Identifying the Causes of Alzheimer’s Disease within Brain Sections

2016

UNIVERSITY OF
LINCOLN

Jacob Carse

CAR11354616

BSc. (Hons) Computer Science

School of Computer Science

University of Lincoln

Supervisor: **Professor Nigel Allinson**

Acknowledgements

This Project would not have been as successful as it is if it without the loving help of those who have aided me throughout the course of this project.

- My loving girlfriend Tasnim Hassan for her outstanding patience, her loving confidence boosts and her willingness to help proof read through every page of this extremely long document.
- The late night Computer Science students that shared a similar late night working regime and helped keep each other going throughout the late night lab sessions.
- The music of Anamanaguchi and Girl Talk as their loud beats helped me stay focused and drown out all distractions.
- Kieran Allinson for obtaining the testing images which have been used throughout the course of the project.
- Last but not least, Professor Nigel Allinson who assisted in inspiring my new found love of Computer Vision and has made this project a joy to work on.

Abstract

To diagnose Alzheimer's disease a pathologist has to go through the timely and bias process of identify plaques and tangles within the Hippocampus. In order to speed up this process and standardise the diagnosis, a system to identify plaques and tangles needs to be built. This has been achieved using a Visual Bag of Words and a Gaussian SVM classifier which utilises SURF feature extraction to classify both the plaques and tangles. Unfortunately because of limitations with accessing labelled image data, the system has yet to be fully evaluated. However with a small image set with estimated labels, the system was able to classify plaques and tangles with an accuracy of 88%.

Contents

List of Tables and Figures

1. Introduction

According to the a report by the House of Commons Library Research (2010, 44-45), there are approximately 10 million people over the age of 65 living within the United Kingdom and it is estimated to increase to 19 million by the year 2050. As this population of over 65s increases, there will be a greater demand for treatments and cures for age associated diseases such as Osteoarthritis (joint pain), Osteoporosis (weak bones) and, namely, Alzheimer's (a form of dementia). Alzheimer's can be especially concerning for aging population as it can interfere with their ability to perform basic daily tasks; unfortunately it is difficult to diagnose and treat. Alzheimer's disease can only truly be diagnosed via analysis post mortem brain sections of the Hippocampus (Alzheimer's Association, 2015). This has put greater pressure on medical research groups to develop new methods of diagnosing Alzheimer's earlier.

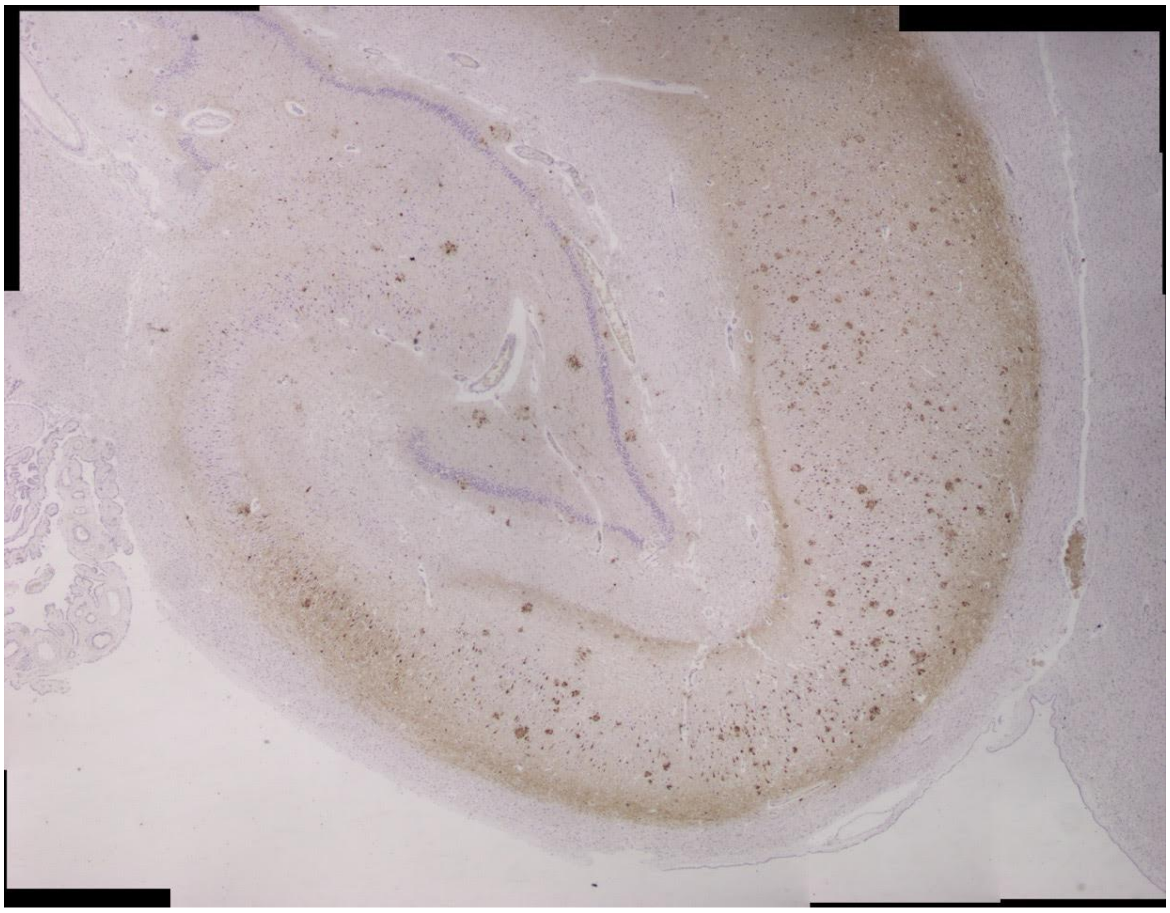

Figure 1. A brains slice showing a Hippocampus with stained plaques and tangles in the grey matter from (Allinson, 2015).

Within the Brain Bank at Addenbrooke's Hospital, operated by their Pathology Department, there consists a large number of donated post-mortem brains. The Pathology Department has the responsibility to analyse each of these brains to find any signs of disease such as Alzheimer's. Alzheimer's can be diagnosed by identifying senile plaques and neurofibrillary tangles within the grey matter of the Hippocampus. Senile plaques are made up of leaked deposits of Amyloid Beta, an amino acid, within the grey matter of the Hippocampus. Senile plaques have been linked to degenerative neural structures as shown in Cras et al. (1991). Neurofibrillary tangles are aggregates of tau protein and are the primary marker of Alzheimer's (Braak and Braak, 1995). These plaques and tangles can be identified by slicing into the Hippocampus then using medical staining to artificially colour the tau proteins within the grey matter to make them easily identifiable. An example of stained plaques and tangles within the Hippocampus can be seen in Figure 1.

Once the staining process has been completed, it allows researchers to analyse the scans of the Hippocampus so that Alzheimer's disease can be diagnosed. Given that analysing brain scans for plaques and tangles is a qualitative process, different analysers can give varying results. This process of having human analysis on brain sections for plaques and tangles can be a long process that may be prone to errors. Other medical imaging tasks have already been automated to greater success. An example of this can be seen in the identification of Diabetic Retinopathy Stages in the Fundus; by using an automated identification system of which the accuracy rate was able to reach 93% (Nayak et al., 2008). This leads to the belief that automating the task of analysing brain sections for plaques and tangles would be beneficial to researchers and other medical professionals as it would save them time and guarantee a degree of accuracy.

The initial evolution of the system used Feature Descriptors such Circular Hough Transformation (Illingworth and Kittler, 1987) that examines the circularity of a feature and the Generalised Hough Transformation (Ballard, 1981) which detects matching features with templates. These descriptors are used to classify the plagues and the tangles within an inputted image. This method is similar to the method performed by Smereka, M and Dulęba, I (2008) where they used a modified Hough Transform to detect low contrast circles to detect cell nuclei in cytological smears. This was found to be ineffective for this system as plaques and tangles have similar features which will cause the system to produce a large amount of false reading when inputting images into the system. This constitutes that a more accurate method of classification was needed in order to produce a system that can identify plaque and tangles.

After the first evolution of development, a new approach was taken which used a form of template matching whereby a template would be created by using the algorithm featured in Bagon et al.'s (2010) paper "Detecting and Sketching the Common". This algorithm would generate an image based on the common features of the plaques and tangles within the image. This would constitute that the more examples the system has to train, the more generic the template would be. The template would then be used another other algorithms to detect similarities across different images. The chosen algorithm can be seen in Shechtman and Irani's (2007) conference paper "Matching Local Self-Similarities across Images and Videos". This algorithm looks at structural features within images and uses correlation surface in order to check for the similarity between images. The evolution of the system didn't prove to be successful as there were issues with their implementation. Implementing the algorithms was an option but due to limited time constrains was not feasible for the project.

The next evolution of the system looks at a form of machine learning by training a classifier which examines common features within a given image set. The system uses a Bag of Words Classifier (Dance et al., 2004) which looks at common features found by using the feature extraction algorithm SURF (Bay et al., 2008). The Bag of Words classifier then examines inputted scans and predicts the classification of the features within the scans. Utilizing a Bag of Words model within medical imaging is a method which has been proven to work before. For example "Using a Bag of Words for Automatic Medical Image Annotation with a Latent Semantic" by Bouslimi et al. (2013) where they use SIFT extraction to find features within images and automatically classify them using a Bag of Words classifier.

Once the system has been built, it will be able to highlight the plaques and tangles within the image and also display a count. The input would have to be in the form of a zoomed in scan of the Hippocampus, an example of this can be seen in Figure 2. This count of plaques and tangles will be able to aid researchers and other medical professionals in the diagnosis of Alzheimer's making better use of their time.

The previously described system which had been produced was unable to be evaluated because labelled test data used to train and then validate the classifier was not available before the deadline for the project. This meant that the project was evaluated with test images where the labels for each of the features were estimated by the author rather than a medical professional. From this estimated evaluation testing, the system against a validation set found the system had an accuracy of 88%, although the system correctly classifies tangles at a rate of 33%, presenting a number of false negatives. These false negatives could lead to someone not being diagnosed despite having Alzheimer's disease. This leads us to the belief that in order to more accurately classify tangles from noise, the training data would need to be correctly labelled and a larger image set should be used.

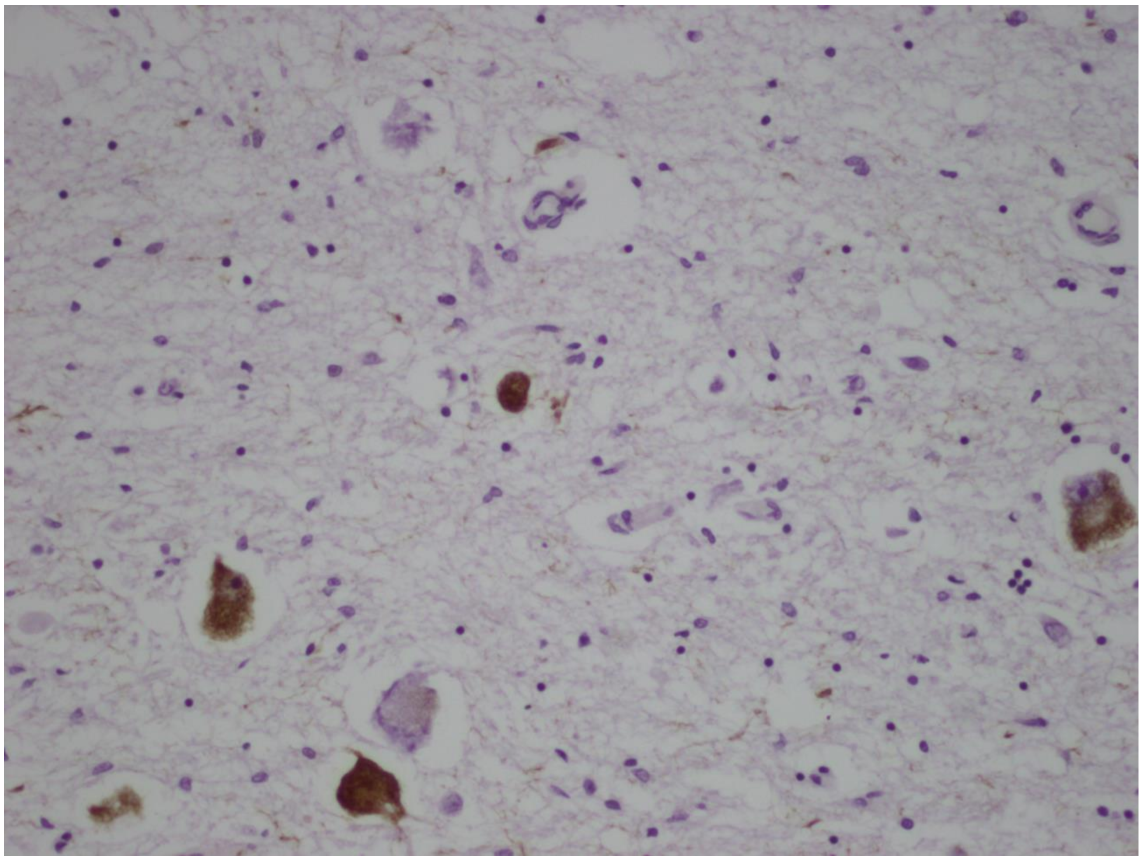
Figure 2. A close up view of the grey matter showing the stained plaques and tangles among other healthy cells from (Allinson, 2015).

2. Literature Review and Background Research

2.1. Literature Review

2.1.1. Alzheimer's

2.1.1.1. Alzheimer's in Society

Within countries with high GDPs, the population of people aged 65 and over has been consistently rising. This rise can be visualised within Table 1, where it is estimated that the population aged 65 and over living within the United States of America will nearly double in size over the next 38 years. The human process of aging can lead to a number of medical complications such as Atherosclerosis, Arthritis and Alzheimer's, these conditions can make life harder for those suffering. These complications are difficult to avoid as they are a by-product of aging and as the elderly population increases, so does the demand on medical services to treat these conditions. "The Aging Population and Its Impact on the Surgery Workforce" is an article from Etzioni et al. (2003) which illustrates that the population aged 65 years and over are 52.59% more likely to require a medical procedure. The article upholds the recommendation that more specialised doctors should be trained in order to meet the predicted demand and further, that more money is too distributed to medical research to find more efficient ways to perform these medical procedures.

