Analysis of Hydride Structures in Micrographs of Zirconium Samples using Image Processing in Python

Group 1: S. Khan, A. Hanson, J. Larkin

May 25, 2021

This report details progress made during the Research Software & Engineering in Python module, in which image processing techniques are utilised to automatically analyse micrograph images of zirconium hydride (H_4Zr) structures in samples of metallic zirconium. Explanation of the method, including relevant images and discussion of code elements are included.

1 Introduction

During the service lifetime of zirconium fuel cladding within a light-water reactor core, several phenomena manifest as a result of both direct neutron irradiation and chemical reactivity with the core coolant. One of the more prominent of these is the formation of zirconium hydride within the bulk material, resulting from the radiolysis of core coolant and subsequent migration of hydrogen through the lattice, which then reacts with the metal and accumulates at grain boundaries. As zirconium hydride is typically more brittle than the lattice itself, this serves to compromise the local integrity of the lattice.

Zirconium, as a HCP lattice, exhibits a specific crystal *texture* which is determined during component fabrication, the result of which is the alignment of the $< 0002 >$ basal planes tangentially around the cladding tube. As hydrides typically accumulate parallel to the basal plane, this is advantageous as it prevents the development of cracks radially through the structure which can result in through-wall cracking, compromising the barrier between nuclear fuel and core coolant. Larger scale cracks are able to form due to the close proximity of hydride structures, permitting a brittle fracture through one hydride region to migrate to a neighbouring region, continuing until a large macroscopic crack manifests. This is herein referred to as *hydride connectivity*.

One beneficial technique for analysing these phenomena is the use of transmission electron microscopy (TEM), which generates two-dimensional micrographs of a thin sample slice and highlights changes in the sample composition, including the presence of hydride zones in a metallic matrix (see Figure 1). It is not known what microscopy technique was used to produce the hydride images in this study, however due to the low resolution of the images and the size of the hydrides it is thought to be TEM.

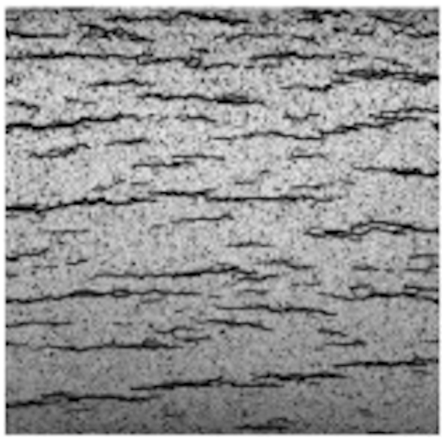
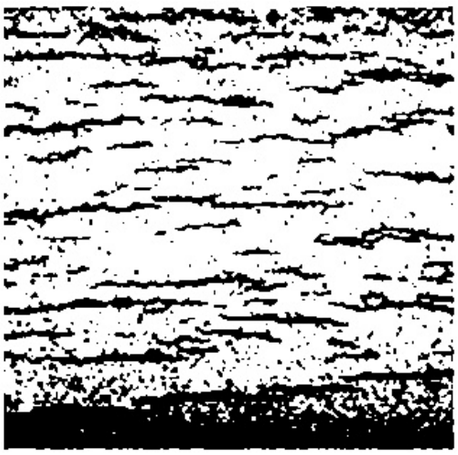
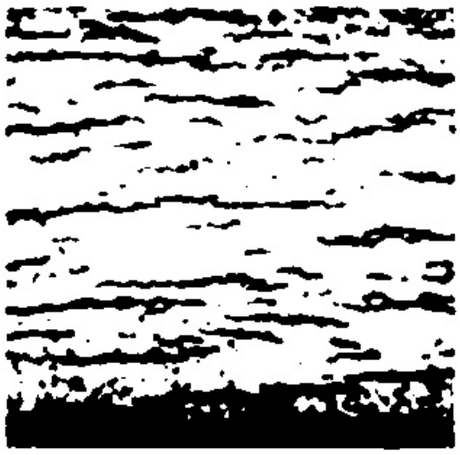


Figure 1: Typical TEM micrograph image of darker hydride bands within a zirconium matrix. The horizontal regions are sporadically connected together by small fractures within the material.

