A MATLAB Program to implement the band-pass method for discovering relevant scales in surface roughness measurement

A Master's Project Report:

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science in Applied Mathematics

By

____

Chukwunomso Agunwamba

Advisors:

____

Professor Homer Walker, Mathematical Sciences Department

____

Professor Christopher Brown, Materials Science and Engineering Department

Table of Contents

Table of Figures

Table of Tables

Acknowledgements

I would like to thank my advisors, Professors Homer Walker and Christopher Brown for their patience, advice, and guidance during the project. I would also like to thank Johann Berglund for providing the surface data discussed in this project and for his explanations. Lastly, I would also like to extend my thanks to Toby and Brendan for the warm welcome they gave me in the lab and their readiness to provide me with resources and to make things work smoothly for me.

Abstract

This project explores how to use band-pass filtering with a variety of filters to filter both two and three dimensional surface data. The software developed collects and makes available these filtering methods to support a larger project. It is used to automate the filtering procedure. This paper goes through the workflow of the program, explaining how each filter was implemented. Then it demonstrates how the filters work by applying them to surface data used to test correlation between friction and roughness [Berglund and Rosen, 2009]. It also provides some explanations of the mathematical development of the filtering procedures as obtained from literature.

CHAPTER 1: Introduction

Project Overview

This project explores how to use band-pass filtering with a variety of filters to filter both two and three dimensional surface data. The software developed collects and makes available these filtering methods to support a larger project. The larger project is done by [Berglund et. al, 2010]. It seeks to use band-pass filtering to find strong correlations. The software described in this report is used to automate the filtering procedure. This report goes through the workflow of the program, explaining how each filter was implemented. Then it demonstrates how the filters work by applying them to surface data provided by [Berglund, 2009]. It also provides some explanations of the mathematical development of the filtering procedures as obtained from literature.

Some Conventions and Definitions

Surface data and filters that depend on a single variable are termed “2D data” and “2D filters” respectively. Those depending on two variables are termed “3D data” and “3D filters” respectively.

Surface roughness is viewed as texture created by a combination of spatial short wavelength sinusoids present in the surface [Muralikrishnan and Raja, 2009, Chapter 3]. There is a reciprocal relationship between wavelength, λ, and frequency f:

$$\lambda = \frac{1}{f}.$$

The wavelength’s relationship to frequency thus makes relevant the use of Fourier decomposition and frequency spectrum filtering to extract surface roughness values at different frequency or wavelength bands. The center wavelength in each wavelength band represents the wavelength scale in which we are extracting roughness information. The length and area scales are examples of other scales used in scale-based decomposition of surface data. These scales are used in the scale sensitive fractal analysis of surface data [Berglund and Rosén, 2009].

Two main examples of parameters that characterize the surface roughness are: average roughness (abbreviated as Ra for 2D and Sa for 3D), root mean square roughness (abbreviated as Rq for 2D and Sq for 3D). Average roughness is the arithmetic average of absolute values of the surface data residuals around the mean line:

$$Ra = \frac{1}{n}\sum_{i=1}^{n}|r(i)|.$$

The root mean square is the square root of the variance of the surface data around the mean line:

$$Rq = \sqrt{\frac{1}{n}\sum_{i=1}^{n} r^2(i)}\ .$$

There are other parameters. The above formulas, along with some others used to characterize a surface, are explained by [Muralikrishnan and Raja, 2009, Chapter 23]. Obtaining the mean line requires low-pass filtering. Thus, filtering participates in the roughness parameter calculations. The larger project extends this procedure by controlling the wavelength band selection where the roughness parameters are calculated. So the same Ra and Rq formulas, for example, can be applied on band-pass filtered surface data and the results studied.

The importance of roughness parameters

One way to study calculated roughness parameters is to correlate them to some topographic controlling processing or topographic dependent attribute, behavior, or performance exhibited by a class of surfaces in which we are interested. These observed phenomena could be for example: friction, adhesion, adsorption, wetting, scattering of light or particles, reflectivity. The correlation subsequently helps in the selection of the value for the roughness parameter adequate for influencing the observed behavior as in engineering design optimization.

Why use band-pass filters on Surface Data?

Band-pass filters enable a type of scale-based decomposition of surface data. Making surface roughness depend on wavelength scale is a particular vehicle to answer the question: “how does scale influence observed topographic phenomena?” Now, instead of only varying the way each surface in our collection is created and calculating the roughness parameters for each, we can also calculate a collection of roughness parameters for data restricted to certain scales. In this case, the scales are wavelength bands of our own choosing. The larger project tests the hypothesis that we will find stronger correlations between roughness parameters and, in particular, the friction coefficient of the surface when we select the appropriate wavelength band.

State of the Art in Surface Roughness Filtering

There are many filters available for surface data filtering. They broadly fall into two categories: envelope filters (E filters) and mean line filters (M filters) (see [Muralikrishnan and Raja, 2009]). We only select a few mean line filters presented in the book. We also make it possible for the user to either employ MATLAB’s fdatool command in designing a custom filter or to specify the filter spatial domain coefficients or matrix for use in the program. They each deal with certain issues such as the ability to avoid introducing edge artifacts, and how oscillatory the mean line result is.

Approach

Chapter 2 introduces the program and explains the main function that provides an interface to user. Chapter 3 explains inputs and outputs of the auxiliary filter functions that are called by the main function. Chapter 4 provides detail about how the filter codes were implemented. Chapter 5 explains the two strategies used to create a band-pass filter with specified characteristics. Chapter 6 discusses some results of applying the filters to data. Finally, chapter 7 concludes the paper summarizing the work and its usefulness.

CHAPTER 2: The Program

This chapter introduces the program and explains the main function that provides an interface to user. It provides two charts. The first one shows the structure of the program. The second shows the project work flow in relation to the larger project. The work flow consists of both the steps the program follows in filtering the data and the preparation of the filtered data files for more analysis work in the larger project.

Surface Data Filtering Procedure using grandforloop.m

The main function that is used to automate the selection of data files and filtering is called grandforloop.m. Table 1 shows how to use it. Apart from accepting inputs through the command window, one can call it without any inputs. In this case, grandforloop.m provides dialogue boxes that enable one to choose the data source folder, the results destination folder, the number of files to analyze, a list of center wavelengths, a corresponding list of bandwidths, and the type of filter to use. (See the dialogue box pictures in Table 2.)

Dialogue boxes are used to:	Available Filters:	Major Inputs
1. Choose the data folder 2. Choose folder where to save calculations 3. Specify center wavelengths and wavelength bandwidths 4. Choose the filtertype	a. Filters available through matlab's FDA toolkit	1. Surface Data 2. filtertype 3. bandwidth 4. center wavelength 5. Band pass method (see chapter 5)
	b. Gaussian Filter	
	c. Zero order Gaussian regression filter	
	d. Second order Gaussian regression filter	
	e. Robust zero order Gaussian regression filter	

	f. Non-periodic Spline Filter	

Table 1: Usage of grandforloop.m

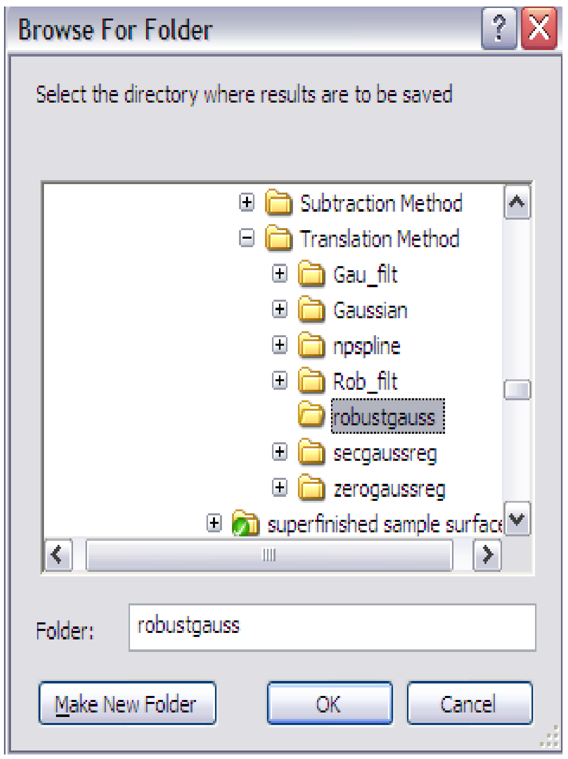
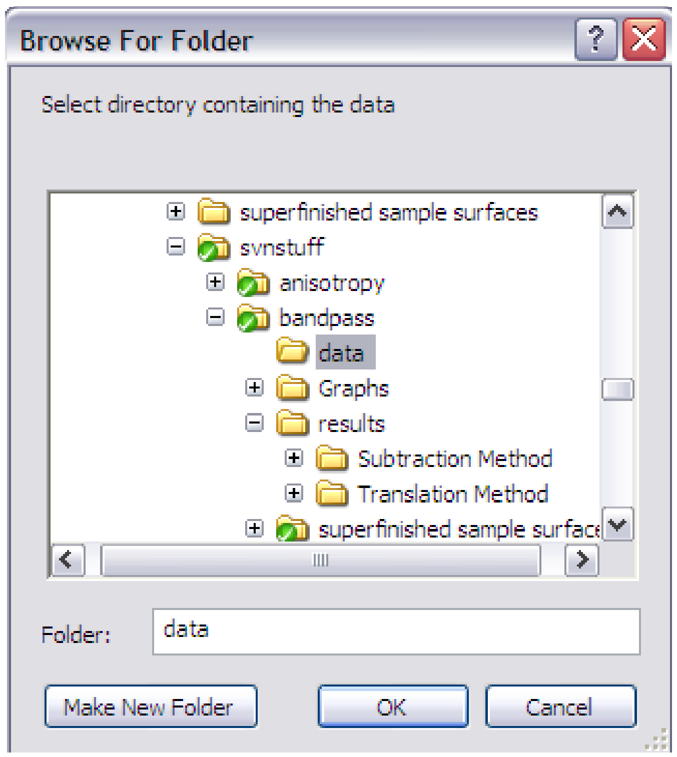
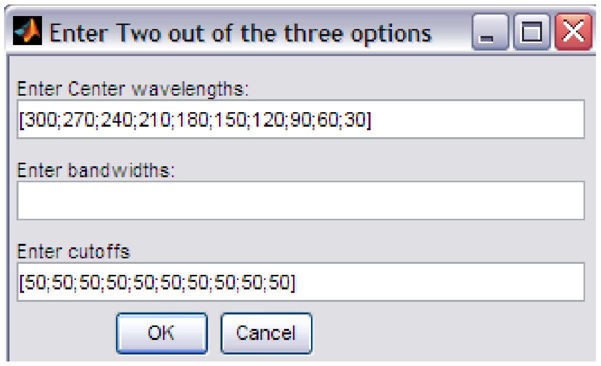
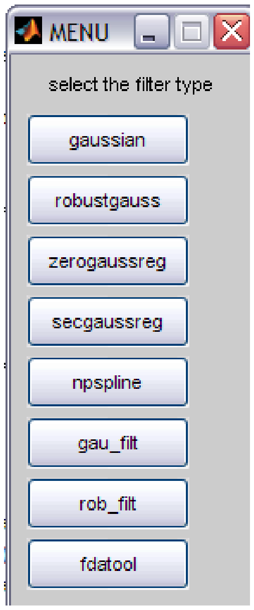
 Browse For Folder Select the directory where results are to be saved Subtraction Method Translation Method Gau_filt Gaussian npspline Rob_filt robustgauss secgaussreg zerogaussreg superfinished sample surface Folder: robustgauss Make New Folder OK Cancel **Figure 1:** Choosing the results destination folder	 Browse For Folder Select directory containing the data superfinished sample surfaces svnstuff anisotropy bandpass data Graphs results Subtraction Method Translation Method superfinished sample surface Folder: data Make New Folder OK Cancel **Figure 2:** Choosing the data source folder
 Enter Two out of the three options Enter Center wavelengths: [300;270;240;210;180;150;120;90;60;30] Enter bandwidths: Enter cutoffs [50;50;50;50;50;50;50;50;50;50] OK Cancel **Figure 3:** Specifying filter parameters	 MENU select the filter type gaussian robustgauss zerogaussreg secgaussreg npspline gau_filt rob_filt fdatool **Figure 4:** Choosing a filter

Table 2: Dialogue boxes numbered in the sequence they are used when calling grandforloop.m

Charts

The chart in Figure 6 below shows the two auxiliary functions called by grandforloop. The function bndfilt.m converts the bandwidth and center wavelength parameters to upper and lower cutoffs or to center frequency and frequency bandwidth depending on what band pass method the user chooses. The chart also shows the level of grandforloop.m's connection to other important auxiliary functions that implement the filtering.

One should note that the function, filterimg.m, shown at the fourth level in the chart, accepts already generated filter coefficients and implements them either via convolution in space or frequency domain multiplication after application of FFT. (See chapter 4 for implementation details.)

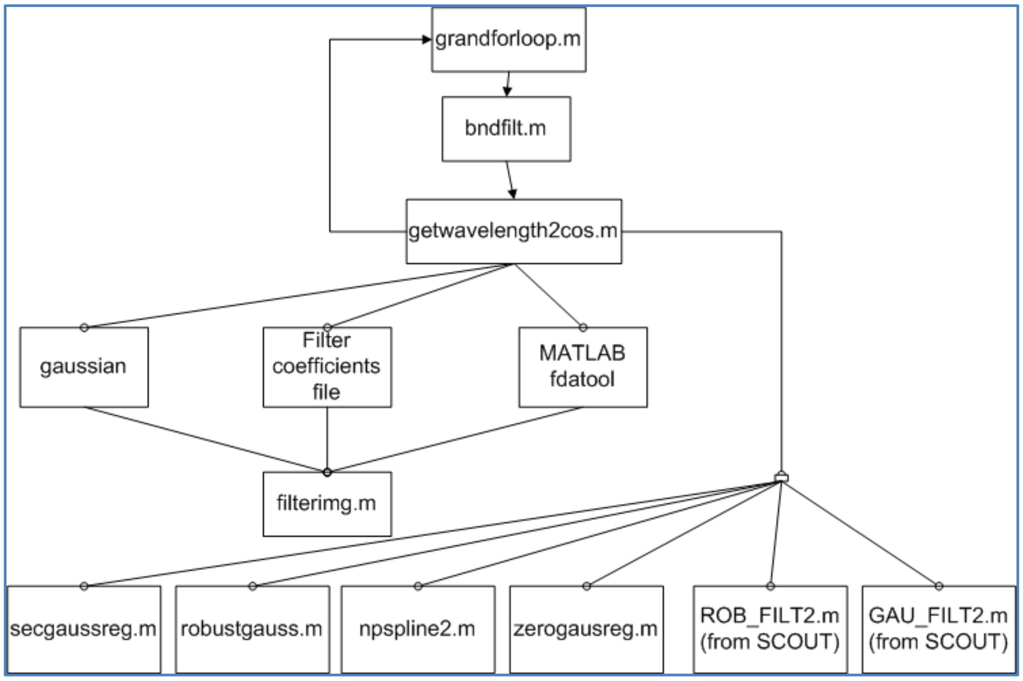


Figure 5: Program chart

The next chart, Figure 7 below, shows the input 3D data is fed into a Gaussian band-pass filter with specified characteristics. Then a band-pass filtering method is used. The filtered data files are saved. Then, MountainsMap software is then used in the larger project to calculate and record roughness parameters for each data file.