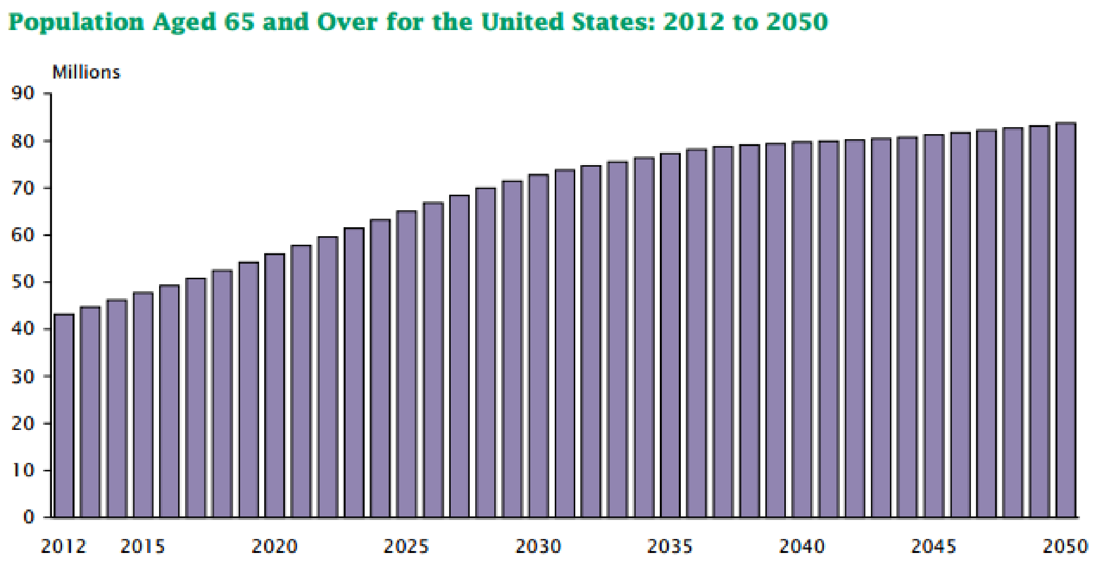

Table 1. A table showing the rising aging population in the United States from (U.S. Department of commerce, Economics and Statistics Administration et al., 2014, p2).

The article "Heart Failure and the Aging Population: An Increasing Burden in the 21st Century" by Stewart et al. (2003) also shows a similar trend in Scotland. The article claims that between 2005 and 2020, the number of the people over 45 years old that will die, as a result of heart failure, is estimated to rise by 39%. Stewart et al. also states that the average number of GPs visits and the general use of medical facilities will also significantly increase. A common theme between the two articles is that they both claim there will be a much greater strain on health services because of the increases in the ageing population. Therefore in order to loosen the strain on health services, more doctors and services would be required to meet these new demands or more efficient technologies would need to be developed to aid the current services.

2.1.1.2. Diagnosis

Diagnosis of Alzheimer's disease can be a strain on medical resources as its diagnosis requires a pathologist to analyse brain scans and can be a long and tedious task. This waste of time and resources could be used in other situations which could help benefit others. The diagnosis of Alzheimer's is described in Ishizuka et al. (1997) and requires pathologists to examine, at high resolution, images of post mortem brain sections and identify deposits of amyloid beta in the hippocampus's grey matter called senile plaques. They will also need to identify neurofibrillary tangles made up of twisted proteins. In order to reduce this pressure upon medical professional, the task can be automated and thus freeing up their time. Computer Vision can be used to automate the task of identifying the plaques and tangles within the high definition medical scans of brain sections. An example of the plaques and tangles which need to be identified can be seen in Figures 1 and 2.

In order to identify the plaques and tangles, a Computer Vision system can be implemented which utilizes the wide range of Computer Vision, techniques, tools and algorithms that are currently available. Braidy et al. (2014) conducted a study of a similar nature where they researched the identification of metal dyshomeostasis at the molecular level in the grey matter of the brain. They were able to accomplish this goal by using a combination of Computer Vision techniques including colour slicing. By utilising colour slicing, they were able to identify the presents of metals such as iron, copper and zinc within the grey matter in microscopic scans of brain sections. This system uses the standard approach to Computer Vision systems and can be divided into the stages, Pre-Processing, Segmentation, Feature Detection and Pattern Classification.

2.1.2. Pre-Processing

Pre-processing is a stage in Computer Vision where images are processed so they become better suited for complex operations to be performed on them.

2.1.2.1. Intensity Transformations

Intensity transformations such as Contrast Stretching and Histogram Equalization are Pre-Processing techniques. These techniques are used in order to make features appear clearer and easier to identify within the image. Intensity Transformation techniques have been implemented in the article by Vovk et al. (2007) as they were trying to improve the quality of intensity inhomogeneity MRI scans. The team tested a number of different Intensity transformations to find one that produced the best result for the MRI scans. This study helped reduce the error rate when analysing MRI images in critical situations, such as, medical diagnosis where inaccuracies can have serious consequences.

2.1.2.2. Morphological Transformations

Morphological Transformations are operations that are performed on binary images as a form of Pre-Processing. Morphological Transformations can be used to erode or dilate a given binary image based on a structuring element, this process has been demonstrated in Figure 4, where a simple binary image is eroded then dilated. Figure 4 also shows how Morphological Transformations can be utilised in noise removal. Morphological operations have been proven to be an effective method to reduce the amount of noise within binary images, as shown in the article by Jamil et al. (2008) where they state “noise can be effectively removed from binary images using combinations of erode-dilate operations.” This shows that Morphological Transformations are one of the most efficient ways of removing noise within binary images.

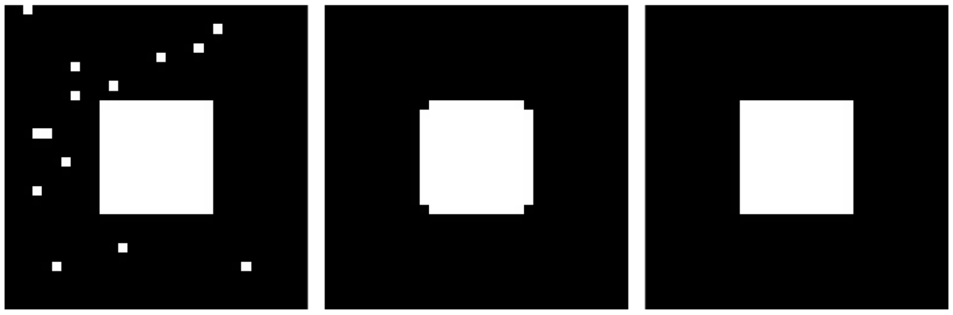
Figure 4. Morphological operation close showing the erosion and dilation steps (Gouillart and Varoquaux, 2011).

2.1.2.3. Filtering

Filtering is a form of Pre-Processing that uses kernels which are applied to the image to add effects such as blurring, sharpening or to detect edges (used to extract features). Noise within images will make it difficult for Computer Vision systems to perform complex operations on, such as Feature Extraction and even small amounts of noise can lead to inaccuracies. The article by Rudin et al. (1992) show that in order to remove certain types of noise from different sources of noise, specific algorithms which are best suited for removing specific types should be used. Rudin et al. have demonstrated this in their study as they produced an algorithm which is best suited for removing variation based noise from images.

The Pre-Processing technique Edge detection can be used to both sharpen images and extract regions from images. An example of sharpening being applied can be seen in the article from Ahmed and Nordin (2011). In this study, they were able to sharpen X-rays of spines to make the features within the images easier for both humans and Computer Vision system to identify. The resulting image of their study can be seen in Figure 3.

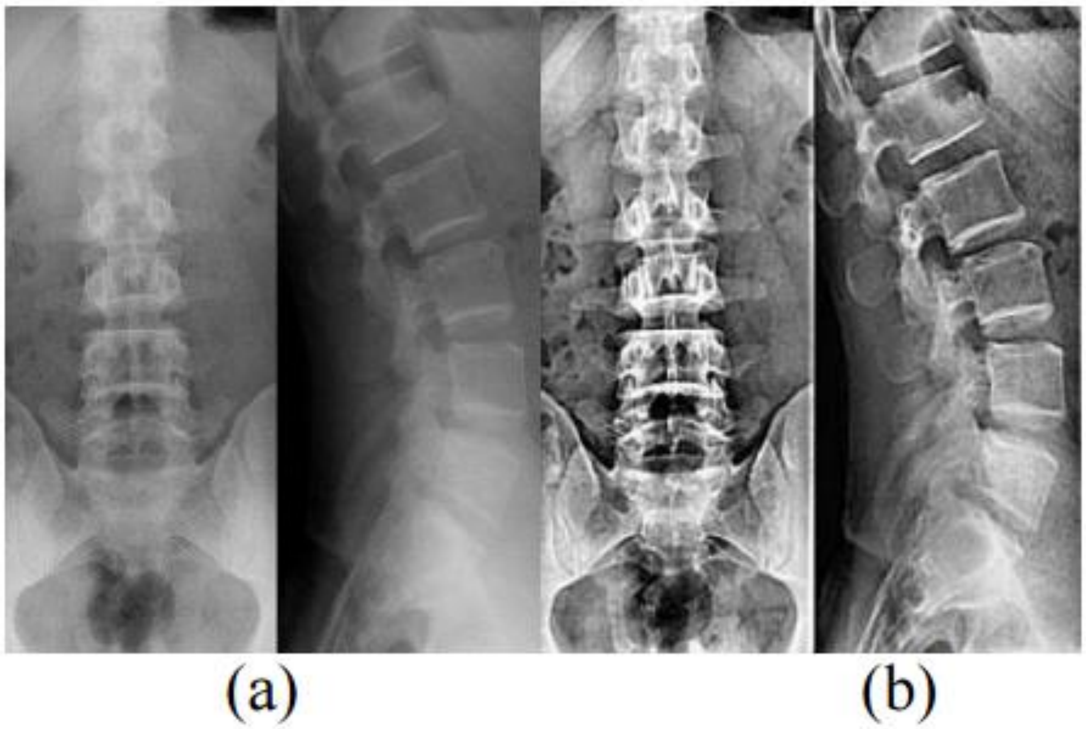

Figure 3. X ray of a spin (a) unsharpened image (b) sharpened image (Ahmed and Nordin, 2011).

2.1.2.4. *Connected Component Labelling*

Connected Component Labelling is a Computer Vision technique, used within Pre-Processing, which is able to detect connected regions within binary image. Connected Component Labelling has been used within the field medical imaging, as seen in the article by Haris et al. (2001) where they were able to locate coronary artery skeletons in the heart. They were able to do this by using a thresholding algorithm to produce a binary image and then perform Connected Component Labelling to find the largest component which was the coronary artery skeleton.

2.1.3. Image Segmentation

Image Segmentation is a Computer Vision process which extracts only the relevant parts of images, allowing a system to focus only on these parts. Segmentation is a difficult process to automate, and automated Segmentation algorithms can be extremely complex and inaccurate. In order to achieve higher quality results, human interaction is required to select the segments but this can be tedious when processing large amounts of images or segmenting multiple features within an image (Liu et al., 2011).

2.1.3.1. Colour Slicing

Colour Slicing is used to segment images based on the colours of each pixel, extracting only the pixels between the certain colour ranges. This technique has been used to extract stained cells from medical images as seen in the article "Colour Thresholding and Objective Quantification in Bioimaging" (Fermin et al., 1992). Fermin et al. describe how the stained cells of interest stand out against the other cells within the image and where Colour Slicing could be performed, leaving only the cells of interest within the image. This shows how useful Colour Slicing is in the field of medical imaging, especially when working on the molecular level where cells of interest can go through a process of staining which makes them stand out as shown in Figure 2.

2.1.3.2. Edge Based Segmentation

Edge based segmentation can be performed on images by first applying an Edge Detection filter such as Canny, Prewitt or Sobel to return the edges of the image. The pixels within the image which are not separated by an edge are added to the same category, segmenting the image into several sections. Edge Based Segmentation is used throughout medical image such as in the article by Tang et al. (2000) where they were able to use the Canny Edge Detector to extract the regions from MRI scans. There are many different edge detectors which are available but Canny was chosen by Tang et al. because it features good localization. A comparison of Edge Detectors was conducted by Maini and Aggarwal (2009) and they found that the Canny Edge Detector to be the best in all situations "The Canny's edge detection algorithm performs better than all these operators under almost all scenarios."

2.1.4. Feature Detection

Feature Detection is the process of identifying features within images and then describing them in the form of some sort of descriptor. The article from Chaudhuri et al. (1989) shows that when a Computer Vision system uses a Feature Detection algorithm, different algorithms are need to be tested against the input to find the most suitable algorithm. In the article, Chaudhuri et al. tests different algorithms to find the best method suited for their system to identify retinal blood vessels. Chaudhuri et al. implemented their own Feature Detection algorithm that works based on directional edges to identify the retinal blood vessels. Feature Detection can be based on features such as at scale, texture, colour, etc.

2.1.4.1. Circular Hough Transform

The Circular Hough Transform can be used to identify circular shapes such as cells within images. This process can be seen in the article by Solainman et al. (1998) where the Aorta, one of the main arteries from the heart, is identified within ultrasound images by finding the feature with highest curvature. The authors found that this was an effective method as the authors described "the resulting circular approximation of the aorta is of high precision".

2.1.4.2. Generalised Hough Transform

The Generalised Hough Transform (Ballard, 1981), a modification to the original Hough Transform, is capable of detecting known shapes by using template matching. In an article by Kassim et al. (1999), several comparisons of different Feature Detection algorithms was conducted and found that overall the Generalised Hough Transform was a robust algorithm for finding shapes. It was further able to detect the shapes with deformities when noise was present. Although they found that the Generalised Hough Transform did not handle changes in orientation and scale very efficiently and could lead to a long processing times as the template needs to be scaled and rotated each time it is checked. The Generalised Hough Transform has been used in medical imaging as shown in the article from Philip et al. (1994) where they use the Generalised Hough Transform, combined with Fuzzy set theory in order to retrieve shapes of organs.

2.1.4.3. SIFT

The SIFT algorithm (Lowe, 1999) works by first detecting features by Finding Scale-Space Extrema, and then performing Key point Localization and Filtering. The algorithm finds local features throughout the image then each feature is described by assigning orientation and removing the effects of rotation and scale, then generating a descriptor based on a histogram of orientations. SIFT has been used within a Medical imaging environment. This can be seen in the article by Zhi et al. (2009) where SIFT was used to find features within medical scans that were used to identify regions. This shows that two sets of SIFT descriptors from two different images can be compared to check the similarities between the images. An example of this can be seen in Figure 5 where the matching features from two images taken in the same location is visualised.

The SIFT algorithm is highly regarded within the Computer Vision community for having a high accuracy for identifying and describing features while still staying reasonably efficient. This can be seen in the blog post by Mailsiewicz (2015) where he states "The world of computer vision research changed almost overnight". It has been stated that the SIFT algorithm may be inaccurate when comparing two images with different lighting conditions. This should not be a problem within a medical imaging environment as lighting conditions during image acquisition do not change under normal circumstances as explained in the article from Carlevaris-Bianco and Eustice (2014) "descriptors such as SIFT have limited lighting invariance".