These micrographs may be processed using appropriate software environments to identify characteristics such as the density of hydride regions within the material, and the level to which connectivity is observed. Here, image processing techniques within the python programming environment are used to identify these properties, chiefly incorporating the *scikit-image* series of algorithms along with originally-developed scripts. For this project, code is developed, implemented and tested to determine the most effective means of isolating and analysing hydride structures within the images. An account of the evolution of the techniques is given, and results are presented and discussed.



(a) The binarised micrograph without application of Gaussian blurring.



(b) The same micrograph with Gaussian blurring applied prior to filtering.

Figure 2: Comparison between identical micrographs both with and without the application of a Gaussian blur algorithm.

2 Initial Image Processing

2.1 The Gaussian Blur

To begin this project, a series of pre-prepared micrograph images were manipulated and several different approaches in python were tested to provide an initial understanding of the basics of image processing. Algorithms were written which could apply different threshold filters to image value arrays once they had been read into python. When the images were read by the *Scikit image* package in python, they could be expressed as a numerical array with *Numpy*. Threshold values can then be applied to these arrays which can convert an image to binary (black and white) based on the threshold, this requires the image to be converted to greyscale initially. Although initial scripts succeeded in removing the zirconium matrix and isolating the hydride regions, a significant quantity of noise was observed caused by aberrations in the original images, this may have arised from sample pitting during electropolishing or foreign contaminants on the material or in the microscope used to take the image. As these aberrations manifested at similar pixel intensities to the hydrides themselves, filtering techniques were not able to eliminate them. Therefore, a *blurring* technique was applied prior to intensity filtering in an attempt to remove the noise artefacts.

Here, the function *cv.GaussianBlur()* is utilised, in which a convolution of a Gaussian function is applied to the pixel intensities within a region to be blurred. Although this results in a slight aberration of boundaries for well-defined regions such as hydrides, the effect is minimal due to the relatively large quantities of both black and white pixels either side of the boundary. However, for small scale regions such as several black pixels against a white background (typical of micrograph noise), the technique is highly effective as the small quantity of black pixels tends to disappear when averaged around a large white surround. Figure 2 highlights the removal of artefacts for an initial example micrograph. Upon testing, this technique proved effective for all trialled images.

Gaussian blur theory
The Gaussian blur is typically used to smooth boundaries in an image where an immediate change in pixel intensity is observed compared to neighbouring pixels. The function typically employs a square matrix of values known as a *kernel*. This kernel is 'superimposed' over the individual pixel values by an algorithm in a *raster* pattern and alters their intensity accordingly.

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} \longrightarrow \begin{bmatrix}0 & 10 & 170 & 220 & 130 & 30 & 10\\ 0 & 12 & 180 & 241 & 122 & 35 & 14\\ 2 & 13 & 176 & 237 & 134 & 43 & 7\\ 2 & 13 & 174 & 236 & 129 & 40 & 7\\ 1 & 11 & 159 & 224 & 132 & 31 & 5\\ 0 & 13 & 178 & 242 & 136 & 44 & 9\end{bmatrix}$$

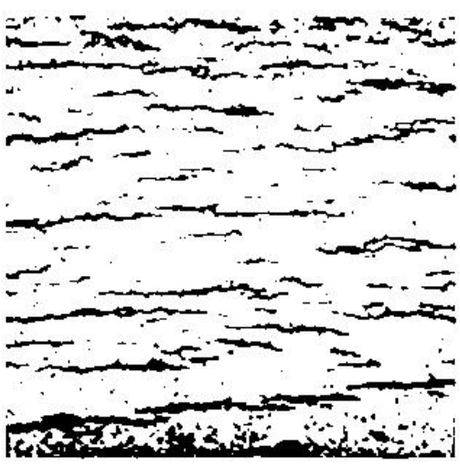
The above example depicts a 3×3 Gaussian blur kernel, although larger dimensions such as 5×5 may also be used. The binary image (represented here by a 7×6 matrix of pixel intensities) is *convoluted* through a specific matrix operation. This alters the pixel intensities of the filtered image as weighted by the values in the kernel. There are two possible ways to adjust the level to which the image is filtered; by increasing the dimensions of the kernel, which increases the range over which the intensities are averaged, or by adjusting the values in the kernel to provide a greater or letter degree of filtering. Each route provides its own merits and drawbacks.

