *y, int n)
```

in the array pointed by b. The other arguments passed to the routine are nb, the size of b, x and y as pointers to the independent and dependent variables and n as the size of those.

So far the routine creates the matrix and the vector in the form required by the regression. We solve the linear equation by employing a numerical analysis technique commonly used in computing: LU decomposition or lower upper decomposition. This technique factors a square matrix, V in our case, into the product of a lower triangular matrix with an upper triangular matrix, the resulting matrix the process being a form of Gaussian elimination. The technique is coded using row permutations technique, known as LU factorization with partial pivoting, in

```
inline void ludcmp(double **a, int n, int *indx, double *d)
```

The function takes `a` as the pointer to the array of pointers representing the Vandermonde matrix, n is size of `a`. The factorized matrix is returned in `a`. The index vector records the row permutations effected by the partial pivoting and d is output as ±d, depending on whether the number of row interchanges is even or odd.

The system is then solved by the `ludcmp()` counterpart routine:

```
inline void lubksb(double **a, int n, int *indx, double b[])
```

which returns the coefficients of the fit in b[].

The fitting process code is exemplified below:

```
/* a B vector isn't necessary, the routine uses the *p vector
   for intermediate calculations */
svander(svm, k*2+1, x, nv);
lsqra(a, svm, k+1);
lsqrb(p, k+1, x, y, nv);
ludcmp(a, k+1, indx, &d);
lubksb(a, k+1, indx, p);
```

Last task for the fitting part is to calculate the residuals of the fit and return them in the profile data; this is accomplished using two routines, `pval()` and `pres()`:

```
inline double pval(double x, int n, double *p, int k)
inline void pres(double x, double *y, int nv, double *p, int k)
```

`Pval()` returns the n-th value of a polynomial function whose coefficients are stored in p. The independent variable is assumed to be equally spaced at interval x and the first element of x is zero; k is the order of the polynomial held in p.

`Pres()` replaces y with the residual values computed from the polynomial p whose order is held in k.

5.3.4. Profile parameters

The average roughness, Ra; average square roughness, Rq; the maximum peak height, Rp; maximum valley height, Rv and total profile height Rt can be calculated from the resulting profile. We have constructed the corresponding functions that return those parameters defined as:

```
double ra(double *Y, int n)
double rq(double *Y, int n)
double rp(double *Y, int n)
double rv(double *Y, int n)
double rt(double p, double v)
```

With the exception of `rt()`, all functions take as argument the pointer to the segment data and the size of the segment; `rt()` simply returns the sum of p and v.

If Rt returns a value larger than `MAX_RT`, defined as 15.0 in 'prtm.h' then the processing is aborted and 'Measurement failed' fault is raised.

Finding the mean profile depth and average maximum height necessitates finding the peaks and valleys of the profile; this task has been ported from the Matlab toolbox into:

```
int extrema(double *vmax, int *imax, int *nmax, double *vmin, int *imin, int *nmin,
     double *Y, int n, double Tp, double Tv)
```

The arguments passed to the function are vmax as pointer to the array of doubles where the peaks values are placed, imax as pointer to the array of the indices were the peaks were found, nmax as pointer to the total number of peaks found, vmin, imin and nmin have a similar meaning for the valleys identified; Y is pointer to the segment, n is the size of the segment; Tp and Tv are the peak and valley thresholds.

It is possible for `extrema()` to return one of the vectors with zero length if the shape of the profile is U shaped (or inverted U), if this is the case the routine returns -1 or -2 respectively. A corresponding fault is raised by the machine state mechanism and the track processing is aborted.

The mean profile depth is returned by

```
double mpd(double *Y, int n)
```

where Y is the pointer to the data segment and n is the size of the segment. The algorithm is implemented as described by ASTM E965 (ASTM International, 2015); however, the

standard assumes that each half of the profile contains at least a peak; our practice has had instances where this wasn't the case. In this case, the maximum profile height is returned by `mpd()`, otherwise the peaks for each profile is used.

The average maximum profile height is calculated by

```
double rz(double *Y, int n, double Tp, double Tv)
```

where Y is the pointer to the data segment and n is the size of the segment. The peak and valley thresholds are passed in arguments Tp and Tv respectively. This routine is also using `extrema()` to find the peaks and the valleys of the profile. As an error detection mechanism, the functions returns -1.0 if no peaks are detected and -2.0 if no valleys are detected.

The segment processing part is exemplified below (we have removed debugging code):

```
for (i=0; i<NO_OF_SEGMENTS; i++) {
    /* fit linear regression to segment */
    pfit(p, x, &data[SEGMENT*i], SEGMENT, ORDER_OF_FIT);

    /* calculate residuals */
    pres(SAMPLE_DISPLACEMENT, &data[SEGMENT*i], SEGMENT, p, ORDER_OF_FIT);

    Ra[i+t] = ra(&data[SEGMENT*i], SEGMENT);
    Rq[i+t] = rq(&data[SEGMENT*i], SEGMENT);
    Rp[i+t] = rp(&data[SEGMENT*i], SEGMENT);
    Rv[i+t] = rv(&data[SEGMENT*i], SEGMENT);
    Rt[i+t] = rt(Rp[i+t], Rv[i+t]);

    /* sanity check, the Rt should be less than 10 mm, failure to do so
     * is an indication of bad cleaning of data */
    if(Rt[i+t]>MAX_RT) {
        pRTM.State = rtmSateFaulted;
        pRTM.Fault = rtmFaultMeasurementFailed;
        return -1;
    }

    /* find thresholds */
    er = extrema(vmax, imax, &nmax, vmin, imin, &nmin, &data[SEGMENT*i], SEGMENT, 0, 0);
    if(er == 0) {
        Tp = TH * maximum(vmax, nmax);
        Tv = TH * minimum(vmin, nmin);
    }
    else if (er == -1) {
        pRTM.State = rtmSateFaulted;
        pRTM.Fault = rtmFaultTooManyPeaks;
        return -1;
    }
    else {
        pRTM.State = rtmSateFaulted;
        pRTM.Fault = rtmFaultTooManyValleys;
        return -1;
    }
```

```
    /* Average maximum height */
    Rz[i+t] = rz(&data[SEGMENT*i], SEGMENT, Tp, Tv);

   /* Mean profile depth */
    MPD[i+t] = mpd(&data[SEGMENT*i], SEGMENT);
}
```

5.3.5. Reporting results: parameters mean and estimated texture depth

The last operation the firmware performs is to report the parameters. We have chosen to display only the mean profile depth, the average maximum height, roughness average and the average square roughness as those are most relevant for the application.

The circular texture meter reports two values for each parameter, one is the mean of all 16 segments obtained; the other one is the estimated texture depth. The means are computed by `meanParameters()`, which simply packs four calls to the routine that returns the average of an array:

```
double mean(double *d, unsigned int n)
```

where d is the pointer to the array of doubles and n is the number of elements in n.

Estimation of the texture depth is following the note made by Elunai et al. (Elunai, Vinod, & Edith, 2011) stating that a texture depth measurement method can be correlated with the sand patch test method by the slope, α, and intercept, β, of a line that fits the best r-square. The slope and intercept are hardcoded in microcontroller flash memory for each parameter and is returned by `estimateDepth()` method, which simply updates the estimates:

