Imperial College London

Automated Analysis of Thin Section Images

August 30th 2019

Applied Computational Science and Engineering MSc
Independent Research Project

by
Richard M. G. Boyne
Email - rmb115@ic.ac.uk
Github - Boyne272
CID - 01057503

Supervisors

Prof. Olivier Dubrule
Imperial College London
Dep. of Earth Science & Engineering
o.dubrule@imperial.ac.uk

MSc Lukas Mosser
Imperial College London
Dep. of Earth Science & Engineering
lukas.mosser15@imperial.ac.uk

Dr Meindert Dillen
Wintershall Dea GmbH
EOR Projects
meindert.dillen@wintershalldea.com

MSc Tobias Thiel
Wintershall Dea GmbH
Exploration & Production
tobias.thiel@wintershalldea.com

This project was carried out over three months as part of a longer internship with Wintershall DEA GmbH at their headquarters in Kassel, Germany.

Abstract

Analysis of geological Thin Section images is common a practice in the petroleum industry, though existing techniques are often labour intensive and have poor consistency for comparison between samples. Here the TSA python package is developed to provided a tool set for automating the the more time consuming processes involved. Specifically it aims to deploy a sequence of image segmentation followed by material clustering of segments by material to obtain property distributions over the image. This is developed in a highly adaptable manner, such that a wide range of possible metrics and features can be used in conjunction with existing data science packages. Preliminary results shows good segmentation via SLIC[1] and MSLIC[2] but poor clustering due to a failure to incorporate texture and color features effectively. This opens itself to further software development and investigation, for which the software's sustainable design and flexibility are well suited.

Code Source

The current software version (first release v1.0.0) is publicly accessible from (1) with the user manual in the repository and also available in the documentation page at (2). Documentation is generated with sphinx in an interactive html format.

1. https://github.com/msc-acse/acse-9-independent-research-project-Boyne272
2. https://msc-acse.github.io/acse-9-independent-research-project-Boyne272/index.html
3. https://boyne272.github.io/acse-9-independent-research-project-Boyne272/index.html a repo mirror of (2)

Contents

Abstract **i**

Code Source **i**

1 Introduction **1**

1.1 Thin Section Analysis 1

1.2 Current Practices 1

1.2.1 Model Analysis 2

1.2.2 Computational Image Processing 2

1.2.3 Machine Learning Techniques 2

1.3 Project Motivation 3

1.4 Proposed Solution 4

1.5 Project Aims 5

2 Software Development **6**

2.1 Development Methodology 6

2.2 Algorithm Theory 7

2.2.1 SLIC and MSLIC 7

2.2.2 Unsupervised Segment Clustering 8

2.3 Software Architecture 9

2.3.1 SLIC and MSLIC 9

2.3.2 Segments 10

2.3.3 AGNES 10

2.4 Run Time Analysis . . . 11
2.5 Sustainability . . . 11
2.5.1 Testing . . . 12
2.5.2 Coding Practices . . . 12
2.6 Technical Specifications . . . 12

3 Software Performance **13**
3.1 Sample Images . . . 13
3.2 Over Segmentation with SLIC and MSLIC . . . 13
3.2.1 Verification . . . 13
3.2.2 SLIC Performance . . . 14
3.2.3 MSLIC Performance . . . 15
3.3 Segment Merging . . . 16
3.3.1 Verification . . . 16
3.3.2 Clustering Feature Selection and Results . . . 16
3.3.3 Edge Confidence and Results . . . 17
3.4 Property Determination . . . 17

4 Discussion and Conclusions **20**
4.1 Discussion . . . 20
4.2 Conclusion . . . 21

Bibliography **23**

Appendices **24**
A Solution Skeleton with Images . . . 24
B Sample Thin Section Sandstone Images . . . 26

Chapter 1

Introduction

1.1 Thin Section Analysis

Geological Thin Sections (TS) are samples of rock cores thinly sliced, encased in epoxy and polished to $\sim 30\mu m$ thick for inspection. Images of these are then collected through a petrographic or electron microscope under plane polarised light, often in sets with different cross polariser angles[3]. Appropriate analysis of TS images can determine a material's composition, cleavage, grain size distribution, silt ratio, porosity and mineral identification[3]. Such properties contribute to the description of the geological environment and geological history at the location of the sample, hence TS analysis is of great importance for Exploration and Production (E&P) in the petroleum industry.

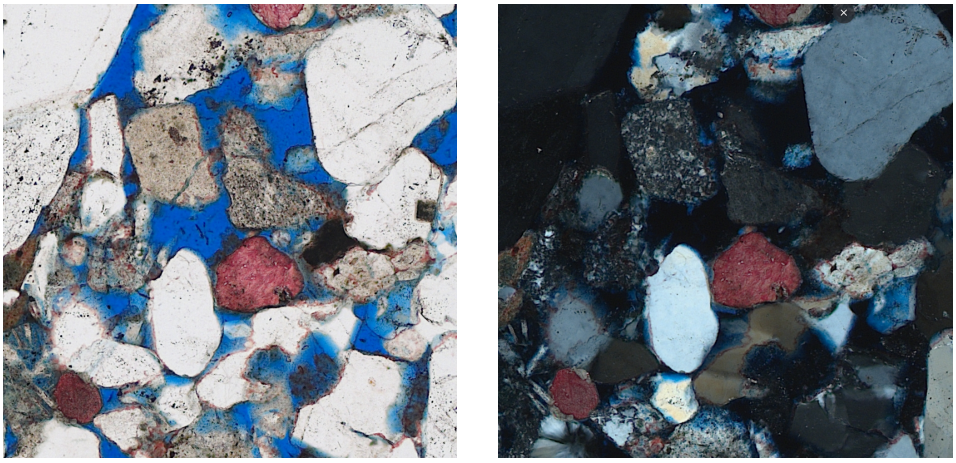


Figure 1.1: Example Thin Section Images of Litharenite Sandstone sample from the coast of Texas, with parallel polarisers (left) and cross polarisers (right).

1.2 Current Practices

TS analysis is often one part in many forms of identification for a mineral sample, often combined with experimental data (such as CO_2 injection[3]). A common issue here is the largely varied nature of rock structures. Even the same mineral type taken from different regions can look very different, and as such caution needs to be taken that analysis can account for this variation.

1.2.1 Model Analysis

Traditionally the method for analysing TS images is manually identifying the properties of several hundred random points in the image, giving an estimated distribution over the whole materials. This is a laborious process which requires an experienced geologist, even more so when multiple polariser angles are considered. It is also highly subjective; increasingly so when the varied nature and often messy, ambiguous structure of real-world samples are taken into consideration. The values obtained from this analysis are generally in good agreement with those from experimental data[4], within the uncertainties from its human element.

1.2.2 Computational Image Processing

A more modern method is to use a combination of multiple image filters, then apply various algorithms such as colour thresholding or watershed segmentation to extract information about the TS image. Such analysis was demonstrated by Berrezueta et al (2019)[5]. However, considering the variation of TS images there is no single routine that can analyse all types, especially since these processes are very sensitive to the choice of filters and hyperparameters used. Hence this has to be implemented manually, with these choices made by the analyst, making this a subjective process. For a given routine there is also large potential for systematic errors, such as regions of significance being just under a chosen threshold. Hence this form of analysis can be difficult to quantify the uncertainty or accurately compare the results from different images.

1.2.3 Machine Learning Techniques

Over the past two decades, there has been a significant increase in the use of Machine Learning (ML) techniques in both research and industry. Although ML methods are becoming common practice in various industries, they are still very new in the petroleum industry. Current operational uses are largely aimed at organising of unstructured data (such as reports, etc) and predictive maintenance.

A common ML problem is the classification of a feature set, hence previous work has attempted to automate the mineral classification part of TS analysis. Młynarczuk et al (2013)[6] approached this with supervised nearest neighbour clustering, achieving test accuracies as high as 99.8% on nine individual rock samples. Cheng et al (2017)[7] utilised deep convolutional networks for this, obtaining test accuracies up to 98.5% using samples from the Ordos Basin. Both these papers outline the importance of considering different colour spaces for image processing (e.g. RGB, HSV, etc.). Whilst the accuracies are high, when one considers the variability of minerals these classifiers have less practical application as a labelled dataset for all relevant regions in an industrial setting is very costly to obtain.