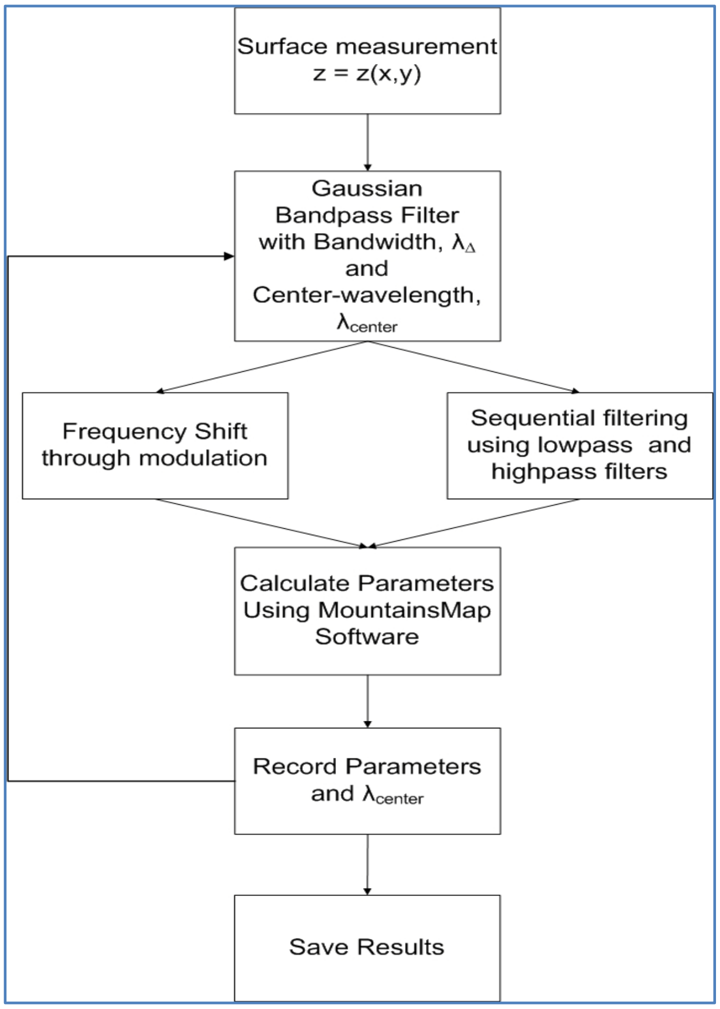

Surface measurement z = z(x,y)
Gaussian Bandpass Filter with Bandwidth, λ_{Δ} and Center-wavelength, λ_{center}
Frequency Shift through modulation
Sequential filtering using lowpass and highpass filters
Calculate Parameters Using MountainsMap Software
Record Parameters and λ_{center}
Save Results

Figure 6: Project Work-Flow Chart

CHAPTER 3: Description of important Inputs and Outputs of Major Functions in the Program

This chapter describes the important inputs and outputs of major functions in the program. Table 2 below lists the inputs and outputs of each function. There some functions have extra inputs that are only used to study them – not varied in the program and not relevant to the actual work the function does. There are also other sub-functions used to enable the functions to work. The comments for each function are pasted in Appendix A. The filtertype input options are given in chapter 4. The explanations for the filters are also given in that chapter.

Also, note that these functions are not the only functions necessary to make the program work. There are other auxiliary functions they call that need to be present whenever one is using these functions. The input surface data format to grandforloop.m, bndfilt.m, and getwavelength2cos.m is in the WPI format. This format requires the readfraxsurf.m file to read it. One should select 1 for the frax input option if one has data in WPI format. Otherwise if one is using a raw data matrix, one should select 0 for the frax input option in files that have the frax option.

Function	Command with Inputs and outputs
grandforloop.m	[IIb,filenames,List, directory_name] = grandforlooptest(filtertype, I, centwave, bndwths, sw, sw2, disgraphs);
bndfilt.m	[IIL,II2L,IIH,II2H,IIb] = bndpfilt(filtertype, I, centwave, Lambdamn, sw, disgraphs);
getwavelengthcos.m	[II, II2] = getwavelength2cos(filtertype, I, centerfreq, Lambdam, frax, Lambda, L, npoints, spacing, disgraphs);

secgaussreg.m	II = secgaussreg(I, n, nv, wc1, wc1v, dx, dxv, lambdacn, lambdacnv, Lambdam, Dirsw);
robustgauss.m	II = robustgauss(I, n, nh, wc1, wc1h, dx, dxh, lambdacn, lambdacnh, Lambdam, alpha, CBt, relCB, iterationNo, disgraphs);
npspline2.m	[x,II6] = npspline2(I, dx, Lambdam(1), wc1);
zerogausreg.m	II = zerogaussreg(I, n, nv, wc1, wc1v, dx, dxv, lambdacn, lambdacnv, Lambdam);
filterimg.m	This Is not intended to be called independently. (The results of each filter can seen by using getfiltereddat.m)
ROB_FILT2.m (Copy of ROB_FILT.m from SCOUT)	[FDATA,FS] = rob_filt (DATA, SPACING, CUTOFF) Used but not written by this project. (See Appendix A.9)
GAU_FILT2.m (Copy of GAU_FILT.m from SCOUT)	FDATA] = Gau_filt (DATA, SPACING, CUTOFF, FTYPE) Used but not written by this project. (See Appendix A.10)

Table 3: Description of program functions

Below is the description of the functions:

(For all the functions using it, the disgraphs input causes the functions to display graphs when they run. This option is suppressed by default.)

The grandforloop.m function:

The filtered data is stored with a name that contains the input surface data filename, bandwidth, center wavelength, and type of filter used. It is also outputted as IIb if the user uses this option in the command window. The input variable, I, is the raw surface data stored in WPI format. The inputs centwave and bndwths are the center wavelength and bandwidths respectively. The input, sw, specifies whether a sequence of low-pass and high-pass filters are used or modulation at a center frequency is used. It is 1 for the first case and 0 for the second one. One specifies how many data files to analyze by setting sw2 to that number.

The bndfilt.m function:

This converts the center wavelength and bandwidth to upper and lower cutoffs or to center frequency and frequency bandwidth. Then it decides whether to use a sequence of low-pass and high-pass filters by calling the filtering function twice, or to use modulation. It provides the filtered and leveled data as IIb to grandforloop.m Lambdamn can either be horizontal and vertical bandwidths or horizontal and vertical low-pass cutoffs.

The getwavelengthcos.m function:

This calls the filter functions. It either uses a sequence of low-pass and high-pass filters by calling the filtering function twice, or it calls the filtering function once, giving it the center frequency for modulation. It outputs the filtered and leveled data as II and the non-leveled and filtered data as II2. The function obtains the number of points, spacing information from the WPI format structure, I. Lambdam is the cutoff wavelength vector for vertical and horizontal directions while Lambda is the Lambdam divided by the spacing. One specifies either Lambdam or Lambda.

General Inputs to the Filters:

The input, I or z, is the raw surface data matrix. The outputted filtered data variable is II. The number of points in the vertical and horizontal directions are n and nv respectively.

(v or h in nv or nh only shows nv and nh are different from n. The vertical direction is assigned to n and to other variables that do not end with v or h.) The angular frequency shift is wc1, dx is the spacing, lambdacn is the cutoff wavelength divided by the corresponding spacing, and Lambdam is the cutoff wavelength vector.

The filter codes: secgaussre.m, robustgauss.m, npsplin2.m, and zerogaussreg.m can be used independent of grandforloop to filter individual surface data files.

Descriptions and Inputs specific to each Filter:

The secgaussreg.m function:

This function performs 2D and 3D second order Gaussian regression filtering. Dirsw provides an option to perform the filtering assumging that the vertical and horizontal directions are independent or to allow the cross-product terms to enter the calculation. (See chapter 4.) The setting, Dirsw = 1, is for no interaction while Dirsw = 2 allows the interaction term.

The robustgauss.m function:

This function uses the robust Gaussian filter. This filter is an iterated - weighted zero order Gaussian regression filter. Its input, alpha, is used to decide how many standard deviations away from the mean of the data to use in calculating the weight. The thresholds, Cbt, relCB, and IterationNo are used to stop the iteration. CBt is applied on the average of the filtered data, relCB limits the error between the current average and the previous average, and iterationNo limits the number of iterations.

The npspline2.m function:

This function implements the nonperiodic spline filter in the vertical direction. So it is called a second time by getwavelength2cos.m in order to filter the horizontal direction.

The zerogausreg.m function:

This function uses the zero order Gaussian regression filter. It only uses the general filter inputs.

The filterimg.m function:

This function is not intended to be called independently. It accepts filter coefficients and performs the linear convolution either in space or by frequency multiplication.

The ROB_FILT2.m and GAU_FILT2.m functions (from SCOUT)**:**

These were written by S. Brinkman, Hannover University and A. Porrino, Brunel University. Their emails are: brinkmann@imr.uni-hannover.de and alessandro.porrino@brunel.ac.uk. These functions were not made specifically for this project. They were simply made available as an option that can be called when executing grandforloop.m. (See the Appendix A.9 and A.10 for further descriptions.)

(SCOUT read and write functions were renamed from Readsdf.m and Writesdf.m to Readsdf22.m and Writesdf22.m respectively in order to avoid conflict with the surfrax read and write files that possess the same name.)

CHAPTER 4: Available Filter Types

This chapter discusses the filter codes used, referring the reader to where the codes originated and explains the modifications that were inserted in each of them. In general, the modifications made to the filter codes were in terms of variable names, extending the method of implementing them to handle both 2D and 3D filtering, and to allow them to handle relatively large data sets.

Filters from MATLAB's FDA toolkit and Saved filter coefficients

To use the MATLAB Filter Design and Analysis Tool, fdatool is typed on the command window or 'fda' is used as the filtertype input variable inside the function that has filtertype as an input. When using the FDA option to design a filter, the new filter is exported as an ascii file (with extension .fcf). One chooses from among the available filters, specifies the design parameters, designs the filter and exports it using the export option under the file menu. Then, before closing the FDA tool GUI, the ascii file is edited by removing both its header and the line that reads: "Numerator:". This way the resulting file contains only the filter coefficients. The GUI is closed so that the calling function can resume.

The calling function will request the user to select the file where the filter coefficients were saved. Then it will use those to filter the data. In addition, if one has already saved the filter coefficients, one can use the string, 'coeff', as the filtertype option.

The Gaussian Filter

Using the SCOUT code:

The formulas and parameter specifications for the SCOUT program are:

GAU_FILT (Z, h, CUTOFF,str)

Z is the raw data, h is a vector containing the spacing in y and x directions, CUTOFF is the 50% lowpass cutoff, and 'str' is a string where a cropping method is chosen. GAU_FILT.m uses the filter2 command from MATLAB. For the cropping method, 'L' or 'l' asks it to use a cropping method provide by MATLAB filter2 command, and ‘H’ or ‘h’ asks it to use the cropping method provided inside the GAU_FILT.m code. An example is (assuming the raw data and parameters are in the workspace):

```
[ZF] = GAU_FILT(z, [dy,dx], [Lamy Lamx],'l');
```

The code creating the filter, G, that SCOUT uses, is given below:

```
Nx=round(lamx/dx/2);
Ny=round(lamy/dy/2);
x=[-Nx:Nx]*dx;
y=[-Ny:Ny]*dy;
[X,Y]=meshgrid(x,y);
beta=log(2)/pi;
G=dx*dy/beta/lamx/lamy;
G=G*exp(-(pi/beta)*((X/lamx).^2+(Y/lamy).^2));
```

Note that it cuts the cutoff in half and uses the round command when creating the x and y coordinates. That is, the cutoff is approximately the filter length here.

Another option uses the American Standard [American Society of Mechanical Engineers 2002]. The formula [Muralikrishnan and Raja, 2009 chapter 5 page 33] for this method is given as:

$$S(x,y) = \frac{1}{\alpha\lambda_{cx}\alpha\lambda_{cy}} \exp\left[-\pi\left(\frac{x}{\alpha\lambda_{cx}}\right)^2 - \pi\left(\frac{y}{\alpha\lambda_{cy}}\right)^2\right], \text{ where } \alpha = \sqrt{\frac{ln2}{\pi}}.$$

The following code lines, in lines 265 to 378 of getwavelength2cos.m, are used to construct the function for this method:

```
    lambdacc = dx*lambdacn; %user defined cutoff wavelength (When it is centered at zero before translation to a different center wavelength)
    lambdacch = dxh*lambdacnh;
    lambdac = Lambdam(1); %user defined cutoff wavelength (When it is
centered at zero before translation to a different center wavelength)
    lambdach = Lambdam(2);
    x1 = (-lambdacc:dx:lambdacc)';
    x1h = (-lambdacch:dxh:lambdacch)';
    N1 = round(x1/dx);
    % plot(N1,'*')
    % pause
    alpha = sqrt(log(2)/pi); %0.4697;
    S1 = (1/(alpha*lambdac)).*exp(-pi*(x1/(alpha*lambdac)).^2);
    S1h = (1/(alpha*lambdach)).*exp(-pi*(x1h/(alpha*lambdach)).^2);
```

This formula and basic code for this filter in 2D is the one provided in [Muralikrishnan and Raja 2009, chapter 5 page 34].

This project modified the code in the following manner. The actual implementation of the 3D capable filter is done in the file filterimg.m. The m-function, filterimg.m, has the two options of filtering in frequency domain and convolution in spatial domain. Each is done by a designated sub-function. The sub-function in line 166, getfiltdatavec, multiplies the FFT of the data and the filter and takes the inverse FFT. It applies the filter in one direction. Therefore, it is called twice to filter the horizontal and vertical directions in sequence. The other sub-function is getfiltdatavec2. It is also called twice in like manner.

Both methods can filter the data in sections when the data is too large. The variables ns in line 168 and nns in lines 175, 240, 252, and 272 determine how many rows and columns to use at a time.

The vector, Lambdam, contains the spatial cutoffs. Note that this method uses twice the cutoff as the filter length and applies the floor function in contrast the SCOUT code. This ensures a symmetric odd length filter. The relatively extra points included might cause the sampled Gaussian's FFT distribution to be more like samples of its continuous Fourier transform.

In the filterimg.m code, n is the data's length in the vertical direction and m1 is the filter's length in the vertical direction. In the event that the filter is longer than the data along any dimension, the filter's discrete Fourier series (DFS) is used instead of the FFT to resample the filter's frequency distribution at 2n-1 points instead of using the n+m1-1 points the FFT would have given. This conditional is implemented in lines 80 to 114 of filterimg.m. (See Appendix B.1 for some explanation about convolution and frequency domain filtering.)

The 3D Zero-Order Gaussian Regression Filter

The zero-order Gaussian regression filter takes care of the edge effect that happens during smoothing. This is illustrated in figure 3 below. Here, a section of a sine wave with random noise added is filtered.

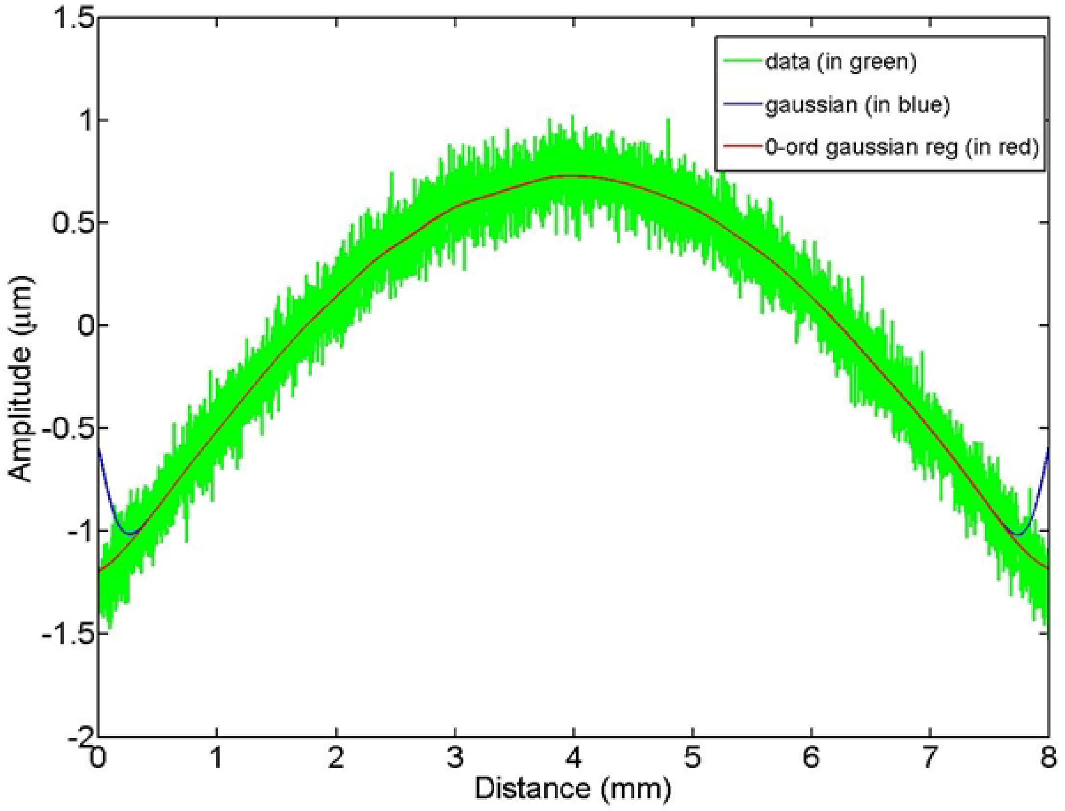


Figure 7: The edge behaviors of the zero order Gaussian regression filter and the regular Gaussian filter [as has been shown in 2D by Muralikrishnan and Raja (2009)]

The zero-order Gaussian regression filter [Muralikrishnan and Raja, 2009, pages 68 to 69 and 73] is obtained by solving for a constant weighting function in 3D that minimizes the discretized energy functional:

$$E(k_x, k_y) = \sum_{p_y=1}^{n_x} \sum_{p_x=1}^{n_y} \left(z(p_x, p_y) - C(k_x, p_x, k_y, p_y)\right)^2 S(k_x, p_x, k_y, p_y)\Delta x\, \Delta y$$

The filtered result, $C(k_x, p_x, k_y, p_y)$, is a constant function here,

$$S(x, y) = \frac{1}{\alpha\lambda_{cx}\alpha\lambda_{cy}} \exp\left[-\pi\left(\frac{x}{\alpha\lambda_{cx}}\right)^2 - \pi\left(\frac{y}{\alpha\lambda_{cy}}\right)^2\right],$$

And $S(k_x, p_x, k_y, p_y)$ is defined as:

$$S(k_x, p_x, k_y, p_y) = S(p_x - k_x, p_y - k_y) = S(p_x - k_x)S(p_y - k_y).$$

The exponential of a sum is factored here.