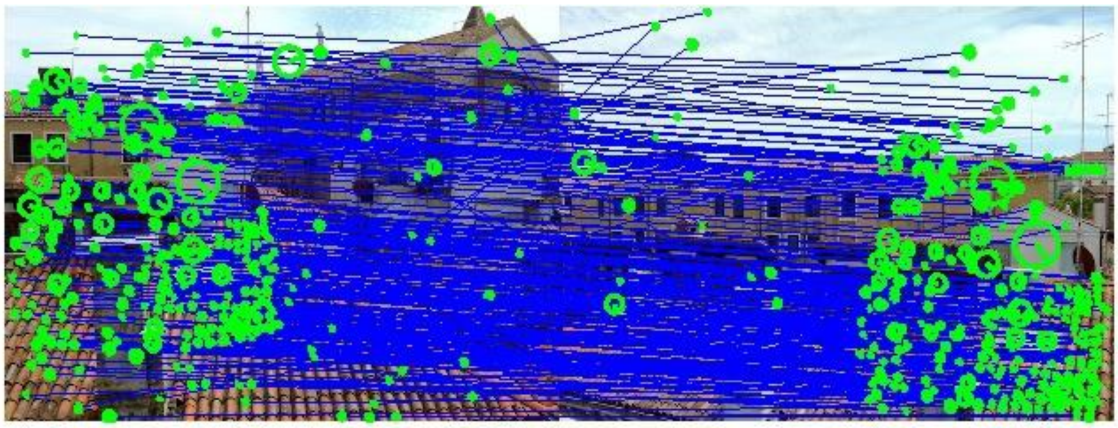

Figure 5. Matching SIFT features within two images (VLFeat, 2013).

2.1.4.4. SURF

The Speeded Up Robust Features algorithm (SURF) (Bay et al., 2008) is a Feature Extraction algorithm that works with a similar method to SIFT but is designed to be more efficient. This can been seen in the article by Panchal et al. (2013) where they compared SIFT and SURF. They concluded that SIFT was able to detect more features with just less than double the matching points than SURF, although during the tests the SURF algorithm ran over three times faster than the SIFT algorithm to find the features. This thus gives rise as to why SURF is used within the field of Robotics and other applications that use live image feed, as computation may be valuable. Within a medical imaging environment, most work is performed on static images and rely very little on live image feeds meaning that SIFT could be better suited for a medical imaging.

SURF has been used within Medical Imaging, for example, in the article by Lukahevich (2011) where they used the SURF algorithm on CT images so they can register them. One of the reasons that SURF is used over SIFT is that SIFT has been patented by its creator, meaning that the use the algorithm within a commercial product requires payment to the creator. SURF is freely available for use and is used throughout many products for this reason, for example, MATLAB only includes an implementation for SURF rather than SIFT.

2.1.4.5. Self-Similarity Descriptors

In the article from Shechtman and Irani (2007), they describe their approach for matching images and videos using internal self-similarities. Their method works by analysing the patterns within a given scene to generate a Self-Similarity descriptor that can then be used to compare the content within images. This method has been used in order to detect certain events in crowded videos as described in Ke et al.'s (2007) article "Patterns can be located anywhere in the scene (in both space and time)". This shows that using Self-Similarities to compare images is a robust method and is able to describe patterns within images.

2.1.4.6. *Sketch the Common*

Sketch the Common is a Feature Detection algorithm from the paper “Detecting and Sketching the Common” by Bagon et al. (2010). The algorithm is able to produce a binary sketch of the common features from a set of inputted images as seen in Figure 6. It is able to produce this by detecting self-similarity descriptors (as seen in 2.1.4.5. Self-Similarity Descriptors) that are common among all inputted images. This has been used other Computer Vision systems such as the one as described in the article by Zhao et al. (2011) where they use the Sketch the Common Algorithms to match shapes between brand logos within commercials. This algorithm could be used to generate a template which could be used within a template matching system and because the algorithm is based on machine learning, the more data that is inputted into a system the more generic the sketch will become.

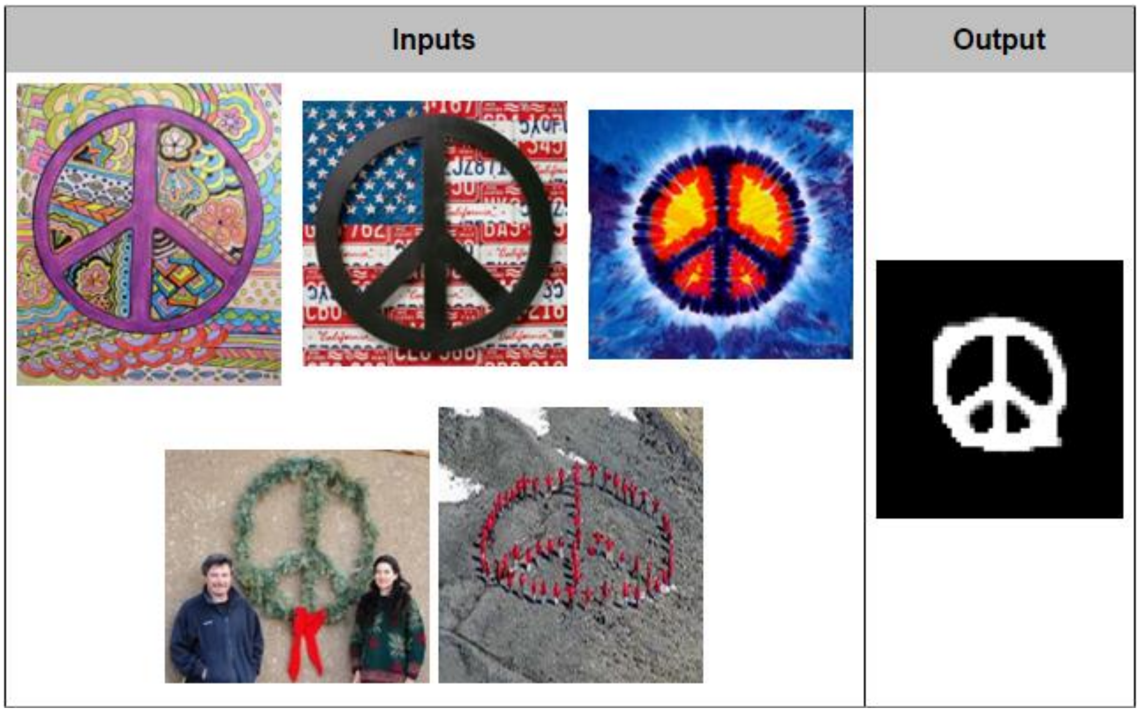

Figure 6. Demonstration on how Sketch the Common algorithm works (Bagon, 2010).

2.1.5. Pattern Recognition

The last step in a Computer Vision system is Pattern Recognition where images are classified based on the exacted features from previous stages. Pattern Recognition can use either supervised or unsupervised machine learning to train a classifier which can predict the classification of images. A supervised Pattern Recognition is trained with labelled data that is inputted into the system. This constitutes that the developer will have to make sure that all the data which is going into the training of a system is correctly classified. Training a supervised system may be time consuming task as data for training needs to be gathered.

2.1.5.1. Support Vector Machines

A Support Vector Machine is a supervised Patter Recognition algorithm that plots the extracted features and draws a boundary dividing the different classifications of features. The boundary is calculated by finding the divider with the largest margins between the types of features, an example of a Support Vector Machine with a Gaussian kernel can be seen in Figure 7. Support Vector Machines have been used in order to detect Microcalcifications within medical scans as shown in the article by El-Naqa et al. (2002) where they stated "The SVM approach yielded the best performance when compared to several existing methods". Support Vector Machines can be used to classify multiple types of features by drawing boundaries between multiple types of features based on sets of features. An SVM classifier can be trained from a Visual Bag of Words as discussed in 2.1.5.2.

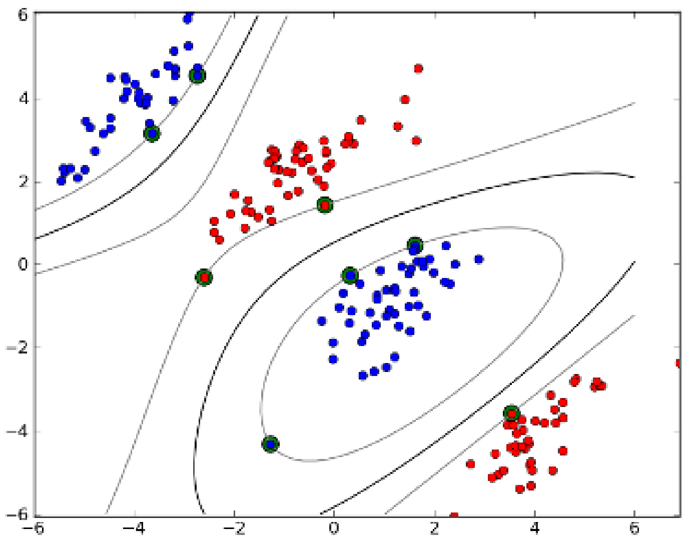

Figure 7. A Support Vector Machine with a Gaussian kernel used to classify feature descriptors (Blondel, 2010).

2.1.5.2. *Visual Bag of Words*

A Visual Bag of Words is an Image Classification model based on the Bag of Words supervised Machine Learning model. A Visual Bag of Words works by keeping a sparse count on the occurrence of different features within an image. These features are usually found by using Feature Detection algorithms, such as SIFT or SURF but can also be found using cruder methods such as random sampling of an image. Once features from the testing data have been found, they then become built into a Visual Vocabulary by using K-means Clustering to group Visual Words. Images that are inputted into the system will then be sampled for features and will compare its features with the Visual Vocabulary to see if it has any matching Visual Words In order to predict a classification. This can be seen in Figure 8 where a Visual Vocabulary are matched with images. An example of a Visual Bag of Words being used within a medical imaging system is shown in an article by Rodriguez et al. (2015). In their article, they claim that by using a Visual Bag of Words they were able to classify multiple types of features within the images they were working with.

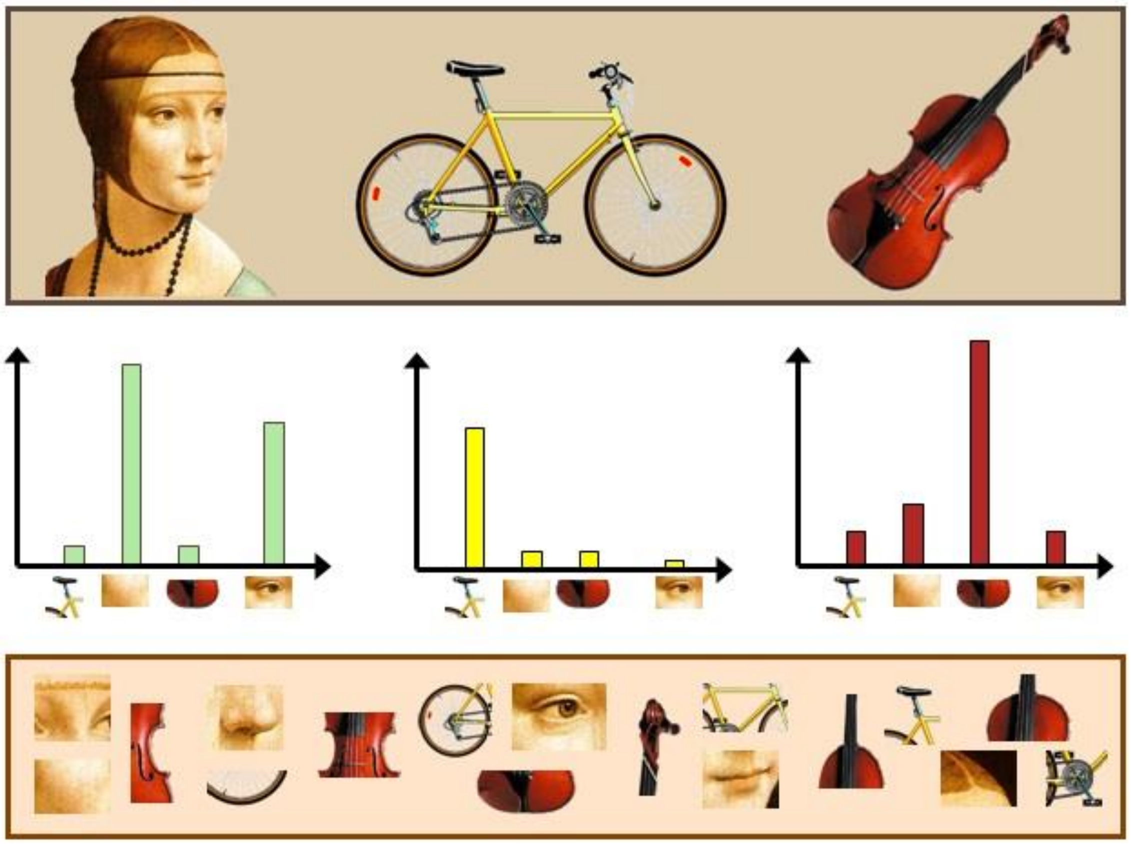
Figure 8. A Visual Bag of Words representing objects as histograms of words occurrences (Levi, 2013).

2.1.6. Tools

2.1.6.1. MATLAB

One of the most renowned Computer Vision tools is MATLAB (MathWorks, 2015) a java based, array oriented programming language that is paired with its own development environment. In the article “Advantages and Disadvantages if Using MATLAB for Solving Differential Equations in Engineering Applications” by Ahmed (2013), he explains that the main reason that MATLAB is used within the engineering sector is because it is fast and simplistic to implement in comparison to other tools. Ahmed also claimed that other tools such as C++ are not as common in the Engineering sector as they are time-consuming to develop, making it unsuitable for Rapid Prototyping a common methodology within the field of Engineering. MATLAB is used within the field of Medical Imagining as shown in Patil and Bhalchandra (2012) where they use MATLAB to identify Brain tumours. MATLAB has been used because, given that it is an array oriented language, it is able to process matrices very efficiently. Processing the images therefore becomes very efficient, making MATLAB capable of performing high level image processing techniques with ease (MathWorks, 2015).