To combat the issue mineral locality, Li et al (2017)[8] developed Festra, a transfer learning approach using networks pre-trained on one region to accelerate the learning in another. This is more practical but still requires some labelled images from a region before classification can occur.

Datasets of segmented images are rare and the large variability of segmentation tasks makes them not often representative, as such unsupervised approaches are often taken. One such approach is to segment the image into its regions of different materials. It is then much faster to analyse by model analysis, or more automated approaches can then be developed to extract the same information. Image segmentation in computer vision is commonly approached as a clustering problem of pixels as feature vectors. This has led to the development of superpixel algorithms[1] where a superpixel is a name given to each pixel group.

Achanta et al (2012)[1] developed a k-means variant for superpixel clustering dubbed Simple Linear Iterative Clustering (SLIC). This was expanded upon and applied to multiple TS images by Jiang et al (2018)[2] as the Multi-SLIC (MSLIC) algorithm. The main issue with a k-means approach is the number of segments must be predefined; since the number of segments is rarely known this is usually overestimated, then followed by some form of segment merging to achieve the desired result. Jiang et al implemented merging using the colour, texture and edge features of each segment under threshold-based merging criteria. This showed F-measures from 62 to 73% for the nine samples inspected (with four polarisation angles per sample). It is however highly sensitive to the threshold hyperparameters and consideration needs to be taken to the samples used to tune them.

1.3 Project Motivation

As the current practices show there is still much to be desired when it comes to automated TS analysis. Specifically, there are currently no methods that can speed up the analysis in such a way that is consistent between samples. From a business perspective, the obvious benefit of such a method would be saving geologists time. More importantly it would also increase the usability of TS based results in E&P, since they can be better compared to previous cases and therefore give improved predictions.

1.4 Proposed Solution

Here we propose a software package which aims to increase automation of TS analysis in a way that reduces human subjectiveness and is valid for a wide range of TS images without large changes in the routine. Large labelled datasets are not available here, which justifies the development of an unsupervised segmentation approach. The software is developed not to completely remove the need for human input, but make it far less ambiguous and laborious. The general outline of this pr workflow is shown in figure 1.2.

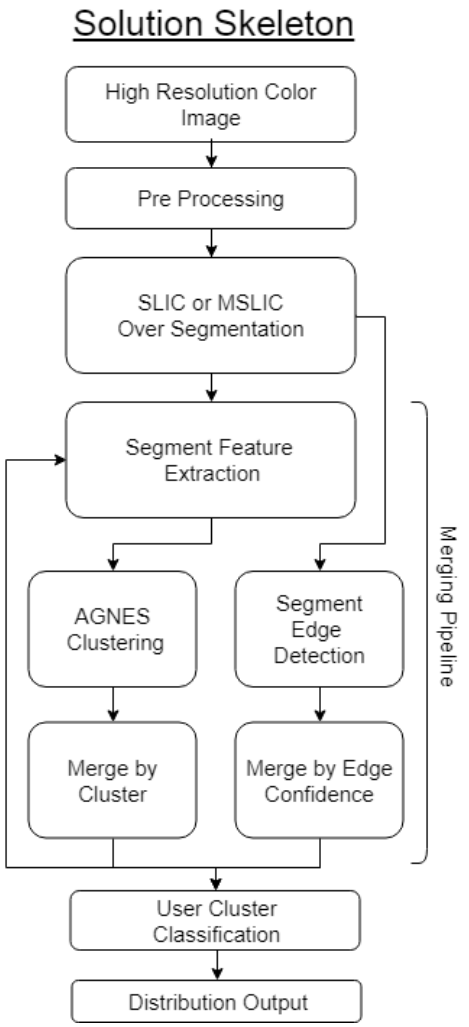


Figure 1.2: Outline of proposed solutions workflow. The output image at every stage in this workflow is shown in appendix A. The merging pipeline region is designed to be easily integrated with other data science pipelines, see section 2.1.

First, the SLIC or MSLIC algorithm is used to over-segment the image such that the boundaries between different material regions are outlined by some of the segments. Due to the recent development of these algorithms, they are implemented from scratch.

This is followed by a pipeline for feature extraction (using the segmentation), then unsupervised clustering of segments associated with the same material. This is one way to identify segments that should be merged, leaving only the different material grains outlined as desired. Another way to identify this is with segment edge analysis, which attempts to determine if the boundaries of a segment are desirable. Either method can be done and then repeated until the desired merged segments are obtained. A significant benefit of this approach is a reduction in the need for arbitrary merging criteria with a high number of sensitive hyperparameters. The hope is that such merging methods will be better able to cope with the variability of TS images, however, this is all dependent on the versatility of the unsupervised clustering algorithm and edge detection methods.

Given the clusters found, a user can identify the material of each cluster, which is relatively fast as the number of clusters is proportional to the number of different materials in the image. Finally, there is a routine to output quantitative parameters associated with the distributions of each cluster. These will include relative compositions and geometric properties such as grain size distribution.

1.5 Project Aims

This project aims to produce the proposed software. Its implementation should be reasonably efficient as TS images are very large, $\mathcal{O}(10^4 \times 10^4)$, therefore inefficient computation or memory usage could be impractical to deploy. Intended users are geologists, hence it must have an interface which is suitable for a less computer-knowledgeable user. As this software solution is likely to be further developed within Wintershall DEA the code clarity and sustainability are the main priority, with optimisation being the second.

Throughout care should be taken to keep the routines as general and adaptable as reasonably possible. The goal is less to solve the issue of TS segmentation as to build a set of tools that can be used to do this. This is partly due to the time frame of this project as finding the exact configuration of features, clustering algorithm and hyperparameters for accurate analysis is a lengthy process. Such configuration would also be biased by the limited image samples available for software development.

The performance of this software is considered concerning the quality of segmentation and clustering as well as resource usage, flexibility and usability. Segmentation and clustering are analysed by qualitative inspection and quantitative comparison of the output distributions to a small sample of TS images that have analysed by experiment. However, these results will need to be taken as more of an indication given the above discussion of not finding the ideal configuration.

This project has a relatively short time frame; as such various software elements can not be implemented here. For example handling the original size TS images is avoided by using smaller sections ($10^3 \times 10^3$), which are still valid for analysis as they are large compared to the grain sizes. This approach naturally lends itself to parallelisation in the future, but this is also not considered here.

Chapter 2

Software Development

Python is the chosen platform here as it is one of the most widely used high-level language and is free to use. As a result it has many pre-existing packages for image processing and data science that are highly optimised. It also requires less training than many other coding languages, meeting the usability requirement of this software. Other than integrating publicly available packages where applicable this software is being developed from scratch. As future development within Wintershall DEA is expected, either improved performance or deployment, code sustainability is a key focus.

Google Colab[9] is chosen as a convenient development environment since it has a bash environment (need for git access), useful development tools (e.g. code completion) and no installation required (useful given company software restrictions).

2.1 Development Methodology

Within the field of data science, current common practice is to use a pipeline of pre-made packages for data pre-processing, feature extraction, analysis (e.g. ML algorithms) and results processing. This pipeline structure is advantageous in many ways, such as:

- Sections can be re-used or exchanged easily, allowing for quicker solution development and algorithm experimentation
- Pipelines are very adaptable due to fairly consistent inputs/outputs at each stage, meaning only a small number of packages is needed to create widely varied software with little development effort
- Hence software packages are very reusable, which allows them to be well developed, highly optimised and widely distributed
- The compartmentalised structure of a pipeline significantly reduces the complexity of code since each part needs only do a relatively small task, making code more sustainable (see "the complexity beast"[10])

This structure is therefore used here and will therefore be usable in conjunction with existing data science packages.

The compartmentalised nature of the outlined solution lends itself to the Agile development methodology[11], in which each software functionality is developed independently in the sequence that it will be used in. This is versatile as it can adapt to changes in the software plan, making it often the method of choice for project with time pressure and/or a reasonable degree of uncertainty. Agile methodology is also good for short projects as even if the full software is not finished, the sections that have been give a still somewhat useful product.