The associated formula for it is obtained as follows:

$$\frac{\partial E}{\partial C} = \sum_{p_y=1}^{n_y} \sum_{p_x=1}^{n_x} 2\left(z(p_x, p_y) - C(k_x, p_x, k_y, p_y)\right) S(k_x, p_x, k_y, p_y)\Delta x\,\Delta y = 0$$

$$\Rightarrow C(k) = C_x(k)C_y(k) = \left[\frac{\sum_{p=1}^{n} z(p_x, p_y)S(k_x, p_x)\Delta x}{\sum_{p=1}^{n} S(k_x, p_x)\Delta x}\right]\left[\frac{\sum_{p=1}^{n} z(p_x, p_y)S(k_y, p_y)\Delta y}{\sum_{p=1}^{n} S(k_y, p_y)\Delta y}\right]$$

Since $S(k_x, p_x, k_y, p_y) = S(k_x, p_x)S(k_y, p_y)$.

The resulting filtering process is not shift invariant due to the fact that the normalizing factors, $\sum_{p=1}^{n} S(k_x, p_x)\Delta x$ and $\sum_{p=1}^{n} S(k_y, p_y)\Delta y$, vary with each (k_x, k_y). The combined factor enables the sum to be a weighted average of only data values and keeps the filtered result from escaping at the ends. This way, we do not need to worry about cropping out corrupted parts of the filtered result as we do with regular convolution with zero-padded data (*see* [Muralikrishnan and Raja, 2009, chapter 3] *for an explanation of discrete convolution*).

The m-file function that implements the zero order Gaussian regression filter is called zerogaussreg.m and the filtertype name for calling it in other functions is the string 'zerogaussreg'. The subfunction that does the convolution is called getzerogaussreg2 in line 105 of zerogaussreg.m. This one uses a forloop similar to the one provided in [Muralikrishnan and Raja, 2009, page 74]. Another subfunction, getzerogaussreg in line 153, is similar to getfiltdatavec in filterimg.m having the ability to do the summations on selected number of rows and number of columns at a time.

The 3D Second-Order Gaussian Regression Filter

The 3D second order Gaussian regression filter is left as an exercise in [Muralikrishnan and Raja, 2009, page 74]. So it was derived and implemented in this project. The filter is obtained by solving for a quadratic weighting function that minimizes the discretized energy functional[1]:

$$E(k_x, k_y) = \sum_{p_y=1}^{n_x} \sum_{p_x=1}^{n_y} \left(z(p_x, p_y) - w(k_x, p_x, k_y, p_y)\right)^2 S(k_x, p_x, k_y, p_y)\Delta x\,\Delta y$$

The filtered result, $w(k_x, p_x, k_y, p_y)$, is a quadratic function determined at each (p_x, p_y) point. Its general expression is found in [Muralikrishnan and Raja, 2009, page 74].

$$w(k_x, p_x, k_y, p_y) = Ax(\mathrm{k}_x, k_y)^2 + By(\mathrm{k}_x, k_y)^2 + Cx(\mathrm{k}_x, k_y) + Dy(\mathrm{k}_x, k_y)$$
$$+Ex(\mathrm{k}_x, k_y)y(\mathrm{k}_x, k_y) + F$$

And $S(k_x, p_x, k_y, p_y)$ is the same Gaussian function defined section 4.3.

We obtain a system of equations when we differentiate to find the minimum:

$$\frac{\partial E}{\partial A} = \sum_{p_y=1}^{n_x} \sum_{p_x=1}^{n_y} 2\left(z(p_x, p_y) - w(k_x, p_x, k_y, p_y)\right) x(\mathrm{k}_x, k_y)^2 S(k_x, p_x, k_y, p_y)\Delta x\,\Delta y = 0$$

$$\frac{\partial E}{\partial B} = \sum_{p_y=1}^{n_x} \sum_{p_x=1}^{n_y} 2\left(z(p_x, p_y) - w(k_x, p_x, k_y, p_y)\right) y(\mathrm{k}_x, k_y)^2 S(k_x, p_x, k_y, p_y)\Delta x\,\Delta y = 0$$

$$\frac{\partial E}{\partial C} = \sum_{p_y=1}^{n_x} \sum_{p_x=1}^{n_y} 2\left(z(p_x, p_y) - w(k_x, p_x, k_y, p_y)\right) x(\mathrm{k}_x, k_y) S(k_x, p_x, k_y, p_y)\Delta x\,\Delta y = 0$$

$$\frac{\partial E}{\partial D} = \sum_{p_y=1}^{n_x} \sum_{p_x=1}^{n_y} 2\left(z(p_x, p_y) - w(k_x, p_x, k_y, p_y)\right) \mathrm{y}(\mathrm{k}_x, k_y) S(k_x, p_x, k_y, p_y)\Delta x\,\Delta y = 0$$

$$\frac{\partial E}{\partial E} = \sum_{p_y=1}^{n_x} \sum_{p_x=1}^{n_y} 2\left(z(p_x, p_y) - w(k_x, p_x, k_y, p_y)\right) x(\mathrm{k}_x, k_y) y(\mathrm{k}_x, k_y) S(k_x, p_x, k_y, p_y)\Delta x\,\Delta y = 0$$

$$\frac{\partial E}{\partial F} = \sum_{p_y=1}^{n_x} \sum_{p_x=1}^{n_y} 2\left(z(p_x, p_y) - w(k_x, p_x, k_y, p_y)\right) S(k_x, p_x, k_y, p_y)\Delta x\,\Delta y = 0$$

[1] Chapter 9.2, page 74 of Computational Surface and Roundness Metrology by Bala Muralikrishnan and Jay Raja. Copyright 2009, Springer-Verlag London Limited

The function, secgaussreg.m, solves this system. It has two options for how to solve it. That option is the input setting Dirsw. Dirsw = 1 will make it use the subfunction, getQM2, which ignores the xy interaction embodied in the equation for $\frac{\partial E}{\partial E}$. On the other hand, changing the setting to 0 will cause it to use another subfunction, getQM. This one solves the entire problem including the interaction. Dirsw = 1 is faster because it uses less number of calculations.

In the secgaussreg.m code, the matrix, M, represents the dot multiplication of the Gaussian with terms in the expression for w. The vector, Q, represents the filtering of the data with the Gaussian function. These two variables, M and Q are obtained by distributing $S(k_x, p_x, k_y, p_y)$ in the equations above.

Also, unlike the zero-order Gaussian regression code, this project allows secgaussreg.m to use only those points falling within the filter cutoffs instead of the entire number of points determined by the horizontal and vertical lengths of the data. This restriction is created in lines 91 to 117. Although this spatial domain windowing helps to speed up the calculation, it increases the oscillation amplitude at non-zero center frequencies when the modulation method is used. When used a low-pass filter, it exhibits better edge behavior than the zero-order Gaussian regression filter.

The Robust 3D Zero-Order Gaussian Regression Filter

The robust zero-order Gaussian regression filter is a weighted zero-order Gaussian regression filter. It is an iterative filter with weights calculated after each iteration. See [Brinkmann et. al, 2000 and 2001] for explanations about the filter. It is used to reduce or remove unwanted spikes from filtered data.

The filtertype is the string ‘robustgauss’. The code used is in robustgauss.m. It is similar to the 2D case provided in [Muralikrishnan and Raja, 2009, page 90]. The actual filtering code is the zero-order Gaussian regression code modified to accept a weighting function called delta before convolution. This function is applied on the data first before the filter is applied. It was extended to handle 3D data by using matrix assignments instead of vector assignments.

The code in robustgauss.m, however, foregoes the original filtering forloop and uses matrix multiplications to apply the filter. The sub-function, getzerogaussregwithdelta given in line 160, applies the filter matrix on the weighted data.

The filter matrix and the delta matrix are created before getzerogaussregwithdelta is called in line 107. The sub-function, getS11 given in line 204 and called in line 101, creates the non-normalized filter matrix. Another sub-function, getS given in line 291 and called in line 106, creates a matrix of normalizing coefficients, taking delta into account. The normalization is applied in lines 186 and 190. This normalization makes the filtering akin to an averaging process.

In order to calculate the weights in the entries of the delta matrix, averages are estimated along each row and each column in line 133. The row averages are repeated across the columns to create a matrix of row averages, called My. Likewise, the column averages are repeated down the rows to make a matrix of column averages, called Mx. These two matrices are added to create an overall matrix of averages that is used in the eventual calculation of the delta matrix in lines 134 to 143.

The Non-periodic Spline Filter

The non-periodic spline filter code in [Muralikrishnan and Raja, 2009, pages 80 to 81] is largely left intact. The m-file is called npspline2.m. The only changes inserted were modulation in lines 24 to 31 and 63. Also, all the vector assignments were made into matrix assignment to allow for when the data is a matrix. The horizontal and vertical directions are kept independent, allowing this filter to be called twice in order to filter both directions.

CHAPTER 5: Two Band-Pass Filtering Techniques

This chapter first presents two methods for generating a band-pass filter. It then goes on to derive equations used to convert from the bandwidth and center wavelength to either upper and lower wavelength cutoffs or to upper and lower frequency cutoffs, or to center frequency and frequency bandwidth.

Using a sequence of low and high pass filters (Method A)

In this case, a low pass filter is first applied using the upper wavelength cutoff, λ_{uc}. Then, that result is filtered with a high pass filter at the lower wavelength cutoff, λ_{lc}. The cutoffs refer to the wavelength where the filter has approximately 50% transmission. The resulting overall 50% cutoff for the new filter is drastically different from the original specifications. This is caused by the resulting multiplication of the previous two filters in the frequency domain. Thus, it is difficult to specify the overall cutoff for this method.

This method was implemented using the SCOUT gau_filt.m code. The code treats the cutoff as the full width of the spatial filter, whereas, the next method treats it as half the spatial width. Two maintain positive upper and lower cutoffs,, they are not allowed to go below twice the spacing. If any of them goes below this threshold, it is set equal to twice the spacing. Likewise, they are not allowed to exceed the data spatial lengths in their corresponding directions. When any of them exceeds the corresponding length, its value is reduced to that length. This is implemented in lines 45 to 87 of the bndfilt.m code.

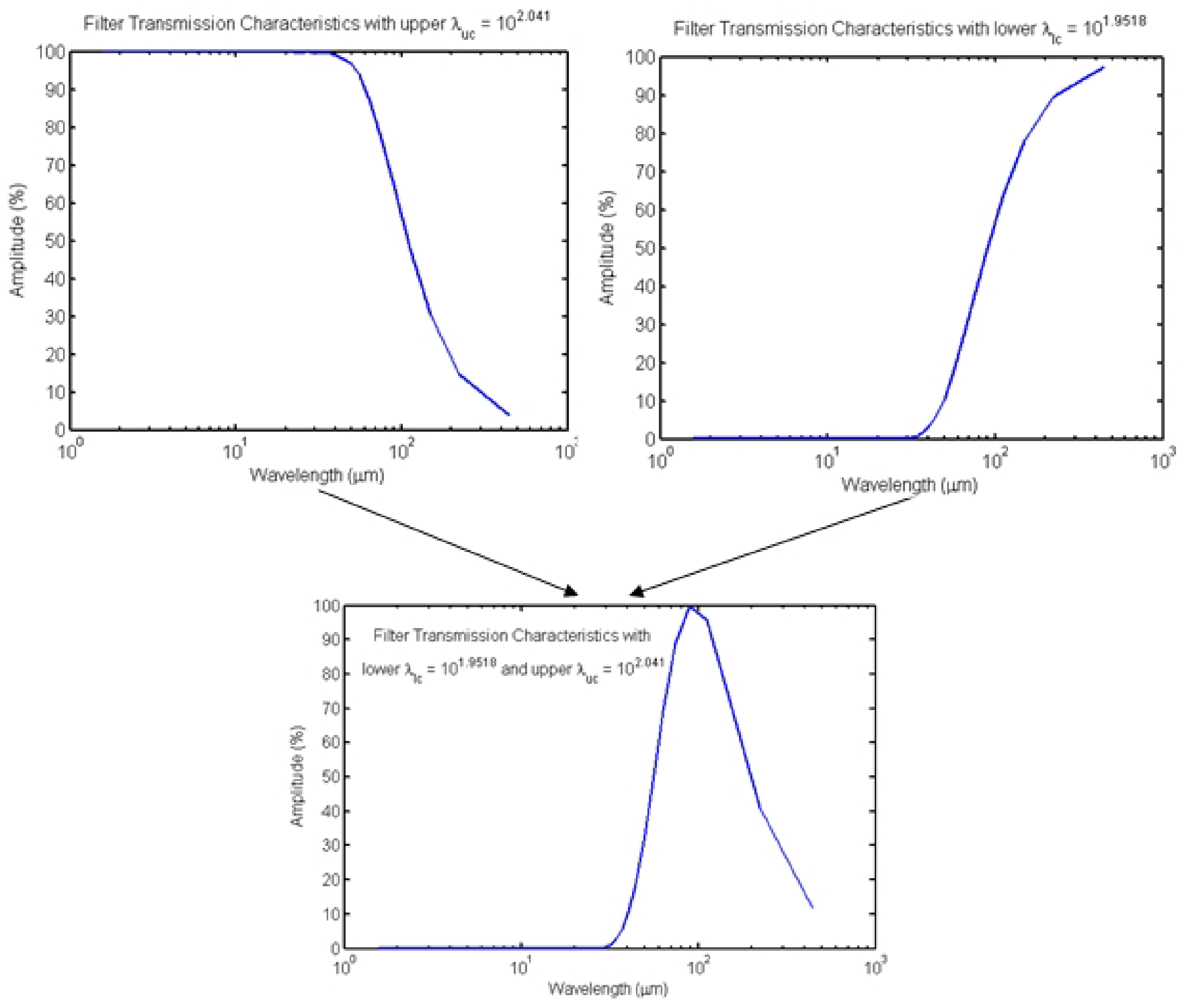


Figure 8: Combination of high and low pass filters to create a band pass filter

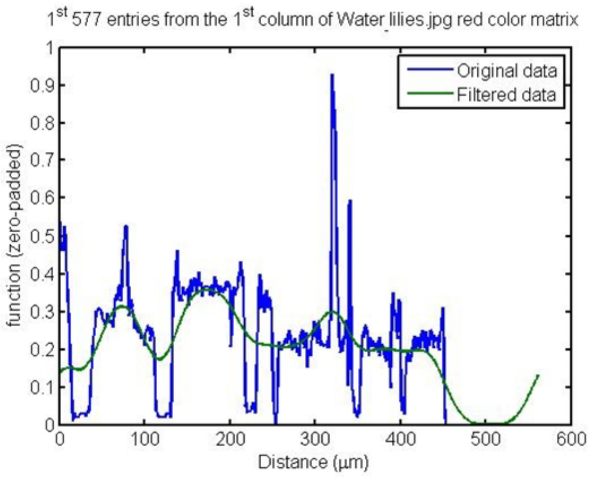


Figure 9: filtered data using traditional Gaussian band-pass

Using cosine modulation to attain frequency shift (Method B)

In this case, the given low-pass filter is spatially modulated with a discrete cosine function at a particular center frequency. This shifts the center of the filter's frequency response from zero frequency to the new center by even symmetry.

When the filter changes as its spatial center is translated during convolution, as can be seen in all the non-Gaussian filters used in this project, the data is modulated instead of the filter. This is accomplished at the beginning of each filtering code and at the end. In this case, the complex exponential is employed instead of the cosine function. Then the real part of the result is taken at the end. This is the same as cosine modulation. See Appendix B for explanations about using spatial modulation to implement frequency shift.

The filter transmission graph using Method A is shown in Figure 10 while the corresponding graph for Method B is shown in Figure 11. These were created using a center wavelength of $\lambda_{center} = 30(\mu m)$ and a cutoff (prior to shifting) of $\lambda_c = 50(\mu m)$. The resulting upper and lower wavelengths cutoffs where: $\lambda_{uc} = 10^{1.644}(\mu m)$ and $\lambda_{lc} = 10^{1.3696}(\mu m)$ respectively. The explanations for how to convert between Method A specifications and Method B specifications are provided in the next section.