```
    etdMPD = aMPD * meanMPD + bMPD;
    etdRz = aRz * meanRz + bRz;
    etdRa = aRa * meanRa + bRa;
    etdRq = aRq * meanRq + bRq;
```

Results are displayed with 2 decimal point precision on the LCD screen until the start button is pressed.

6. Results

As mentioned, the parameters obtained with the circular texture meter can be correlated with the sand patch texture depth using slope, α, and intercept, β, of a line that minimises the mean root square error. The current practice of correlating the parameters observed in literature is to take a series of measurements from a diversity of sites and plot the results obtained against the sand patch depths taken from the same site. While this approach is widely accepted, we have identified a few shortcomings: [1] the parameters returned from sites with similar sand patch depths are noticeably varying; [2] the number of points measured for the same sand patch depths is somehow small; and [3] only one sand patch depth is taken from a site despite poor reproducibility noted in literature. Considering those, we have adopted a different method of estimating the slope and intercept of the correlation line.

Our method consists in measuring a flat surface with a microtexture small enough to be filtered by our algorithms that will return the intercept β for each parameter. For line fitting we need at least another point to be evaluated. We have obtained this point taking measurements from a well defined road surface with the size of one meter square. The *x* variable, representing the sand patch texture depth, is derived from the mean of multiple sand patch sampling equally spaced in the square. The *y* variable, calculated separately for each parameter, is obtained by averaging the segments resulting from taking multiple profile tracks within the sampled area. This method considers variations in both sand patch texture depths and the parameters returned by the circular texture meter. We have taken 9 samples of sand patch depths to determine *x* and 752 segments to determine *y*.

We are interested in the performance of the mean profile depth (MPD) and three amplitude parameters, average maximum height, Rz, average roughness, Ra and average square roughness, Rq. The first and last parameters are accepted methods of parametrising road surface texture; Rq is similar to the sensor measured profile depth, the difference being the sampling interval and the length of the profile. Rz is a modified parameter adapted from surface metrology practice.

The flat surface used to estimate the intercept is the laminated top of our workbench lab. We have recorded 10 tracks slightly moving the circular texture meter between

measurements to obtain a random distribution of the results. The 160 segments resulted are used to estimate the intercept for each parameter. We have used both first order and second order fitting to the segment profiles.

As mentioned, for slope estimation we carried out measurements over an area of 1m square. The road portion chosen is the inner wheel path of a street with low vehicle traffic where the chip seal has not been replenished in the last 10 years, thus the embedment of chips in binder is well settled. The area presents damage to some extent such as flushed surface and dis-embedded chips from the substrate which would be a normal occurrence for a surface with this age.

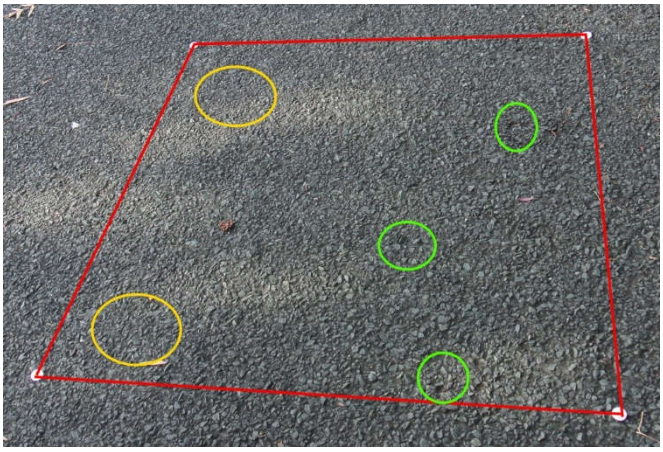


Figure 35. Square meter test. Test area enclosed in red. Flushed surfaces are encircled with orange while missing chips areas are encircled with green.

The sand patch test was performed as prescribed by New Zealand Transport Agency, distributed in 3 rows and 3 columns over the surface, the first row and column being at the bottom right in the **Figure 35**, columns progressing upward and rows leftward. The results obtained are presented in Table 1. It can be noted that the missing chips areas from the test area are reflected in the depths obtained, row 1, column 1 and 2 while the flushed areas are evident in column 3, rows 2 and 3.

Prior to the sand patch test we have acquired profiles with the circular texture meter in order to avoid contamination with sand. The area was cleaned of debris and two profiles were acquired, one in clockwise direction, and one in counter clockwise direction at about 100 mm intervals until the whole test surface was acquired. One profile track consists of 2500 points at a sampling interval dx of about 0.345 mm.