2.2 Algorithm Theory

2.2.1 SLIC and MSLIC

The superpixel algorithm SLIC is a variant on k-means clustering specifically designed for images. Pixels are considered feature vectors in a 5D position-colour space (i.e. X, Y, C1, C2, C3, where Cx is a colour channel). Clusters are restrained to local regions by binning pixels into a regular grid, so that each pixel only able to join a cluster from one of the neighbouring bins. This prevents large clusters forming across large regions of the image, often leading to contour like segments that are difficult to interpret. It is also has the advantage of reducing the number of calculations needed as only local pixel-centroid distances need by calculated, This makes the complexity is independent of the number of segments (excluding the re-centring calculation).

As k-means is a distance-based approach an appropriate distance metric in position-colour space is required. Achanta et al[1] proposed the following:

$$D^i_{p,q} = \sqrt{\sum_{x=1}^{3}(Cx_p - Cx_q)^2} + \gamma\sqrt{\frac{K}{N}}\sqrt{(X_p - X_q)^2 + (Y_p - Y_q)^2} \tag{1}$$

Where $D^i_{p,q}$ is the distance between pixels p and q in image i, N is the total number of pixels, K is the number of segments and γ is a chosen spatial bias. This equation acts as the euclidean distance on a linearly scaled position-color space.

If we have multiple different images of the same object (such as the different polariser angles in a TS image set) then using information from all images in a single clustering should result in a better image segmentation. We could therefore use a combination of the above distances in each image, giving the MSLIC method. Jiang et al[2] gives an example combination metric:

$$D^*_{p,q} = \max_i \left(D^i_{p,q}\right) \tag{2}$$

where D^* is the combined distance used in the k-means algorithm. This was found as the preferred metric since it prioritises more confident boundaries in all images. Whilst equations 1 and 2 are established methods they are not the only choices for these metrics, hence they should be changeable by the user.

2.2.2 Unsupervised Segment Clustering

For the clustering of segments with common material, there are many possible choices of an algorithm. The pipeline structure is such that feature vectors for each segment are extracted from the initial segmentation and once clustered they are passed back into the next stage of the pipeline. This allows any externally sourced unsupervised clustering algorithm to be used within this software's framework, improving adaptability.

For this project, Agglomerative clustering (AGNES[12]) was used since it is capable of determining the number of clusters to form within a given tolerance. This is a required functionality here since the number of different grain types is unknown. It is also able to cope well with outliers, which are often present in TS images.

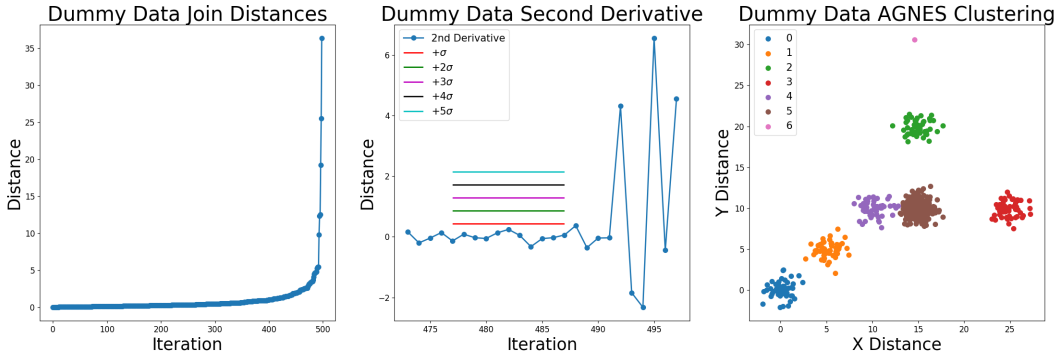


Figure 2.1: An example AGNES clustering on dummy data. Left shows the distances of each cluster merger until all clusters have been merged. Middle shows the second derivative of the last 25 merges along with standard deviations, the sudden increase at 492 corresponds to the 7 clusters in the dummy data (6 groups and 1 outlier). Right shows the resultant clustering in the dummy data from std = 3σ termination.

AGNES operates by initially considering each individual vector a cluster, then merging the two nearest clusters repeatedly. The method varies by how the new cluster's distance is calculated and what termination criteria is used. Here a new cluster's distance to anyone vector is the maximum of the distances between the vector and two unmerged clusters. This is often best at separating meaningful clusters as it takes into account the span of a cluster as well as it's proximity, preventing a few large clusters from growing too fast. Termination should occur when the distance between the nearest clusters significantly increases, which is seen as an elbow in the iteration-distance curve shown in figure 2.1. One way of identifying the elbow is a

consequential spike in the second derivative. For the dummy data shown this spike is very clear, but that is not always the case so other termination criteria should be considered too.

2.3 Software Architecture

Python, being an Object-Oriented language, lends itself to classed-based structures. These keep data and processes for a single task neatly together. Separate classes are made for SLIC, MSLIC, segment feature extraction and merging, AGNES, segment analysis and image processing (for pre-processing the original TS image). Some key design elements of these classes are described below.

2.3.1 SLIC and MSLIC

The SLIC class iterates through a process similar to the classical k-means algorithm in three-steps:

1. calculate the distance between each local centroid-vector pair
2. update the clusters by assigning each vector to its nearest centroid
3. update the position of the centroids based on their constituent vectors

The advantage of breaking down the iteration like this means that working on multiple images (MSLIC) can be easily implemented by having multiple SLIC instances running concurrently. A 1.5 stage is then inserted where distances in each instance are combined by the desired metric, hence MSLIC is a wrapper around the SLIC implementation.

Stage 1 is most time-consuming as this has to be done for every local centroid-pixel pair. Pytorch tensor operations help optimise this through their parallelised structure, further leveraged by doing as many calculations in a single operation as possible. MSLIC is additionally optimised by ensuring its SLIC instances do not repeat calculations (e.g. step 2 is only calculated once and copied to each instance).

As mentioned in section 2.2.1 the SLIC distance metric and MSLIC distance combination metric should be changeable by the user. This is done by optionally passing an externally defined function and arguments to the class initialisation. Another important feature of these classes is there a wide range of visualisation methods which are essential for the analysis of different configurations and metrics.

2.3.2 Segments

Segments are processed by having a class for individual segments and a wrapper class holding them all together. In this way, methods are logically divided between actions of an individual segment, such as finding its neighbours, and actions on the overall image, such as merging segments.

For merging, every neighbour of a segment needs to be identified. Furthermore, determining if an edge is present requires the specific bordering pixels. This is implemented by iterating over every pixel in the image, a computationally demanding, but less so than SLIC or MSLIC, hence this operation was not notably optimised.

To keep this process flexible the user defines a function and arguments to extract features for each segment, allowing any combination of processed images to be used to as clustering features. The same approach is taken to edge detection, except the function iterates on boarder pixels and returns an edge confidence value instead. Merging is then done between adjacent segments an edge confidence below a given threshold and/or segments the same cluster. Since this is more a framework for analysis rather than a predefined solution visulisation of the results is important with several versatile plotting methods implemented.

2.3.3 AGNES

This class operates only on the features extracted from the segments, as such can be considered a stand-alone tool. Since the iterations are identical regardless of the termination criteria the algorithm is iterated until all vectors belong to a single cluster, storing the pairings and distances as they occur. From this the clusters can be formed with the desired termination criteria, here three criteria have been implemented:

1. by variation of the second distance derivative above a given standard deviation
2. by pairing distance above a given maximum
3. to a specified number of clusters

The first method advantageous for variable feature values as it interprets when to terminate by the merge distances themselves. This is less effective if a smaller feature sample is used, hence the second option allows clustering by an arbitrary distance.

The main optimisation here is to leverage scipy.linalg methods for fast computation of distances. Clustering is a less computationally demanding compared to segmentation as there are much fewer segment features than pixels, hence scipy was chosen over Pytorch for simplicity of implantation.

2.4 Run Time Analysis

As SLIC (and hence MSLIC) is the most computationally demanding task here the time complexity for SLIC should be representative of the whole software. SLIC is shown to scale linearly with image size in figure 2.2, as is expected from the algorithm. For grid size (i.e. number of segments) the time fluctuates somewhere closer to a square root relationship, which is less the the linear scaling expected from classic k-means. This is likely a result of needing to recalculate centroids positions at every iteration. MSLIC scales in the same manner and additionally be proportional with the number of images being used.