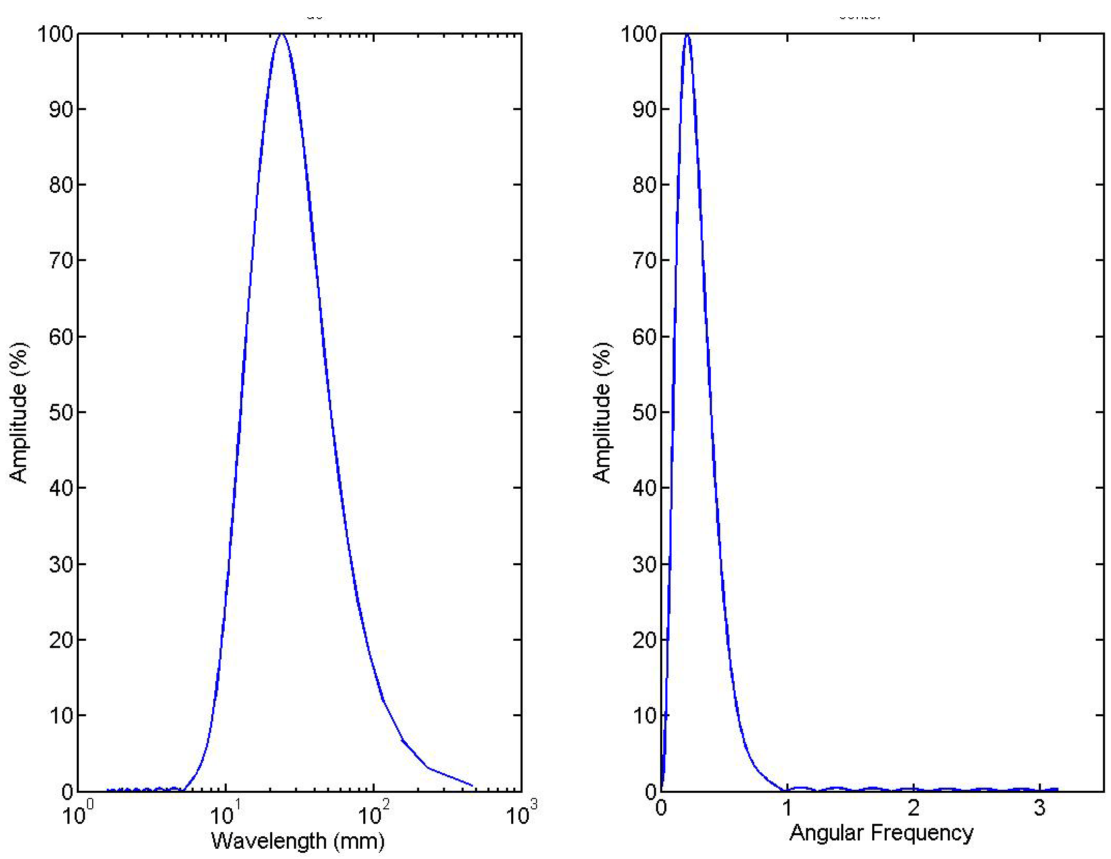


Figure 10: Gaussian filter using Method A

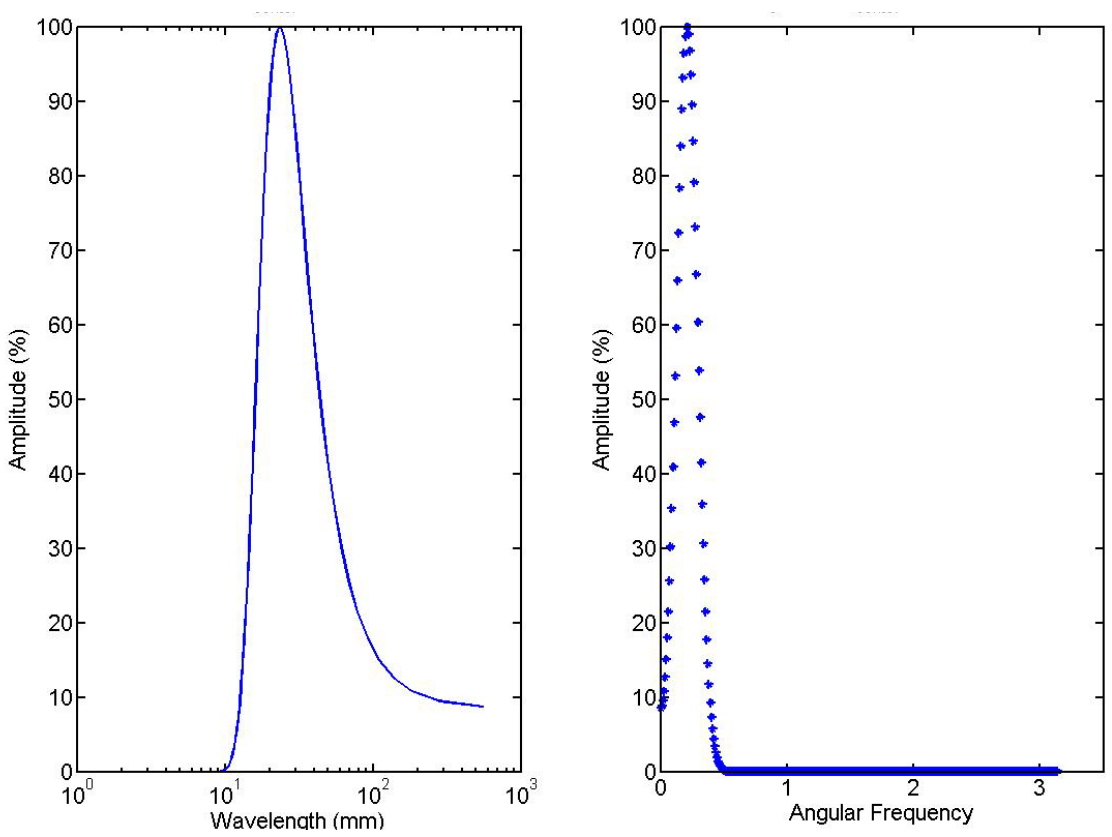


Figure 11: Gaussian filter using Method B

Notice how the left tail in Method A relatively resists crossing the vertical axis compared to Method B. This is due to the restriction on its upper wavelength cutoff. Method B appears to cross more readily due to aliasing. On the other hand, Method B is both narrower in general and has more precise upper and lower wavelength cutoffs. Method A's upper and lower wavelength cutoffs are less precise because the cutoff transmission of one of the filters involved in the sequence is always corrupted by the nearby values of the other filter by multiplication. This multiplication makes the overall wavelength cutoffs to be different from the specification. (See Appendix B.1 for the relationship between spatial domain convolution and frequency domain multiplication)

Formulas for converting between band-pass parameter specifications

In this case, the given low-pass filter is spatially modulated with a discrete cosine function at a particular center frequency. This shifts the center of the filter's frequency response from zero frequency to the new center by even symmetry.

Care can be taken to control the distribution's width on the wavelength axis. Simply translating the distribution's center frequency will preserve its width on the frequency axis, but will not do so on the wavelength axis. For the same frequency bandwidth, the wavelength bandwidth at a higher center wavelength is larger than the wavelength bandwidth at a lower center wavelength. We will study and create formulas that help to control this issue.

Let the spatial length be $L = N\Delta x$, where N is the number of points and Δx is the sampling interval. Let $k \in \mathbb{N}$ be the discrete frequency variable. The formula to used to convert from the discrete frequency axis to the wave axis is: $\lambda_k = \frac{N\Delta x}{k}$ with $1 \leq k \leq floor\left(\frac{N}{2}\right)$. The formula used to convert from the discrete frequency axis to the angular frequency axis is: $\omega_k = \frac{2\pi k}{N}$. To obtain the angular frequency cutoff from the spatial cutoff, one uses $\omega_c = \frac{2\pi\Delta x}{\lambda_c}$. For a filter with N points, the observable wavelength range is $2\Delta x \leq \lambda \leq \frac{N\Delta x}{2}$ [Muralikrishnan and Raja, 2009, page 26].

One might ask, what is the range available for the cutoff λ_c? At high center frequencies, the lower limit of λ is too small especially when considering we are using a discrete band-pass filter created by frequency translation and with its number of points limited by $\frac{2\lambda_c}{\Delta x}$. The filter's width in the frequency domain can be too wide with the resulting aliasing drastically altering the distribution's shape and bandwidth. We also have a similar problem at the higher limit for λ. In this case, we run the risk of only studying the frequency characteristics of the filter instead of obtaining useful frequency information from the data.

To control these problems, we use the Gaussian as a model filter and calculate the minimum and maximum positions for the center frequency for a given bandwidth that satisfies 50% transmission at the cutoffs.

To get the minimum center frequency, we apply the following process:

Given the wavelength bandwidth $\Delta\lambda$ and the discrete frequency half bandwidth Δk, we want to find both $k_{centermin}$ and λ_c, where $\lambda_c = \lambda_{cmax}$ is the low-pass cutoff before the frequency shift is applied.

Set the upper and lower frequency cutoffs as:

$$(1)\ k_{uc} = k_{centermin} + \Delta k$$

$$(2)\ k_{lc} = k_{centermin} - \Delta k = 1 = k_{min}$$

with λ_c defined as:

$$\lambda_c = \frac{N\Delta x}{\Delta k}$$

(This cutoff is half the wavelength bandwidth)

Then:

$$\Delta\lambda = \frac{N\Delta x}{k_{lc}} - \frac{N\Delta x}{k_{uc}} = \frac{N\Delta x}{1} - \frac{N\Delta x}{k_{uc}} = N\Delta x - \frac{N\Delta x}{k_{centermin} + \Delta k}$$

$$\frac{N\Delta x}{k_{centermin} + \Delta k} = N\Delta x - \Delta\lambda$$

$$\Rightarrow \Delta k = \frac{N\Delta x}{(N\Delta x - \Delta\lambda)} - k_{centermin} = \frac{N\Delta x}{(N\Delta x - \Delta\lambda)} - (1 + \Delta k)$$

by substitution of equation (2).

$$\Rightarrow \Delta k = \frac{N\Delta x}{2(N\Delta x - \Delta\lambda)} - \frac{1}{2}$$

by isolation. We get both λ_c and $k_{centermin}$ as:

$$\lambda_c = \frac{N\Delta x}{\Delta k} = \frac{N\Delta x}{\frac{N\Delta x}{2(N\Delta x - \Delta\lambda)} - \frac{1}{2}}$$

$$k_{centermin} = 1 + \Delta k$$

The minimum angular center frequency, with a left tail satisfying a 50% transmission at $k = 1$ is then $\omega_{centermin} = 2\pi k_{centermin}$. Likewise, the maximum angular center frequency that satisfies that condition is $\omega_{centermax} = 2\pi k_{centermax}$. The corresponding center wavelength we use is the center of the wavelength interval between the lower and upper wavelength cutoffs:

$$\lambda_{uc} = \frac{N\Delta x}{k_{lc}}$$

$$\lambda_{lc} = \frac{N\Delta x}{k_{uc}}$$

Of course, the wavelength peak of the Gaussian is still obtained by the conversion:

$$\lambda_{peak} = \frac{2\pi\Delta x}{\omega_{center}}$$

Now, let us find the maximum center frequency satisfying a 50% transmission at $k = \left\lfloor\frac{N}{2}\right\rfloor$. (Note that due to the inverse relation between wavelength and frequency, we used $k = 1$ when we were calculating the minimum because that corresponds to the maximum observable finite wavelength. Likewise, the minimum observable wavelength corresponds to the maximum discrete frequency.)

Given $\Delta\lambda$ and Δk, we want to find $k_{centermax}$ and λ_c, where $\lambda_c = \lambda_{cmin}$ is the low-pass cutoff before the frequency shift is applied.

Set the upper and lower frequency cutoffs as:

$$(3)\ \ k_{uc} = k_{centermax} + \Delta k = \left\lfloor\frac{N}{2}\right\rfloor = k_{max}$$

$$(4)\ \ k_{lc} = k_{centermax} - \Delta k$$

Then:

$$\Delta\lambda = \frac{N\Delta x}{k_{lc}} - \frac{N\Delta x}{k_{uc}} = \frac{N\Delta x}{k_{centermax} - \Delta k} - \frac{N\Delta x}{\left\lfloor\frac{N}{2}\right\rfloor}$$

$$\frac{N\Delta x}{k_{centermax} - \Delta k} = \frac{N\Delta x}{\left\lfloor\frac{N}{2}\right\rfloor} + \Delta\lambda$$

$$\Rightarrow \varDelta k = \frac{N\Delta x}{\left(\frac{N\Delta x}{\left\lfloor\frac{N}{2}\right\rfloor} + \varDelta\lambda\right)} + k_{centermax} = \frac{N\Delta x}{\left(\frac{N\Delta x}{\left\lfloor\frac{N}{2}\right\rfloor} + \varDelta\lambda\right)} + \left(\left\lfloor\frac{N}{2}\right\rfloor - \varDelta k\right)$$

by substitution of equation (3).

We get both λ_c and $k_{centermax}$ as:

$$\lambda_c = \frac{N\Delta x}{\frac{N\Delta x}{\left(\frac{N\Delta x}{\left\lfloor\frac{N}{2}\right\rfloor} + \varDelta\lambda\right)} + \left(\left\lfloor\frac{N}{2}\right\rfloor - \varDelta k\right)}$$

$$k_{centermax} = \left\lfloor\frac{N}{2}\right\rfloor - \varDelta k$$

Note that the transformation from the frequency axis to the wavelength axis is nonlinear. In the MATLAB code used, the discrete frequency cutoffs are applied before the shift is implemented. To help preserve a certain wavelength width, the spatial cutoff, λ_c, must be changed before shifting. (That is, the frequency bandwidth of the filter should be changed as the distribution is being translated on the frequency axis). The wavelength width preserving cutoff formula (in the absence of aliasing) is calculated in the following manner:

Given the wavelength bandwidth $\Delta\lambda$ and the center frequency k_{center}, we want to find both the discrete frequency half bandwidth Δk and λ_c, where λ_c is the low-pass cutoff before the frequency shift is applied.

Set the upper and lower frequency cutoffs as:

$$(5)\ k_{uc} = k_{center} + \varDelta k$$

$$(6)\ k_{lc} = k_{center} - \varDelta k$$

Then:

$$\Delta\lambda = \frac{N\Delta x}{k_{lc}} - \frac{N\Delta x}{k_{uc}} = \frac{N\Delta x}{k_{center} - \varDelta k} - \frac{N\Delta x}{k_{center} + \varDelta k}$$

$$\Rightarrow \Delta\lambda = \frac{N\Delta x 2\, \Delta k}{k^2{}_{center} - \Delta k^2}$$

by substitution of equations (5) and (6). We then obtain the quadratic equation

$$\Rightarrow \Delta k^2 + \frac{N\Delta x 2\, \Delta k}{\Delta\lambda} - k^2{}_{center} = 0$$

by rearrangement. We get both λ_c and Δk as:

$$\Delta k = -\frac{N\Delta x}{\Delta\lambda} + \sqrt{\left(\frac{N\Delta x}{\Delta\lambda}\right)^2 + k^2{}_{center}}$$

(the positive root is chosen here)

$$\lambda_c = \frac{N\Delta x}{\Delta k}.$$

Finally, we want to be able to determine the center frequency k_{center} and the wavelength bandwidth $\Delta\lambda$, given λ_c and λ_{center}. This is useful when one wants to relate the translation method to the subtraction method.

Set the upper and lower wavelength cutoffs as:

$$(7)\ \lambda_{uc} = \frac{N\Delta x}{k_{lc}}$$

$$(8)\ \lambda_{lc} = \frac{N\Delta x}{k_{uc}}$$

Define the center wavelength as:

$$\lambda_{center} = \frac{\lambda_{uc} + \lambda_{lc}}{2} = \frac{N\Delta x}{2}\left(\frac{1}{k_{lc}} + \frac{1}{k_{uc}}\right)$$

by substituting equations (7) and (8).

Then:

$$\lambda_{center} = \frac{N\Delta x}{2}\left(\frac{1}{k_{center} - \Delta k} + \frac{1}{k_{center} + \Delta k}\right)$$

by substituting equations (5) and (6).

$$\Rightarrow \lambda_{center} = \frac{N\Delta x k_{center}}{k^2{}_{center} - \Delta k^2}$$

Define $\Delta\lambda$ as:

$$\Delta\lambda = \lambda_{uc} - \lambda_{lc} = \left(\frac{N\Delta x}{k_{center} - \Delta k} - \frac{N\Delta x}{k_{center} + \Delta k}\right) = \frac{N\Delta x 2\Delta k}{k^2{}_{center} - \Delta k^2}$$

We then obtain the following quadratic equation:

$$k^2{}_{center}\lambda_{center} - N\Delta x k_{center} - \Delta k^2\lambda_{center} = 0$$

by rearrangement. We get k_{center} as:

$$k_{center} = \frac{N\Delta x + \sqrt{(N\Delta x)^2 + 4\lambda^2{}_{center}\Delta k^2}}{2\lambda_{center}}$$

(The positive root is chosen here.)