In the context of this report, both routes to alter the Gaussian mask were trialled. The method proposed by S. Khan involved using a module named *OpenCV*, which is a cross-platform open source module that focuses on image processing techniques such as as greyscale to binary conversion, Gaussian and median blurring, filtering and thresholding. One approach of clarifying hydrides in the zirconium matrix was to initially blur and remove noise from the image followed by application of a pixel filter value as a method

of binarising (i.e. identifying a threshold value between 0 and 255). The idea behind this approach was to allow a user to specify threshold and kernel blur values by calling a single function, *image_conv*, which would display a clear binarised image with applied Gaussian blur. A before and after histogram is also displayed to show how the previous 8-bit image is filtered to a binary image with 0 and 255 as integer values. Image blur was increased by altering the dimensions of the mask overlaying the micrograph image. This served to significantly reduce the noise observed in Figure 2a, however the method relies primarily on an odd-integer input to specify square dimensions of 1,3,5..., etc. This can reduce the versatility of the technique as it limits control over the amount of blurring achieved for a specific region of pixels.

A. Hanson proposed a similar technique but instead utilising the *Scikit image* or *skimage* package. In this method the values within the Gaussian blur are adjusted through the use of altering the standard deviation from the mean used by the filter. Here, a larger specified value for σ will produce a larger range of values within the kernel and generate a more intense blurring, whilst lower σ values will generate higher *relative* pixel intensities at the fringes of the kernel and thus reduce the level of blurring, see Figure 3 (after binary thresholding). It was thought that this method proposed was more suited for the purpose of this study, as python code written in skimage allowed the gaussian blur σ to be adjusted individually for each image.

Using this approach, a function carries out the gaussian blur on an image array that has been read into python via *skimage*, then the sigma values from the middle column of Table 1 are specified in each case. There was also an additional input, *truncation* value, which truncates the gaussian filter after the set amount of standard deviations. This was kept at a constant value of 3 throughout the study, as this was the default value recommended in skimage and also data within 3σ's accounts for 99.9% of the data concerned. After this, images were ready for binary thresholding in the next function.



(a) Image 'chu1' filtered using a Gaussian mask where σ=0.5

(b) Image 'chu1' filtered with σ=1.

Figure 3: Post-processing of the same micrograph with differing values for σ. It is evident that higher values of σ remove a larger quantity of noise between hydrides, but also result in aliasing of hydride boundaries and systematic removal of some regions.

2.2 Binary thresholding

Applying the same parameters in a Gaussian filter to different images tends to yield largely different results for the quantity of noise reduction in a range of images. In the sample image set provided for this investigation, many TEM micrographs exhibited a high degree of peripheral darkening, which leads to difficulties in applying an effective intensity threshold without suffering losses to the visible hydrides. As such, a threshold and standard deviation must be manually agreed upon for each image to ensure maximum hydride retention whilst eliminating a sufficient quantity of noise. This presents a significant limitation of the approach as large quantities of micrographs may become time-intensive to analyse. In addition, the systematic removal of small hydride regions may generate a significant source of error when cross-comparing images. Practically, this can cause an over- or underestimation of hydride quantity within a sample and difficulties may arise in subsequent connectivity analysis.