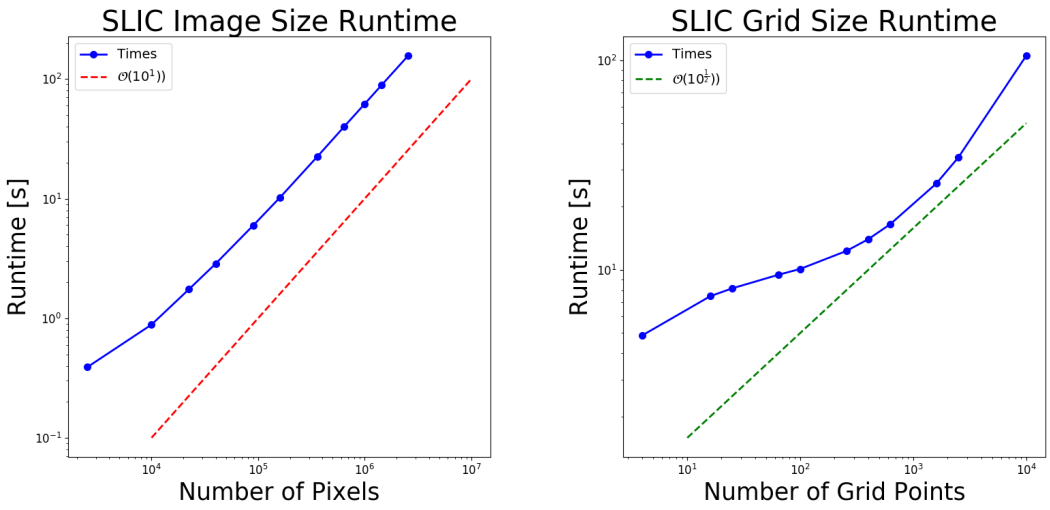


Figure 2.2: Runtime for SLIC implantation for 10 *iterations. Left shows varying image size with a 1st order line of reference. Right shows varying grid size with a half order line of reference.*

Memory usage was not analysed here as Google Colab's overhead masked any memory variation between image sizes. However, this shows the memory demands for the image sizes tested are not problematic, though this may not be the case for larger image sections. It is worth noting that although SLIC is the dominating computational demand at the scales investigated here it may not be case for larger image sections, as other regions may scale worse. However time restrictions don't allow for a full analysis of this possibility, though it should be considered in future work.

2.5 Sustainability

Where possible existing code packages were used, especially for computationally demanding tasks where optimisation is most significant. However, care was also be taken to keep external dependencies of this software low, as few non-standard library packages as possible were used. This helps reduce software maintenance and potential compatibility issues.

2.5.1 Testing

Testing is key for sustainable code, with integrated testing being the ideal solution for the efficient development of multi-platform software. However, this has significant setup time and overhead which due to delays with beginning this project and its short time frame was deemed too costly. Instead, tests are written in an integrated manner and run manually with Pytest at the start of each day (when code is imported to Google Colab from the repository), making this a scheduled testing methodology.

Unit tests are developed and used throughout the development cycle as common practice. Due to the compartmentalised nature it is logical also do acceptance tests with expected outcomes for simple cases, which are particularly useful when improving and optimising code.

2.5.2 Coding Practices

Often code performance is a top priority, however this package is developed with the potential for future development in mind. Therefore code clarity is prioritised over performance within reason. As a general practice each completed section is repeatedly revised to be improved upon. This includes anything from changing variable names or redesigning function structure to something then known to be more logical. Whilst this is labour intensive it iteratively improves the software to be clearer and easier to follow.

Other general good practices are adopted, such as Git version control via a GitHub repository (ensuring the use of git-flow), clear documentation (the numpy doc-string practices used and compiled by sphinx[13]), using Pylint on all code and hiding internal objects not intended user interaction.

2.6 Technical Specifications

As mentioned dependencies were kept as low as possible and always open source; only scikit-image[14] and PyTorch[15] and standard library packages (numpy, scipy, etc.) are used. The hardware requirements for $(10^3 \times 10^3)$ image sections are low enough for use on any modern computer. It is however recommended that the chosen environment is optimised for Pytorch (e.g. Google Colab), to best utilise vector based performance optimisations. Package formalisation is implemented for pip instillation, though distribution is kept only to GitHub.

Chapter 3

Software Performance

3.1 Sample Images

TS images are in general challenging to segment, largely due to very similar regions to be segmented, often with noisy textures which are difficult to differentiate. As such code is developed using a simpler image of a butterfly (seen in figure 3.1). This image is chosen as on the macroscopic scale it is far easier to segment into regions of butterfly, flower and sky. Additionally, the wing patterns closely resemble many of the challenging features seen in TS images making, it a suitable comparison.

Once developed the code is validated using a small set of TS images of offshore sandstone, chosen for its large porosity and distinct grains. Samples are chosen from a wide range of locations to include regional variations (see appendix B for locations and sample images). As this dataset is small and restricted is is only sufficient to be a proof of concept, which is appropriate here given optimal hyperparameter and feature selection is not considered.

3.2 Over Segmentation with SLIC and MSLIC

3.2.1 Verification

Unit tests are used to ensure the outputs at each computation stage are not incorrect in any obvious ways, for example ensuring no segments are empty at any stage. Several of these were implemented in an error-driven way, in that whenever a bug was encountered a unit test made to ensure it did not reappear.

Due to the lack of pre-segmented images, there cannot be any form of quantitative assessment on the over-segmentation alone, hence it is very hard to verify that the result is correct. Some simple cases can be imaged where the results are easily predictable, such as an image with a known number of uniform regions. The output mask of this over-segmentation follows these criteria. Hence a result base test is implemented which checks a segmentation is preserved after being used as the image, which provides some degree of verification here.

3.2.2 SLIC Performance

As previously mentioned both the distance and combination metrics are definable by the user, thus these are hyperparameters to be selected. If one uses the defaults for both metrics (Achanta et al[1] distance metric and maximum distance combination) there is instead the spatial bias to consider. The image binning (i.e. the number of segments) is also to be chosen. None of these is analysed in detail, but changing them has relatively obvious implications, as shown in figures 3.1.

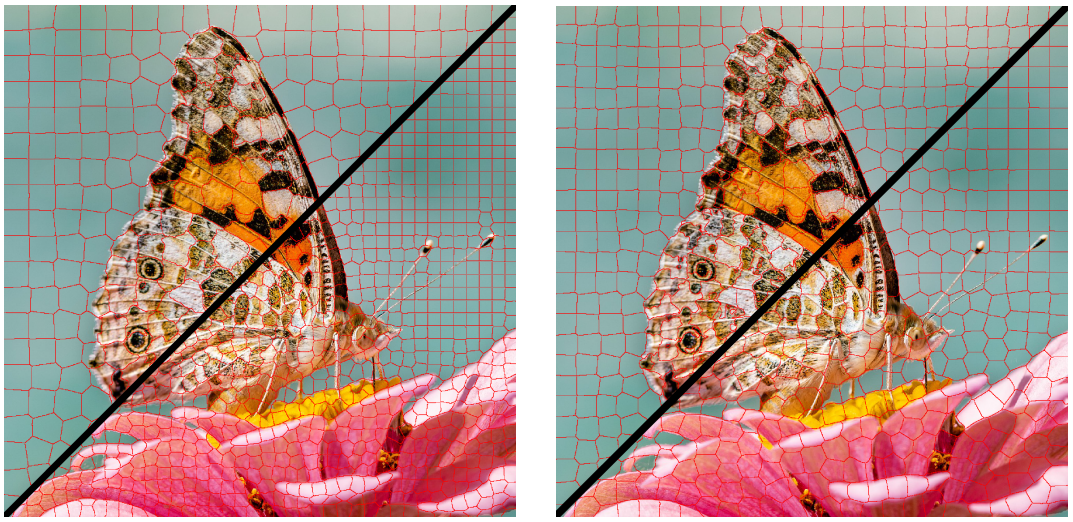


Figure 3.1: (a) Left: 10 iteration comparison of bin grid resolution (20,20) above and (40,40) below. In some of the sky regions the original grid is unchanged due to no colour variation. (b) Right: 10 iteration comparison of low spatial bias above and high spatial bias below; a lower bias allows for more complex segment shapes.