To finish, recall that:

$$\lambda_c = \frac{N\Delta x}{\Delta k} \Rightarrow \Delta k = \frac{N\Delta x}{\lambda_c}.$$

λ_{center} is given and we can get Δk from λ_c, which is also given. Using k_{center} and Δk, $\Delta\lambda$ is calculated as:

$$\Delta\lambda = \frac{2N\Delta x\ \Delta k}{k^2{}_{center} - \Delta k^2}$$

This is possible because we derived formulas for calculating k_{center} and Δk. So, λ_c is now related to both λ_{center} and $\Delta\lambda$ and we have a more precise control. These formulas are implemented in the auxiliary function, uniformwidths2.m.

CHAPTER 6: Application and Discussion

This chapter displays results of filtering surface data with the chosen filters explained in chapters 4 and 5. The two roughness parameters, Sq and Sa, given in chapter 1, are calculated for each. The filtering is performed at a list of center wavelengths and bandwidths using both methods discussed in chapter 5 for creating a band-pass filter.

With each filter, the data file, 1-1.sdf from [Berglund, 2009], is filtered at the specified bandpass parameters listed in Table 4. Sa and Sq are calculated and listed in Table 5, Table 6, and Table 7. Then these Sa and Sq values are plotted against the center wavelength in Figure 12 and Figure 13 respectively. The units for the parameters are in micrometers. Because it takes too long to analyze a file, results from secgaussreg.m are not included below.

cutoffs (μm)	Wavelength Bandwidths(μm)	Center Wavelengths(μm)
50	552.0797	300
50	492.3099	270
50	432.5971	240
50	372.9657	210
50	313.4556	180
50	254.1381	150
50	195.153	120
50	136.8154	90
50	80	60
50	28.1025	30

Table 4: Band-pass filter parameters

Center Wavelengths		gau_filt			gaussian	
		Sa(μm)	Sq(μm)		Sa(μm)	Sq(μm)
30		0.219055250	0.384200620		0.011731102	0.019064977
60		0.425065290	0.671016220		0.116983690	0.147262740
90		0.618700780	0.906102110		0.227072980	0.277813960
120		0.800990320	1.107939800		0.301787180	0.365818700
150		0.975889510	1.295359800		0.352639290	0.425571370
180		1.134419200	1.460630200		0.388862770	0.468075920

210		1.268853300	1.599655900		0.415791130	0.499647480
240		1.381489300	1.716203000		0.436524760	0.523946120
270		1.488103900	1.827859300		0.452951060	0.543192170
300		1.588292700	1.934124800		0.466272900	0.558797630

Table 5: gau_filt and gaussian filtertype Sa and Sq results

Center Wavelengths		npspline			rob_filt	
		Sa(μm)	Sq(μm)		Sa(μm)	Sq(μm)
30		0.016064355	0.046289900		0.149259190	0.226675380
60		0.106549930	0.183543260		0.307970380	0.457068480
90		0.216319700	0.323927120		0.522087560	0.743229900
120		0.302489810	0.429647130		0.753932290	1.009561100
150		0.366243440	0.506533370		0.982360550	1.259065700
180		0.413981440	0.563596420		1.173314000	1.467393900
210		0.450640650	0.607182340		1.339133700	1.649587800
240		0.479514460	0.641387570		1.482813800	1.808357700
270		0.502773860	0.668869180		1.600346700	1.939260800
300		0.521876680	0.691394090		1.697610900	2.048328000

Table 6: npspline and rob_filt filtertype Sa and Sq results

Center Wavelengths		robustgauss			zerogassreg	
		Sa(μm)	Sq(μm)		Sa(μm)	Sq(μm)
30		0.012943818	0.026146479		0.013269034	0.026535190
60		0.128308370	0.171900270		0.128463260	0.172524390
90		0.246356730	0.313674320		0.246579930	0.314445620
120		0.325975570	0.408423800		0.326292470	0.409303290
150		0.380039420	0.472529760		0.380425550	0.473488230
180		0.418499590	0.518044860		0.418942370	0.519061700
210		0.447066850	0.551812010		0.447553840	0.552874160
240		0.469052220	0.577779170		0.469573710	0.578877030
270		0.486464940	0.598335080		0.487015320	0.599461640
300		0.500581740	0.614994660		0.501157130	0.616144850

Table 7: robustgauss and zerogauss filtertype Sa and Sq results

Center Wavelengths		gaussian (using a sequence of low-pass high-pass and filters)	
		Sa(μm)	Sq(μm)
30		0.2128624900	0.3758557100
60		0.4197805300	0.6647651300

90		0.6158307500	0.9038451500
120		0.8043552000	1.1134129000
150		0.9803887900	1.3013416000
180		1.1443979000	1.4730001000
210		1.2967199000	1.6320191000
240		1.4373813000	1.7797502000
270		1.5661078000	1.9160325000
300		1.6824037000	2.0400595000

Table 8: gaussian filtertype Sa and Sq results (using a sequence of low-pass and high-pass filters)

The Sa and Sq curves seem to be very similar to each other. The gau_filt and rob_filt filtertypes used the sequence of low-pass and high-pass filters to create a band-pass filter. The other filters used the modulation method to translate the center frequency from zero to the new one corresponding to the center wavelength.

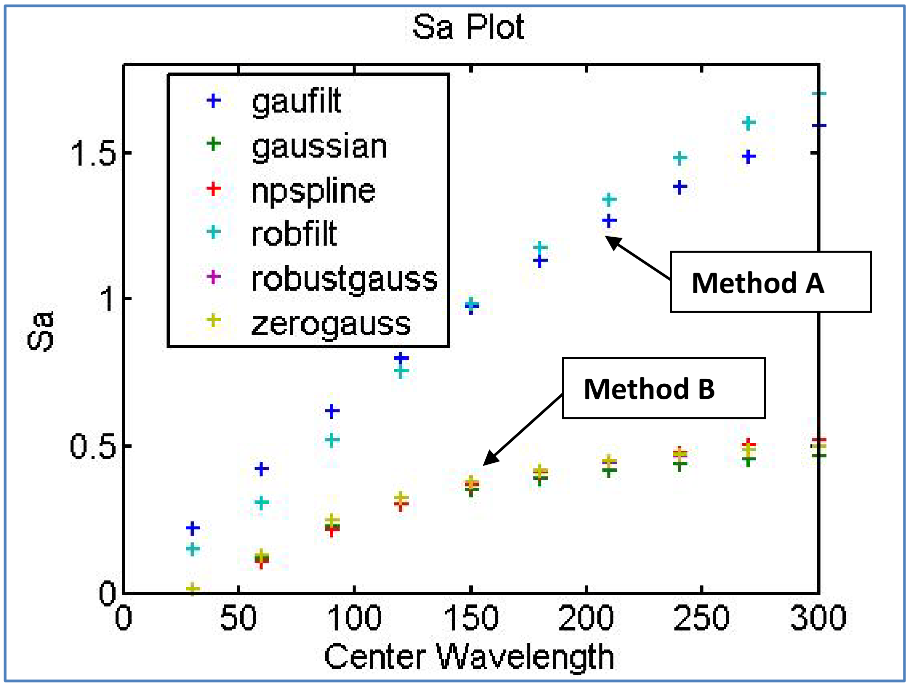


Figure 12: Sa versus center wavelength plots of filtered data

In Figures 12 and 13, the curves produced by SCOUT's Gaussian (gaufilt legend, with gau_filt as the filtertype) and robust Gaussian (robfilt legend, with rob_filt as the filtertype) codes are colored dark blue and cyan respectively. These two curves were produced using Method A, which is band-pass filtering with a sequence of low-pass and high-pass filters as discussed in chapter 5. The other curves are produced using Method B, which is band-pass filtering with the center frequency translation method.

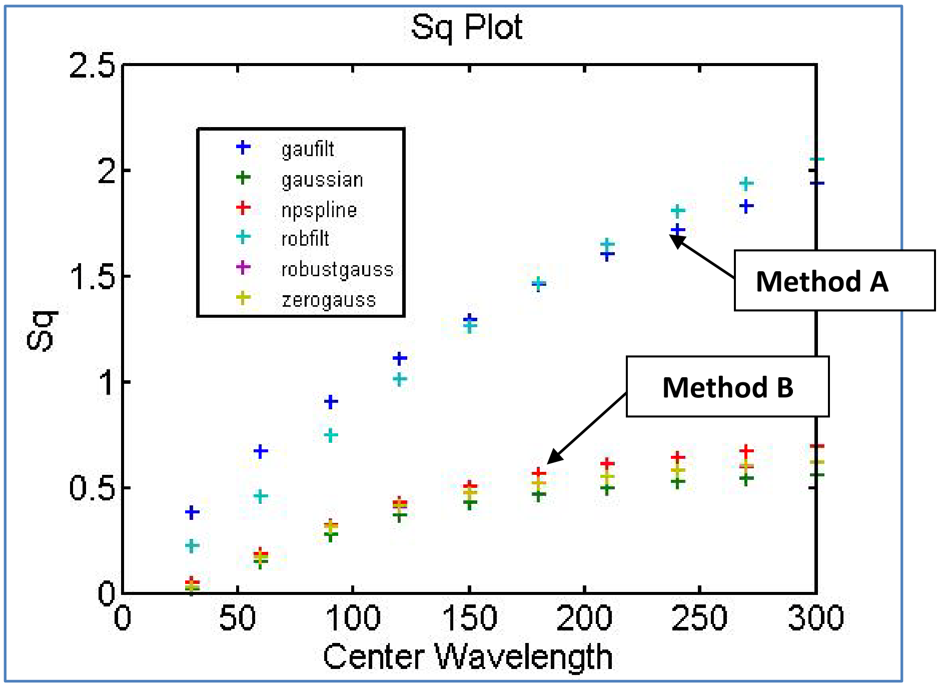


Figure 13: Sq versus center wavelength plots of filtered data

The gau_filt and the rob_filt graphs are growing faster than the others. They seem to have a higher limit close than the rest as the center wavelength is increased. This seems to be mostly due to the band-pass method because when the gaussian filtertype was used with the sequence of filters, it gave results that were very similar to the one from gau_filt. This is shown in Figure 14 and Figure 15. For the gaussian filitertype, the corresponding table for this band-pass method is Table 8.

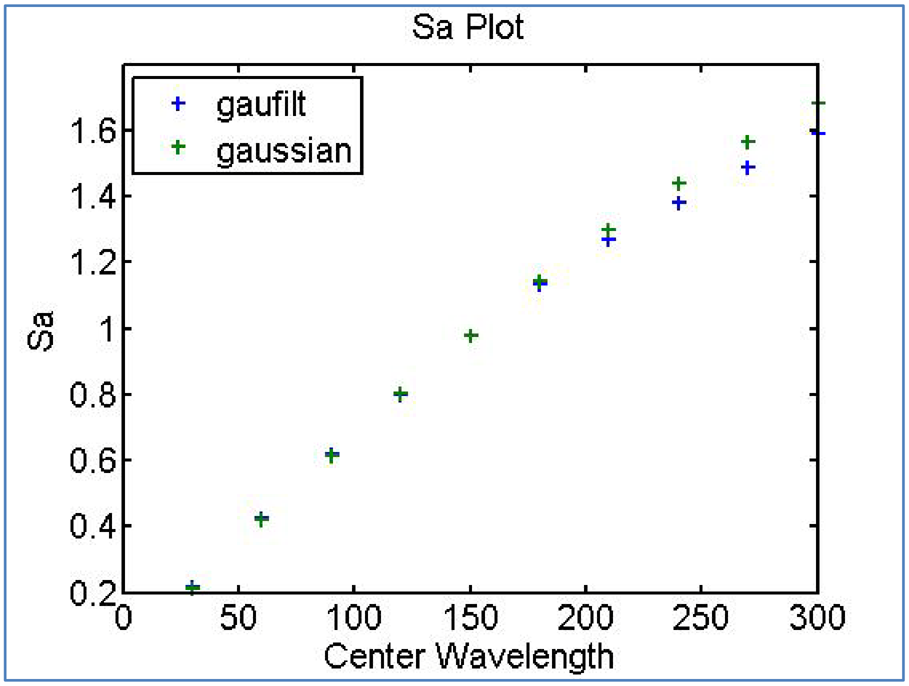


Figure 14: Sa plots for the gau_filt versus the gaussian filtertype, using a sequence of low-pass and high-pass filters

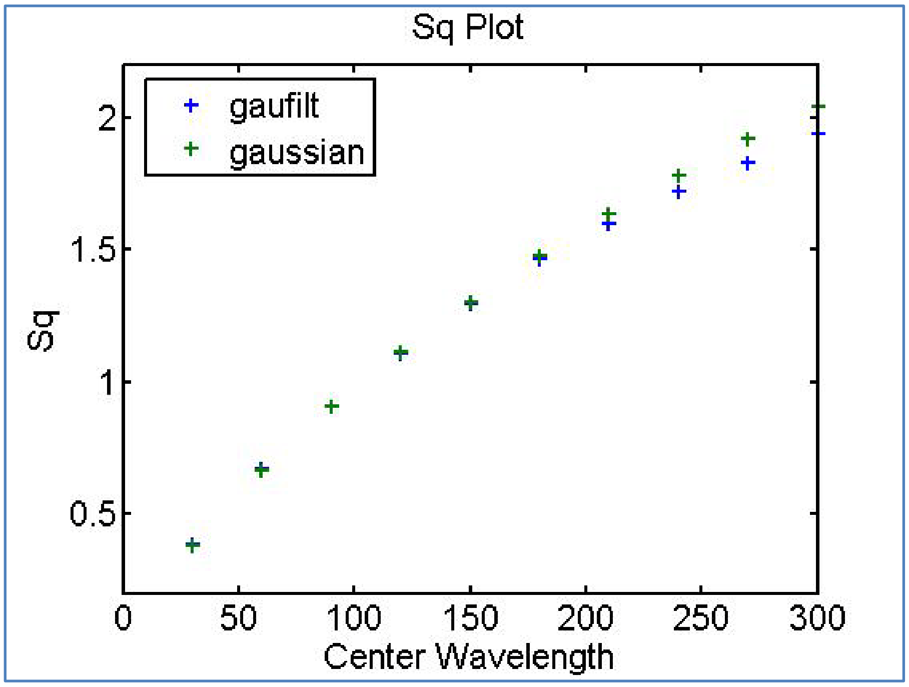


Figure 15: Sq plots for the gau_filt versus the gaussian filtertype, using a sequence of low-pass and high-pass filters

The following figures below give a pictorial illustration of the effect of band-pass filtering on surface data. These were create with a cutoff of 50μm and a center wavelength of 120μm. The last figure is original surface data.

The SCOUT filters, gau_filt and rob_filt look more similar to the original because they are not modified to use the center frequency translation method. They do not attenuate low frequencies as well as the other filtering method does. This is due to the fact that the overall filter, obtained by the sequence of low-pass and high-pass filtering, has a broader frequency distribution than that obtained by center frequency translation. (See Figures 10 and 11 in chapter 5 for a comparison of the filters' shapes.)