The mean depth of the sand patch data set sample is found to be $\bar{x}_{SP} = 1.9835$ mm with a standard deviation of $s_{SP} = 0.1616$ mm. Assuming a normal distribution, we would expect 95% of the measurements to fall between 1.6603 and 2.3067 mm. The relative standard deviation is r$sd_{SP} = 0.0815$ mm representing an 8.15% coefficient of variation.

The defects in the measured area are well represented by measurements, marked by a 2.3187 mm depth where aggregates were dislodged from the binding substrate (column 1, row 1) and around 1.8 mm where the area was flushed (column 3, rows 2 and 3).

Table 2. **Mean texture depths (mm) by sand patch method for the square meter test**

	Column 1			Column 2			Column 3		
	Diameter		MTD	Diameter		MTD	Diameter		MTD
Row 1	155	160	**2.3187**	165	170	**2.0243**	175	165	**1.9827**
	155			167.5			170		
Row 2	160	170	**2.1047**	170	175	**1.9256**	190	170	**1.7685**
	165			172.5			180		
Row 3	170	170	**1.9827**	175	170	**1.9256**	180	175	**1.8187**
	170			172.5			177.5		

The bounds of 95% confidence intervals for this sample mean and standard deviation are:

	μ	σ
lower bound	1.8593	0.1091
upper bound	2.1077	0.3095

6.1. Mean profile depth

The mean profile depth descriptive data is presented in Table 3 for the intercept calculation and in Table 4 for the square meter test. For visual analysis, we have used a Matlab box plot to show the quartile ranges and one histogram for each fit used; the bin widths are 0.01 mm for intercepts and 0.05 mm for square meter test. The plots are presented in Figure 36 for intercept and Figure 37 respectively for square meter test.

While the difference of the means for the flat surface is not very large, the differences between distributions' histograms is significant as can be observed in Figure 36. We attribute this to the waviness of the tracks arising from unavoidable misalignments between measured plane and sensor plane; a first order fit will fail to remove the waviness

in some of the cases. The behavior is also present in square meter test histograms although the differences in distribution are not as significant.

It should be also pointed that measuring the flat surface might return negative results for MPD; which would not be expected. This is due to the way the mean profile depth is calculated by our Matlab toolbox: the algorithm considers the peaks identified in the profile rather than the largest values in the segment (see Figure 31 where the peak at about 70 mm is negative), therefore if a first order fit is used, it is possible for the extrema algorithm to identify peaks with negative values. If a second order fit is used, this situation does not occur.

Table 3. Mean profile depth results from a flat surface

	First order fit	Second order fit
Mean	0.0514	0.0229
Standard deviation	0.0448	0.0211
Relative standard deviation	0.8708	0.9231
Mean confidence interval	0.0444	0.0196
	0.0584	0.0262
Standard deviation confidence interval	0.0404	0.0190
	0.0503	0.0237
First quartile Q1	0.0222	0.0116
Third quartile Q1	0.0731	0.0238
Median	0.0408	0.0176
Interquartile range IQR	0.0509	0.0122

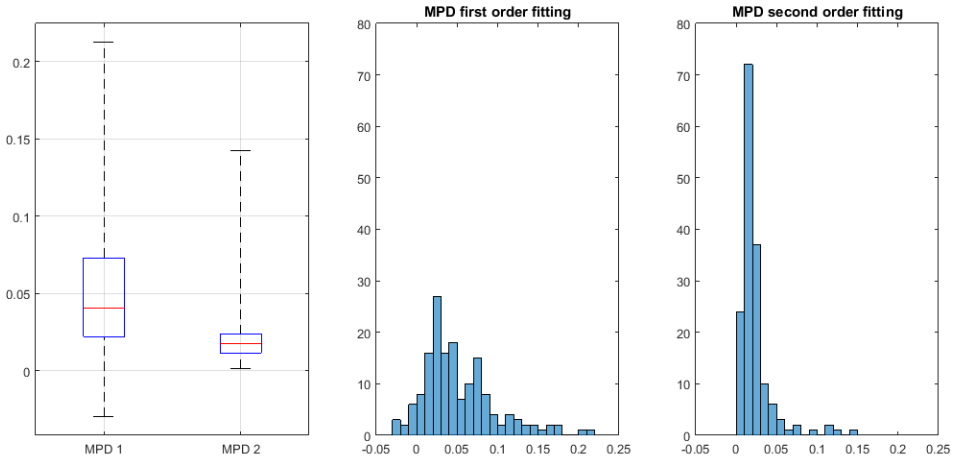


Figure 36. MPD box plot (left) and histograms (bin width = 0.01) for a flat surface data.

Table 4. Mean profile depth results from square meter test

	First order fit	Second order fit
Mean	0.9616	0.8631
Standard deviation	0.3173	0.2879
Relative standard deviation	0.3300	0.3336
Mean confidence interval	0.9389	0.8425
	0.9844	0.8837
Standard deviation confidence interval	0.3020	0.2741
	0.3342	0.3033
First quartile Q1	0.7381	0.6671
Third quartile Q1	1.1504	1.0253
Median	0.9166	0.8283
Interquartile range IQR	0.4123	0.3582

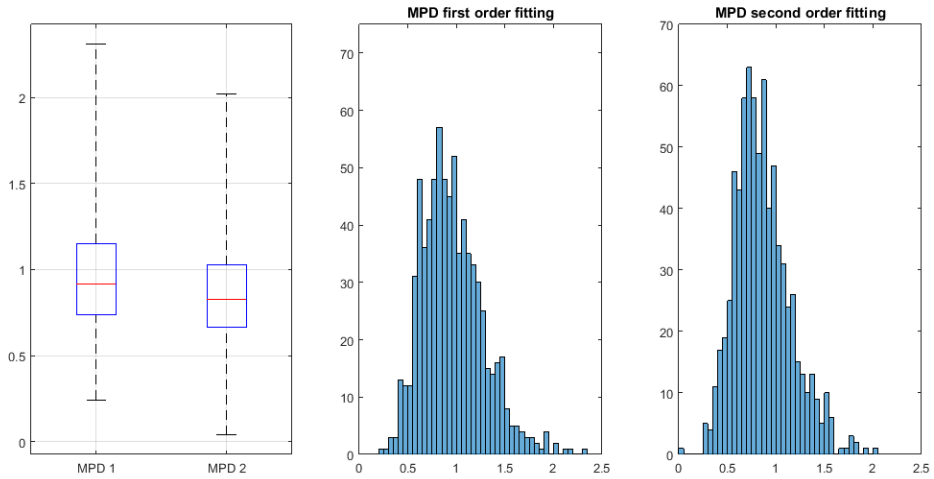


Figure 37. MPD box plot (left) and histograms (bin width = 0.05) for square meter test.

6.2. Average maximum height

In comparison with MPD, the mean intercept value for average maximum height is larger for both first order and second order fits; as it can be observed in Table 5. The distribution histograms show the same pattern observed for the MPD where the second order fit has a much smaller distribution compared with the first order fit, this is shown in Figure 38. The presence of possible outliers is noted for the second order fit where, in fact, some of the segment values has increased compared with the first order fit.