For the sample set of images; *chu1, chu2, chu3, chu7, chu11*, threshold and filter values (namely = σ) were agreed upon based on their perceived effectiveness as follows. Thresholding is a very simple operation and is applied with a simple python function that reads in the previously blurred image array, plots a respective histogram of it's values, and then applies the following operation, based on the specified threshold;

$$\text{In respective image array: } value > threshold = true$$

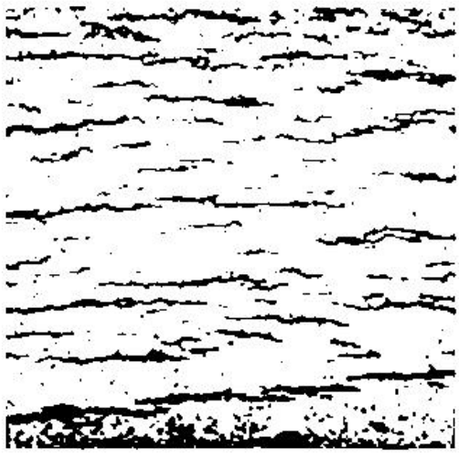
This sets any values in the image array greater than said threshold to true, numpy in python is then able to interpret this and convert this to an integer array of 1's and 0's which produces the binary images as displayed further on. In this case, a value of 1 is equal to white, when read in by skimage.

Two sets of parameters were selected for each image conversion to highlight the effect of gaussian blurring and thresholding on the output images. As such, the images filtered with the lower set of values are referred to as *low-pass* (LP) and those filtered through higher values are denoted *high-pass* (HP), see Table 1. It was decided to proceed with the low-pass binary images produced in an effort to reduce both excessive noise and excessive hydride omission.

Low-pass Images			
Image Reference	σ (Gaussian blur)	Intensity Threshold Value	Gaussian Truncation Value
chu1	0.5	0.4	3
chu2	0.5	0.35	3
chu3	0.5	0.5	3
chu7	1.0	0.4	3
chu11	1.0	0.4	3
High-pass Images			
Image Reference	σ (Gaussian blur)	Intensity Threshold Value	Gaussian Truncation Value
chu1	1.0	0.5	3
chu2	1.0	0.4	3
chu3	1.0	0.4	3
chu7	1.0	0.6	3
chu11	1.0	0.5	3

Table 1: Tested parameters gaussian blur σ and binary threshold (see later sections) for the studied images using the *scikit image* approach. This was split into low-pass and high-pass to investigate the effect of altering parameters.

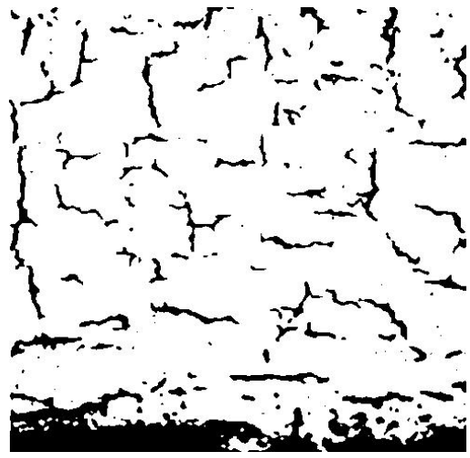
These values generated the following pre-processed images shown in Figure 4.