2.1.6.2. MATLAB Alternatives

Although MATLAB is the tool of choice among Computer Vision researchers, depending on the situation other tools may be better suited for development compared to MATLAB. Due to MATLABs licencing restrictions, anything developed within MATLAB will be limited in terms of how the system can be maintained and upgraded. Joel Granados encountered issues with MATLAB’s licences as described in his blog post “Using Matlab was a mistake.” (Granados, 2012) where he was not able to access a system he had been creating because of licencing issues he was unable to correct. Alternative tools can be used to create a Computer Vision system which, in some cases, can be more efficient and can be maintained and upgraded without any fear of extra costs being involved. Two suitable alternatives to MATLAB are Octave (2015) and SCILAB (Scilab Enterprises, 2015); both are open source and are able to bring the ease of use and simplicity of MATLAB to a wider audience. Almeida et al. (2012) wrote a comparison of all three tools and claimed that SCILAB performed the best of the three although had a much lower accuracy compared to the two other tools.

2.1.6.3. OpenCV

Tools such as MATLAB, Octave and SCILAB use interpreters to execute code. OpenCV is a C/C++ library used to develop Computer Vision systems in C++ or Python (Itseez, 2015). OpenCV is the fastest Computer Vision library that is available for C++ as shown in Table 2; this is down to the low level that OpenCV works and its inclusion of functions that run in parallel using CUDA (Nvidia, 2016) and OpenCL (Khronos Group, 2016). OpenCV has been used within complex medical imaging systems such as the system to remove noise from medical imaged based on the Total Variation Algorithm, developed by Li and Que (2011). OpenCV was used because its superior performance which is a result of C++ compiling straight to machine code, removing any intermediaries when running the system. OpenCV is Open Source which therefore is freely available online so that any system developed in it can be edited and maintained by anyone without needing to pay any licencing fees as all source code is available for free.

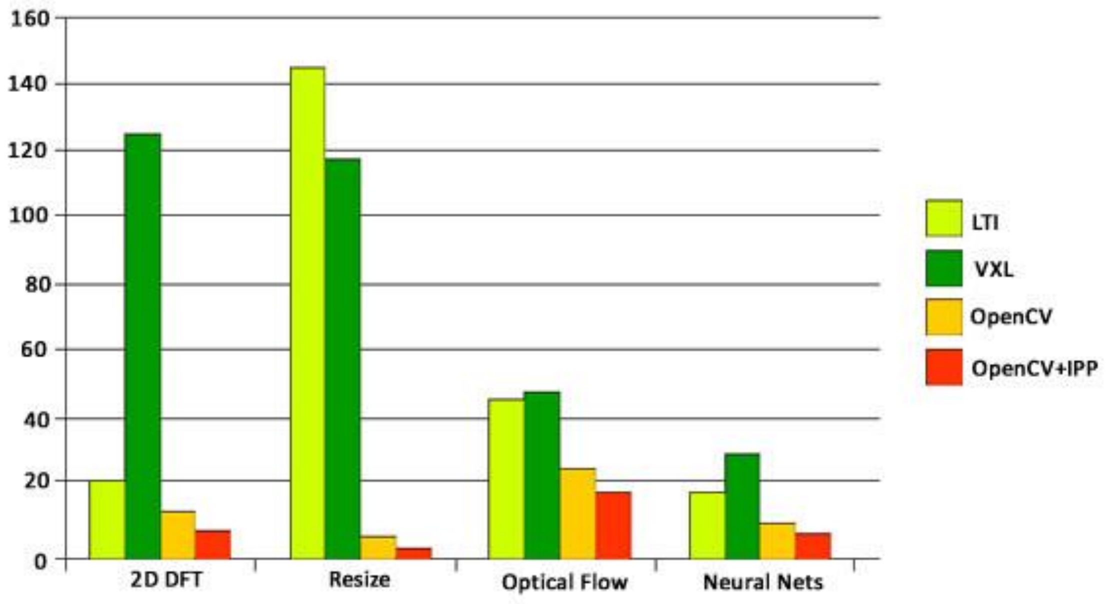

Table 2. Comparison of time take for different C++ computer vison libraries perform tasks (Bradski and Kaehler, 2008, p8,).

2.1.7. Software Methodology

When approaching a software project, the methodical approach of the development needs to be planned out to ensure that the system is able to be developed efficiently. The importance of methodologies to software development has been discussed in a blog post on Toolbox.com by Craig Broyowich (2010) in which he states that software methodologies are the key to structuring deliverables of a project. Due to the projects strict deadlines, an efficient methodology needs to be planned in order to ensure that a deliverable are produced within an allocated time scale.

2.1.7.1. Waterfall

Due to the nature of the requirements of the project, Waterfall's linear approach to development would not be an appropriate for use in this project. The drawbacks of Waterfall have been explained in a paper comparing Waterfall to Agile development by Hou et al. (2004) "Waterfall has a number drawbacks, such as inflexibility in the face of changing requirements, and a highly ceremonious processes irrespective of the nature and size of the project." This is stated because Waterfall follows a linear development which flows from one stage to another but because of the research focused nature of the project, this linear methodology will not fit.

2.1.7.2. Evolutionary

Due to the adaptive nature and fast delivery time required by the project, an Evolutionary methodology may be the best suited methodology for the project. This is because Evolutionary uses Rapid Application Development where the requirements on how a software system should be built is based on development of software "As a result knowledge gained from the development process itself can feed back to the requirements and design of the solution" (Brooks, 1986). The Evolutionary methodology is where a system is developed over periods of rapid prototypes in order to find the solution best suited for the project. Using Rapid Application Development can help find the best solution for a problem but because it is built up of evolutions where development may have to begin from the ground up, this approach may be time consuming (Sundberg, 2015).

2.1.7.3. Incremental

The Incremental methodology is a software development methodology where the development of an article is performed in stages and each stage is a Waterfall-like process with its own induvial steps, this can been seen in the Table 3. It is also stated that an incremental system is best used when working on a system that is designed to have multiple features or levels within a system, for example, a game with multiple levels or a website with multiple pages (Pressman, 2010). In the article from Misra (2010), he explains that Computer Vision systems can be implemented using the incremental software methodology. Each iteration of the system would review and then improve the accuracy of the system.

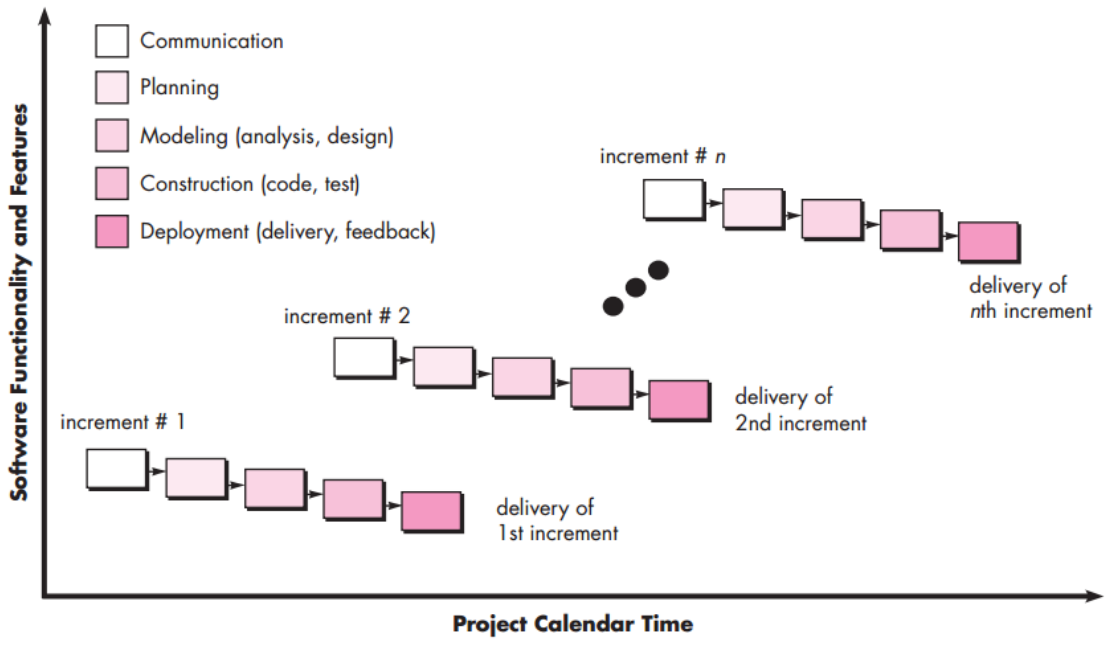
Communication
Planning
Modeling (analysis, design)
Construction (code, test)
Deployment (delivery, feedback)
increment # n
delivery of nth increment
increment # 2
delivery of 2nd increment
increment # 1
delivery of 1st increment
Software Functionality and Features
Project Calendar Time

Table 3. Diagram of the Incremental Methodology from (Pressman, 2010, p42).

2.2. Literature Review Summary

The Literature Review featured several Pre-Processing algorithms and their applications. Pre-Processing can be performed within a Computer Vision system when necessary to make the quality of an image match the standard that is required. This could be in the form of removal of noise from an image like in Rudin et al. (1992) or the sharpening of an image as seen in Ahmed and Nordin (2011). The type of Pre-Processing that would be used within a system would be dependent on the images input into the system and how they are treated inside the system.

From looking at all the stages of a Computer Vision system, it is clear that there is no straight forward way to approach the implementation of such a system. However the most obvious choice on how the system should work would be via Colour Slicing. This is because the plaques and tangles within the image have been stained, making them easily identifiable by their colour as seen in Figure 2. The article by Fermin et al. (1992) has a similar situation where they were able to segment stained cells using Colour Slicing.

Once the segmentation step has been completed, there are a number of different approaches that could be used to implement the system, for example, basic Feature Detection algorithms could be used such as Circular Hough Transform as seen in Solainman et al. (1998). A more robust method could be used to detect shapes such as the Generalised Hough Transform which works by template matching shapes as seen in Philip et al. (1994). Template matching could also be implemented by matching self-similarities across a template and an image as seen in Ke et al. (2007). Machine Learning could also be used in order to classify plaques and tangles as seen in Rodriguez et al. (2015) where they used a Visual Bag of Words to classify medical images.

3. Aims and Objectives

3.1. Aim

"To automate the process of identifying senile plaques and neurofibrillary tangles from high resolution scans of the Hippocampus, in order to aid in the diagnoses of Alzheimer's disease."

This aim encompasses everything which is needed to be completed to produce a system that is useful in the field of medical imaging. This aim defines what sort of input the system will be working with and high resolution images of the Hippocampus, whilst simultaneously being vague on what the output of the system will be. The aim further shows that the system is not attempting to diagnoses Alzheimer's but instead, will aid in the identification of senile plaques and neurofibrillary tangles which would be used to diagnose Alzheimer's.

3.2. Objectives

"To produce a Pre-Processing solution that will remove noise from the input images without modifying its content."

Noise can be the source of lots of issues within a computer vision system; this constitutes that the system will need to complete checks on all images, confirming that they are noise free to avoid errors. It must be noted that when removing noise, there is a possibility that the content within the image can be damaged due to the methods of removing noise. The system will therefore have to ensure that noise within the image is removed but also the content is not altered too much in order to avoid false results.

"To segment the image by colour, extracting all of the stained cells in the image."

By using Colour Segmentation, a new image will be created from the extracted pixels which will fit within a certain colour range from the original image. Given that all the stained proteins within brain scans are all similar colours and stand out from the rest of the image, they can be extracted leaving only the plaques and tangles. Thus, any operations performed on the image later in the system will be performed on only the plaques and tangles.

"To perform Feature Detection on the image to identify its features."

Feature Detection techniques are used to find unique features within an image and then create a descriptor that describes the detected feature. There are many methods of feature extraction and descriptors, to which the most accurate for the system will need to be chosen to make the most effective feature detection possible. By using Feature Detection, the descriptors generated can then be used to compare and identify plaques and tangles.

"To classify the identified features as either plaques or tangles."

Classification is performed in order to identify the type of a feature. There are many different ways to classify a feature from a basic review of a descriptor of the feature or using a Pattern Recognition algorithm to classify the features. Classifying the features would indicate if a feature is a plaque or a tangle.

3.3. Optional Objectives

"To refine the system by implementing it using OpenCV to improve efficiency."

An OpenCV C++ implementation of the system would be more efficient than a MATLAB implementation. OpenCV is an open source library, meaning that it is freely available to anyone. This makes the system more easily maintained, upgraded or repurposed compared to the limitations of MATLAB. These reasons make an OpenCV C++ implementation more desirable but OpenCV can take longer to develop. Due to the experimental nature of the project, a development environment which is easy to develop in is required for the initial iterations.

"To implement a simple, user friendly user interface for the system."

If the system is to be installed at Addenbrooke's Hospital's Pathology Department, a good user interface will be required so that the researchers would be able to use it to aid in the diagnosis of Alzheimer's.

4. Methodology

4.1. Project Management

4.1.1. Scheduling

To make sure that the project is progressing at the rate required to meet the strict deadlines, a project schedule needs to be drafted. In order to accomplish the aim of the project, milestones were created for each individual goal which is needed to reach the aim i.e. Development of artefact. The process described in Grave (2009) specifies the process that was taken to identify the tasks which are required to meet milestones. These tasks were then ordered by finding their dependencies and making sure that when each task is started, all of its dependencies have been completed. Once this process had been completed, each of the tasks had a timescale allocated, based on the tasks complexity (Sharp and Howard, 1996).

To add a greater level of flexibility to the schedule, each of the tasks were assigned a priority. Tasks with a lower priority can be skipped in case any issues occur whilst working on higher priority tasks, and still keep to the defined project milestones. A similar practice can be seen in the Agile Methodology SCRUM where tasks within the Project Backlog are ranked by importance to the project (Ambler, 2014). This allows the important tasks to the project to be developed, and optional tasks are only started if there is enough time.

From this process, the resulting project schedule can be seen in Appendix 1. The project schedule was then visually represented in the form of a Gantt chart which can be used to monitor the overall progress of the project. The resulting Gantt chart can be seen in Appendix 2. The Gantt chart was then tracked by using the project management software Microsoft Project (Microsoft, 2016). This software allows for a project’s tasks and milestones to be tracked in the form of an interactive Gantt chart which can be used to observe the overall progression of the Project.

4.1.2. Risks

As part of the management of the project, any issues that have the possibility to occur throughout the course of the project need to have mitigations planned to reduce their negative impact to the project. To plan mitigations for the risks, the risks first need to be identified and the process of identifying risks requires an understanding of the nature of each task in the project schedule. Once the risks have been identified, their likelihood of its occurrence needs to be found as well as the potential impact to the project. By identifying the project risks, it helps with the understanding of how issues may arise during the course of the project; this can help avoid these issues during the course of the project.