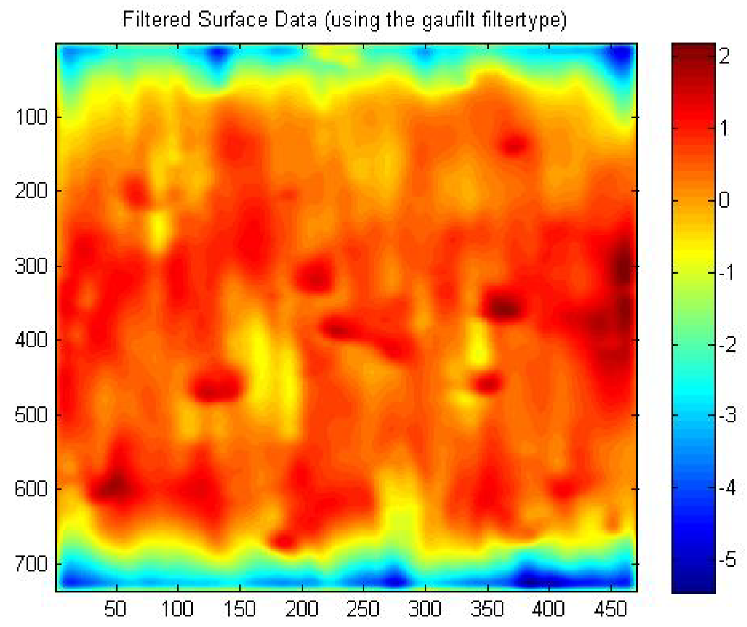


Figure 16: Filtered surface data using the SCOUT Gaussian filter

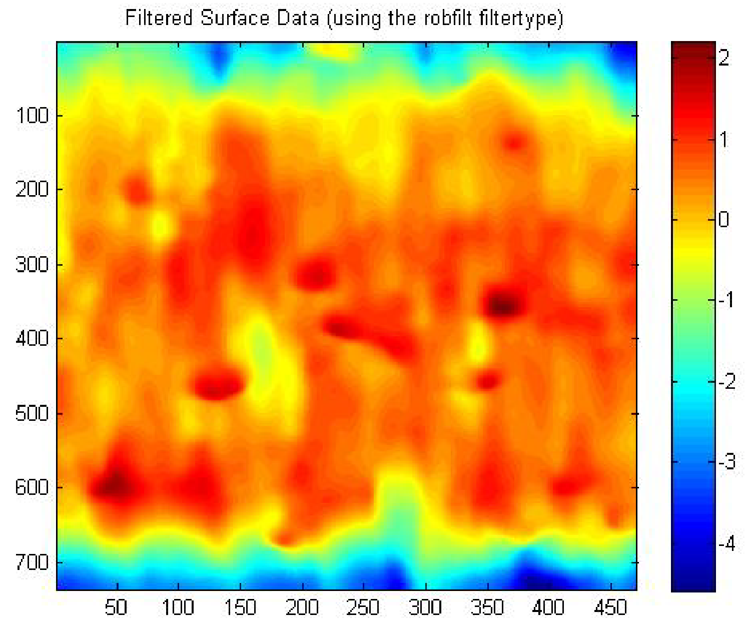


Figure 17: Filtered surface data using the SCOUT robust Gaussian filter

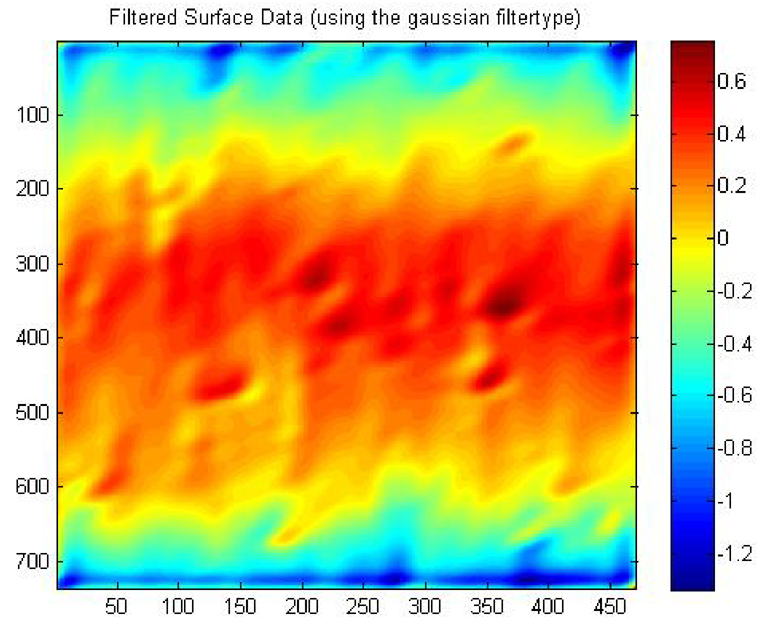


Figure 18: Filtered surface data using the non-SCOUT Gaussian filter

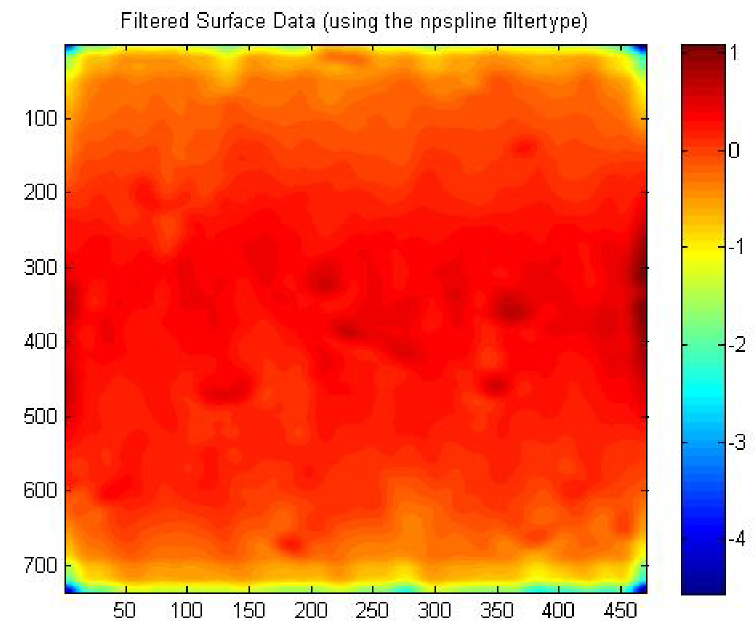


Figure 19: Filtered surface data using the non-periodic spline filter

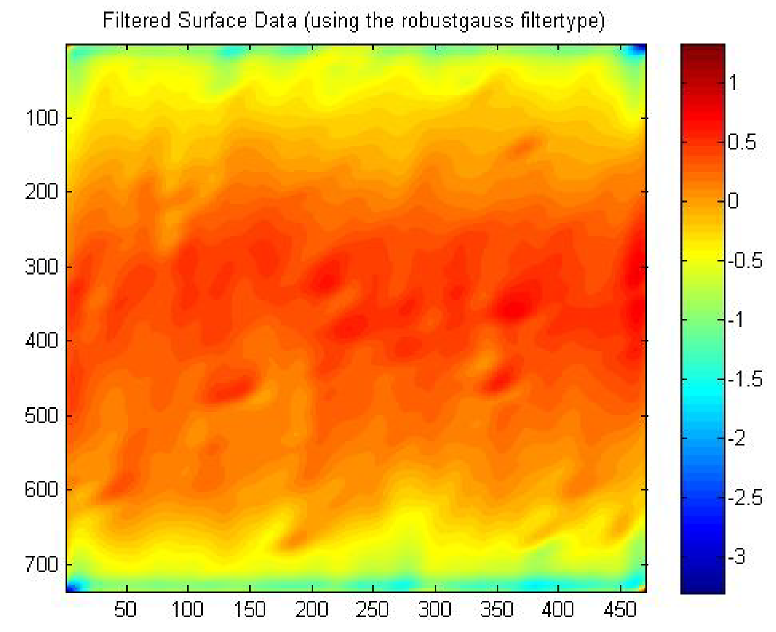


Figure 20: Filtered surface data using the non-SCOUT robust Gaussian filter

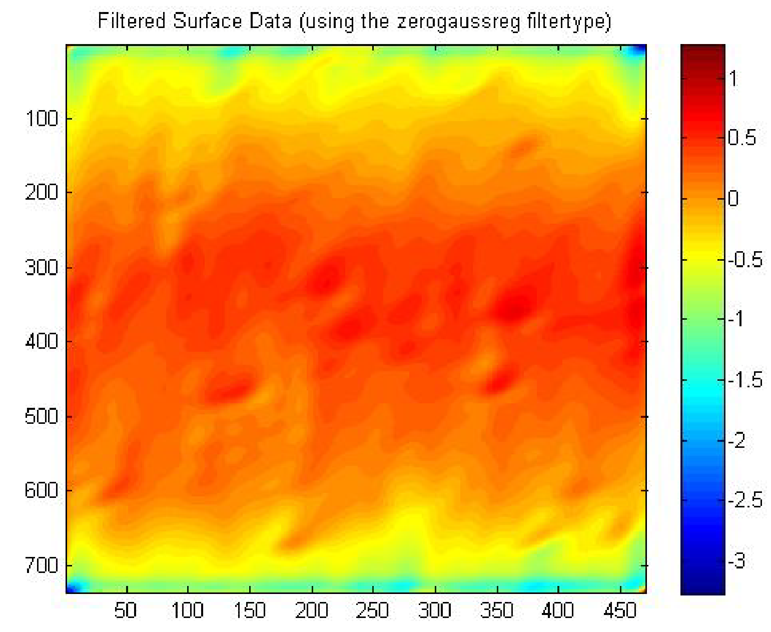


Figure 21: Filtered surface data using the 0-order Gaussian regression filter

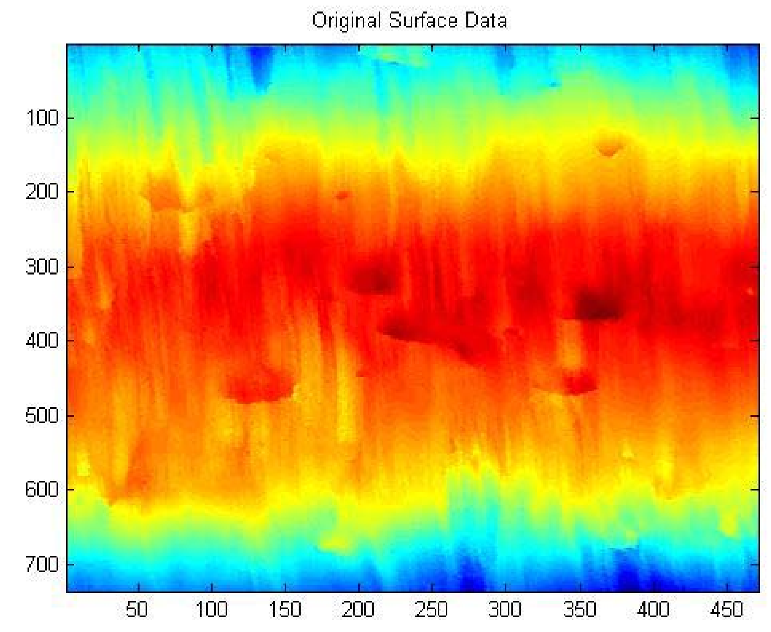


Figure 22: Original raw surface data

CHAPTER 7: Conclusion

Band-pass filtering with a variety of filters to filter both two and three dimensional surface data was explored. The software that collects and makes available these filtering methods to support a larger project, done by [Berglund et al. 2010], was developed. It uses band-pass filtering to find strong correlations [Berglund et al 2010]. The software described in this report is used to automate the filtering procedure. The workflow of the program, explaining how each filter was implemented, has been presented. The filters have been demonstrated by applying them to surface data provided by Berglund (2009). The two methods for forming a band-pass filter were shown to produce different curves when using either the Sa or the Sq roughness parameters. Explanations of the mathematical development of the filtering procedures as obtained from literature have been given. One filtering procedure, the second-order Gaussian regression filter, was developed as part of this project. Additionally, all filtering procedures from the literature were extended from 2D to 3D.

APPENDIX A: Description of important Inputs and Outputs of major functions in the Program

This chapter contains comments for the major functions called in the program (See chapter 3 for further explanations).

The grandforloop.m function:

```
 % Example of how to use the command: [IIb,filenames,List,
% directory_name] =
% grandforlooptest(filtertype,I,centwave, bndwths,sw,sw2,disgraphs);


% sw2 is used to specify how many files are analyzed. Otherwise, it
% will use all the WPI format data files present in the data folder.
% Example: let the center wavelengths be:[200 105] and the bandwidth be
% 50.
% When sw = 0 (filtering with frequency translation)
% grandforlooptest('gaussian',I,[200 105],
% 50,0);
% When sw = 1 (filtering with addition and subtraction method)
% grandforlooptest('gau_filt',I,[200 105],50,1);
% IIb = bandpass filtered data
```

The bndfilt.m function:

```
 % Example of how to use the command: [IIL,II2L,IIH,II2H,IIb] =
% bndpfilt(filtertype,I,centwave,Lambdamn,sw,disgraphs);


% To use translation by modulation, let use sw = 0
% To use the subtraction method, let sw = 1.
% II = leveled
% II2 = non-leveled
% L and H modifying IIL and IIH refers to low-pass and high-pass
% filtered data respectively when sw = 1.
% IIb = bandpass filtered data
```

The getwavelengthcos.m function:

```
 % Example of how to use the command: [II, II2] =
% getwavelength2cos(filtertype,I,centerfreq,Lambdam,...
%     frax,Lambda,L,npoints,spacing,disgraphs);


% Inputs:
% I is the data (in WPI format). You can either input it from the
% workspace
% or use the dialogue box option to select the data file.
% L = [Lx,Lxh] physical profile length (vertical and horizontal
% respectively).
% Lambdam = [Lambdamx, Lambdamxh] actual cutoff wavelength in the
% vertical
% and in the horizontal directions
% Lambda = [lambdacn,lambdacnh] integers used to make cutoff wavelength
% lambdacn = for cutoff wavelength in vertical direction (calculated as
% lambdacn*dx)
% lambdacnh = for cutoff wavelength in horizontal direction (calculated
% as lambdacnh*dxh)
% frax = 1 or 0. method used to get data. Decide whether to get
% fraxsurf file by typing 1 or
% load a raw data matrix by typing 0.
% This mfile currently applies the Gaussian filter to an image.
% dispgraphs = 1 to display graphs are displayed to show how the filter
% behaves on a segment of the data.
% centerfreq = [n1,n2] where n1 is the multiple of the vertical
% center frequency and n2 is the multiple of the horizontal
% center frequency.

% other variables used later:
% numpoints = actual number of points on the profile line
% m1 = size(S1,1) % vertical length of filter
% m1h = size(S1,2) % horizontal length of filter
% spacing = [dx,dxh]
% npoints = [numpoints, numpointsh] number of points in the vertical
% and horizontal directions
% I is the data
% II = leveled
% II2 = non-leveled
```

The secgaussreg.m function:

```
% Uses second order Gaussian regression filter
% Example of how to use the command: II =
% secgaussreg(I,n,nv,wc1,wc1v,dx,dxv,lambdacn,lambdacnv,Lambdam,Dirsw);

% Dirsw=1; %assuming no xy interaction (much faster and is default)
% Dirsw=2; %assuming xy interaction (much slower)

% For the other inputs see chapter 3 and the above description for getwavelengthcos.m
% function.
```

The robustgauss.m function:

```
% Uses the robust Gaussian filter
% Example of how to use the command: II =
% robustgauss(I,n,nh,wc1,wc1h,dx,dxh,lambdacn,lambdacnh,Lambdam,alpha
% ,CBt,relCB,iterationNo,disgraphs);
% alpha = coefficient used in determining CB determine
% CBt = threshold on elements of the CB matrix
% relCB = threshold on the absolute difference between the max of two
% consecutive CB matrices
% iterationNo = the maximum number of iterations

% CB is an estimate of the standard deviation using the median absolut
% deviation times alpha. CB is used to calculate weights applied to the % filter before
% the next iteration. For explanations about the meaning
% of CB, see: [Brinkmann et al. (2000) pp. 122-132]2,3

% (It might not make sense to use this filter at very high center
% frequencies since you 'introduce' sharp peaks and valleys). However,
% if a non zero center frequency is used, the iteration compares the
% subsequent results to the first filtered result.

% For the other inputs see the above description for getwavelengthcos.m
% function.
```

The npspline2.m function:

```
% Uses the nonperiodic spline filter
% Example of how to use the command:
% [x,II6] = npspline2(I,dx, Lambdam(1),wc1); %filter vertical direction
% [xh,II6] = npspline2(II6.',dxh, Lambdam(2),wc1h).'; %filter
```

[2] Brinkmann, S., Bodschwinna, H. and Lemke, H. W. 2000, 'Development of a robust Gaussian regression filter for three-dimensional surface analysis', Proceedings of the X International Colloquium on Surfaces, Chemnitz University of Technology. Chemnitz, pp. 122-132.

[3] Brinkmann, S., Bodschwinna, H. and Lemke, H. W.: Accessing roughness in three-dimensions using Gaussian regression filtering', International Journal of Machine Tools & Manufacture 41 (2001) 2153–2161

```
% horizontal direction
% II6 = II6.'; %restore the original orientation
%
%
% For the other inputs see the above description for getwavelengthcos.m
% function.
```

The zerogausreg.m function:

```
 % Uses zero order Gaussian regression filter
% Example of how to use the command:
% II = zerogaussreg(I,n,nv,wc1,wc1v,dx,dxv,lambdacn,lambdacnv,Lambdam);
%
% For description of the inputs see the above description for
% getwavelengthcos.m function
```

The filterimg.m function:

```
 % Is not intended to be called independently. (The results of each
% filter can seen by using getfiltereddat.m)
```

The ROB_FILT2.m function (Copy of ROB_FILT.m from SCOUT):

```
 %  [ FDATA,FS ] = rob_filt ( DATA, SPACING, CUTOFF )
%
%   Authors : S. Brinkmann, Hannover University (DLLs)
%             A. Porrino, Brunel University (Matlab)
%   e-mail  : brinkmann@imr.uni-hannover.de
%             alessandro.porrino@brunel.ac.uk
%   Version : 1999.10.18
%
%  Input Variables:
%
%   DATA      : Surface Data
%   SPACING   : X,Y spacing (m unit) in one of the following forms:
%              - [X_SPACING,Y_SPACING]
%              -  XY_SPACING  (if X_SPACING = Y_SPACING )
%   CUTOFF    : X,Y Direction Cut-off lengths (m units) in one of the following forms:
%              - [X_CUTOFF,Y_CUTOFF]
%              - XY_CUTOFF  (if X_CUTOFF = Y_CUTOFF )
%
%  Output Variables:
%
%   FDATA     : Filtered Data
%   FS        : Filtered Spacing
%
% NOTE: The ROB_FILT libraries were kindly supplied by Hannover University
%      to the AUTOSURF consortium
```

SCOUT read and write functions were renamed to Readsdf22.m and Writesdf22.m respectively in order to avoid conflict with surfrax read and write files that possess the same name.