The square meter results, presented in Table 6, reveal a mean much closer to the sand patch depth mean value, 1.53 mm for the first order fit and 1.35 mm for the second order fit, however, the interquartile range between the fits is not differing significantly. A visual comparison between square meter histograms shows an improved distribution of segments values.

Table 5.Average maximum height results from a flat surface

	First order fit	Second order fit
Mean	0.0847	0.0435
Standard deviation	0.0369	0.0275
Relative standard deviation	0.4359	0.6319
Mean confidence interval	0.0789	0.0392
	0.0904	0.0478
Standard deviation confidence interval	0.0333	0.0248
	0.0415	0.0309
First quartile Q1	0.0542	0.0291
Third quartile Q1	0.1059	0.0497
Median	0.0834	0.0388
Interquartile range IQR	0.0517	0.0207

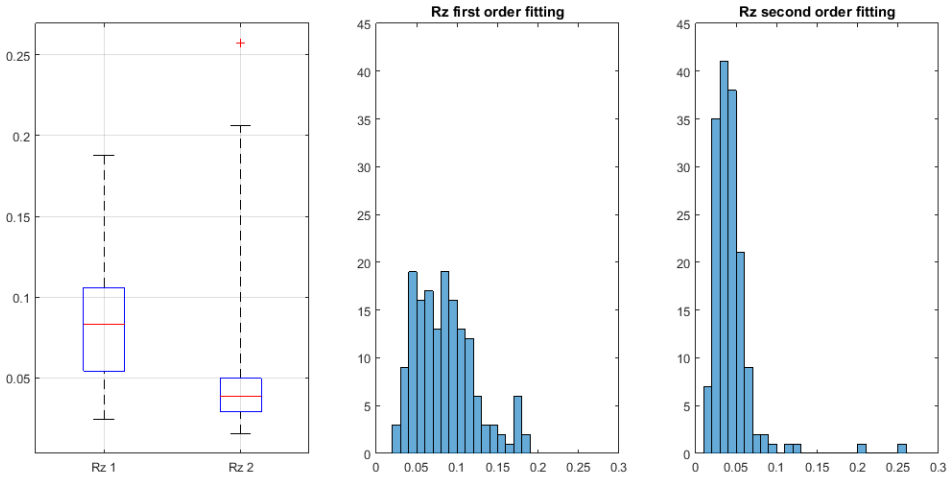


Figure 38.Rz box plot (left) and histograms (bin width = 0.01) for a flat surface data.

Table 6. Average maximum height results from square meter test

	First order fit	Second order fit
Mean	1.5254	1.3527
Standard deviation	0.4987	0.4512
Relative standard deviation	0.3269	0.3335
Mean confidence interval	1.4897	1.3204
	1.5611	1.3850
Standard deviation confidence interval	0.4747	0.4295
	0.5253	0.4752
First quartile Q1	1.1797	1.0307
Third quartile Q1	1.7889	1.5932
Median	1.4454	1.2744
Interquartile range IQR	0.6093	0.5626

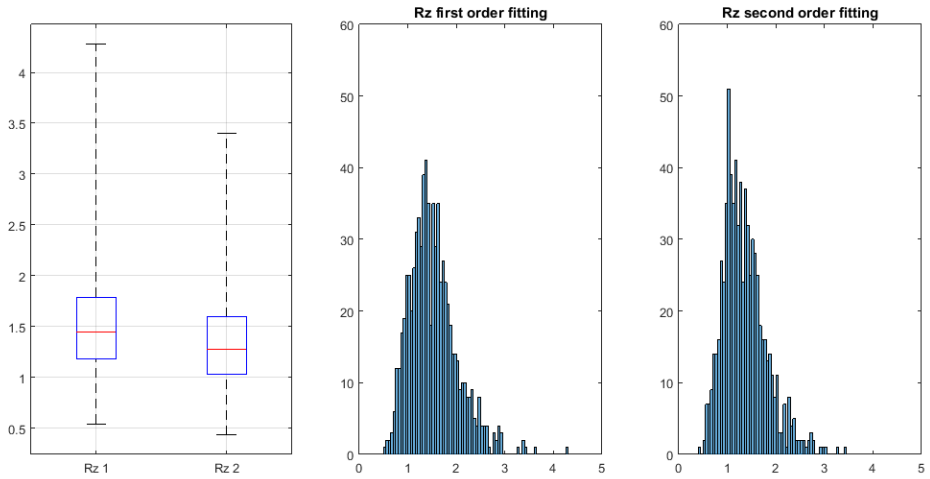


Figure 39. Rz box plot (left) and histograms (bin width = 0.05) for square meter test.

6.3. Average roughness

The means for average roughness intercept and square meter test – Table 7 and Table 8 – are the smallest from the parameters tested, first order mean intercept being 0.037 mm and the square meter test being 0.515 mm. As a result, the number of histogram bins is much reduced for both fits as in can be seen in Figure 40 and Figure 41.

The square meter test histograms in Figure 41 are showing little improvement in distributions between first and second order fit; however, it should be noted that the median and mean values for both fits have close values.

Table 7. Average roughness results from a flat surface

	First order fit	Second order fit
Mean	0.0368	0.0146
Standard deviation	0.0159	0.0062
Relative standard deviation	0.4316	0.4206
Mean confidence interval	0.0343	0.0137
	0.0393	0.0156
Standard deviation confidence interval	0.0143	0.0055
	0.0178	0.0069
First quartile Q1	0.0221	0.0106
Third quartile Q1	0.0517	0.0175
Median	0.0347	0.0137
Interquartile range IQR	0.0296	0.0068

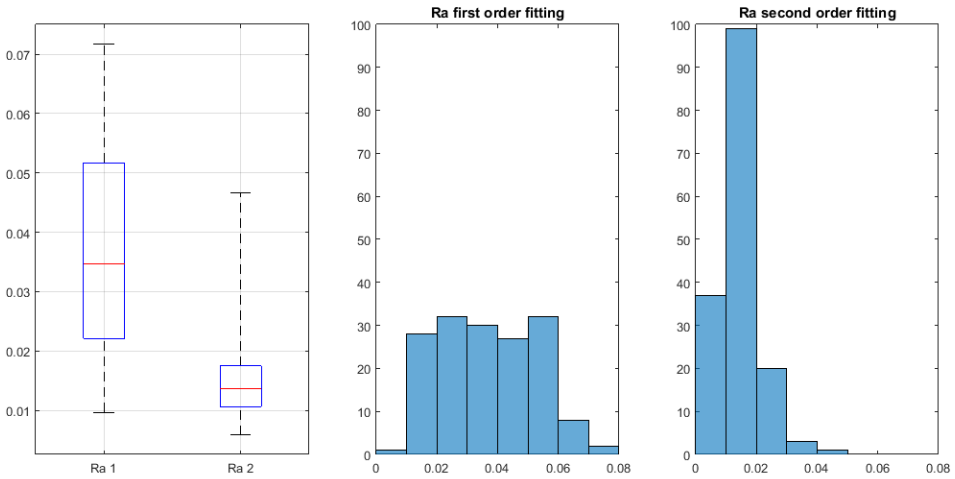


Figure 40. Ra box plot (left) and histograms (bin width = 0.01) for a flat surface data.

Table 8. Average roughness results from square meter test

	First order fit	Second order fit
Mean	0.5146	0.4481
Standard deviation	0.1583	0.1384
Relative standard deviation	0.3076	0.3088
Mean confidence interval	0.5032	0.4382
	0.5259	0.4580
Standard deviation confidence interval	0.1507	0.1317
	0.1667	0.1458
First quartile Q1	0.4013	0.3480
Third quartile Q1	0.6067	0.5220
Median	0.4889	0.4299
Interquartile range IQR	0.2054	0.1741

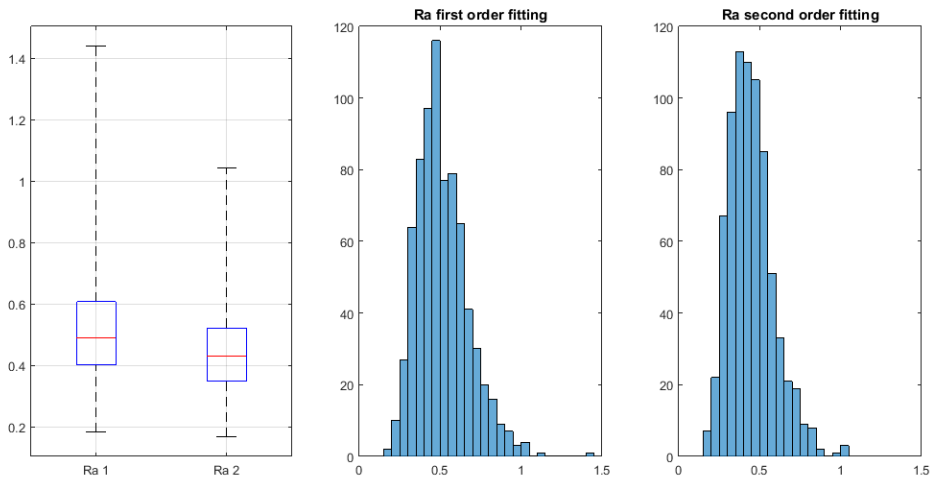


Figure 41. Ra box plot (left) and histograms (bin width = 0.05) for square meter test.

6.4. Average square roughness

Average square roughness data results, presented in Table 9 and Table 10, are following closely those of average roughness with slightly larger values obtained for both intercept and the square meter test; behavior also confirmed graphically from the histogram plots in Figure 42 and Figure 43 respectively.

Table 9. Average square roughness results from a flat surface

	First order fit	Second order fit
Mean	0.0439	0.0183
Standard deviation	0.0181	0.0076
Relative standard deviation	0.4134	0.4163
Mean confidence interval	0.0411	0.0171
	0.0467	0.0195
Standard deviation confidence interval	0.0163	0.0069
	0.0204	0.0086
First quartile Q1	0.0273	0.0134
Third quartile Q1	0.0603	0.0214
Median	0.0412	0.0175
Interquartile range IQR	0.0331	0.0080

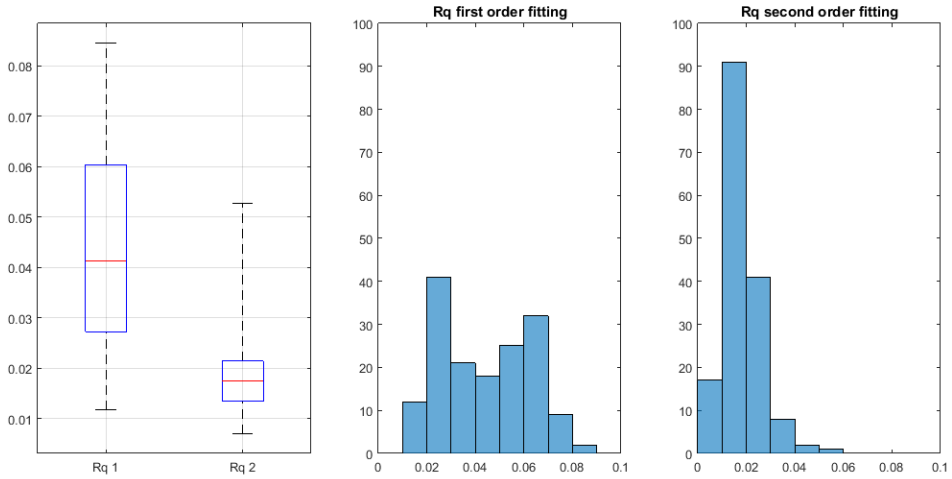


Figure 42. Rq box plot (left) and histograms (bin width = 0.01) for a flat surface data.

Table 10. Average square roughness results from square meter test

	First order fit	Second order fit
Mean	0.6274	0.5493
Standard deviation	0.1870	0.1663
Relative standard deviation	0.2981	0.3027
Mean confidence interval	0.6140	0.5374
	0.6408	0.5612
Standard deviation confidence interval	0.1780	0.1583
	0.1970	0.1751
First quartile Q1	0.4984	0.4286
Third quartile Q1	0.7318	0.6395