(a) *chu1(LP)*: σ=0.5, T=0.4

(b) *chu1(HP)*: σ=1.0, T=0.5

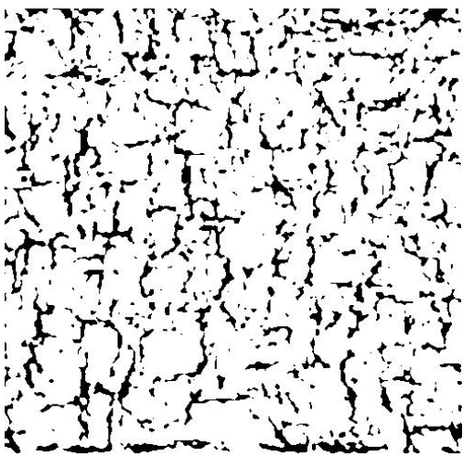
(c) *chu2(LP)*: σ=0.5, T=0.35



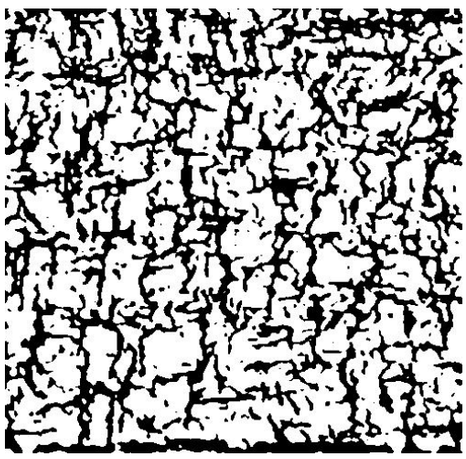
(d) *chu2(HP)*: σ=1.0, T=0.4

(e) *chu3(LP)*: σ=1.0, T=0.4

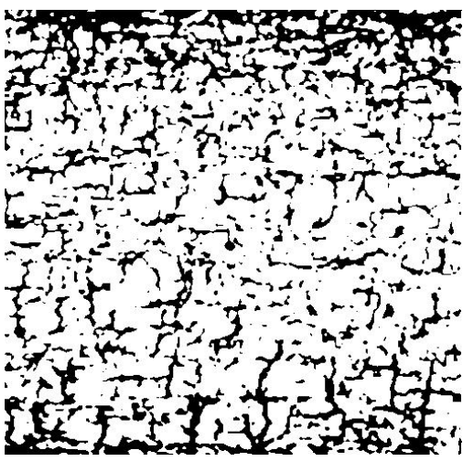
(f) *chu3(HP)*: σ=0.5, T=0.5



(g) *chu7(LP)*: σ=1.0, T=0.4



(h) *chu7(HP)*: σ=1.0, T=0.6



(i) *chu11(LP)*: σ=1.0, T=0.4

(j) *chu11(HP)*: σ=1.0, T=0.5

Figure 4: Pre-processed micrographs from the sample image set. Two images with differing threshold and σ values are included for each micrograph.

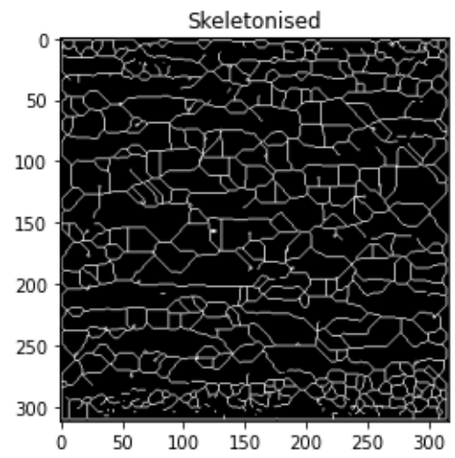
3 Initial Trial Methods to Assess Hydride Connectivity

Following pre-processing, techniques were trialled in an attempt to analyse the level of connectivity between regions of H_4Zr in the micrographs [1], [2], [3].

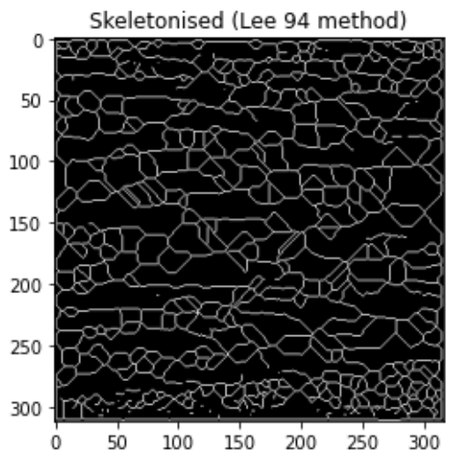
3.1 Skeletonise function from Sci-kit Image