The risk mitigation for the project has been planned in the form of a Risk Matrix as seen in Appendix 3. The risk Matrix includes heading for the issue, the likelihood the issue will occur, the potential impact on the project and how the risk can be mitigated. Risk matrices have been used throughout organisations to mitigate risks as seen in NHS risk management (National Patient Safety Agency, 2008). It must however be noted that the use of risk matrices have been criticized in academia as seen in the article "What's wrong with risk matrices?" (La, 2008). The article claims that that risk matrices can lead to inaccuracies when classifying the nature of risks which can then lead to poor quality mitigations.

Although the Risk Matrix did aid in the avoidance of issues because of the change in the software methodology, the project's requirements changed rapidly and this meant that the risks also changed. One of the issues that occurred was the training data of images which were not labelled by a medical professional; this held that the classifier could not be trained with correct data without estimating the classification of the feature. Given that the classifier was not able to be trained sufficiently without the correctly labelled data, it would mean that the system's accuracy would not meet the standard as expected of the project.

This issue did not have any mitigation planned because the changing software environment meant that the issue was not identified initially. Identification of risks within a changing software development environment, such as Evolutionary development, can be difficult because the requirements for the project can change drastically day to day. This has been discussed in the conference paper by Hossain et al. (2009) where it is stated that risks matrices constantly would need to be updated as a project evolves in order to keep up.

4.2. Software Development

The project's software methodology chosen was to be the most efficient to work with the development of a Computer Vision system. As Computer Vision systems are separated into stages (Pre-Processing, Segmentation, etc.), the development of the artefact can be performed in a separate development cycles, known as Incremental software development. This has been shown to be an efficient methodology for Computer Vision systems in the article from Misra (2010) where an investigation was conducted on the efficiency of the Incremental methodology with Computer Vision systems.

Initially, the Incremental Software development methodology was chosen to be used for the development of the artefact. For each stage in the system, such as pre-processing and segmentation, a separate development cycle would occur with its own design, implementation, testing and evaluation. This methodology makes sure that each stage of the system is fully working and up to specification which is required by the project before continuing to develop the next stage. This methodology requires accurate planning and a clear definition on how the system will work as stated in ISTQB (2016). Due to the experimental nature of the project, defining a clear path of development can be difficult. This flaw in the chosen methodology was identified shortly after the development for the project was started. A new approach for the development methodology was therefore required.

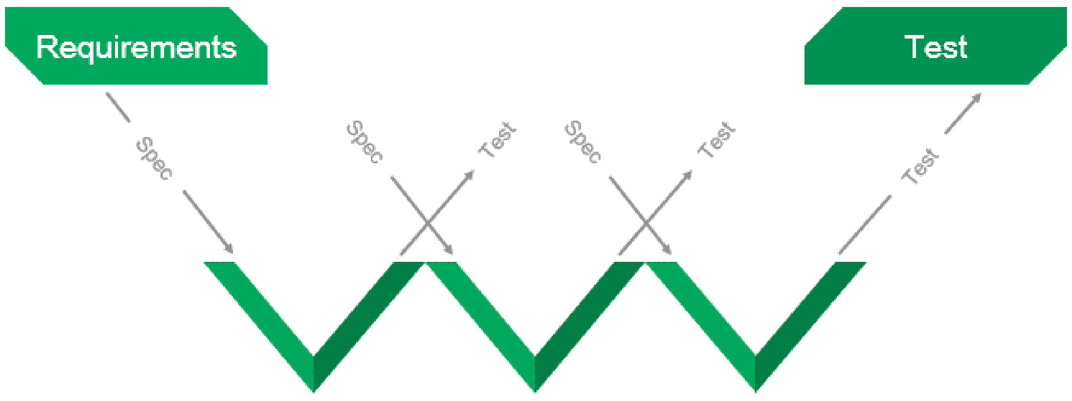

Figure 9. Diagram of Evolutionary Development (Johnson, 2015).

This led to the Evolutionary software development methodology to be used for the development of the system. The Evolutionary methodology allows for rapid prototyping with many prototypes (evolutions) being developed very quickly, each trying new approaches on how to create a solution. This is useful because the project has no defined method to approach a solution and different methods can be tried in rapid succession. Each evolution works as a test to see the validity of the given method within the system. This methodology can also save time by reducing the amount of time planning and designing, as parts of the system can be reused in different evolutions. Kelly (2013) explains that this methodology will save time as methods that appear not to be producing a working solution can be scrapped early in it development.

In the initial evolution of the project, a small selection of different Feature detectors were tested to see their suitability for comparing features in order to then classify them. The two different algorithms were tested to find the most suitable for the plaques and tangles. The algorithms were the Circular Hough Transform and the Generalized Hough Transform. When it was found that either algorithm was suitable, a new method was found by combining two algorithms; one to sketch the common features of a set of images into a template image and another to test the similarity of the template to the input images. While using these two state of the art algorithms, they seemed the ideal choice for the system but due to issues with their implementations, it made it impossible to use in the system.

This constituted that another evolution was needed to be developed to build the system. Using the Pattern Recognition algorithm Visual Bag of Words, the features in the input images can be classified using a SVM. This solution was used in the final solution and was only made possible because of the Rapid Prototyping that the Evolutionary methodology allows for. Each Evolution changed how the solution classified the plaques and tangles while still keeping the initial steps of the evolutions the same, cutting down on the time required to produce evolutions.

4.3. Tools

4.3.1. Matlab

MATLAB (MathWorks, 2016) is one of the most renowned tools for Image Processing and Computer Vision, and is industry standard for research. MATLAB is an array oriented programming language based on C++ and Java developed by MathWorks that has been able to become the industry standard because of the large code base, ease of use and its efficiency for operating on matrices. Many academics, across the field of Computer Vision, use MATLAB to demonstrate their work such as Bagon et al. (2010) where the Sketching the Common algorithm has been demonstrated using MATLAB. An example of MATLAB being used in the field of Medical Imaging is in Patil and Bhalchandra's (2012) system to identify Brain tumours. This has been built with MATLAB to implement their high level algorithms in an environment where it is easy to rapid prototype and experiment.

In the article "Advantages and Disadvantages if Using Matlab for Solving Differential Equations in Engineering Applications" (Ahmed, 2013), the writer explains that the main reason that MATLAB is used within the engineering sector is that it is fast and simplistic compared to other tools available. The writer of the article claims that other tools such as C++ are not as common in the Engineering sector as they can be time-consuming to implement and test in. This is partly because MATLAB is interpreted and does not need to be compiled before running an application. MATLAB also includes interfaces to work with other development environments such as .NET, Java and Python. This means that parts of an application, such as graphical user interfaces, can be developed in different development environment such as C# as Windows Forms (Microsoft, 2016) and the complex functionality can be implemented in MATLAB.

4.3.2. Image Processing Toolbox

The Image Processing Toolbox (MathWorks, 2016) for MATLAB is a collection of functions of standard Image Processing algorithms. The toolbox features functions that are used within the Pre-Processing section of Computer Vision systems such as Morphological Transformations and Connected Component Labelling. The Image Processing contains functions used for analysis of images that can be used in order to find regions within image, for example, bounding boxes. MathWorks Image Processing Toolbox was chosen to be used over other Image Processing library as the library is fully supported in MATLAB and has the most support available. Other libraries with similar functions have no clear advantage over the Image Processing Toolbox, for example, the Octave Image Library (Draug, 2015) is a similar library for MATLAB and Octave. The Octave Image Library is less supported online and in order for the system to work, it will need to be installed as well as MATLAB for it to run.

4.3.3. Computer Vision Systems Toolbox

MATLAB has a Computer Vision Systems Toolbox (MathWorks, 2016) which features functions used for Feature Detection and Pattern Recognition. For example, it includes a group of function that are used to build a train a Bag of Visual Words Image Classifier. These functions use Machine Learning techniques to build a classifier and then use the classifier to predict the classification of images. The Computer Vision Systems Toolbox also includes Feature Detection functions such as SURF, MSER and FAST. This toolbox has been chosen over other Computer Vision libraries because it is the largest one available for MATLAB. However while it is the largest library available, some implementations of algorithms are not part of the tool box such as the SIFT Feature Detection algorithm.

4.3.4. VLFeat

The VLFeat (VLFeat, 2013) is another library for MATLAB that features functions to perform complex Computer Vision algorithms such as SIFT Feature Extraction. The library also includes functions that allow for SIFT features to be compared across images returning lists of matching descriptors. This library has been used as it is the most recommended implementation of the SIFT algorithm for MATLAB.

4.3.5. OpenCV

OpenCV is a powerful Computer Vision library built in C/C++ that can be used to develop in C++ or Python. OpenCV has a large range of tools and functions available to be used and because it has been written in C/C++, the functions have better performance compared to other Computer Vision tools such as MATLAB. OpenCV also utilizes CUDA to power several functions. This means that OpenCV can work in parallel GPUs, increasing the speed of functions. The benefits of CUDA can be seen in Table 4. Despite OpenCVs advantages with performances, its implementation within C++ requires a longer development period as compared to tools such as MATLAB. Thus, OpenCV is an unsuitable environment for Evolutionary development as prototypes would take an increased time to develop. However because of the performance increases, OpenCV can be used to develop software for release.

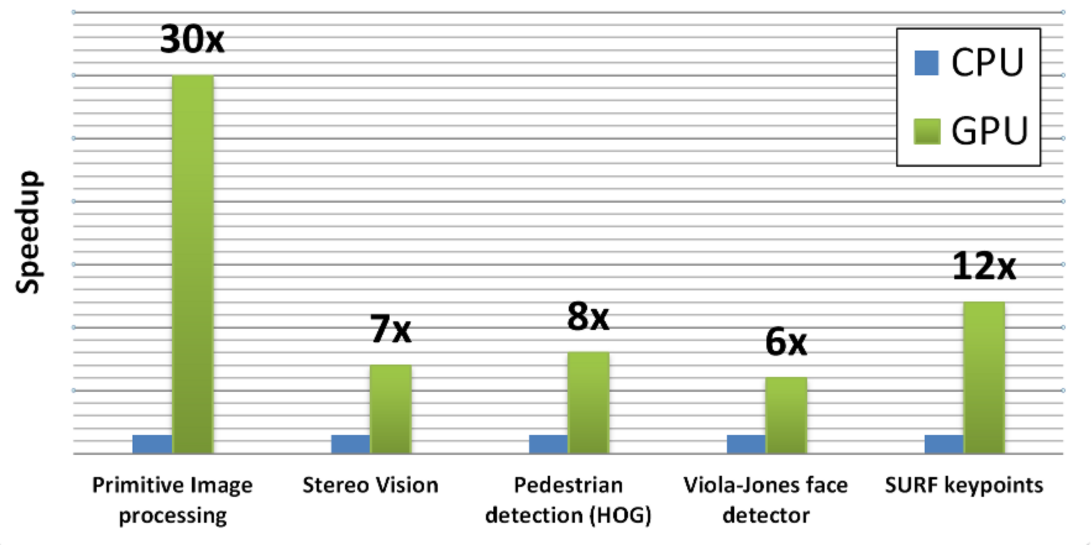

Table 4. Chart showing the speed difference when using CUDA OpenCV functions (Itseez, 2016).

4.3.6. Visual Studio

Visual Studio (Microsoft, 2016) is a powerful IDE build by Microsoft which is used to develop applications in a various range of languages. Visual Studio has powerful tools that aid in the development of C++ code within the IDE and also has Microsoft's Visual C++ Compiler (Microsoft, 2016) built into the IDE. This makes developing C++ systems much more convenient with features such as error identification and diagnostic tools. Visual Studio also allows for libraries to be added to the environment, making it easier to include them into a project. Visual Studio is most supported IDE for C++ development, compared to other alternative IDEs such as NetBeans (Oracle, 2016) and CLion (Jet Brains, 2016). Both have large communities but lack Visual Studio's vast feature set.

4.3.7. GitHub

GitHub (GitHub, 2016) is a Version Control platform based upon Git that allows for robust Version Control which is hosted on the GitHub platform. GitHub allows for every iteration of the project to be uploaded to GitHub with Git making note of all the changes to the code between each commit to GitHub. This makes sure that all changes made within the system are well documented and if there any changes to the code that cause issues, the code can always be reverted to a previous version when the system was working. GitHub also acts as a backup for the project as all the source code is hosted on GitHub's external servers; this means that if anything happens to the source code, the latest commit can be pulled from GitHub. Version control is highly recommended when developing any software projects as seen in the article "Why use Version Control?" (SoundSoftware, 2011). The article discusses how using Version Control is an easy way to manage how a piece of software is developed. GitHub was chosen above any other version control platform as it is the most user friendly Version Control platform and also hosts the largest community compared to others such as Bit Bucket. GitHub also includes a student package that gives students five free private repositories that have been used in the project.

4.3.8. Microsoft Project

Microsoft Project (Microsoft, 2016) is a piece of Project Management software that has been used throughout the project to track its progress. It was used by copying the Gantt chart that contains all the information about the tasks into the program. With this program, the progress of the project was easily visible by viewing colour coded charts and comments which were made to each of these tasks.

4.4. Research Methods

In order to validate a Medical Imaging system, there is a lot of different methods that can be used to validate a system. This is explained in the paper by Krupinski and Jiang (2008) where they list the different levels of evaluation and the most efficient methods to assess each level. As this system is not a diagnostic system and only a system to aid diagnosis, the system will only be tested on the first level known as Technical Efficiency.

4.4.1. Technical Efficiency

Accordingly to Krupinski and Jiang (2008), the technical accuracy of the system would be the most suitable way to evaluate the validity of a project. This can be done by dividing the input data into two sets, one for training and another for validation. The ratio that the data is split would depend on the number of available data in the input but would most likely use the Pareto Principle (Kiremire, 2011) where the data uses an 80/20 split. Once the validation set has been confirmed, the system would then be compared the results of the system processing the validation set to the labels confirmed for the validation set resulting in a confusion matrix.

This confusion matrix will show how often the system classifies certain features as different classifications. For example, a confusion matrix would map how many times the system identified a tangle as a tangle and a tangle as a plaque. This shows the number of true positives, false positives, true negatives and false negatives which appear within the system as well as an average accuracy rating. From the number of false positives and negatives, several metrics can be extracted. The metrics that will be used within project are Accuracy, Error Rate, True Positive Rate, False Positive Rate, Specificity, and Precision. ROC curves can also be generated to demonstrate how effective the system can be by visualising the number of false positives vs true positives.

5. Design and Development

5.1. First Evolution

5.1.1. Design

The system that is being developed is a Computer Vision system to which it follows the standard pattern of Computer Vision systems, starting with Pre-Processing. Given that the input image are of a consistent standard and have all originated from the same source with consistent lighting conditions, there is therefore no need to perform any initial pre-processing. Image Segmentation is the second stage in Computer Vision systems; this needs to be performed on the images in order to extract the plaques and tangles into new binary image showings, containing only their shape. This separates the data that is needed by the system from the other irrelevant data.