Median	0.6067	0.5299
Interquartile range IQR	0.2334	0.2109

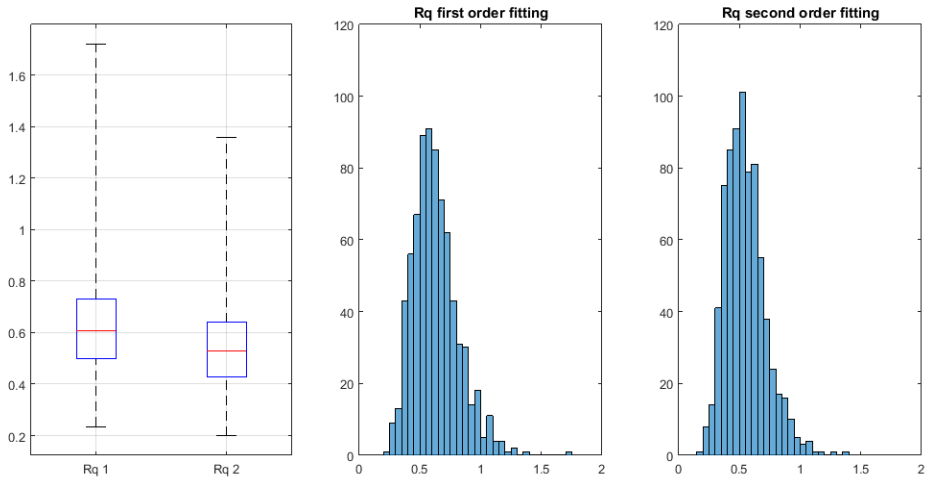


Figure 43. Rq box plot (left) and histograms (bin width = 0.05) for square meter test.

6.5. Estimated slope and intercept from the square meter test

The ultimate goal of the circular texture meter is to provide an estimate of the texture depth. We have used the means obtained from the flat surface segments as the intercepts. As expected, the values obtained for each parameter are positive, therefore to map a parameter value to an estimated depth, the intercept β is negative in all cases. It should be noted that the magnitudes of intercept values are not varying significantly between methods, lying between 37 and 85 micrometers for the first order fit approach and 15 and 44 micrometers for the second order fit approach. In contrast, the magnitudes of slope values have large differences. The mean intercept and slopes for each parameter are shown in Table 11.

We also consider the 95% confidence intervals for both mean intercept and slopes, presented in Table 12. Of particular interest is the range of the slope values, obtained, the best performing being the average maximum height with 115 micrometers interval for the first order fit and 125 micrometers for the second order fit; in comparison, the average roughness has a confidence interval about three times larger.

Graphically, the estimated correlation parameters have been plotted in Figure 44. It can be observed that average maximum height has a slope closest to 1, making it suitable for road surface texture depth. In contrast, the average roughness and average square roughness parameters present a very steep slope: a small error in measurement will lead in large errors in estimation.

Table 11. Estimated slope and intercept

	First order fit		Second order fit	
	Intercept β	Slope α	Intercept β	Slope α
MPD	-0.0514	2.1792	-0.0229	2.3607
Rz	-0.0847	1.3767	-0.0435	1.5151
Ra	-0.0368	4.1517	-0.0146	4.5763
Rq	-0.0439	3.3993	-0.0183	3.7355

Table 12. 95% confidence intervals for estimated slope and intercept

	First order fit		Second order fit	
	Intercept β	Slope α	Intercept β	Slope α
MPD	-0.0444	2.0786	-0.0196	2.2595
	-0.0584	2.2763	-0.0262	2.4579
Rz	-0.0789	1.3179	-0.0392	1.4513
	-0.0904	1.4332	-0.0478	1.5763
Ra	-0.0343	3.9652	-0.0137	4.3802
	-0.0393	4.3314	-0.0156	4.7646
Rq	-0.0411	3.2452	-0.0171	3.5737
	-0.0467	3.5480	-0.0195	3.8909

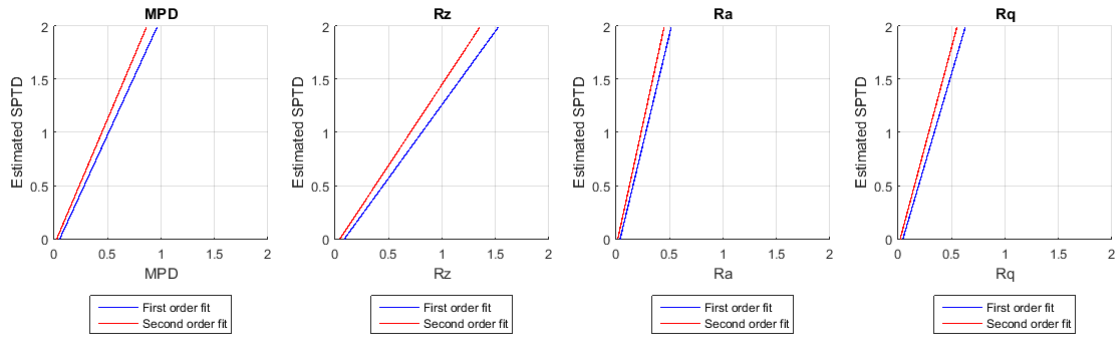


Figure 44. Graphic representation of slope and intercept

6.6. Correlation with sand patch texture depth

Correlation with sand patch was done in the same manner as current practice is: a track is recorded on site then the texture depth is evaluated using sand patch method on the same site. The readings were obtained courtesy of Fulton Hogan from 45 sites; the results are recorded in Table 13.

The correlation with the sand patch texture depth was calculated for each parameter using a linear regression where the intercept was constrained to the value found by measuring the flat surface; this approach results in a smaller r-square for the fit. The correlations are graphically presented in Figure 45 to Figure 48.

All parameters show an improved correlation between first order fit approach and second order approach. Also, the slopes obtained from correlations are smaller than those obtained from the square meter test.

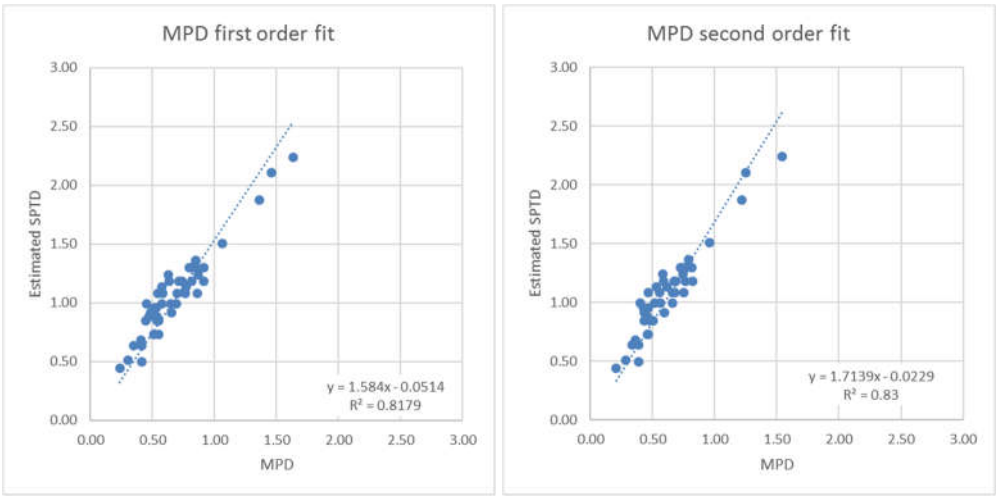


Figure 45. MPD vs SPTD correlation

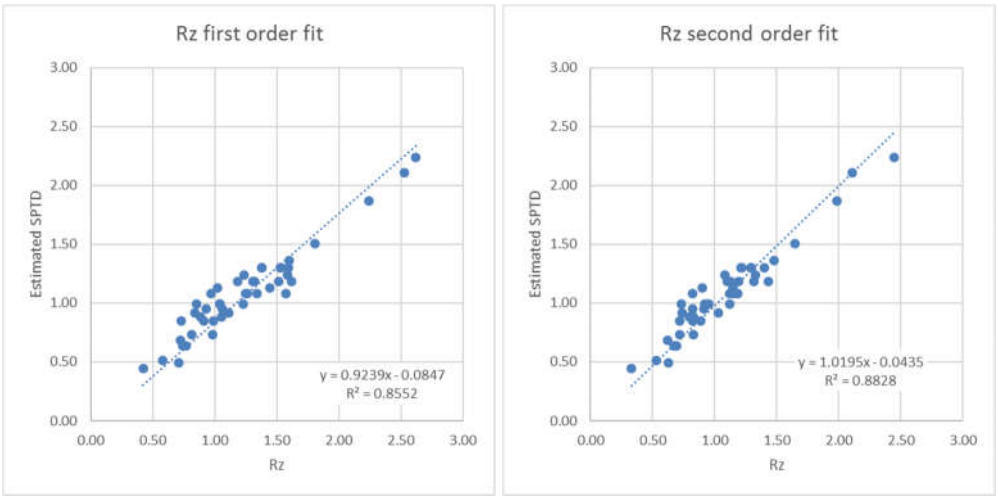


Figure 46. Rz vs SPTD correlation

The mean profile depth obtained with the standardized method prescribed by ASTM E965 is showing the smallest r-square: 0.8179. This is improved to 0.8300 for the second order fit. The slopes found are 1.5840 for first order fit and 1.7139 for the second order fit.

The average maximum height is showing a better correlation with the sand patch method with a r-square of 0.8552 for the first order fit method and 0.8828 for the second order. Also it should be noted that the slopes, 0.9239 and 1.0195, are the closest to unity for this parameter indicating suitability for use as a texture depth estimator.

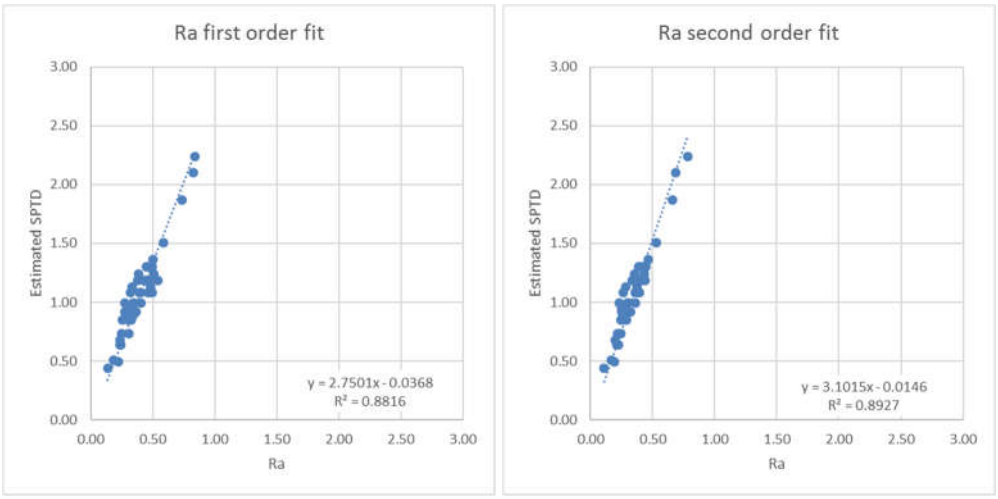


Figure 47. Ra vs SPTD correlation

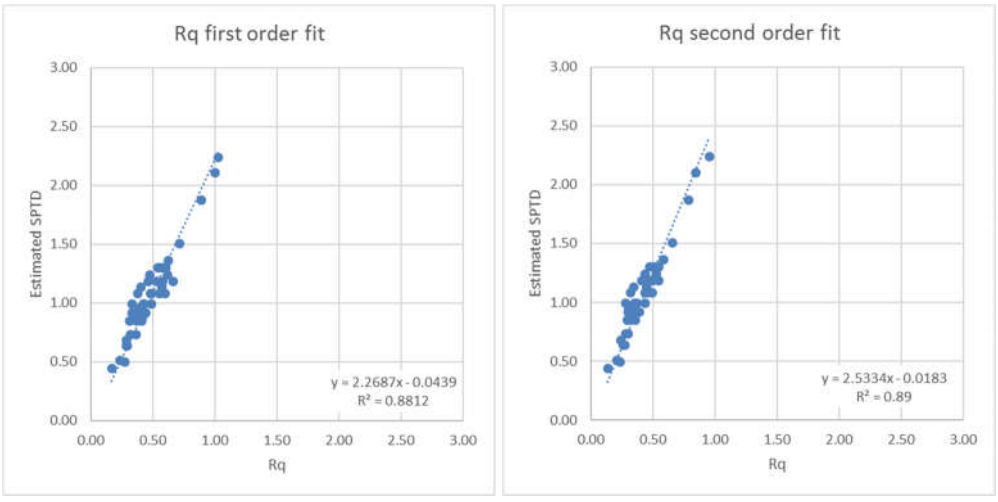


Figure 48. Rq vs SPTD correlation

The last two parameters, average roughness and average square roughness have the highest r-square for both first order fit and second order fit although the difference is not