The GAU_FILT2.m function (Copy of GAU_FILT.m from SCOUT):

```
 %  [ FDATA ] = Gau_filt ( DATA, SPACING, CUTOFF, FTYPE )
%
%   Authors : N. Amini, Volvo Technological Development
%           A. Porrino, Brunel University
%   e-mail  : VCC9.AMINI@MEMO.VOLVO.SE
%           alessandro.porrino@brunel.ac.uk
%   Version : 1999.11.10
%
% variables:
%
```

```
%   DATA      : Surface Data
%   SPACING   : X,Y spacing (m unit) in one of the following forms:
%              - [X_SPACING,Y_SPACING]
%              -  XY_SPACING  (if X_SPACING = Y_SPACING )
%   CUTOFF    : X,Y Direction Cut-off lentgths in one of the following forms:
%              - [X_CUTOFF,Y_CUTOFF]
%              -  XY_CUTOFF  (if X_CUTOFF = Y_CUTOFF )
%   FTYPE     : String to indicate lowpass or highpass, 'L'=Lowpass, 'H'=Highpass
%
%  Output Variables:
%
%   FDATA     : Filtered Data
```

(SCOUT read and write functions were renamed from Readsdf.m and Writesdf.m to Readsdf22.m and Writesdf22.m respectively in order to avoid conflict with the surfrax read and write files that possess the same name.)

APPENDIX B: Frequency Domain and Spatial Domain Filtering

B.1 Discrete Convolution

Frequency domain filtering of data can be represented as convolution in the spatial domain. The convolution operation becomes useful especially when the filter's spectrum changes as it changes position.

We are going to treat the x and y directions as independent, allowing the 3-D filters to be products of 2-D filters. In 2-D, the convolution output is represented as[4]:

$$Z(n) = \sum_{\tilde{k}=-\infty}^{\infty} z(\tilde{k})h(n-\tilde{k})$$

for a (spatially) invariant system. Here, $z(n)$ is the data and $h(n)$ is the filter (also called the system function). We are using the discretized spatial variable, n from $x = n\Delta x$. $Z(n)$ is the filtered data. When the system function varies with location then we only have the superposition summation[5]:

$$Z(n) = \sum_{\tilde{k}=-\infty}^{\infty} z(\tilde{k})h(n,\tilde{k})\,.$$

Discrete frequency domain filtering corresponds to multiplication of the discrete Fourier transforms (DFT) of the filter and the data sequences. We are using finite length filters and data. The Fourier transform (FT) of a finite length signal is a 2π periodic function defined as[6]:

$$zf(\omega) = \sum_{n=0}^{N-1} z(\tilde{k})e^{-i\omega\tilde{k}}$$

with $-\pi \leq \omega \leq \pi$ and the corresponding N-point DFT is the uniformly spaced samples of the FT:

[4] Page 76 of Digital Signal Processing Principles, Algorithms, and Applications, 3rd Edition by John G. Proakis and Dimitris G. Manolakis. Copyright 1996 by Prentice-Hall, Inc.
[5] Page 76 of Digital Signal Processing Principles, Algorithms, and Applications, 3rd Edition by John G. Proakis and Dimitris G. Manolakis. Copyright 1996 by Prentice-Hall, Inc.
[6] Pages 401 of Digital Signal Processing Principles, Algorithms, and Applications, 3rd Edition by John G. Proakis and Dimitris G. Manolakis. Copyright 1996 by Prentice-Hall, Inc.

$$zf(\omega_k) = \sum_{n=0}^{N-1} z(\tilde{k})e^{-i\omega_k\tilde{k}}$$

with $\omega_k = \frac{2\pi k}{N}$. The inverse DFT (IDFT) is:

$$\tilde{z}(\tilde{k}) = \frac{1}{N}\sum_{k=0}^{N-1} zf(\omega_k)e^{i\omega_k\tilde{k}}$$

So the IDFTs of samples of the Fourier transforms will yield N-periodic sequences of infinite length. $\tilde{z}(\tilde{k})$ is the periodic repetition of $z(\tilde{k})$. This repetition makes the linear convolution become akin to circular convolution[7]. That is, this process is similar to placing the spatial data on a circle while the filter slides around the circle during the convolution. So, to avoid the filter meeting old data points when it goes past the last data point, the data is zero-padded. Sufficient zero-padding then makes the frequency domain filtering result the same as what would have occurred if linear convolution was performed in the spatial domain. (In order to reduce computation time, the Fast Fourier transform method (FFT) is used to compute a signal's DFT and IDFT in the filtering codes instead of direct implementation of the DFT and the IDFT from definitions).
The equivalence between zero padded circular convolution and frequency domain multiplication is shown by the following calculation:

Let z and h have lengths N and M respectively with $z(n)$ and $h(n)$ equal to zero for negative n and for $n > N - 1$ and $n > M - 1$ respectively. Define:

$$z(n) * h(n) = \sum_{\tilde{k}=-\infty}^{\infty} z(\tilde{k})h(n-\tilde{k})$$

Truncate the sum because z and h have finite lengths:

$$z(n) * h(n) = \sum_{k=0}^{\infty} z(\tilde{k})h(n-\tilde{k}) = \sum_{k=0}^{n} z(\tilde{k})h(n-\tilde{k})$$

Then apply the effect of zero-padding by inserting zeroes into the sum:

[7] Pages 415 to 416 of Digital Signal Processing Principles, Algorithms, and Applications, 3rd Edition by John G. Proakis and Dimitris G. Manolakis. Copyright 1996 by Prentice-Hall, Inc.

$$z(n) \ast h(n) = \sum_{\tilde{k}=0}^{N-1} z(\tilde{k})h(n-\tilde{k}) = \sum_{\tilde{k}=0}^{N+M-2} z(\tilde{k})h(n-\tilde{k})$$

Now, take the $N+M-1$ point DFT of $z(n) \ast h(n)$:

$$Zf(\omega_k) = \sum_{n=0}^{N+M-2} e^{-i\omega_k n} Z(n) = \sum_{n=0}^{N+M-2} e^{-i\omega_k n} z(n) \ast h(n)$$

$$Zf(\omega_k) = \sum_{n=0}^{N+M-2} e^{-i\omega_k n} \left[\sum_{\tilde{k}=0}^{N+M-2} z(\tilde{k})h(n-\tilde{k}) \right],$$

Where

$$\omega_k = \frac{2\pi k}{N+M-1} \; and \; 0 \le k \le N+M-1$$

Interchange summations:

$$Zf(\omega_k) = \sum_{\tilde{k}=0}^{N+M-2} z(\tilde{k}) \left[\sum_{n=0}^{N+M-2} h(n-\tilde{k})\, e^{-i\omega_k n} \right]$$

Truncate the inner summation since $h(n)$ equal to zero for negative argument:

$$Zf(\omega_k) = \sum_{\tilde{k}=0}^{N+M-2} z(\tilde{k}) \left[\sum_{n=\tilde{k}}^{N+M-2} h(n-\tilde{k})\, e^{-i\omega_k n} \right]$$

Apply a change of variables using $l = n - \tilde{k}$:

$$Zf(\omega_k) = \sum_{\tilde{k}=0}^{N+M-2} z(\tilde{k}) \left[\sum_{l=0}^{N+M-2} h(l)\, e^{-i\omega_k (l+\tilde{k})} \right]$$

Factor out $e^{-i\omega_k \tilde{k}}$ from the inner sum:

$$\Rightarrow Zf(\omega_k) = \sum_{\tilde{k}=0}^{N+M-2} z(\tilde{k}) e^{-i\omega_k \tilde{k}} \left[\sum_{l=0}^{N+M-2} h(l)\, e^{-i\omega_k l} \right]$$

Separate the independent summations into a product of sums:

$$\Rightarrow Zf(\omega_k) = \left[\sum_{\tilde{k}=0}^{N+M-2} z(\tilde{k}) e^{-i\omega_k \tilde{k}} \right] \left[\sum_{l=0}^{N+M-2} h(l)\, e^{-i\omega_k l} \right]$$

$$\Rightarrow Zf(\omega_k) = zf(\omega_k) hf(\omega_k)\,,$$

Where $zf(\omega_k)$ and $hf(\omega_k)$ are defined as:

$$zf(\omega_k) = \left[\sum_{\tilde{k}=0}^{N+M-2} z(\tilde{k})e^{-i\omega_k\tilde{k}}\right]$$

$$hf(\omega_k) = \left[\sum_{l=0}^{N+M-2} h(l)\, e^{-i\omega_k l}\right]$$

This way the multiplication done in the frequency domain is related to zero-padded linear convolution in the spatial domain. Sometimes, as with the Gaussian filter, the filter is a symmetric function about the zero position. In this case, filter is 'circularly' padded with zeroes. The first half of the filter is sent to the end of the vector and the middle point along with the second half is kept at the beginning of the modified vector. That is, if $h(n) = \tilde{h}(n + n_o)$, $n_o \in \mathbb{N}$, $h(n) = 0$ for negative n and for $n > M - 1$ respectively. Then let $= l - n_o$, we get $h(n) = \tilde{h}(l)$. The $N + M - 1$ point inverse DFT of the sampled Fourier transform of $h(n)$ will have a discrete period of $N + M - 1$. The following calculations illustrate this zero-padding method:

Let

$$hf(\omega_k) = \left[\sum_{n=0}^{N+M-2} h(n)\, e^{-i\omega_k n}\right] = \left[\sum_{n=0}^{N+M-2} \tilde{h}(n + n_o)\, e^{-i\omega_k n}\right]$$

Apply a change of variables $l = n + n_o$:

$$hf(\omega_k) = \left[\sum_{l=-n_o}^{N+M-2-n_o} \tilde{h}(l)\, e^{-i\omega_k(l+n_o)}\right]$$

Break the sum into these two parts:

$$hf(\omega_k) = \left[\sum_{l=-n_o}^{-1} \tilde{h}(l)\, e^{-i\omega_k(l+n_o)} + \sum_{l=0}^{N+M-2-n_o} \tilde{h}(l)\, e^{-i\omega_k(l+n_o)}\right]$$

Factor out $e^{-i\omega_k n_o}$:

$$hf(\omega_k) = e^{-i\omega_k n_o}\left[\sum_{l=-n_o}^{-1} \tilde{h}(l)\, e^{-i\omega_k(l+(N+M-1))} + \sum_{l=0}^{N+M-2-n_o} \tilde{h}(l)\, e^{-i\omega_k l}\right],$$

with

$$e^{-i\omega_k(N+M-1)} = 1\ .$$

Let

$$\tilde{l} = (N+M-1) + l$$

Then,

$$hf(\omega_k) = e^{-i\omega_k n_o}\left[\sum_{\tilde{l}=(N+M-1)-n_o}^{(N+M-1)-1} \tilde{h}\left(\tilde{l}-(N+M-1)\right)e^{-i\omega_k\tilde{l}} + \sum_{l=0}^{N+M-2-n_o} \tilde{h}(l)\, e^{-i\omega_k l}\right]$$

Therefore we have that $\tilde{h}(l)$, which is found in

$$\sum_{l=0}^{N+M-2-n_o} \tilde{h}(l)\, e^{-i\omega_k l}$$

, is the zero-padded second half, and $\tilde{h}\left(\tilde{l}-(N+M-1)\right)$, which is found in

$$\sum_{\tilde{l}=(N+M-1)-n_o}^{(N+M-1)-1} \tilde{h}\left(\tilde{l}-(N+M-1)\right)e^{-i\omega_k\tilde{l}}$$

, is the first half now moved to the other end of the vector. That is, $\tilde{l}$ denotes the new positions for the points taken from the first half of the filter.

The artificial phase factor, $e^{-i\omega_k n_o}$, is thrown away and the zero position is redefined as the location of the filter point that was at n_o. The new filter is created as:

$$h(l) = \begin{cases} \tilde{h}(l) & for \quad 0 \le\ l\ \le \left\lfloor\frac{M}{2}\right\rfloor + 1 \\ 0 & for \quad \left\lfloor\frac{M}{2}\right\rfloor + 2 \le\ l\ \le (N+M-2) - n_o \\ \tilde{h}\big(l-(N+M-1)\big) & for \quad (N+M-1) - n_o \le\ l\ \le (N+M-2) \end{cases}$$

This filter is from the same periodic function from which $\tilde{h}$ came. We simply 'chose' a different interval.

Interpreting the Filtered Result

At this point we should notice that the purpose of filtering here is to modify the frequency or wavelength content of the data, and not to add extra artificial points to it.

So, we must throw away points resulting from the zero-padding of both the data. These artificial points do not correspond to when the summation involved only data points. So, we measure how far the convolving filter has to slide before it is completely within the domain of the data. This corresponds to sliding by $\left\lfloor\frac{M}{2}\right\rfloor$ places at the beginning and at the end. This way, we remove M points if the filter is of even length or we remove $M-1$ points if it is of odd length.

B.2 Using Spatial Modulation to implement frequency Shift

What is important during modulation is the relative shift of the modulated function to the other function involved in convolution. It is either that the filter's center frequency is moved to the desired location or the data's distribution falling within the region of interest is translated into the filter's pass-band and translated back after filtering. Modulating the filter becomes difficult when it varies as its spatial center is being shifted during convolution. So the modulation is transferred to the data in this case. We move from an expression modulating the filter to an expression modulating the data:

$$Z(n) = z(n) \ast [h(n)cos(\omega_0 n)] = \sum_{\tilde{k}=0}^{n} z(\tilde{k})h(n-\tilde{k})Re\left(e^{i\omega_0(n-\tilde{k})}\right)$$

$$\Rightarrow Z(n) = Re\left(\sum_{\tilde{k}=0}^{n} z(\tilde{k})h(n-\tilde{k})e^{i\omega_0(n-\tilde{k})}\right)$$

Factor out $e^{i\omega_0(n)}$ and rearrange factors inside the summand:

$$Z(n) = Re\left(e^{i\omega_0 n}\sum_{\tilde{k}=0}^{n}\left[z(\tilde{k})e^{-i\omega_0\tilde{k}}\right]h(n-\tilde{k})\right)$$

Also, note that one can also work backwards in the following way to obtain the same result. First, the last two steps above are repeated with more explanations to introduce expressions that will connect back to the first equation:

Define:

$$Z(n) = Re\left(\sum_{\tilde{k}=0}^{n} z(\tilde{k})h(n-\tilde{k})e^{i\omega_0(n-\tilde{k})}\right)$$

Factor out $e^{i\omega_0(n)}$:

$$Z(n) = Re\left(e^{i\omega_0 n}\sum_{\tilde{k}=0}^{n} z(\tilde{k})h(n-\tilde{k})e^{-i\omega_0\tilde{k}}\right)$$

Expand the complex exponentials:

$$Z(n) = Re\left(\left(cos(\omega_0 n) + isin(\omega_0 n)\right)\sum_{\tilde{k}=0}^{n} z(\tilde{k})h(n-\tilde{k})\left(cos(\omega_0\tilde{k}) - isin(\omega_0\tilde{k})\right)\right)$$

Distribute the summation:

$$Z(n) = Re\left(\sum_{\tilde{k}=0}^{n} z(\tilde{k})h(n-\tilde{k})\left[\left(cos(\omega_0 n)cos(\omega_0\tilde{k}) - isin(\omega_0 n)isin(\omega_0\tilde{k})\right) + \left(isin(\omega_0 n)cos(\omega_0\tilde{k}) - cos(\omega_0 n)isin(\omega_0\tilde{k})\right)\right]\right)$$

Simplify:

$$Z(n) = Re\left(\sum_{\tilde{k}=0}^{n} z(\tilde{k})h(n-\tilde{k})\left[\left(cos(\omega_0 n)cos(\omega_0\tilde{k}) + sin(\omega_0 n)sin(\omega_0\tilde{k})\right) + i\left(sin(\omega_0 n)cos(\omega_0\tilde{k}) - cos(\omega_0 n)sin(\omega_0\tilde{k})\right)\right]\right)$$