Predominant edges such as the butterfly-sky right wing edge are outlined very well, and superpixels are also able to pick out the varied patters inside the wings. However, not all features are isolated, such as the antennas. This may be a consequence of locality the restriction preventing the segment sizes becoming too small or the initial grid shifting too much. Also the gradual colour change at some boundaries prevent accurate identification, such as the butterfly-sky left wing edge. This affect is reduced by decreasing the spatial bias, though this cause a more contour like segmentation. When one considers coarser segmentation of this image into sky, flower and butterfly these issues seem less significant.

A short investigation into convergence showed that after approximately 5 iterations segment edges were not moving enough to be detected by inspection. Termination criteria by a fraction of clusters changing would be a reasonable choice here, however due to time restraints it was not implemented. Instead double this, 10 iterations, was used throughout to keep the chance of early termination low.

A key difficulty SLIC segmentation is shown in figure 3.2, where texture features near a boundary cause regions inside one segment be grouped in the neighbour segment (i.e. the neighbour segment is disjointed). This causes significant issues when it comes to detecting segment edges later. One solution is to split any disjointed segments into multiple continuous segments, then merging segments under a certain size with their largest (or in many cases only) neighbour. However, to reduce the computation required for this process it is preferable to also pre-process the image with a short-range gaussian blur, which removes the fine texture features causing this effect. This results in a few pixel shifts of some segment boundaries, which is likely to be less significant in the overall segmentation.

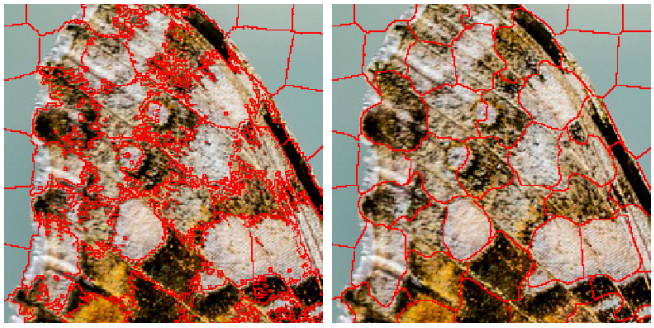


Figure 3.2: Butterfly wing tip after SLIC with the original image (left) and Gaussian blurred image (right). The fine texture details near the boundary edges are not seen in the right-hand image though the boundaries themselves are relatively unchanged.

3.2.3 MSLIC Performance

A comparison of SLIC and MSLIC under the maximum distance combination metric is shown in figure 3.3. For each image individually the SLIC algorithm gives visually a better result as it can adhere to each image separately. Also a slight offset between the different polarization images makes the MSLIC results less accurate. The cause of this image offset is not clear, but correcting this is outside the scope of this software solution (though such correction greatly improve the MSLIC segmentation). If both images features are to be used for segment merging the MSLIC segmentation is a more appropriate choice as the segmentation mask is the same for both images.

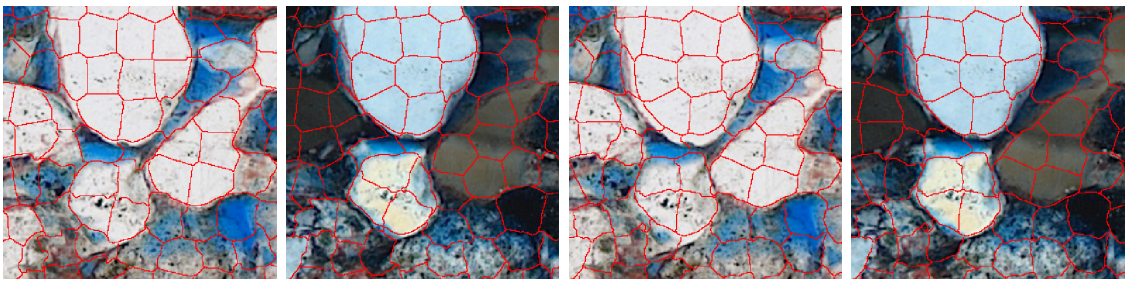


Figure 3.3: Comparison of SLIC performed on each image individually (left two) and MSLIC using both images (right two) for the same region of image. The MSLIC images have the same segmentation mask, as such are suitable for feature extraction from both images.

3.3 Segment Merging

3.3.1 Verification

Once again as there is no pre-existing segmentation it is not possible to do any form of quantitative verification on the segment merging alone. Unit tests on dummy segmentation masks ensure that the segment objects are correctly identifying their neighbours, the bordering pixels and post merging segments. The same dummy mask allows for ensuring feature extraction and edge confidence managing is correct. The AGNES routines, on the other hand, are far easier to verify with result based testing, as dummy data with a specified number of clusters and outliers can be easily made and tested with.

3.3.2 Clustering Feature Selection and Results

The main user choice here is the selection of features to cluster with. A significant issue with AGNES (or any other distance-based clustering algorithm) is how to scale the input features. Each feature needs a variation representative of its importance for material identification. If only one type of feature is compared, average colour for example, then these criteria are met as all features are pixel values, so have the same scale. However, if other feature types are combined, such as texture from Gabor filtered images[12], this is not the case, hence a clustering may be either dominated or unaffected by each feature type. Sadly one can not simply normalise each feature as then noisy or less relevant features (say the green channel which is often less predominant) are scaled up in the clustering.

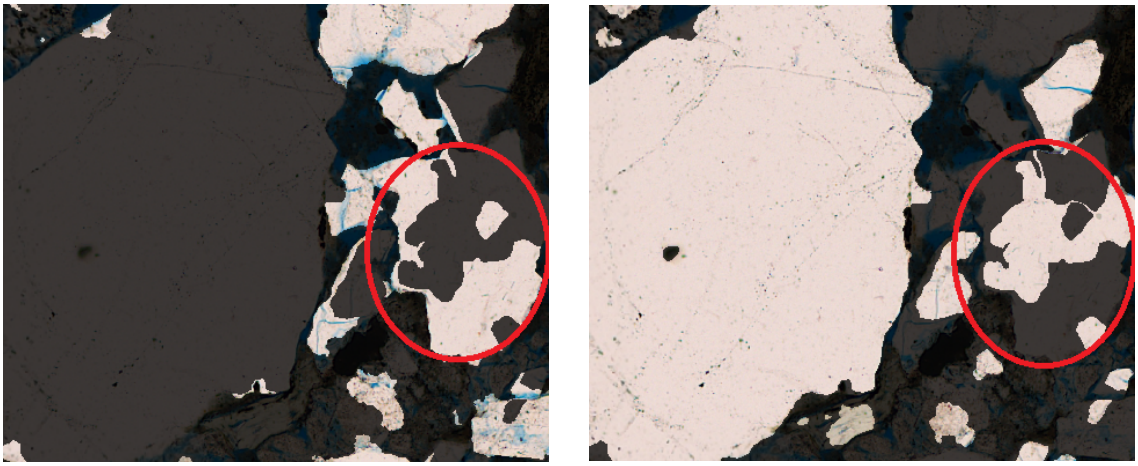


Figure 3.4: Two example clusters from a color based clustering of Californian sandstone. Outlined is a region of rock separately clustered, despite being of the same material.

The curse of dimensionality[16] must also be considered here. Adding more features tends to lead to all vectors being close together, tending to form a single large cluster with several outliers. The straightforward way to overcome this is to tune the feature scaling manually for a given image set, however, such arbitrary parameter tuning is exactly what we are trying to avoid. A different solution could be to do separate clustering for each feature type, then combine the clusters somehow (e.g. voting).

If only colour based clustering is used then there is a good distinction between pore and rock, with silt sometimes being separately clustered but often being confused between the two (see appendix A for an example). More specific distinctions prove problematic, as textures, natural colour variation and differences in the polar image often cause parts of the same grain to be clustered separately (shown in figure 3.4). As consequence grains are often under merged, meaning determined grain geometry distributions are inaccurate.

3.3.3 Edge Confidence and Results

The other user choice is the edge confidence calculation. This also proved problematic, as the segments with clear edges are often indiscernible compared to rock textures in the image (see figure 3.5a). As such the use of edge confidence in merging often prevents segments with high contrast textures form merging without preventing seperate grains of the same material from merging.

