Git Project: Image Analysis and FEM

G. Bowker, S. Lister, J. Collins, S. Rehman, F. Livera

0.1 Introduction

Simple finite elements in Python (SfePy) uses finite element methods to solve coupled partial differential equation (PDE) in systems up to three dimensions. SfePy is a powerful software that allows complex physical problems to be coded quickly and easily. It has been used successfully in a variety of disciplines, ranging from biomechanical modelling [1] to the computational analysis of acoustic transmission coefficients [2].

In this report, the input file to the SfePy software is a microstructural image which must first be 'cleaned' through segmentation, mesh generation and noise reduction. It can then be imported into the software as a mesh file, where boundary and initial conditions are applied. Fields are then created which can be used to define variables which may be 'unknown field', 'test field' or 'parameter field' [3] and the material properties are defined.

0.2 Aims and Objectives

0.3 Image Analysis

A key requirement for this project is to build a python script which will allow JPG images to be read as an array, which can then be manipulated and processed with python packages such as Sci-Kit Image (skimage) [4]. Several microstructure images have been used in this project, from a variety of sources including previous research data, scientific literature and the provided database, all with full and sufficient permissions. Images were then pre-processed accordingly through various techniques such as segmentation, removal of noise and measurement, before being imported into FE simulations to be used to create a mesh. This section will further detail how this process was undertaken and broken down through the use of test cases to ensure that the python script and skimage were performing accurately and as desired.

The first test case that was carried out was processing an image of a ferritic microstructure taken from the Materials Science and Engineering: An Introduction by Callister [5]. Firstly, an understanding of how to navigate and operate the skimage package was developed, as the software was new to the authors, through testing different algorithms and operations independently on the test image. This included manipulating image colour and contrast, converting the image to grayscale and cropping the image to remove the scale bar. Other features that were tested at this stage was the use of filters such as the Sobel filter, which is commonly used in image processing to emphasise the edges in an image, aiding edge detection algorithms. A watershed transform was another image processing technique that was trialled at this stage, where the image is treated as a topographical map, with brightness determining elevation of each point and then identifying the lines which run across the peaks to segment the map. Finally, these operations were combined to process the image as effectively as possible, before the measuring tools were used to estimate average grain size.

Secondly, in test case 2, the effectiveness of the code developed in the first test case was applied to an image of a thermally etched steel microstructure showing prior austenite grains, taken from previous research by Joshua Collins and authorised by their academic supervisor, Dr Ed Pickering. This microstructure image, as seen below, had previously been the subject of average grain size analysis via the linear intercept method. The average grain size would now be calculated the image processing python script and the two results compared to assess the accuracy of the approach.

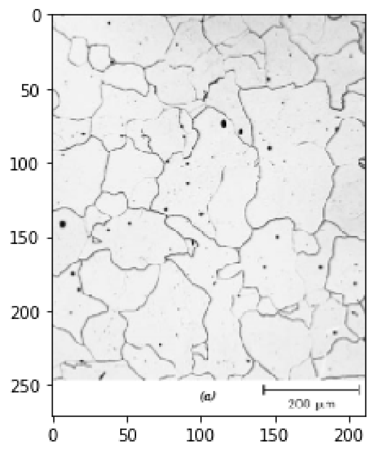

Figure 1: A figure showing the initial microstructure image taken from [5].

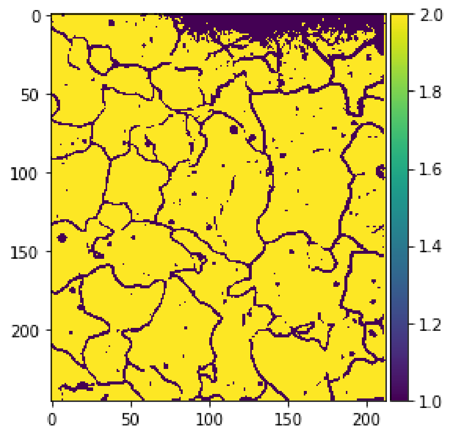

Figure 2: A figure showing the final image after segmentation and processing.