as high as the first two parameters. The slopes obtained from correlations are higher, signifying that those parameters might not be suitable as a texture depth estimator.

We would like to note that the slopes for all parameters, with the exception of mean profile depth, have a similar percentage reduction compared with the square meter test, on average the correlation has produces a slope approximately 67.2% smaller than the slope obtained with square meter test. The slope for MPD is reduced to 72.6% of the square meter test.

A limitation of the correlation approach in finding the slope is the lack of results where the sand patch texture depth is higher than 1.5 mm, therefore the diversity of the upper half of the data is poor represented.

Table 13. Sand patch texture depth and corresponding profile parameters.

Site Name	SPTD	First order fit				Second order fit			
		MPD	Rz	Ra	Rq	MPD	Rz	Ra	Rq
355 Royal Road	2.24	1.63	2.62	0.84	1.03	1.54	2.44	0.78	0.95
335 Royal Road	1.87	1.36	2.24	0.74	0.89	1.22	1.98	0.66	0.79
8 Spargo Road	1.30	0.80	1.53	0.47	0.57	0.74	1.30	0.43	0.53
8 Spargo Road	1.36	0.85	1.60	0.50	0.62	0.79	1.48	0.47	0.58
24 Claverdon Drive	1.30	0.81	1.38	0.45	0.54	0.72	1.21	0.39	0.47
21 Claverdon Drive	1.18	0.74	1.32	0.44	0.53	0.68	1.10	0.36	0.46
4 Reinga Place	1.08	0.70	1.25	0.39	0.49	0.66	1.12	0.36	0.45
3 Reinga Place	0.73	0.55	0.98	0.31	0.37	0.46	0.83	0.25	0.30
5 Reinga Place	1.30	0.81	1.38	0.45	0.55	0.77	1.22	0.39	0.48
10 Reinga Place	0.99	0.45	0.85	0.27	0.33	0.40	0.73	0.23	0.28
151a Cowill Road	1.24	0.63	1.24	0.38	0.47	0.58	1.08	0.35	0.44
170 Cowill Road	1.08	0.58	1.26	0.40	0.48	0.56	1.14	0.36	0.44
129 Cowill Road	0.51	0.30	0.58	0.19	0.23	0.28	0.53	0.17	0.21
126 Cowill Road	0.73	0.51	0.81	0.25	0.32	0.47	0.72	0.22	0.28
113 Cowill Road	1.24	0.87	1.59	0.50	0.62	0.74	1.33	0.43	0.53
108 Cowill Road	1.51	1.06	1.81	0.58	0.72	0.96	1.65	0.53	0.66
37 Helleur Road	0.95	0.53	1.06	0.34	0.40	0.46	0.91	0.28	0.35
108 Helleur Road	0.92	0.66	1.11	0.37	0.44	0.59	1.03	0.32	0.39
9 Helleur Road	1.18	0.63	1.19	0.38	0.46	0.59	1.13	0.34	0.41
Selwood Road	0.92	0.48	0.84	0.28	0.33	0.43	0.74	0.25	0.30
Selwood Road	0.68	0.41	0.72	0.23	0.28	0.36	0.62	0.20	0.24
Selwood Road	1.30	0.91	1.54	0.49	0.60	0.81	1.29	0.42	0.51
1 Kervil Avenue	1.08	0.77	1.34	0.46	0.55	0.68	1.16	0.38	0.46
2 Kervil Avenue	0.99	0.65	1.04	0.34	0.43	0.56	0.96	0.30	0.37
32 Kervil Avenue	1.13	0.57	1.02	0.33	0.41	0.53	0.90	0.28	0.35
34 Kervil Avenue	1.18	0.72	1.30	0.43	0.53	0.67	1.19	0.39	0.48
34 Kervil Avenue	1.13	0.77	1.44	0.48	0.57	0.61	1.15	0.37	0.45
Shamrock Lane	0.85	0.55	0.99	0.33	0.41	0.50	0.89	0.29	0.36
Shamrock Lane	0.88	0.47	0.88	0.31	0.38	0.45	0.79	0.26	0.32
Shamrock Lane	0.85	0.45	0.73	0.26	0.31	0.43	0.72	0.24	0.29
2/4 Shamrock Lane	0.64	0.35	0.74	0.23	0.29	0.34	0.67	0.21	0.26
5 Shamrock Lane	0.64	0.41	0.77	0.24	0.30	0.39	0.69	0.23	0.28
1 Gladfiels Lane	1.08	0.54	0.97	0.32	0.38	0.46	0.82	0.26	0.32
2 Gladfield Labe	1.30	0.86	1.59	0.49	0.60	0.77	1.40	0.44	0.54
Gladfield Lane	1.08	0.86	1.57	0.49	0.60	0.75	1.19	0.39	0.50
Gladfield Lane	0.99	0.58	1.04	0.35	0.42	0.52	0.92	0.29	0.35
Gladfield Lane	1.18	0.82	1.51	0.47	0.58	0.82	1.44	0.44	0.55
Gladfield Lane	1.18	0.92	1.62	0.54	0.67	0.76	1.31	0.42	0.50
114 Kervil Avenue	2.10	1.46	2.53	0.83	1.00	1.25	2.11	0.69	0.84
10 Mead Street	0.88	0.54	1.05	0.34	0.41	0.46	0.83	0.28	0.34
3 Mead Street	0.95	0.52	0.93	0.32	0.39	0.42	0.82	0.25	0.31
3 Mead Street	0.44	0.24	0.42	0.14	0.17	0.21	0.33	0.11	0.14
80 Mead Street	0.50	0.42	0.71	0.22	0.27	0.39	0.63	0.19	0.24
80 Mead Street	0.99	0.69	1.23	0.40	0.49	0.66	1.12	0.36	0.44
83 Mead Street	0.85	0.54	0.91	0.30	0.37	0.48	0.82	0.26	0.32

7. CONCLUSIONS AND FUTURE WORK

The objective of our research was to design and build a portable device able to estimate road surface texture depth using a profile acquired with a laser displacement sensor. The method chosen falls under the circular texture meters (CTM) category where a surface profile is acquired using a laser displacement sensor mounted on an arm that is rotated around an axis. The mechanical design follows the specifications from current practice established by a number of standards in Australia, USA and Europe. Profile processing is realized on board by a purpose built embedded system using a 16-bit processor and the results are available to the operator on an LCD display. In addition, profile data can be saved on an optional SD card, placed on the main printed circuit board, for processing and validation purposes.