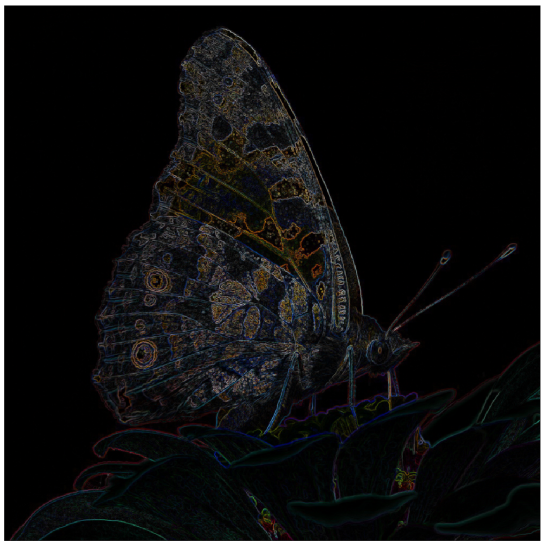
An example of using calculating edge confidence is shown in figure 3.5. For regions with few textural features such as the sky and flower petals, this method identifies most merging segments well. Inside the wing are several regions that that incorrectly determine an edge to be present, such as the central orange region. It is possible that the right combination of filters and calculation of edge confidence could overcome this, as such the developed software would be helpful for easy experimentation of this. This issue is a significant problem in TS images as they have most regions have texture.

3.4 Property Determination

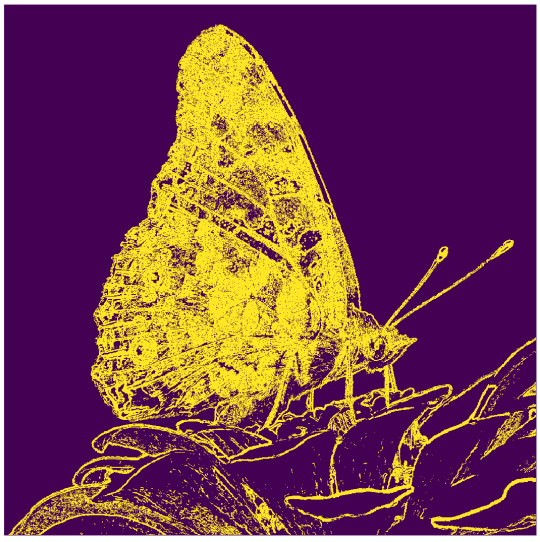
The determination of material properties from thin sections is the only form of quantitative validation currently available for this software. Since grain distributions would not be accurate here, coarse labelling is used to obtain porosity values only. For each sample three random $10^3 \times 10^3$ pixel sections were processed independently. As only porosity is being found adjacent grains of the same material can be merged, hence there is no need for edge confidence calculations.

For the labelling stage, there was an average of 14 clusters to be labelled, which took no more than 2 minutes per image due to the use of python's input function

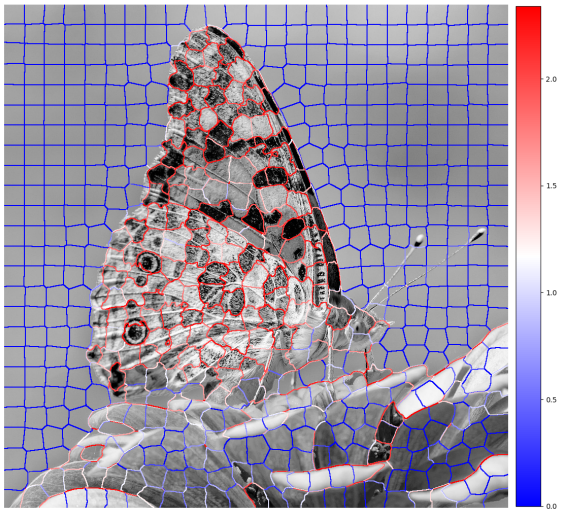
Figure 3.5: An example sequence of merging by edge confidence



(a) Scharr edge filters[2] on each color channel independently. Texture features are more predominant than the desired edges.



(b) Grey scale of figure 3.5a made binary on a threshold of 0.05.



(c) Edge Confidence where confidence is defined as the fraction of segment edges that overlap with figure 3.5b, then scaled by the standard deviation of all segments edges.

(d) The resultant segmentation from figure 3.5c with merging threshold 0.5. Textured regions such as those inside the butterfly wing are not merged.

to create a basic user interface. There is some issue with ambiguous clusters due to silt segments often being clustered with pores, largely due to pore/silt features too small to be segmented separately leading to a clustering of mixed regions. Similarly there are pores too small to be segmented between rock grains. Qualitatively there is overall a good distinction between rock and pore, as shown in figure 3.6.

Determined porosity values along with the feature and hyperparameters used are given in figure 3.7. In many ways, these results are premature, as there is still much improvement required for the method. However they are not promising either; the

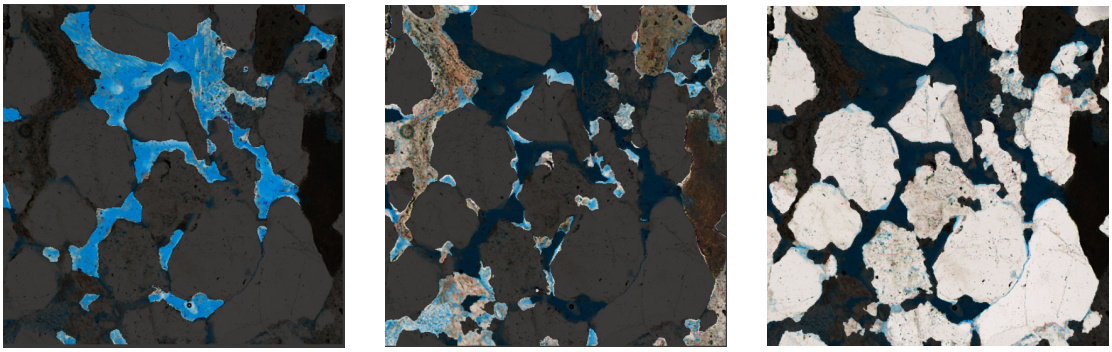


Figure 3.6: Thailand coarsely labelled clusters into pore (left), silt (middle) and rock (right). Some silt regions have significant porous space present, making it ambiguous which label it should be given.

general disagreement could be down to several factors, including:

- the number and size of sample images being too small to correctly represent the overall image
- correction factors may be needed between measured porosity and TS image porosity
- ambiguous user identification of some silt-pore clusters
- pores too small to be seen at the chosen bin grid resolution (would also explain the tendency to under estimate)
- average colour features are not enough to full distinguish pore, silt and rock

Figure 3.7: Determined porosity for regional sandstones compared to experimental values. The specific process included: 3 pixel gaussian blur on the initial image, $\gamma = 0.5$ spatial MSLIC bias, 40×40 pixel bin grid, disjoint segments split with 50 pixel minimum segment size, AGNES clustering at 3σ using features of average channel color in each image (6 total).

Region	Avg	Std	Actual	Diff
Indonesia	28.41	3.25	22.0	6.41
Gabon	13.3	0.98	20.7	-7.4
Norway	14.14	0.77	21.9	-7.76
Texas	16.05	2.38	21.0	-4.95
Thailand	19.51	4.66	19.6	-0.09
California	10.9	4.33	15.1	-4.2

It is highly likely that they could be improved by a better choice of clustering features and larger images (or more samples). Whether in an optimal configuration it would be accurate enough for the intended use is still not clear.

Chapter 4

Discussion and Conclusions

4.1 Discussion

The SLIC and MSLIC implementations give reasonable segmentation results, with difficulties mainly related to computational efficiency, since this is the most demeaning algorithm in the developed software. Whilst these have been reduced through the use of vectorised operations, the run time will likely prove problematic if significantly larger sections of the TS image are to be analysed. As such focusing on improving this, either by parallelisation or reducing code inefficiencies should be considered for future work.

