Image Processing – Methodology Log

Joshua Collins

June 4, 2020

1 Image Processing Logs

See below the steps for image processing:

1.1 Tested 'Skimage' - python image processing package (link: https://scikit-image.org/) (17/04/2020)

- See Image-Processing/testcase_IP_V1.0.ipynb
- Tested algorithms on an image of a ferritic microstructure taken from Materials Science and Engineering: An Introduction, W. D. Callister, 6th Edition.
- Testing included:
  - Manipulating image colour and contrast
  - Using filters (e.g. sobel filter) and transforms (e.g. watershed)
  - Combining manipulation, filters and transforms to segment the image into separate labelled grains
  - Use skimage measuring tools to measure properties of segmented grains (i.e. measured areas and used this to estimate and average grain size)
- Testcase 1 primarily used to test skimage algorithms and understand the image manipulation interface
- Re-named to 'testcase_IP_V1.0' on 23/04/2020

1.2 Tested 'Skimage' processing and subsequent measuring tools on a previously analysed microstructure (21/04/2020)

- See Image-Processing/testcase_IP_V2.0.ipynb
- Tested algorithms on a thermally etched steel micrograph showing prior austenite grains (image was taken by Josh and approved by Dr Ed Pickering [lead research supervisor])

- Linear intercept method previously used to determine grain size of 24.9 microns
- Test case was split into various steps (see notebook):
  - Importing Packages
    - Imported various skimage sub-packages as well as os, numpy, and matplotlib.pyplot
  - Reading the Image
    - Image was read, type and shape was determined before manipulation
  - Manipulating the Image
    - Image cropped to region of highest focus
  - Segmenting the Image
    - Used a combination of a Sobel filter and a watershed transform to segment the image and label each grain
  - Measuring Grain Size
    - Skimages' 'measure' was used to determine average grain size (assuming all grains were perfect circles)
    - Pixels were ceonverted to microns using the scale bar on the image
    - Average grain size found
- Measured grain size = 32.7 microns
  - Discrepancy due to technical assumptions (i.e. grains = circles) and overestimation of grain boundary width during segmentation

1.3 Edited testcase 2 (23/04/2020)

- See Image-Processing/testcase_IP_V2.0.ipynb history log
- Edited section 3. Manipulating the Image
  - Adjusted image exposure to help differentiate between grain boundary and grain interior
- Re-ran rest of notebook
- Change resulted in a new measured grain size = 23.5 microns
  - approx. 1.4 microns difference between experimentally determined value

1.4 Edited testcase 2 (28/06/2020)

- See Image-Processing/testcase_IP_V2.0.ipynb history log
- To produce a binary image of the segmented grains
- Determined that the best place to obtain this image is before labelling
  - When the image is named 'segmentation' (after a watershed transform is applied)
- Binary image required for meshing using Gmsh software
- Conclusions: happy with image processing notebook to successfully process images for meshing and subsequent FEM
  - Possible automation of image processing could be coded in python

1.5 Used image processing notebook to produce binary image of Ti64 microstructure and then meshed binary image using Gmesh software (01/06/2020)

- See Image-Processing/testcase_IP_V3.0.ipynb, Ti64_binary_V1.0.png & Ti64_V1.0.mesh file
- Ti64 image was obtained from Pratheek with his consent
- Uses skimage packages previously mentioned to produce a binary image of segmented grains and saves to repository (image called: Ti64_binary_V1.0.png)
- Exported image into Gmsh software (https://gmsh.info/) to create a square-based mesh of image for future FEM analysis (mesh called: Ti64_V1.0.mesh)

1.6 Used image processing notebook to produce binary image of a single grain in a Ti64 microstructure and then meshed binary image using Gmesh software (01/06/2020)

- See Image-Processing/testcase_IP_V4.0.ipynb, Ti64_binary_V2.0.png & Ti64_V2.0.mesh file
- Ti64 image was obtained from Pratheek with his consent
- Additional skimage sub-packages used to isolate a single grain
  - Used 'threshold' filters and 'clean_border' segmentation to remove surrounding grains from the image
- Image saved to repository as binary image (named: Ti64_binary_V2.0.png)
- Used Gmsh to create square-bsaed mesh of image for FEM (mesh called: Ti64_V2.0.mesh)

- Conclusions: happy with image processing testcases (both image manipulation on python notebook and meshing in Gmsh). End testing.

2 Current Work

2.1 Producing binary images for FEM

2.2 Automate python notebook