Estimation of texture depth is achieved using the industry standard mean profile depth (MPD) parameter along with three other parameters borrowed from surface metrology: average roughness (Ra), average square roughness (Rq) and average maximum height (Rz). The last parameter method of computing was redefined by us to suit the short CTM profiles.

The efficiency of the parameters was determined by the extent of how parameters correlate with texture depth obtained with the sand patch method characterised by the slope α and the intercept β of the linear relation that result in the best r-square for the parameter. We have used two methods in the post processing of the profile prior to parameter calculation: the first approach, in line with standardized practice, uses the fitting of a first order line to the profile segments; the second approach considers the inevitable misalignments of the measured surface plane and the plane where the displacement sensor is lying, therefore a second order fit is applied to the profile segments.

The intercept was derived from mean of the parameters measured over a flat surface for both first order fit and second order fit methods. The slope was estimated firstly from a test area of one square meter where sand patch texture depths were obtained and 47 tracks (one track is two circle profiles: one in forward direction and one in reverse) were recorded. Secondly, the slope was estimated from tracks acquired from a variety of sites

using the correlation method with the intercept being constrained to the value found from the flat surface measurements.

The results show that all parameters have an improved r-square when a second order fit is applied to the profile segments. The mean profile depth parameter has recorded the smallest r-square when the first order fit was applied, the highest correlations being obtained by the average roughness and average square roughness parameters. The slopes estimated with the correlation method are on average 67% percent smaller than the slopes projected with the square meter test. The most suitable parameter as a texture depth estimator is the average maximum texture depth which records a slope of 1.0195 when the second order fit method is used. The average roughness and average square roughness parameters exhibit high value slopes therefore those are less desirable as estimators.

Considering both the coefficient of determination of the correlations and the slope, the circular texture meter returns best estimates from the average maximum height when a second order fit is applied to the profile segments.

Some limitations of the overall approach should be noted. Firstly, compared with the volumetric sand patch method, the profiles acquired with the CTM lack data density; even with two profiles acquired there are only 16 profile segments available for estimation. The square meter test has shown the means tend to be smaller when a larger number of profiles are acquired. Secondly, the results are more influenced by the defects in surface especially where the aggregates are dislodged from the binder substrate, as shown by the histograms computed for the square meter test.

The correlation with the sand patch was done with the assumption that the method was most accurate predictor of texture depth. This is not always the case as the results are dependent on operator skill, the lower limit of texture depth is deemed to be 0.5 mm and we lacked data points for depths between 1.5 to 2.5 mm.

The mean profile depth algorithm uses peaks value rather than maximum value in a segment, thus the returned parameter can be smaller in some cases where the profiles amplitude is small, especially under 0.3 mm.

Further work includes development of the algorithms for peak detection, improvement of texture depth estimation by considering the shape parameters such as skewness and kurtosis of a profile segment and increasing data density either by taking multiple profiles over the same surface or using 3D scanning techniques.

References

Adams, J. M., & Kim, R. Y. (2013). Mean Profile depth analysis of field and laboratory traffic-loaded chip seal surface treatments. *International Journal of Pavement Engineering, 15*(7), 645-656.

Aktas, B., Gransberg, D., Rienner, C., & Pittenger, D. (2011). Comparative analysis of Macrotexture Measurement Tests for Pavement Preservation Treatments. *Journal of the Transportation Research Board*, 34-40.