The segment merging infrastructure has the desired functionality and adaptability. This leads to the significant issue of choosing the appropriate way to use this versatility for the intended TS images. Various aspects of the clusters formed here show promise, such as reasonable groupings of porous space, however, there are significant issues with clustering of textured regions. Further experimentation of image features, with particular focus on trying to incorporate texture-based features, is needed. The current software is well suited for such an investigation due to it's range of analysis tools, particularly visulisation methods. A similar investigation should investigate edge features, as determining the edge of grain these textures proved equally problematic. Further potential may lie a supervised learning approach, though this would require manual segment labelling to obtain a training dataset.

These merging issues make determination of segment geometry distributions unobtainable and result in inaccurate quantitative porosity values on the tested TS images. Other factors such as sample size or hyperparameter selection will also contribute to this inaccuracy. Considering these images are a small subset of what should be analysed by this software it is uncertain whether the desired TS automation can be achieved in the manner set out here.

Overall the software is flexible and well compartmentalised, likely making of use for either current TS analysis or future software development. However, this flexibility comes with some undesired effort on the users' behalf. In particular, the how to define the extraction functions needed may not be immediately obvious to a user, hence documentation and a user manual with examples has been published.

4.2 Conclusion

The desired software outlined in figure 1.2 has been implemented, in a manner that is highly adaptable to the user and suitable for future development. Under preliminary investigation, some elements perform well such as the SLIC and MSLIC over-segmentation, whereas the unsupervised segment clustering and edge detection sections show a significant need for improvement. Further investigation in these regions is needed, particularly concerning feature selection and hyperparameter tuning. The existing software is well suited for future investigations due to its adaptability and range of analysis features.

Early results are inconclusive about the viability of unsupervised clustering to give the desired Thin Section segmentation. If this proves not possible, the compartmentalised nature of the software will likely make it useful in the development of future TS analysis tools. This is especially true for the SLIC and MSLIC implantation, as these have shown good results and reasonable computational efficiency.

Bibliography

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 2274–2282, Nov 2012.

[2] F. Jiang, Q. Gu, H. Hao, N. Li, B. Wang, and X. Hu, "A method for automatic grain segmentation of multi-angle cross-polarized microscopic images of sandstone," *Computers & Geosciences*, vol. 115, pp. 143 – 153, 2018.

[3] M. Raith, R. P, and R. J, *Raith M. M., Raase P. and Reinhardt J. (2012). Guide to Thin Section Microscopy. Second edition, 127 pp. e-Book ISBN 978-3-00-037671-9 (PDF).* Open Access Publication, 01 2012.

[4] C. Reedy, J. Anderson, T. J. Reedy, and Y. Liu, "Image analysis in quantitative particle studies of archaeological ceramic thin sections," *Advances in Archaeological Practice*, vol. 2, pp. 252–268, 11 2014.

[5] E. Berrezueta, M. J. Domínguez-Cuesta, and Ángel Rodríguez-Rey, "Semi-automated procedure of digitalization and study of rock thin section porosity applying optical image analysis tools," *Computers & Geosciences*, vol. 124, pp. 14 – 26, 2019.

[6] M. Młynarczuk, A. Górszczyk, and B. Ślipek, "The application of pattern recognition in the automatic classification of microscopic rock images," *Computers & Geosciences*, vol. 60, pp. 126 – 133, 2013.

[7] G. Cheng and W. Guo, "Rock images classification by using deep convolution neural network," *Journal of Physics: Conference Series*, vol. 887, p. 012089, aug 2017.

[8] N. Li, H. Hao, Q. Gu, D. Wang, and X. Hu, "A transfer learning method for automatic identification of sandstone microscopic images," *Computers & Geosciences*, vol. 103, pp. 111 – 121, 2017.

[9] Free to use research development environment hosted by google, available at `https://colab.research.google.com/notebooks/welcome.ipynb#recent=true`.

[10] P. Domingos, *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World.* New York, NY, USA: Basic Books, Inc., 2018.

[11] G. Kumar and P. Bhatia, "Impact of agile methodology on software development process," *International Journal of Computer Technology and Electronics Engineering (IJCTEE)*, vol. 2, pp. 2249–6343, 08 2012.

[12] O. Demir and B. Dogan, "Unsupervised image segmentation using textural features," *International Journal of Signal Processing Systems*, vol. 5, pp. 112–115, 09 2017.

[13] G. Brandl, *Sphinx Documentation Release 1.8.0.* Sphinx Python Documentation Generator, Sep 2018. available at urlhttp://www.sphinx-doc.org/en/master/intro.html.

[14] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. a. Yu, "scikit-image: image processing in python," *PeerJ*, vol. 2, p. e453, June 2014.

[15] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

[16] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Series in Statistics, Springer, 12 ed., Jan 2017.

Appendices

A Solution Skeleton with Images

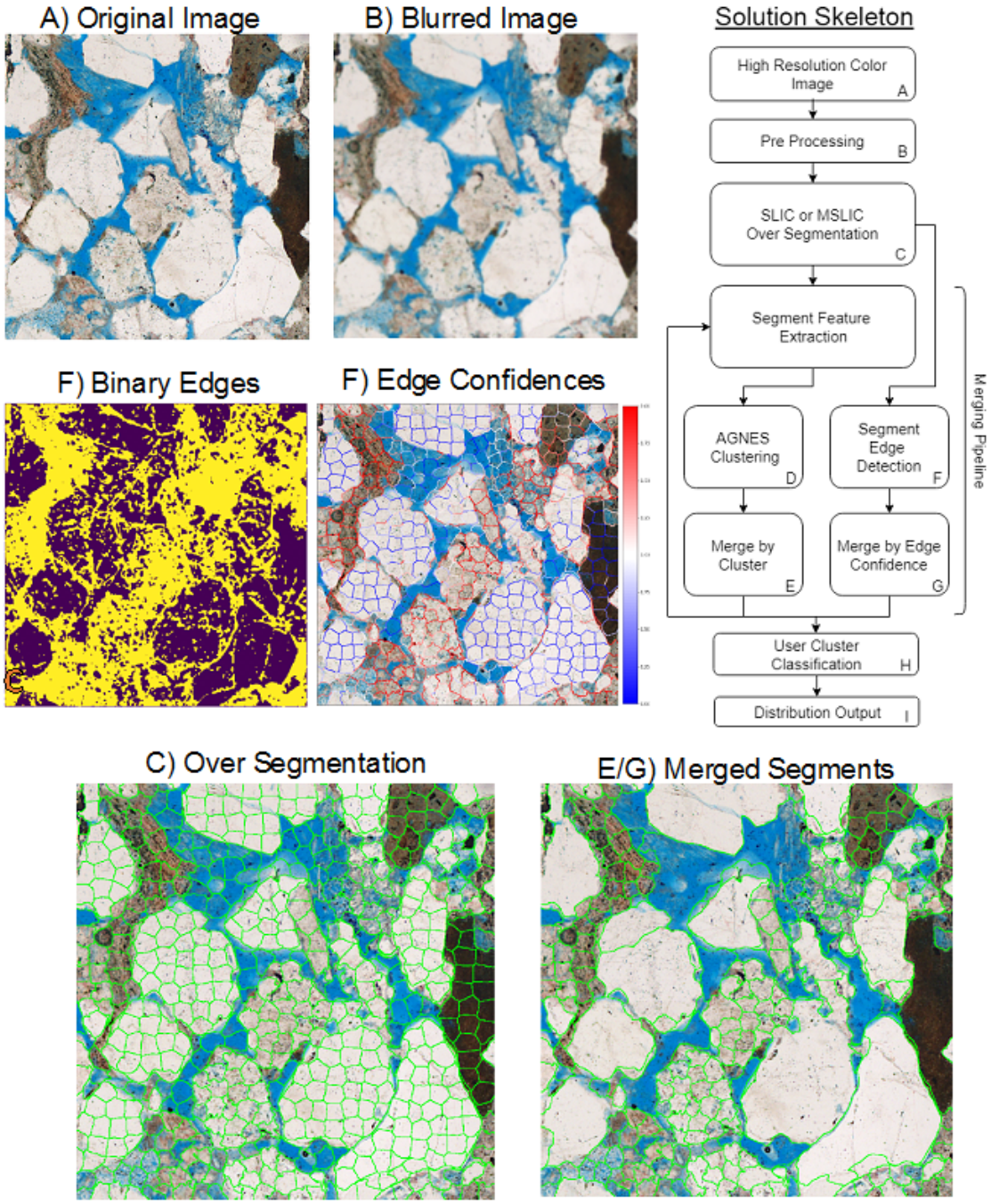


Figure 1: The solution skeleton outlined in section 1.4 now with images showing the process at every stage (note images are not in order due to irregular size and some are in figure 2).