Skeletonisation was implemented as an initial approach for obtaining a simplified path for connected hydrides. This function would theoretically reduce the hydride map to a connected graph of thin lines (approximately one pixel in thickness). Using *skimage.morphology*, the skeletonise() function was tested on image 'chu1'. From the demo code produced, an initial reading of the file from the local directory was performed followed by specifying the image size. Using *skblur* and *skbinary*, 'chu1' was binarized and blurred to give the output as shown in Figure 4a. It is interesting to note that there were different variants to the skeletonise() function. One output (Zha84) initially takes the edges of an object (a hydride in this case) and iteratively removes pixels, stopping when the neighbouring pixel is black (where the intensity is zero) [4]. As long as the image in Figure 4 (a) is inverted, the result is a series of single white pixel lines. Another output (Lee94) uses an 'octree' data structure which is more suitable for 3D images. Like Zha84, this algorithm also performs sweeping iterations over the image where pixels are continuously removed until the condition is satisfied where two iterations produce the same image and there is no change in the pixel count [4]. Figure 5a and Figure 7b shows the results of Zha84 and Lee94 methods respectively, which were applied to the binarised 'chu1'image from Figure 4a.

These results, however, are unsuitable for analysing connectivity as it appears that the matrix has been identified as the object rather that the hydrides themselves. Crucially, the skeletonise function does not provide actual data which can be applied to find suitable microstructural characteristics. Therefore, it was decided to assess an alternative characteristic of the microstructure to determine the hydride characteristics (discussed later). Finally, Figure 6 below illustrates a representation of how *medial_axis* works by identifying the edges of the hydrides and then prints a skeleton (colour map=magma) [4]. In this case the skeleton is performed on the matrix rather than the hydride, this may be because of the initial 'chu1' binary image array, as these values are not inverted.



(a) Zha84 method of skeletonisation applied to 'chu1'.



(b) Lee94 method of skeletonisation applied to 'chu1'.

Figure 5: Processing image 'chu1' with two skeletonisation methods.

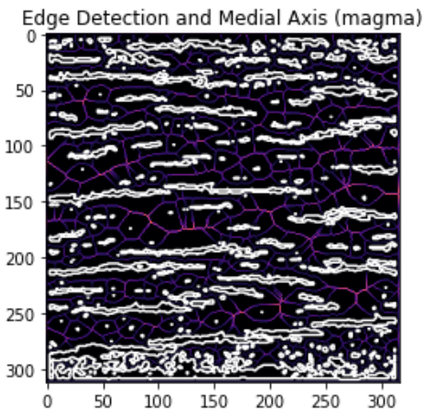
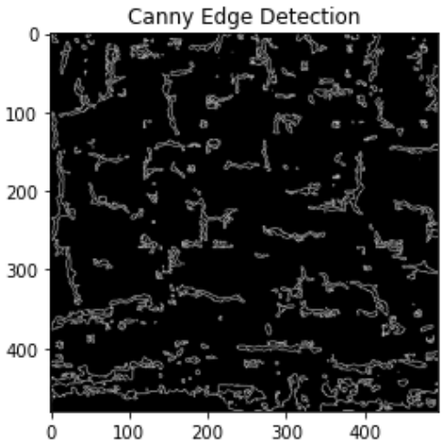


Figure 6: Skeletonisation approach with medial axis pattern printed in a magma colour map, again applied to 'chu1'.

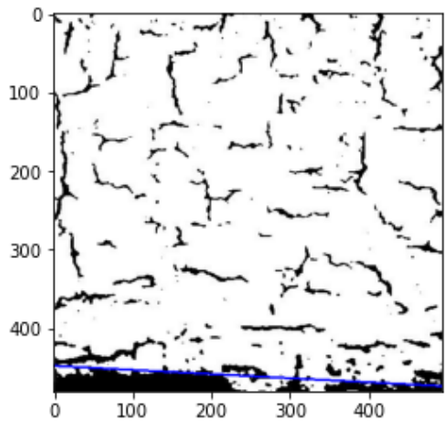
3.2 Canny Edge Detection and the Hough Line Transform

The *Hough Line Transform* relies on the detection of pixels within a series of lines superimposed onto a sample image. Rather than a typical first-order equation $y = mx + c$, the method defines these lines as the perpendicular normal to a line of angle θ from the top-left origin.