Due to the plaques and the tangles both containing stained tau proteins, they both share a similar colour that stands out against the grey matter of the Hippocampus to which the most effective method of segmentation would be Colour Slicing. This would produce a binary image based on if a pixel's colour matches a specified range. This is so that when the plaques and tangles are being processed, no erroneous data is brought forward. A binary image is produced as it shows only the shapes of the features within the image which will be used to generate the descriptors of the plaques and tangles. The result from the Colour Slicing may then require Pre-Processing to remove any noise that may be left over from coloured specs within the image.

As the image is in binary form, Morphological Transformations could be used to get rid of noise. This is a superior method for removing noise as more traditional methods of removing noise involve applying filters to images before Colour Slicing that could damage the unique shapes of the plaques and tangles. Once the images have been confirmed to be free of noise, the system can then begin to segment each individual plaque and tangle into separate images.

For each individual plaque and tangle that is segmented into an individual image, a descriptor needs to be calculated which describes the unique shape of the image. This can be achieved by using Feature Detections algorithms such as a Circular Hough Transform which is able to identify the circularity of the shapes. When a new image is inputted into the system, the circularity will be checked against the average circularity of plaques and tangles in order to then classify the features into plaques or tangles. If this algorithm fails to meet expectations then a more robust algorithm may be required such as the Generalised Hough Transform that will find the similarity against a template and images.

5.1.2. Colour Slicing

In order to perform Colour Slicing, the colour range of the plaques and tangles would need to be identified. Given that all the images are of a similar standard, the same colour range can be applied to each image without having to perform any Pre-Processing. When images are read into MATLAB, they use the RGB colour spectrum with three channels each representing intensity of either Red, Green or Blue. MATAB's inbuilt colour picker can be used to find average RGB values for the plaques and tangles. After finding the colours of plaques and tangles from across multiple test images, the RGB colour ranges that best represents the plaques and tangles were Red 40 – 160, Green 10 – 110 and Blue 10 – 110. The RGB colour spectrum was used rather than the HSI colour spectrum because MATLAB tools does not include any HSI colour conversions. This would mean a third party MATLAB HSI converter would need to be used and from searching MathWork's File Exchange (2016), there are no reliable converters that offer reliable results or efficient performance.

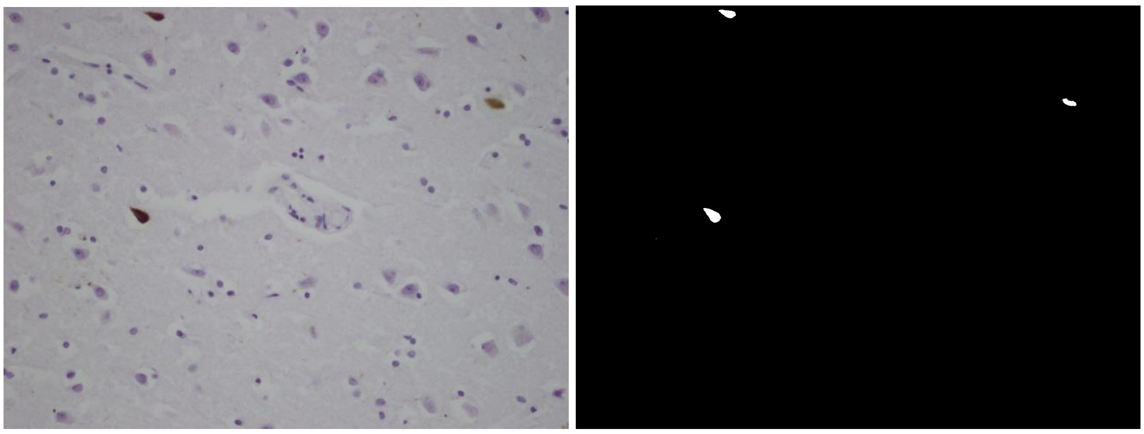
Figure 10. (a) Shows a less noisy test image (b) Result of Colour Slicing the less noisy test image.

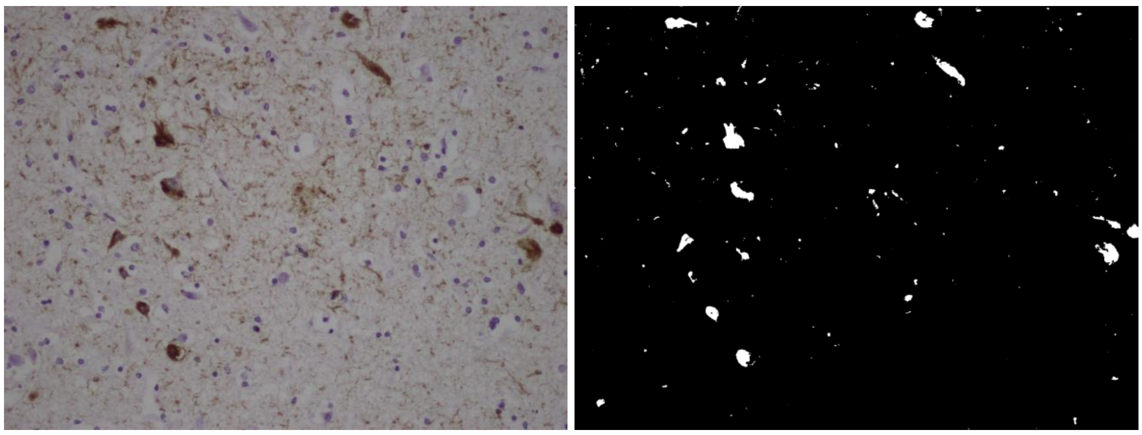
Figure 11. (a) A noisy test image (b) Result of Colour Slicing the no test image.

The result of the Colour Slicing process is a binary image with only the shapes of the plaques, tangles and other small pieces of noise present. This has been accomplished by creating a binary image the same size as the original image with all pixel values were set to 0. The system then loops through every pixel in the original image and checks to see if the pixel's colour fits the specified colour range. If the pixel fits the colour range, the position of the pixel in the binary image is set to 1. Two examples of the results of this process can be seen in Figures 10(b) and 11(b). The result in Figure 10 shows how efficient Colour Slicing can be when there is minimal noise within the image however Figure 11 shows how noise can affect the result leaving a large amount of noise within the binary image.

5.1.3. Noise Reduction

Once the Colour Slicing has been completed, the resulting binary images contained only plaques, tangles and small specs of noise from small objects originally in the image. This noise can lead to inaccuracies further down within the system as their unpredictable shape may be identified as a plaque or tangle. As the images are binary images and the noise are small, specs morphological operations can be used as explained in 2.1.2.2 Morphological Transformations.

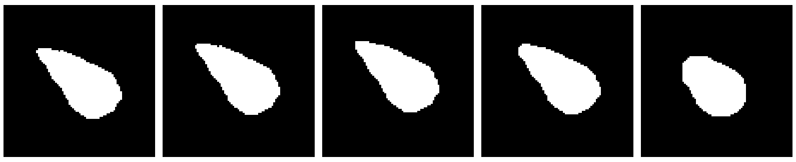
Figure 12. (a) Original binary shape (b) 1x1 disc (c) 3x3 disc (d) 5x5 disc (e) 10x10 disc

To perform Morphological Transformations, a structuring element needs to be selected which is able to remove the noise from the image that also does not alter the unique shape of the plaques and tangles. Due to the circular edges of the shapes of the plaques and the tangles, the disc structuring element was chosen compared to the rigidness of other alternatives. The size of the disc would then need to be selected. After a series of tests, the results of which can be seen in Figure 12, a 3x3 disc structure was chosen as it was able to remove the majority of the noise from the image whilst still leaving the distinctive features of the shape. The results of the Morphological Transformations can be seen in Figure 13 compared to the noise seen in Figure 10 (b) and Figure 11 (b).

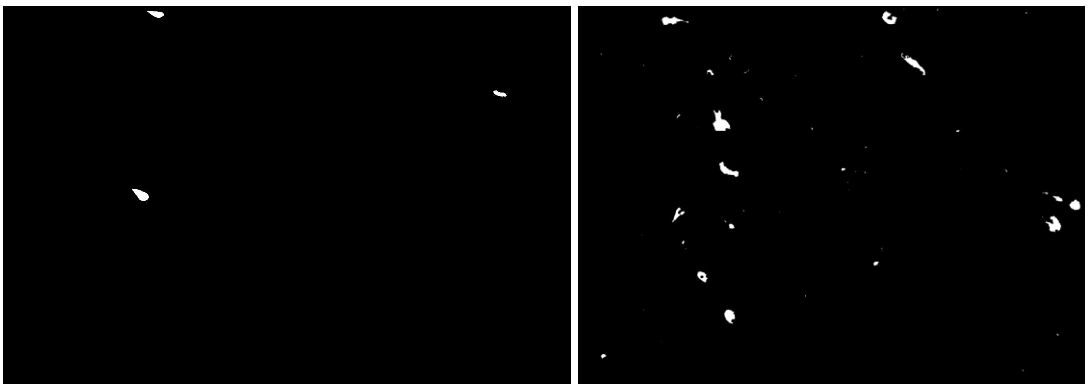
Figure 13. (a) Morphological processed noisy test image (b) De-noised noisy test image.

5.1.4. Induvial Segmentation

To segment each of the induvial plaques and tangles into separate images, the system would first need to locate the features within the image. This can be done using the Computer Vision Technique Connected Component Labelling as seen in 2.1.2.4. Connected Component Labelling. This will produce a matrix that will contain shapes of induvial labels. From the label matrix, the individual shapes can be segmented into separate images, the result of this process can be seen in Figure 14.

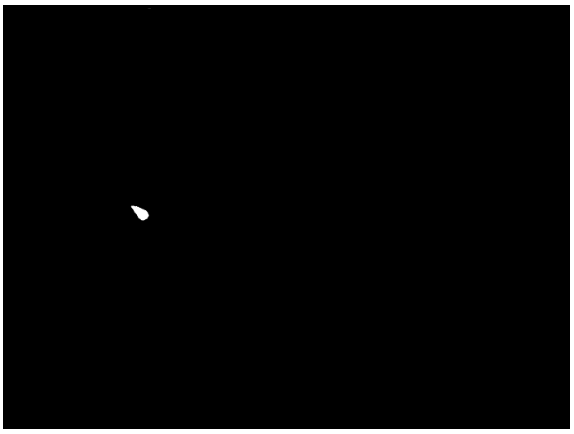

Figure 14. The result of extracting shapes from Connect Component Labelling.

Once this process has been completed, any surrounding empty area needs to be removed to save resources when the image is processes further. To accomplish this, a Bounding Box has been used to find the coordinates of the points which enclose the entire content of the binary image. Using these coordinates, the image can be sampled and a small box contained only one individual feature can be retrieved from the image. An example of the result of this can be seen in Figure 15 where a tangle has been sampled from Figure 14 using this method.

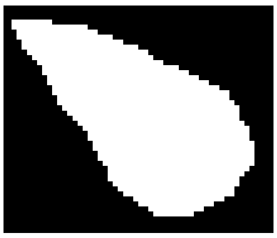

Figure 15. A segmented tangle.

5.1.5. Circular Hough Transform

Feature Detection algorithms can then be performed on the segmented plaques and tangles to extract the descriptors. When working with medical scans that include cells, the best method of Feature Extraction is the Circular Hough Transform as seen in 2.1.4.1. This is because the Circular Hough Transform works by analysing edge information of features by using an edge detection filter to produce edge information. A number of different Edge detectors are available such as Canny, Sobel and Prewitt but to choose one for the system the study by Maini and Aggarwal (2009) was reviewed where they compared a large number of edge detection algorithms. The study claims that the Canny algorithm was the most robust for general Edge Detection in Computer Vision systems.

Once Edge Detection has been applied to the image, the Circular Hough Transform can then be performed to retrieve the descriptor on circularity from each individual plaque and tangle. MATLAB includes a built in Circular Hough Transform function "imfindcircles" (MathWorks, 2016) in the Image Processing Toolbox. This function returns a metric that indicates the circularity of the identified circles within the image. Given that each of the images contain only one feature, the Circular Hough Transform would only find one circle and would then return one metric. From a combination of the metrics for each type of feature (plaques and tangles), an average for each is calculated from every available test image in order to create two average metrics.

```
foreach feature in features:
      edge = Canny(feature);
      circularity = imfindcircles(edge);
      tangle = abs(circularity - tangleCircularity);
      plaque = abs(circularity - plaqueCircularity);

      if tangle < plaque
            tangleCount++;
      if tangle > plaque
            plaqueCount++;
```

Figure 16. Pusdocode for classifying plaques and tangles using the Circular Hough Transform.

When an image is inputted into the system, it would segment the features into individual binary shapes. A circularity metric will then be calculated for each of the features in the separate images. These metrics will then be compared against the average circularity metrics of the plaques and tangles to predict the classification of the feature to see if it is a plaque or tangle. This is achieved by creating two variables from the differences between the circularity and the average circularities of the plaques and tangles. The absolute values of these variables are then found so that any negative numbers become positive. The differences are then compared, and the lower of the two scores then determines the classification of the feature. The pusdocode for this process can be seen in Figure 16.

This method caused issues as the calculated averages for the plaques and tangles from the test images are very similar, due to plaques and tangles both having a large range of circularity with a lot of crossover. The system also is not able to account for any features that are not plaques and tangles as their circularity metric would be unpredictable. This means that circularity is not a suitable metric for identifying plaques and tangles, therefore a more robust metric is needed.

5.1.6. Generalised Hough Transform

After finding that the Circular Hough Transform did not perform as expected, a new form of Feature Detection which is capable for detecting more robust shapes was required. After looking at other forms of Feature Detection, the Generalised Hough Transform was decided to be used as it is able to identify certain shapes by using Template Matching. The Generalised Hough Transform works well to locate robust shapes within an image but can be computational heavy when needing to anticipate for rotation and scale. A further discussion on the Generalised Hough Transform can be found in 2.1.4.2. Generalised Hough Transform. To use the Generalised Hough Transform, an implementation of the algorithm is needed as MATLAB does not include it in their toolboxes. After searching on Mathwork's File Exchange, a suitable implementation was found as written by Eppel (2015). This function takes two inputs, an image and a template built of fully connected binary contours. This would mean, in order to use this algorithm, a suitable template would need to be produced for both classifications.