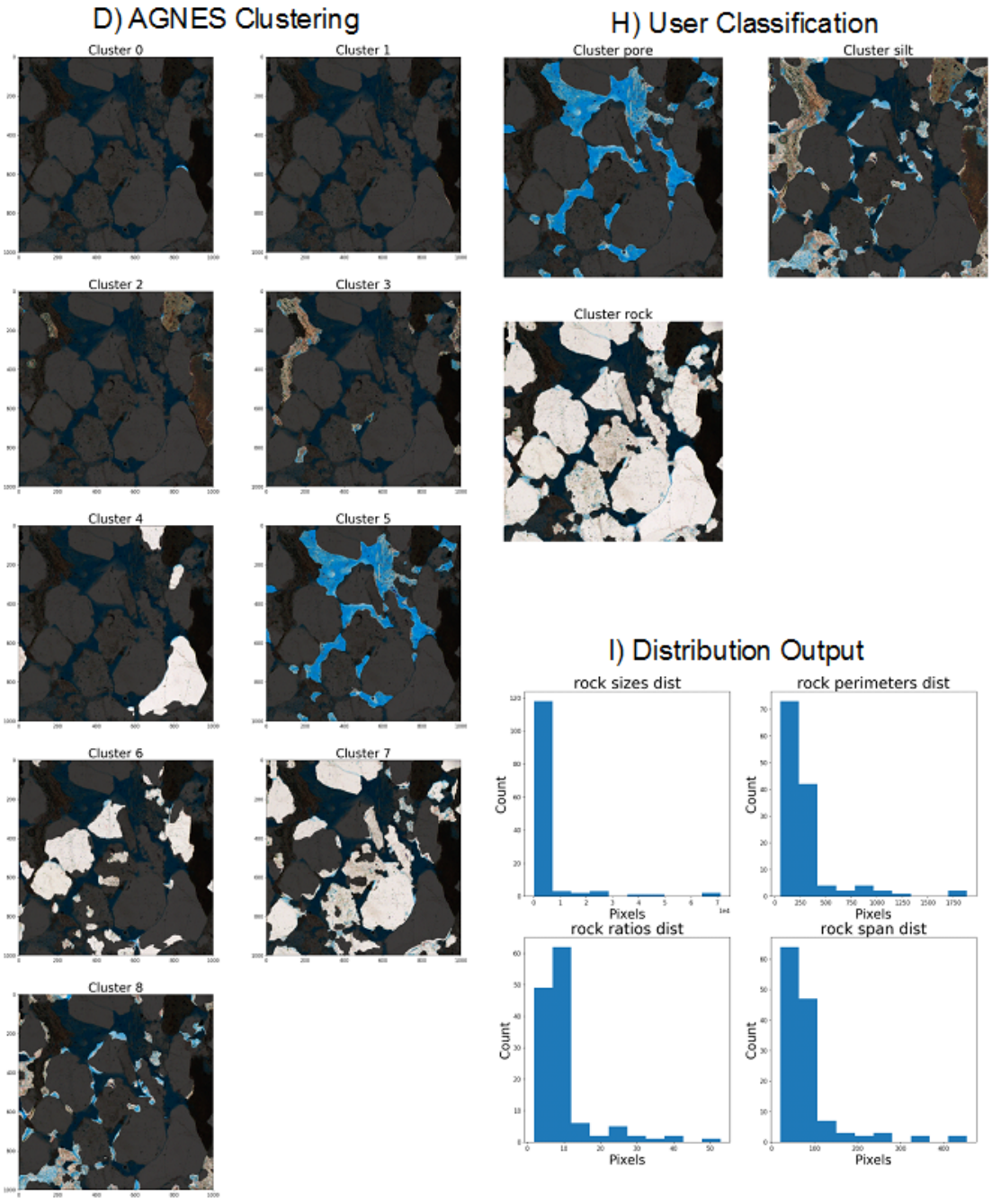


Figure 2: Solution images process images as in figure 1. Interest should be shown the AGNES clustering and the regions of confusion there such as pore and silt being grouped in the bottom left.

B Sample Thin Section Sandstone Images

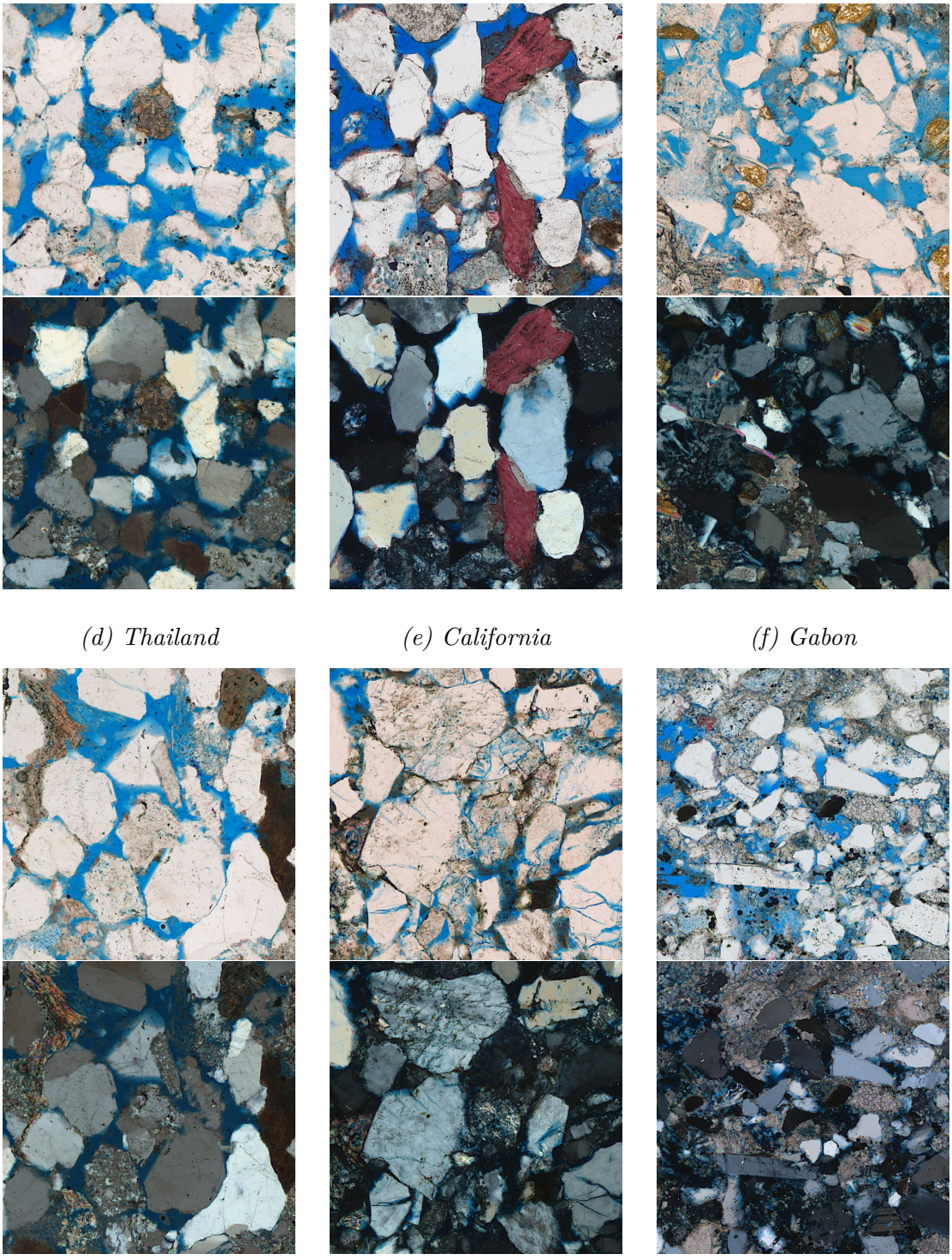


Figure 3: Samples of the TS image set from each region with parallel polarisers (above) and cross polarisers (below). From each region there are two more of each image type that is not shown here.