The image was processed using many of the techniques tested out previously, with the image first being read before being cropped to remove unfocused areas of the micrograph. The image was then segmented via the application of the Sobel filter, application of markers and then a watershed transform to fill in regions of the elevation map before the grains could finally be segmented and labelled individually. Finally, the image is converted back to gray scale, to allow each grain to be assigned a phase number associated with material properties. This will allow the "image" data to be exported as a numpy array and saved as a Comma Seperated Value (CSV) file for easy importation into the Gmsh meshing software later.

As mentioned previously the successfulness of the second test case was assessed by comparing the calculated average grain size from the python algorithm in comparison to that previously measured using the linear intercept method. To do this, the area of each segmented grain size was measured and then the average diameter of each grain can be estimated, by assuming the grains are perfect circles. Although this is a large assumption which is clearly not true, it yielded excellent results as an average grain size of 23.5 microns was estimated, with the previously measured average of 24.9 microns. Such a small difference in average grain size clearly demonstrates the viability of this approach and provides reassurance that the algorithm should be applied to other images to further test the virtue of it.

Test case 3 was then commenced, seeking to apply the algorithm to another image, this time a brightfield image of an $\alpha + \beta$ microstructure of the Ti-6Al-4V alloy, provided by Dr Pratheek Shantraj and shown below. The same procedure as in the previous test case was applied to the image and the image was successfully converted into a segmented, binary image and then exported into the Gmsh software in order to create a square based mesh for subsequent FEM analysis.
Test case 4 then took the same image from the previous test case and processed and segmented a single grain for meshing and FEM analysis. This analysis was done with the use of the additional skimage sub-packages; threshold filters and clean border segmentation to remove the surrounding grains from the image. The binary, single grain image seen below demonstrates that this approach is valid for application to and isolation of individual grains, as well as a multitude of grains as demonstrated in test case 3. The successful application of the 4 test cases allowed the authors to

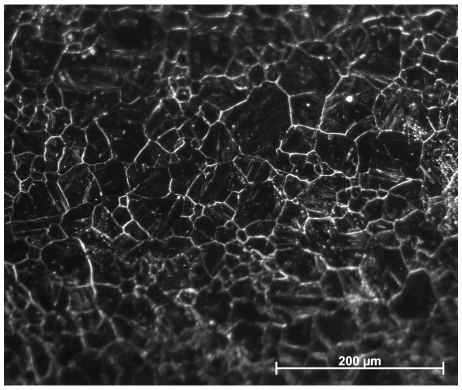

Figure 3: A figure showing the initial SA508 thermally etched steel microstructure image.

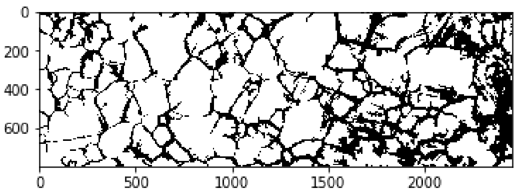

Figure 4: A figure showing the final steel microstructure image after segmentation and processing.

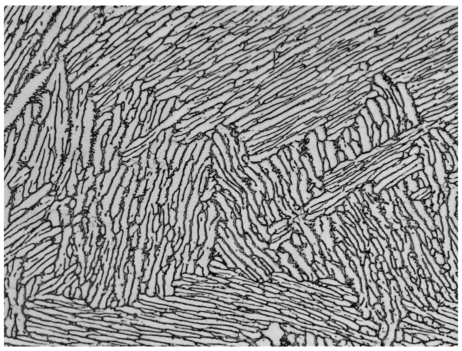

Figure 5: A figure showing the initial Ti-64 microstructure image.