Analog Devices. (2005). AD7694 Datasheet. Analog Devices.

ASTM. (2015). Standard practice for Calculating Pavement Macrotexture Mean Profile Depth. West Conshohocken, PA: ASTM International.

ASTM. (2015). Standard TestMethod for Measuring Pavement Texture Drainage using an Outflor Meter. West Conshohocken, PA: ASTM.

ASTM International. (2015). ASTM E965-15 Standard Test Method for Measuring Pavement Macrotexture Depth Using a Volumetric Technique. West Conshohocken, PA: ASTM International.

ASTM International. (2015). Measuring Pavement Macrotexture Proprieties Using the Circular Texture Meter. West Conshohocken, PA: ASTM International.

Austroads. (2007). Pavement surface texture measurmeent with a laser profilomeer. Sydney: Austroads.

Austroads. (2008). *Review of Surface Texture Measurement Method of Seal Desing Input.* Sydney: Austroads.

Austroads. (2008). *Review of Surface Texture Measurment Method for Seal Design Input.* Sydney: Austroads.

Austroads Inc. (2009). *Guide to Asset Management Part 5g: Texture.* Sydney: Austroads Inc.

Autroads. (2008). AG:PT/T250 Modified surface texture depth (Pestle method). Sydney: Austroads.

Berge, T., & Viggen, E. M. (2014). An investigation of the relationship between texture and tyre/road noise for different types of road surfaces and passenger car tyers. *Inter-Noise.* Melbourne.

Department of Transport and Main Roads. (2012). Test Method Q705B. Syndey: Department of Transport and Main Roads Queensland.

Elunai, R., Vinod, C., & Edith, G. (2011). Asphalt concrete surfaces macrotexture determination from still images. *IEEE transactions on intelligent transportation systems, 12*(3), 857-869.

Fisco, N. R. (2009). Comparison of Macrotexture Measurerment Methods. Ohio: Ohio State University.

Gabriele Bitelli, A. S. (2012). Laser Scanning on Road Pavements: a New Approach for Characterising Surface Texture. *Sensors*, 9110-9128.

Hanson, D. I., & Prowell, B. D. (2004). *Evaluation of ciruclar texture meter for measuring surface texture of pavements.* Auburn, Alabama: National Center for Asphalt Technology.

Hanson, F. M. (1934). Bituminous surface treatment of rural highways. *New Zealand Soc Civic Engineer Proc*, *21*, pp. 89-220.

Johnsson, R., & Odelius, J. (2012). Methods for road texture estimation using vehicle measurements. *Proceedings of ISMA 2012-USD 2012*, (pp. 1573-1582).

Mainroads Western Australi. (2012). Texture depth. Test method WA 311.1-2012. Perth: Mainroads Western Australia.

McGhee, K. K., & Flintsch, G. W. (2003). *High-speed texture measurement of pavements.* Chalottesville, Virginia: Virginia Transportation Research Council.

Microchip. (2007). MCP601/1R/2/3/4 Single supply CMOS Op Amps. Microchip.

Microchip. (2010). PIC24FJ256GA11 Family Datasheet. Microchip.

Microchip. (2012). MCP1525/41 Datasheet. Microchip.

Muralikrishnan, B., & Raja, J. (2009). *Computational surface and roundness metrology.* Charlotte, NC: Springer.

National cooperative highway research program. (2005). *Chip Seal Best Practices A sysnthesis of Highway Practice.* Transportation Research Board of the National Academies.

New Zealand Roading Division; New Zealand Road Research Unit. (1968). *Manual of ealing and paving practice.* Wellingcton: National Road Board.

Pidwerbesky, B., Gransberg, D., & Stemprok, R. (2006). *Road surface texture measurement using digital image processing and information theory.* Wellington: Land Transport New Zealand.

Sick. (2015). OD1B100C5OI14 Short range distance sensor datasheet.

ST Electronics. (2003). L293D Datasheet. ST Electronics.

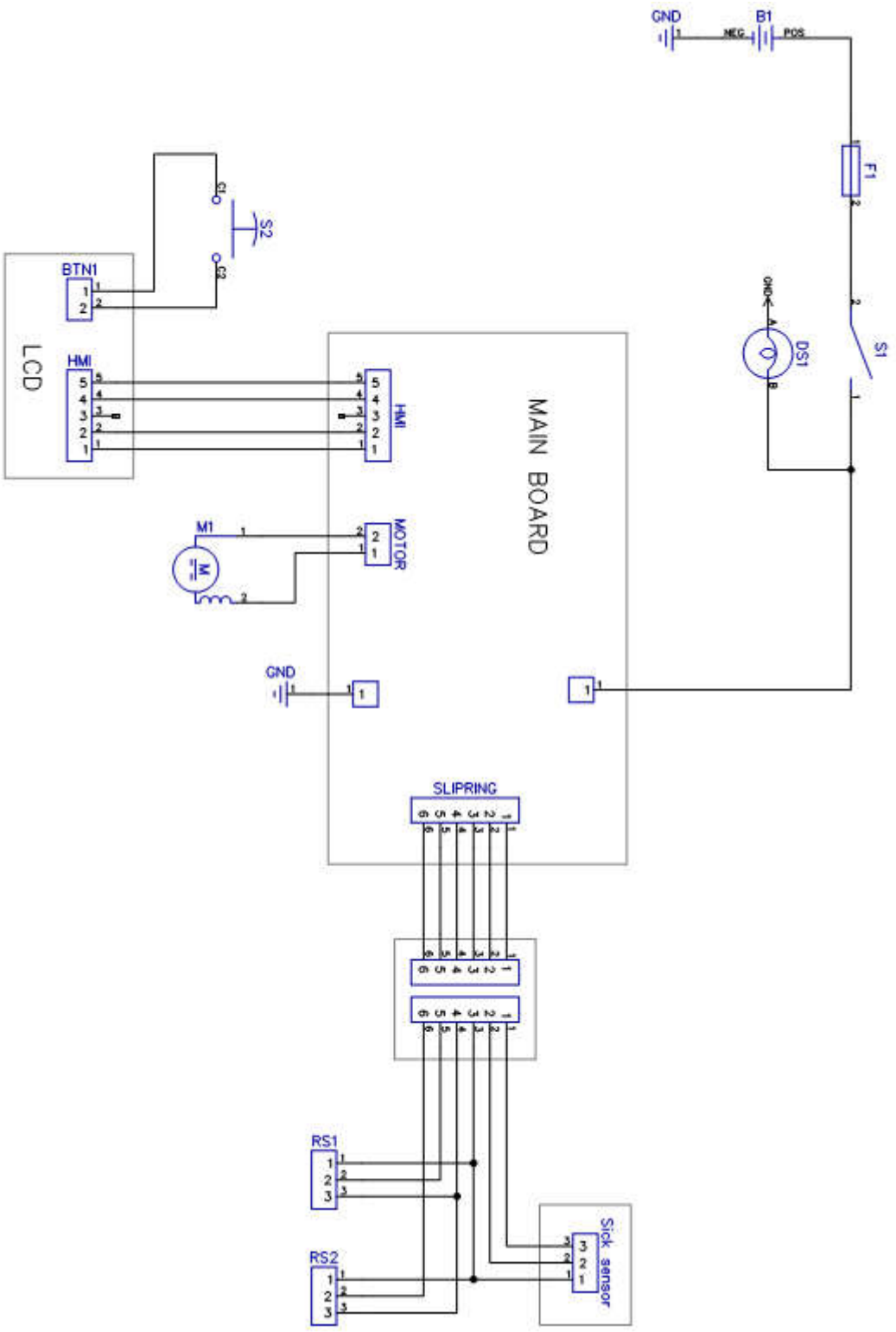
Tabenkin, A. (2007, November). The ABCs of Rz. *Quality*(11), 40.

Tomov, M., Kuzinovski, M., & Trajcevski, N. (2010). Function on gaussian and 2RC filters to determine the roughness profile in real non-periodic and periodic surfaces. *Trends in the development of machineryt and associated technology* (pp. 9-12). TMT.

Transit New Zealand. (1981). Standard test procedure for measurement of texture by the sand circle method. Wellington: Transit New Zealand.

Transit New Zealand. (2005). *Chipsealing in New Zealand.* (J. Towler, & J. Heine, Eds.) Wellington: Transit New Zealand, Road Controlling Authorities, Roading New Zealand. doi:625.850993-dc 22

Annex A. Electrical schematic



Annex B. Main board schematic