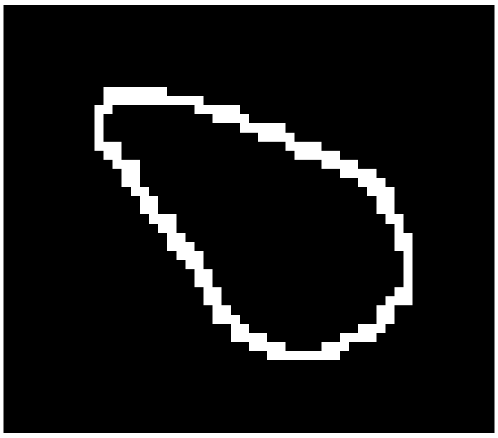

Figure 17. Basic Template for Tangles based off tangle within Figure 11(a).

The function by Eppel returns a metric indicating the similarity between the feature and the template. Thus in order to use the algorithm within the system, not much change will need to be changed. To use the algorithm however, generic templates for plaques and tangles would need to be created. For the purposes of development, a basic template was used created from a tangle from Figure 10(a) by using the Canny edge detector algorithm, this template can be seen in Figure 17. The pusdocode for the implementation that uses the Generalised Hough Transform can be seen in Figure 18.

```
foreach feature in features:
     edge = Canny(feature);
     tangleMatch = find_object_in_image(edge, tangleTemplate);
     plaqueMatch = find_object_in_image(edge, plaqueTemplate);
     tangle = abs(tangleMatch - tangleAverage);
     plaque = abs(plaqueMatch - plaqueAverage);

     if tangle < plaque
          tangleCount++;
     if tangle > plaque
          plaqueCount++;
```

Figure 18. Pusdocode for classifying plaques and tangles using the Generalised Hough Transform.

The Generalised Hough Transform works using template matching, identifying unique abstract shapes predefined using templates. Given the large variety of shapes that plaques and tangles can be, the creation of two generic templates that encompass the general shape of plaques and tangles is a very difficult task. The creation of these templates may still not be effective because of the large range of shapes that plaques and tangles can be, an example of this can be as seen in Figure 19. With the simple tangle template that had been created for development purposes, it was unable to identify the tangle it was created from. This is because the implementation by Eppel does not account for the scale differences between features and the template. Due to these two factors, it can be deemed that a Generalised Hough Transform approach is not suitable for the system and a new approach would be needed.

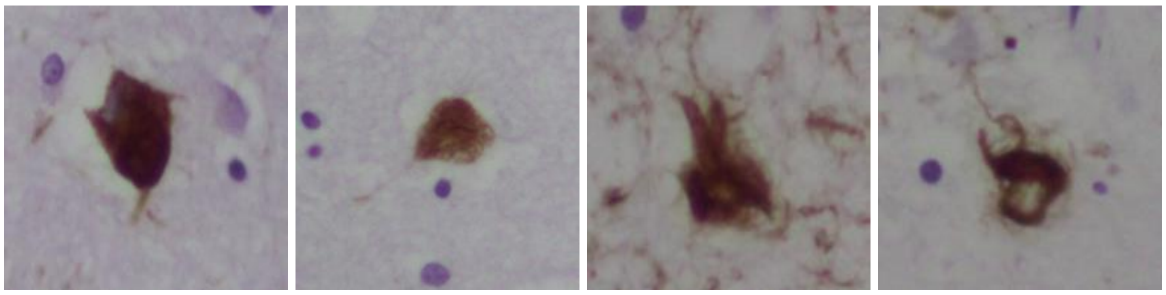

Figure 19. A collection of tangles showing the variety of the shapes.

5.2. Second Evolution

5.2.1. Design

The previous evolution used basic Feature Detection algorithms to identify the plaques and tangles via shape. The first algorithm implemented was the Circular Hough Transform which was used to identify the circularity of the shapes as a form of identifying them. Given that the shapes of the plaques and tangles can drastically vary, as seen in Figure 19, circularity is not the best way to classify them. Once this method was found not to work, a more robust method of shape identification was needed to be used and the Generalised Hough Transform was chosen. This is a more robust algorithm for identifying unique shapes but can be inefficient when having to account for rotation. The implementation of the Generalised Hough Transform used within the first evolution did not account for scale differences between features and the template.

This leads to the belief that a more robust template matching algorithm, which is capable of accounting for scale and rotation, could be used to identify the plaques and tangles. This would require two different algorithm; one that is capable of matching templates to images and producing similarity metric, and another to produce a template based on a set of input images. The Sketch the Common algorithm (Bagon et al., 2010) has been chosen to be used to create two generic templates - one for plaques and another for tangles. The Sketch the Common algorithm works by detecting and then sketching common features across, then inputs images into a binary image, this has been discussed further in 2.1.4.6. Sketch the Common. The algorithm is a form of machine learning that trains on data to produce a generic template. Further depending on the number of input, images will change how generic the resulting templates will be.

Once the templates have been created, an algorithm which is able to match a given template to an image is required. The algorithm described in "Matching Local Self-Similarities across Images and Videos" (Shechtman and Irani, 2007) is capable of identifying templates within images and has been discussed in 2.2.4.5. Self-Similarity Descriptors. The method of comparing Self-Similarities looks at the internal patterns within an image's features and compares them to the patterns from other images such as a template. This method is able to avoid the issues that were present within the previous evolution and is able to function despite differences in orientation and scale.

Given that the two algorithms are modem whereby they were created in 2007 and in 2010, there are not many implementations available. Implementing the two algorithms would be a big task that would be out of scope for the project. This would mean that the project would have to fit around the implementations that already exist. The Sketching the Common algorithm has an implementation created by its designers for MATLAB. The Self-Similarity algorithm has an implementation built by Charfield et al. (2009) for the University of Oxford written in C++ with a MATLAB wrapper.

The system would work in a similar way to the previous evolution but would use the two templates generated by the Sketching the Common algorithm. The method described in the article by Shechtman and Irari (2007) would then be used to compare the templates to the induvial segmented tangles and plaques, and would calculate the similarity. From this similarity, the system would predict the identity of the feature and then tag it accordingly. This purposed system flow can been seen in Figure 20.

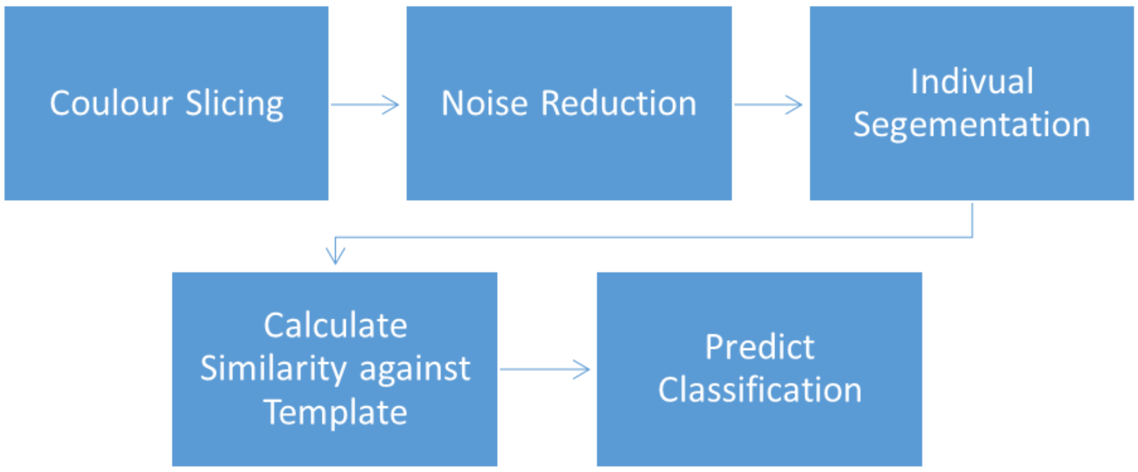

Figure 20. The flow of Processes within the proposed system in the second evolution.

5.2.2. Sketch the Common

For the evolution, the initial step is to create two templates - one of a plaque and another of a tangle. This was achieved by using the MATLAB code from Bagon et al. (2010) available from their website. Their code has been written in a combination of MATLAB and C++ using the OpenCV library. Therefore in order to use their code, it will need to be compiled using MATLAB’s mex compiler. In the code’s read me file, it instructed that the implementation only worked on Linux systems as some elements within the code are not supported in others. It also stated that in order to compile the code, it required the use of the GCC compiler.

Given that the platform which the system was being developed on was Windows machine, it meant that a Linux distribution needed to be installed to compile and run the code. Ubuntu 14 (Canonical, 2016) one the most popular Linux distribution was chosen to be installed. This is because Ubuntu is one of the most popular Linux distributions and also supports a large range of software including MATLAB. There were issues when installing Ubuntu on the hardware used for the project as the installation process had problems with the dual graphic configuration hardware. This meant that a traditional Linux installation was not possible and either a new piece of hardware was required or a virtual desktop was needed to be created on top of the current Windows installation.

Using the software VirtualBox (Oracle, 2016), a virtual desktop was created with an Ubuntu 14 image. MATLAB was then installed onto this desktop and the mex compiler was used to compile the code. The mex compiler gave a warning, stating that the version of the GCC compiler installed was not supported by MATLAB and could cause errors. After the mex compilation had been completed, the example script which was included in the Sketching the Common package was ran. While running the example script, an error accrued within the Sketching the Common code meaning the system was not able to continue. Further investigation into this found that the problem was caused when trying to run the section of the code written in C++. This led to the belief that having an unsupported version of the GCC compiler caused the errors.

This led to the task of removing the current version of the GCC compile and installing a previous version that is supported by MATLAB. The code was compiled without any warnings but then when the sample code was ran, the same errors during runtime occurred. From this we can assume that a fault with the how the code was compiled and ran could be caused by the Virtual machine. To test this, a new piece of hardware was then borrowed, Ubuntu and MATLAB were installed and GCC was downgraded to a supported version. This led to the same result and as code still encountered errors during runtime on the native Ubuntu install. This could either mean that the code could not run on Ubuntu and a different Linux distribution was required or that the code was faulty. Trying to get the implementation to work took a much longer time than expected and is still not guaranteed to work even after more testing. Therefore a new approach is needed in order to implement the system as no more time can be spared on this method that may not work.

5.3. Third Evolution

5.3.1. Design

The previous evolution of the system used the Sketch the Common algorithm to produce two templates which was used with a template matching method against the system's input images. This algorithm used supervised machine learning to produce generic templates of the plaques and tangles. Given that the system uses supervised learning, it means that the more data that the algorithm has to be trained to make generic templates. As templates become more generic, the more plaques and tangles will be accurately identified. While this method of approach seemed the ideal choice, implementation issues caused the system not to be implemented. Despite this the idea of using machine learning was brought forward from the previous evolution, this consequently led to the utilisation of Visual Bag of Words and SVM to be used to classify the features.

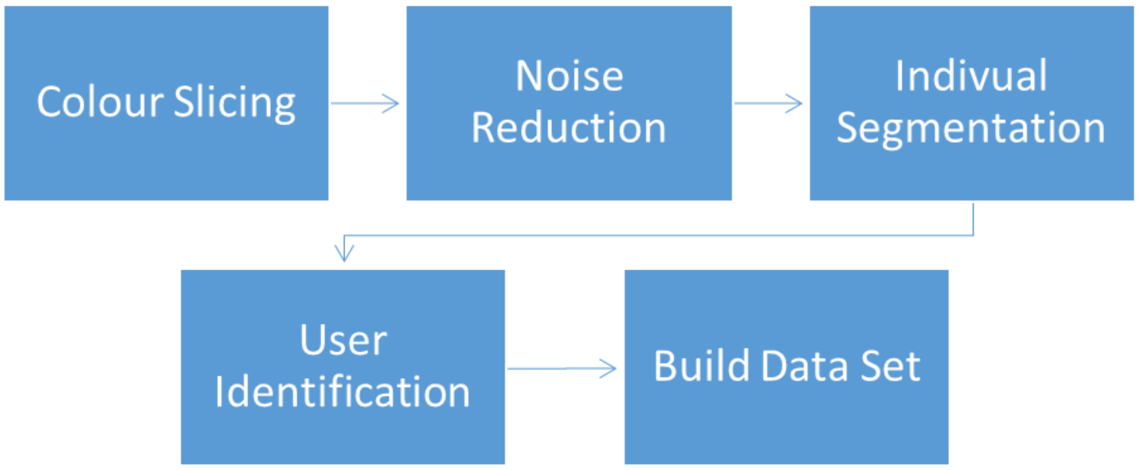

Figure 21. The flow of Processes within the proposed secondary system to build training data.

A secondary system would be required to build a data set which would be used to train a Visual Bag of Words Classifier. The secondary system would have to be able to produce two sets of images that contain only the individual plaques and tangles. Each of the individual plaques and tangles would have to keep their original colour and have their scale normalised. To build the two data sets, the secondary system would have similar initial steps to the previous evolution but once individual plaques and tangles have been segmented, the user would then have to classify them as either plaque or tangle. This proposed method can be seen in Figure 21. With these data sets, a Visual Vocabulary can be built using the Visual Bag of Words algorithm. A Feature Extraction algorithm such as

SIFT or SURF is used to extract features from the images, then K-means clustering is performed to reduce the number of features. SIFT has been chosen to be used as it is more accurate than SURF and there are no requirements for performance in the project. Once the Visual Vocabulary has been built up, it can then be plotted within a linear SVM classifier.

Once a classier has been created, the system can then be developed and would share the initial steps of the secondary system. The steps would be the same until each of the plaques and tangles have been individually segmented then the classifier would be used to predict the classification of the features. This purposed method can be seen in Figure 22. The system would need to be able to be retrained as new data becomes available keeping the system as accurate as it can be.

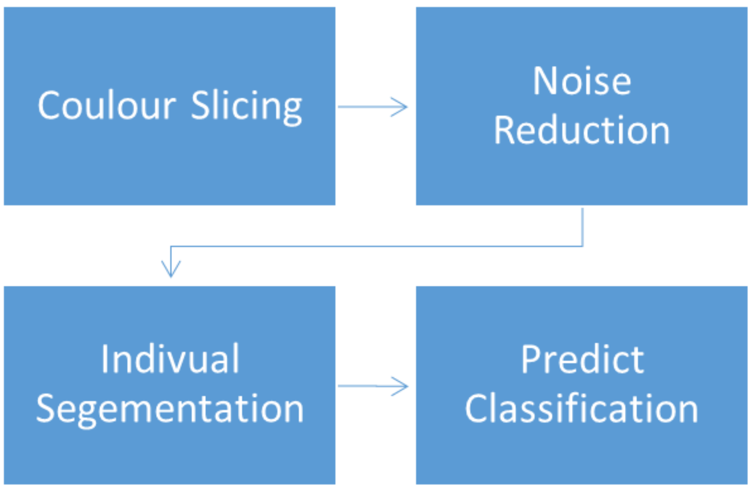

Figure 22. The flow of Processes within the proposed system that uses a classifier.