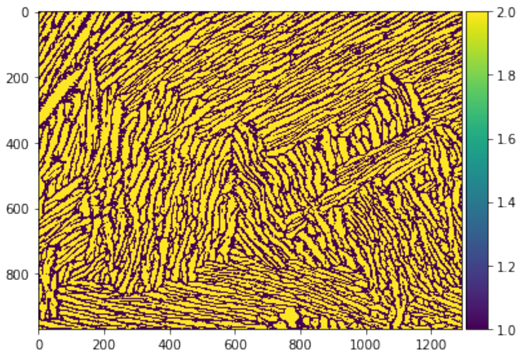

Figure 6: A figure showing the final Ti-64 microstructure image after processing.

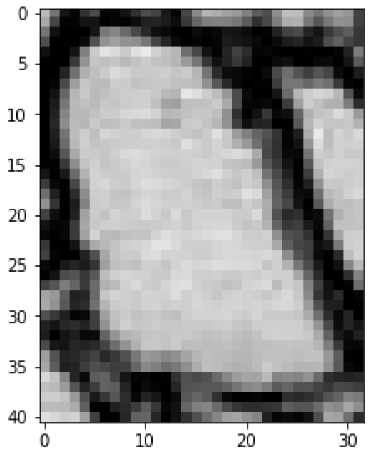
Figure 7: A figure showing the initial image of a single grain of the Ti-64 microstructure.

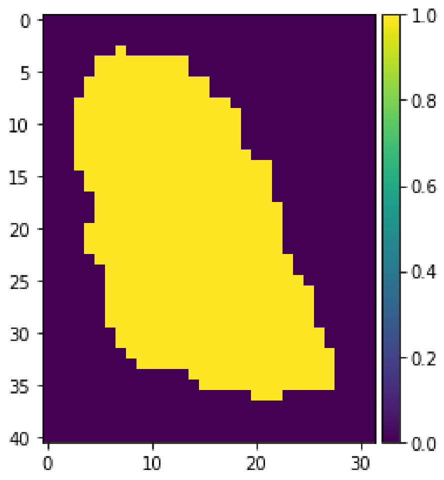
Figure 8: A figure showing the final image of a single grain of Ti-64 after segmentation and processing.

gain a high level of confidence in the algorithm and therefore move on to the next stage of the project, meshing and FE.

Finally, the rigorously tested image processing code was compiled into a single function, to allow it to be easily applied to any image. The function termed 'process image' takes 5 input arguments and returns a binary image that can be inputted straight into the OOF2 program for meshing and simulations. OOF2 was used for the final function as opposed to Gmsh as during testing it became clear that it provided a more robust method for this particular application, which will be discussed further in the next section. The input arguments are as follows; file $\bar{t}$he input image, e = exposure value, u/v are maximum and minimum threshold parameters respectively, and output = file name for the output file. The function therefore reads the input image, converts it to grayscale (if it is currently a rgb file) and then adjusts the exposure as per the input value of e. The sobel filter is then applied using the values of u and v as the max/min markers to isolate grains and separate them from grain boundaries. Following this the watershed transform is applied to create a binary image which is then saved in the specified output file. The function permits the user to quickly change the input arguments and see the results easily, allowing comparisons to be made and the best possible output to be produced for the next step, meshing.

0.4 Finite Element Modelling

Test!

0.5 Results

Test!

Bibliography

[1] Robert Cimrman and Eduard Rohan. Two-scale modeling of tissue perfusion problem using homogenization of dual porous media. *International Journal for Multiscale Computational Engineering - INT J MULTISCALE COMPUT ENG*, 8:81–102, 01 2010.

[2] E. Rohan and V. Lukeš. Homogenization of the acoustic transmission through a perforated layer. *Journal of Computational and Applied Mathematics*, 234(6):1876 – 1885, 2010. Eighth International Conference on Mathematical and Numerical Aspects of Waves (Waves 2007).

[3] Robert Cimrman, Vladimír Lukeš, and Eduard Rohan. Multiscale finite element calculations in python using sfepy. *Advances in Computational Mathematics*, 45(4):1897–1921, 2019.

[4] Stéfan van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and The scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 2014.

[5] William D Callister Jr. Materials Science and Engineering - An Introduction (5th ed.). *Anti-Corrosion Methods and Materials*, 47(1), 2000.