Now, take the real part:

$$Z(n) = \sum_{\tilde{k}=0}^{n} z(\tilde{k})h(n-\tilde{k})\left[\left(cos(\omega_0 n)cos(\omega_0\tilde{k}) + sin(\omega_0 n)sin(\omega_0\tilde{k})\right)\right]$$

Apply angle addition formula for cosine:

$$\Rightarrow Z(n) = \sum_{\tilde{k}=0}^{n} z(\tilde{k})h(n-\tilde{k})cos\left(\omega_0(n-\tilde{k})\right)$$

$$\Rightarrow Z(n) = \sum_{\tilde{k}=0}^{n} z(\tilde{k})h(n-\tilde{k})Re\left(e^{i\omega_0(n-\tilde{k})}\right) = z(n) * [h(n)cos(\omega_0 n)]$$

APPENDIX C: The Discrete Fourier Transform of the Gaussian

We have the 3-D Fourier transform[8,9] of the Gaussian defined as:

$$Sf(f_x, f_y) = \iint_{\mathbb{R}^2} S(x, y)\mathrm{e}^{-i(2\pi f_x x+2\pi f_y y)}dxdy,$$

with

$$S(x, y) = \frac{1}{\alpha\lambda_{cx}\alpha\lambda_{cy}}\exp\left[-\pi\left(\frac{x}{\alpha\lambda_{cx}}\right)^2 - \pi\left(\frac{y}{\alpha\lambda_{cy}}\right)^2\right].$$

The following calculations below show how to obtain the formula for the transform:

$$Sf(f_x, f_y) = \iint\limits_{\mathbb{R}^2} \frac{1}{\alpha\lambda_{cx}\alpha\lambda_{cy}}\mathrm{e}^{\left[-\pi\left(\frac{x}{\alpha\lambda_{cx}}\right)^2-\pi\left(\frac{y}{\alpha\lambda_{cy}}\right)^2\right]}\mathrm{e}^{-i(2\pi f_x x+2\pi f_y y)}dxdy$$

$$Sf(f_x, f_y) = \left[\int\limits_{\mathbb{R}^1} \frac{1}{\alpha\lambda_{cx}}\mathrm{e}^{\left[-\pi\left(\frac{x}{\alpha\lambda_{cx}}\right)^2\right]}\mathrm{e}^{-i(2\pi f_x x)}dx\right]\left[\int\limits_{\mathbb{R}^1} \frac{1}{\alpha\lambda_{cy}}\mathrm{e}^{\left[-\pi\left(\frac{y}{\alpha\lambda_{cy}}\right)^2\right]}\mathrm{e}^{-i(2\pi f_y y)}dy\right]$$

Complete the square:

$$Sf(f_x, f_y) = \left[\int\limits_{\mathbb{R}^1} \frac{1}{\alpha\lambda_{cx}}\mathrm{e}^{\left[-\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cx}}\right)^2\left[\left(x+\frac{i(2\pi f_x)(\alpha\lambda_{cx})^2}{2\pi}\right)^2+\left(\frac{(2\pi f_x)(\alpha\lambda_{cx})^2}{2\pi}\right)^2\right]\right]}dx\right]$$

$$\times\left[\int\limits_{\mathbb{R}^1} \frac{1}{\alpha\lambda_{cy}}\mathrm{e}^{\left[-\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cy}}\right)^2\left[\left(y+\frac{i(2\pi f_y)(\alpha\lambda_{cy})^2}{2\pi}\right)^2+\left(\frac{(2\pi f_y)(\alpha\lambda_{cy})^2}{2\pi}\right)^2\right]\right]}dy\right]$$

Factor out the constants:

$$\Rightarrow Sf(f_x, f_y) = \left[\mathrm{e}^{-\left[\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cx}}\right)^2\left(\frac{(2\pi f_x)(\alpha\lambda_{cx})^2}{2\pi}\right)^2+\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cy}}\right)^2\left(\frac{(2\pi f_y)(\alpha\lambda_{cy})^2}{2\pi}\right)^2\right]}\right]\left[\int\limits_{\mathbb{R}^1} \frac{1}{\alpha\lambda_{cx}}\mathrm{e}^{\left[-\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cx}}\right)^2\left(x+\frac{i(2\pi f_x)(\alpha\lambda_{cx})^2}{2\pi}\right)^2\right]}dx\right]$$

$$\times\left[\int\limits_{\mathbb{R}^1} \frac{1}{\alpha\lambda_{cy}}\mathrm{e}^{\left[-\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cy}}\right)^2\left(y+\frac{i(2\pi f_y)(\alpha\lambda_{cy})^2}{2\pi}\right)^2\right]}dy\right]$$

[8] Chapter 5, page 33 of Computational Surface and Roundness Metrology by Bala Muralikrishnan and Jay Raja. Copyright 2009, Springer-Verlag London Limited
[9] Chapter 8, page 56 of Computational Surface and Roundness Metrology by Bala Muralikrishnan and Jay Raja. Copyright 2009, Springer-Verlag London Limited

Now, by Gauss we have the following integration:

$$\int_{\mathbb{R}^1} \mathrm{e}^{-x^2} dx = \sqrt{\int_{\mathbb{R}^1} \mathrm{e}^{-x^2} dx \int_{\mathbb{R}^1} \mathrm{e}^{-y^2} dy} = \sqrt{\iint_{\mathbb{R}^2} \mathrm{e}^{-(x^2+y^2)} dxdy}$$

$$= \begin{matrix} lim \\ R \to +\infty \end{matrix} \sqrt{\int_0^{2\pi}\int_0^R r\mathrm{e}^{-r^2} dr\, d\theta} = \begin{matrix} lim \\ R \to +\infty \end{matrix} \sqrt{2\pi \int_0^R r\mathrm{e}^{-r^2} dr}$$

$$= \begin{matrix} lim \\ R \to +\infty \end{matrix} \sqrt{\pi \int_0^{R^2} \mathrm{e}^{-\mathrm{u}} du} = \begin{matrix} lim \\ R \to +\infty \end{matrix} \sqrt{\pi\left[\mathrm{e}^{-\mathrm{u}}\big|_0^{R^2}\right]} = \sqrt{\pi}\,,$$

After substitutions and some simplifications, that:

$$\begin{cases} \displaystyle\int_{\mathbb{R}^1} \mathrm{e}^{\left[-\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cx}}\right)^2\left(x+\frac{i(2\pi f_x)\alpha\lambda_{cx}}{2\pi}\right)^2\right]} dx = \left(\frac{\alpha\lambda_{cx}}{\sqrt{\pi}}\right)\int_{\mathbb{R}^1} \mathrm{e}^{-u^2} du = \left(\frac{\alpha\lambda_{cx}}{\sqrt{\pi}}\right)\sqrt{\pi} \\ \displaystyle\int_{\mathbb{R}^1} \mathrm{e}^{\left[-\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cy}}\right)^2\left(y+\frac{i(2\pi f_y)\alpha\lambda_{cy}}{2\pi}\right)^2\right]} dy = \left(\frac{\alpha\lambda_{cy}}{\sqrt{\pi}}\right)\int_{\mathbb{R}^1} \mathrm{e}^{-u^2} du = \left(\frac{\alpha\lambda_{cy}}{\sqrt{\pi}}\right)\sqrt{\pi} \end{cases}$$

Therefore,

$$Sf(f_x, f_y) = \left[\left[\left(\frac{1}{\alpha\lambda_{cx}}\right)\left(\frac{1}{\alpha\lambda_{cy}}\right)\right]\mathrm{e}^{-\left[\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cx}}\right)^2\left(\frac{(2\pi f_x)(\alpha\lambda_{cx})^2}{2\pi}\right)^2+\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cy}}\right)^2\left(\frac{(2\pi f_y)(\alpha\lambda_{cy})^2}{2\pi}\right)^{2\,2}\right]}\right]\left[\left(\frac{\alpha\lambda_{cx}}{\sqrt{\pi}}\right)\left(\frac{\alpha\lambda_{cy}}{\sqrt{\pi}}\right)\right]\pi$$

$$\Rightarrow Sf(f_x, f_y) = \mathrm{e}^{\left[-\left(\left(\frac{(\alpha\lambda_{cx})^2}{\pi}\right)\left(\frac{2\pi f_x}{2}\right)^2+\left(\frac{(\alpha\lambda_{cy})^2}{\pi}\right)\left(\frac{2\pi f_y}{2}\right)^2\right)\right]} = \mathrm{e}^{\left[-\left(\frac{1}{\pi}\right)\left(\left(\frac{\alpha\lambda_{cx}2\pi f_x}{2}\right)^2+\left(\frac{\alpha\lambda_{cy}2\pi f_y}{2}\right)^2\right)\right]}$$

Apply the transformations $f_x = \frac{1}{\lambda_x}$ and $f_y = \frac{1}{\lambda_y}$ to also get:

$$Sf(\lambda_x, \lambda_y) = \mathrm{e}^{\left[-\left(\frac{1}{\pi}\right)\left(\left(\frac{2\pi\alpha\lambda_{cx}}{2\lambda_x}\right)^2+\left(\frac{2\pi\alpha\lambda_{cy}}{2\lambda_y}\right)^2\right)\right]} = \mathrm{e}^{\left[-(\pi)\left(\left(\frac{\alpha\lambda_{cx}}{\lambda_x}\right)^2+\left(\frac{\alpha\lambda_{cy}}{\lambda_y}\right)^2\right)\right]}.$$

To obtain the discrete Fourier transform, first, we replace both $f_x\Delta x$ and $f_y\Delta$y with ω_{k_x} and ω_{k_y}respectively in the integral,

$$Sf\left(f_x, f_y\right) = \left[e^{-\left[\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cx}}\right)^2\left(\frac{(2\pi f_x)(\alpha\lambda_{cx})^2}{2\pi}\right)^2+\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cy}}\right)^2\left(\frac{(2\pi f_y)\left(\alpha\lambda_{cy}\right)^2}{2\pi}\right)^2\right]} \right] \left[\int_{\mathbb{R}^1} \frac{1}{\alpha\lambda_{cx}} e^{\left[-\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cx}}\right)^2\left(x+\frac{i(2\pi f_x)(\alpha\lambda_{cx})^2}{2\pi}\right)^2\right]} dx \right]$$

$$\times \left[\int_{\mathbb{R}^1} \frac{1}{\alpha\lambda_{cy}} e^{\left[-\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cy}}\right)^2\left(y+\frac{i(2\pi f_y)\left(\alpha\lambda_{cy}\right)^2}{2\pi}\right)^2\right]} dy \right]$$

To get the discrete version, this integral needs to be truncated and discretized. Start by replacing x with $n_x\Delta x$ and y with $n_y\Delta y$, and making the summand, $S\left(n_x\Delta x, n_y\Delta y\right)$, nonzero for $0 \leq n_x \leq N_x$ and $0 \leq n_y \leq N_y$ and zero outside the region. Then restrict the continuous variables, ω_{k_x} and ω_{k_y}, to the sample points at $\omega_{k_x} = \frac{2\pi k_x}{N_x+M_x-1}$ and $\omega_{k_y} = \frac{i2\pi k_y}{N_y+M_y-1}$ where $0 \leq k_x \leq N_x + M_x - 2$ and $0 \leq k_y \leq N_y + M_y - 2$. Call the new function $\tilde{S}f\left(\omega_{k_x}, \omega_{k_y}\right)$.

Algebraically these are the steps:

$$Sf\left(\omega_{k_x}, \omega_{k_y}\right) = e^{-\left[\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cx}}\right)^2\left(\frac{\omega_{k_x}}{\Delta x}(\alpha\lambda_{cx})^2\right)^2+\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cy}}\right)^2\left(\frac{\omega_{k_y}}{\Delta \mathrm{y}}\left(\alpha\lambda_{cy}\right)^2\right)^2\right]} \left[\int_{-\infty}^{+\infty} \frac{1}{\alpha\lambda_{cx}} e^{\left[-\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cx}}\right)^2\left(x+i\frac{\omega_{k_x}}{\Delta x}(\alpha\lambda_{cx})^2\right)^2\right]} dx \right]$$

$$\times \left[\int_{-\infty}^{+\infty} \frac{1}{\alpha\lambda_{cy}} e^{\left[-\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cy}}\right)^2\left(y+i\frac{\omega_{k_y}}{\Delta \mathrm{y}}\left(\alpha\lambda_{cy}\right)^2\right)^2\right]} dy \right]$$

by substitution.

The integrals have been shown to be equal to one, leaving us with:

$$Sf\left(\omega_{k_x}, \omega_{k_y}\right) = e^{-\left[\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cx}}\right)^2\left(\frac{\omega_{k_x}}{\Delta x}(\alpha\lambda_{cx})^2\right)^2+\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cy}}\right)^2\left(\frac{\omega_{k_y}}{\Delta \mathrm{y}}\left(\alpha\lambda_{cy}\right)^2\right)^2\right]} = e^{-\left[\pi(\alpha\lambda_{cx})^2\left(\frac{\omega_{k_x}}{\Delta x}\right)^2+\pi\left(\alpha\lambda_{cy}\right)^2\left(\frac{\omega_{k_y}}{\Delta \mathrm{y}}\right)^2\right]}$$

Therefore, the discrete Fourier transform of the samples is:

$$\tilde{S}f\left(\omega_{k_x}, \omega_{k_y}\right) = e^{-\left[\pi(\alpha\lambda_{cx})^2\left(\frac{\omega_{k_x}}{\Delta x}\right)^2+\pi\left(\alpha\lambda_{cy}\right)^2\left(\frac{\omega_{k_y}}{\Delta \mathrm{y}}\right)^2\right]} \left[\sum_{n_x=0}^{N_x+M_x-2} \frac{1}{\alpha\lambda_{cx}} e^{\left[-\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cx}}\right)^2\left(n_x\Delta x+i\frac{\omega_{k_x}}{\Delta x}(\alpha\lambda_{cx})^2\right)^2\right]} \Delta x \right]$$

$$\times \left[\sum_{n_y=0}^{N_y+M_y-2} \frac{1}{\alpha\lambda_{cy}} e^{\left[-\left(\frac{\sqrt{\pi}}{\alpha\lambda_{cy}}\right)^2\left(n_y\Delta y+i\frac{\omega_{k_y}}{\Delta \mathrm{y}}\left(\alpha\lambda_{cy}\right)^2\right)^2\right]} \Delta y \right]$$

The expression, $\frac{\tilde{S}f\left(\omega_{k_x},\omega_{k_y}\right)}{\left[\substack{max\\k_x}\left[\tilde{S}f\left(\omega_{k_x},\omega_{k_y}\right)\right]\right]\left[\substack{max\\k_y}\left[\tilde{S}f\left(\omega_{k_x},\omega_{k_y}\right)\right]\right]}$, is the frequency domain representation of the actual filter used in the Gaussian filtering code in this project. This type of substitution is generally valid either when the error,

$$\left|Sf\left(\omega_{k_x},\omega_{k_y}\right)-\tilde{S}f\left(\omega_{k_x},\omega_{k_y}\right)\right|,$$

is sufficiently small, or when the function, $S(x,y)$, is band-limited in frequency and with the discrete data samples being collected at a sampling rate that satisfies the Nyquist criterion[10,11].

[10] Chapter 4, page 26 of Computational Surface and Roundness Metrology by Bala Muralikrishnan and Jay Raja. Copyright 2009, Springer-Verlag London Limited

[11] Pages 269 to 276 and 396 to 398 of Digital Signal Processing Principles, Algorithms, and Applications, 3rd Edition by John G. Proakis and Dimitris G. Manolakis. Copyright 1996 by Prentice-Hall, Inc.

References

Berglund J., Sandvick, Sweden. Personal communication (2009)

Berglund J., Agunwamba C., Powers B., Brown C. A., Rosén B.-G. 2010, 'On discovering relevant scales in surface roughness measurement – An evaluation of a band-pass method', SCANNING: The Journal of Scanning Microscopies.

Berglund J, Rosén B.-G.: A Method Development for Correlation of Surface Finish Appearance of Die Surfaces and Roughness Measurement Data. Tribology Letters, vol 36 (2009).

Brinkmann S., Bodschwinna H. and Lemke H. W. 2000, 'Development of a robust Gaussian regression filter for three-dimensional surface analysis', Proceedings of the X International Colloquium on Surfaces, Chemnitz University of Technology. Chemnitz, pp. 122-132.

Brinkmann S., Bodschwinna H. and Lemke H. W.: Accessing roughness in three-dimensions using Gaussian regression filtering', International Journal of Machine Tools & Manufacture 41 (2001) 2153–2161

Muralikrishnan and Raja, 2009: Computational Surface and Roundness Metrology. Copyright 2009, Springer-Verlag London Limited

Proakis J. G. and Manolakis D. G.: Digital Signal Processing Principles, Algorithms, and Applications, 3rd Edition. Copyright 1996 by Prentice-Hall, Inc.