It has been established that Hough transform is useful in detecting shapes using mathematical forms such as cartesian coordinates. For the case of straight lines, *Hough space* can be used to identify the distance from the image origin to a line and the angle of the line relative to a specified axis. This description is for the OpenCV, *cv2*, module which uses the 'cv2.houghlines' function. The main advantage of using Hough transform when put up against skeletonise is that it can return rho and theta values in a 2D array. Rho is the distance which is taken as positive if the line is above the origin and negative if the line is below the origin. Theta is the angle measured from the horizontal axis in the counter-clockwise direction. 'Hough.transform' uses the coordinate system (rho, theta) in a voting system [5]. Here, a straight horizontal black line can be visualised in a white background image. If the start point of the line is at an initial value of (40, 90), 'Hough.transform' will detect this and begin searching the image to find matching the number of (40,90) cells. The maximum count for this cell would indicate that a line lies on this coordinate [5]. When using the function, Canny edge detection was used to better visualise the hydrides. This function can also simply threshold and binarise an image, without allowing the user to change truncation or sigma values. After testing, the following results were given, as shown in Figure _ (a) and (b).



(a) Canny edge detection on image 'chu2'



(b) Attempted Hough transform on image 'chu2'

Figure 7: Canny edge detection and hough.transform() used on image 'chu2' taken from Figure 4c.

4 Radial Hydride Fraction to Assess Connectivity

4.1 Initial Approach using Hough Line Transform

Due to difficulties in implementing a viable technique for determining the level of connectivity of hdyrides within test micrographs, the focus of the project was adjusted to assist in determining the likelihood of through-wall cracking and subsequent mechanical failure of analysed zirconium samples. As previously highlighted, the rate of degradation of fuel cladding is dependent on the orientation of hydrides within the grains. However as observed many samples exhibit hydrides oriented both radially and tangentially; a ratio is therefore needed to assess the contributions of the grain orientations to the likelihood of failure by through-wall cracking. Known as the *Radial Hydride Fraction* (RHF), this Initial approach, hough line transform over the whole image

4.2 Refined Approach

Explain methodology that we use to obtain ACTUAL RHF - image slicing and the box.

5 Results and Discussion

5.1 Resulting Radial Hydride Fractions

Results

5.2 Discussion

Discussion of results

6 Conclusions

6.1 On Hydride Structure Analysis

6.2 Recommendations

References

[1] R. K. Sharma, A. K. Bind, G. Avinash, R. N. Singh, A. Tewari, and B. P. Kashyap, "Effect of radial hydride fraction on fracture toughness of CWSR Zr-2.5Nb pressure tube material between ambient and 300C temperatures," *Journal of Nuclear Materials*, vol. 508, pp. 546–555, 2018. [Online]. Available: https://doi.org/10.1016/j.jnucmat.2018.06.003

[2] P. C. A. Simon, C. Frank, L. Q. Chen, M. R. Daymond, M. R. Tonks, and A. T. Motta, "Quantifying the effect of hydride microstructure on zirconium alloys embrittlement using image analysis," *Journal of Nuclear Materials*, vol. 547, p. 152817, 2021. [Online]. Available: https://doi.org/10.1016/j.jnucmat.2021.152817

[3] S. Sunil, A. Gopalan, A. K. Bind, R. K. Sharma, T. N. Murty, and R. N. Singh, "Effect of radial hydride on delayed hydride cracking behaviour of Zr-2.5Nb pressure tube material," *Journal of Nuclear Materials*, vol. 542, p. 152457, 2020. [Online]. Available: https://doi.org/10.1016/j.jnucmat.2020.152457

[4] Scikit-image, "Skeletonize — skimage v0.19.0.dev0 docs." [Online]. Available: https://scikit-image.org/docs/dev/auto_examples/edges/plot_skeleton.html

[5] OpenCV, "OpenCV: Hough Line Transform," July 2013. [Online]. Available: https://docs.opencv.org/master/d6/d10/tutorial_py_houghlines.html

A : Making the code user friendly

Small description on how functions were written to make the process easy for the user.