5.3.2. Secondary System

The secondary system will go through the Colour Slicing and Noise Reductions steps developed in previous evolutions. The system will then segment each individual plaque and tangle by first using Connected Component Labelling to segment individual plaques and tangles into separate images. The system will then calculate coordinates from bounding boxes for each of the separate images. From these coordinates, the system can go to the original image and sample it from the coordinates leaving only the plaque, tangle and other ambiguous features, an example of this can be seen in Figure 23(a). The image of the segmented plaques and tangles can then be segmented by colour, removing all colour within the image that is not between a given colour range. This is because when Feature Extraction is performed on the images, none of the background should extracted. For the same reason, the size of the image needs to be normalised so that the Feature Extraction algorithm does not include scale when searching for unique features. An example tangle that has been Colour sliced and Normalised can be seen in Figure 23 (b).

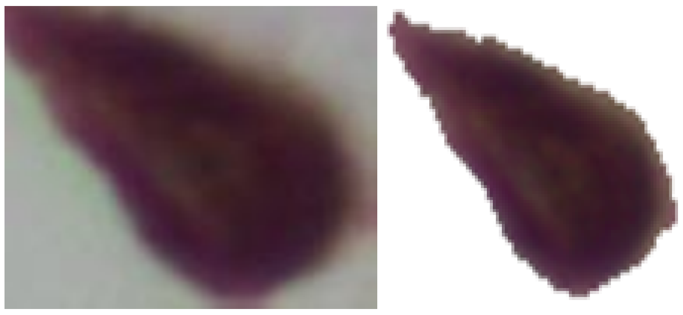
Figure 23. (a) Segmented tangle from original input image (b) Segmented tangle with Colour Slicing and Normalisation applied.

The system would then need to add each of the extracted plaques and tangles into two different images sets. This has been done by writing the extracted features into folders on the hard disk then generating an Image Set with three categories (plaques, tangles and noise) from the three folders. This has been done so that when the system is being processed, the images can be read from the hard disk rather than the whole image set being moved into system memory. This allows for the larger image sets to be processed easier and act as a way to build up data for training permanently on a computer system. The image set is then normalised to make the number of images in each category even by removing entries randomly from the larger category.

5.3.3. Training the Classifier

To create a classifier, a Visual Bag of Words needs to be created using the image set produced by the Secondary System. A Visual Bag of Words can be created by using the "bagOfFeatures" function that is included in MATLAB's Computer Vision Systems toolbox. This function will first detect all the features in the image set by using a Feature Extraction algorithm. MATLAB's Visual Bag of Words function uses MATLAB's implementation of the SURF algorithm, although the function allows for other Feature Extraction algorithms to be inputted as an argument.

This constitutes that an implementation of the SIFT algorithm could be used within the Visual Bag of Words. The VLFeat library includes an implementation of the SIFT algorithm for MATLAB. This was inputted into the Visual Bag of Words as an argument but it was found that the implementation of SIFT was not compatible to work with the Visual Bag of Words because it was unable to handle "nargin". This meant that SURF was chosen to be used rather than SIFT because of the issues with its implementation. The features extracted from the Feature Extraction algorithm will then be clustered using the clustering algorithm K-means. From this, a Visual Vocabulary is built from the centres of each cluster, with this Visual Vocabulary a classifier is ready to be trained.

Once a Visual Vocabulary has been created, it can then be used to train a Linear SVM as described in 2.1.5.1. Support Vector Machines. This can be done using another function that is part of MATLAB's Computer Vision Systems Toolbox "trainImageCategoryClassifier". This function takes a set of training images and a Visual Vocabulary to create a Gaussian SVM classifier that can then be used to predict the classification of features. A Gaussian SVM has been chosen as it is the most robust method for classifying multiple classifications. The produced classifier can then be evaluated to test its accuracy by testing it against a validation image set. To do this, the image set is divided into two sets training and validation with an 80/20 split as described in 4.4.1 Technical Efficiency. This is performed before the Visual Bag of Words is calculated so the system has no knowledge of the validation set. Evaluation of the classifier can be done using the MATLAB function "evaluate" that outputs a confusion matrix.

5.3.4. Classification

The main system will use the classifier that has been produced to predict the classification of the features within an inputted image. The system will first segment the input images to produce a series of individual plaques and tangles images such as the ones seen in Figure 23(b). While these images are being segmented, the system will calculate and record the centres of each feature. Once the features have been segmented, their classification is then predicted using MATLAB’s classification predication function “predict” which will use the classifier to predict the classification.

Once the features have been classified as either plaques, tangles or noise, the System will then display the original inputted image with the centres of the identified features labelled with a yellow spot for plaques and green spot for tangles. An example of this output can be seen in Figure 24 where a classifier with an accuracy of 60% had been trained with 5 images was able to detect 2/3 plaques and tangles within the image. The system will also output a numeric count of plaques and tangles that have been identified within the image.

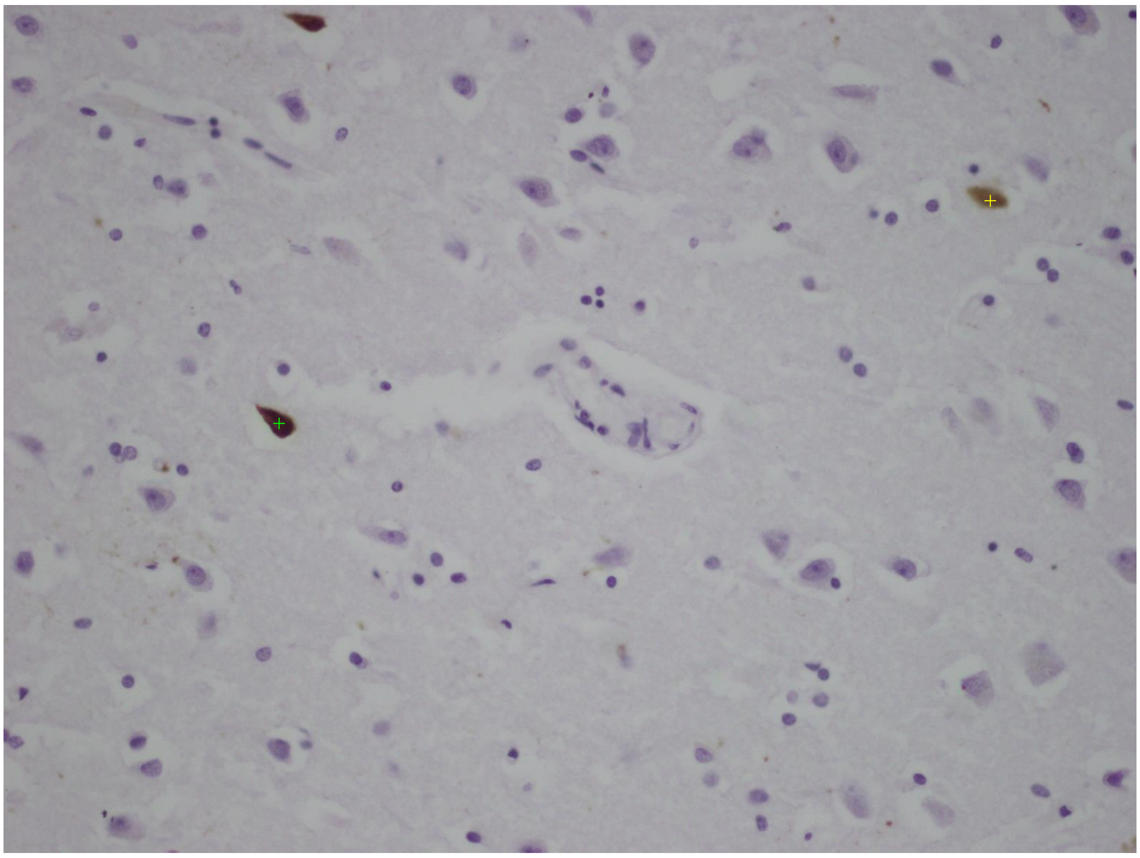
Figure 24. An example output from a classifier that has been trained with 5 images with an accuracy rating of 60%.

6. Conclusion

6.1. Artefact Evaluation

6.1.1. Fabricated Data

To evaluate the artefact, the classifier need to be trained on the test image set to evaluate the classifier. This was not possible at the time of writing because the test images had not been labelled by a medical professional, and therefore the classifier could not be trained correctly and a full evaluation is currently delayed. The importance of labelled data has been shown in the article by Chapelle et al. (1999) where it explains that without labelled data the SVM has to rely on unsupervised learning that can cause issues. This means during the period of waiting for the images to be labelled, a basic evaluation of the system can be conducted to evaluate how the system can classify different shapes.

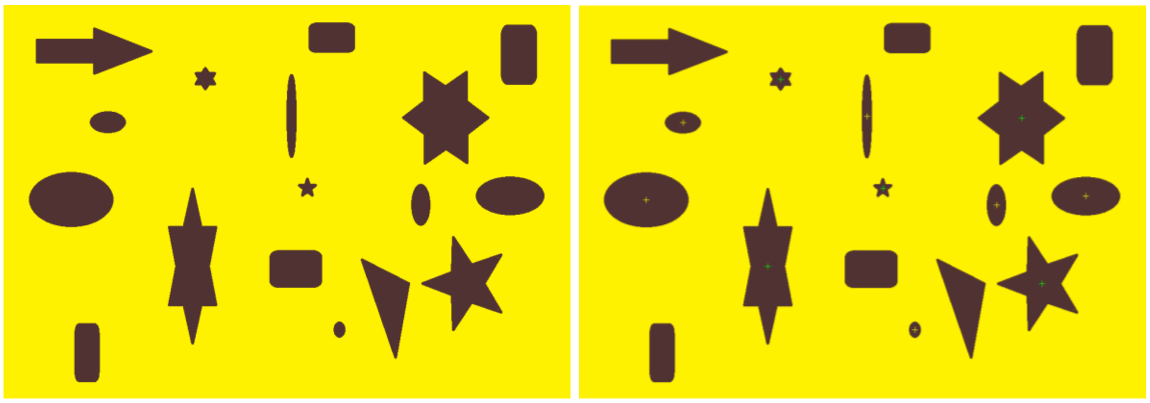

Figure 25. (a) One of the generated test images used to evaluate the system. (b) The result when using the classifier.

To test the basic suitability of the system and to identify any potential weaknesses, a series of test images were produced. These test images were designed to feature three classifications of shapes: stars, circles and noise. Each of these features have been given colour so that the system will not discount the features during the Colour Slicing process, the background has been coloured differently in each image so that it is removed. Ten images were produced, which can be seen in Appendix 4, each featuring a combination of stars, circles and other shapes, all with varying scales and orientations. An example of a generated image can be seen in Figure 25(a). After training the system to identify the stars and circles, the classifier can be evaluated to see how efficiently the system differentiated between the different types of shapes. Metrics were then produced based on how the system classified a validation set of images to show how the system performed.

An image set was built from the features of the generated test images and was then split into two image sets - one for training and one for validation. The training set is then used to build a Visual Vocabulary and train a Gaussian SVM classifier. The classifier is then evaluated against the validation set which is made up of 20% of the image data (3 images). The evaluation produced the Confusion matrix seen in Table 5 that specifies the classification it assigned each known feature in the validation set.

	Star	Circle	Noise
Star	1.00	0.00	0.00
Circle	0.00	1.00	0.00
Noise	0.33	0.00	0.67

Table 5. Confusion Matrix for the Generated Testing Images.

From the generated Confusion matrix, metrics about the classifier can be extracted as seen in Table 6. From looking at the results from the evaluation on the classifier, it can be seen that the number of false positives from the classifier is the reason why the accuracy is only 89% instead of 100%. This is likely due to the fact that the classifier was only trained using 8 images and given that the noise shapes vary, they may have not been trained in the classifier. The True Positive score for the classifier showed that the classifier was able to correctly classify all stars and circles correctly. An example of an image being run through the system with the trained classifier can be seen in Figure 25(b). This demonstrates the effectiveness of classifying shapes by using a Visual Bag of Words and a Gaussian SVM.

	Star	Circle	Overall
Accuracy	(3 + 5)/9 = 0.89	(3 + 6)/9 = 1.0	(0.88 + 1.0)/2 = 0.89
Error Rate	(1 + 0)/9 = 0.11	(0 + 0)/9 = 0.0	(0.11 + 0.0)/2 = 0.05
True Positive	3/3 = 1.0	3/3 = 1.0	(1.0 + 1.0)/2 = 1.0
False Positive	1/6 = 0.16	0/6 = 0.0	(0.16 + 0.0)/2 = 0.08
Specificity	5/6 = 0.83	6/6 = 1.0	(0.83 + 1.0)/2 = 0.91
Precision	3/4 = 0.75	3/3 = 1.0	(0.75 + 1.0)/2 = 0.87

Table 6. Metrics generated from the classifier trained with the Generated Testing images.

6.1.2. Real Data

During the development of the system, there were issues with obtaining labels for the test images. At the current time of writing, the labels have not yet been obtained and the deadline for the project is fast approaching. This meant that an evaluation on the real test data, where the feature's labels have been estimated by the author rather than a medical professional, needed to be conducted. This could lead to inaccuracies in the system as some features may be labelled incorrectly and confuse the classifier. By using this method of evaluation, it means that all the results are currently just estimations on how the system would work with a real labelled data set.

	Tangle	Plaque	Noise
Tangle	0.34	0.00	0.66
Plaque	0.00	1.00	0.00
Noise	0.00	0.00	1.00

Table 7. Confusion Matrix for the Real Testing Images.

Once the image set, generated from 11 images, was built up and labelled by the author (The labelled data can be seen in Appendix 7), the classifier was then trained and evaluated with this data using an 80/20 split on the image set. The system was trained with 17 images of each feature classification (51 in total) and evaluated with 3 of each (9 in total). The resulting Confusion matrix can be seen in Table 7 and the evaluation metrics generated from the Confusion matrix can be seen in Table 8.

	Tangle	Plaque	Overall
Accuracy	(1 + 6)/9 = 0.77	(3 + 6)/9 = 1.0	(0.77 + 1.0)/2 = 0.88
Error Rate	(0 + 2)/9 = 0.23	(0 + 0)/9 = 0	(0.23+ 0)/2 = 0.11
True Positive	1/3 = 0.33	3/3 = 1.0	(0.33 + 1.0)/2 = 0.66
False Positive	0/6 = 0.0	0/6 = 0.0	(0.0 + 0.0)/2 = 0.0
Specificity	6/6 = 1.0	6/6 = 1.0	(1.0 + 1.0)/2 = 1.0
Precision	1/3 = 0.33	3/3 = 1.0	(0.33 + 1.0)/2 = 0.66

Table 8. Metrics generated from the classifier trained with the Real Testing images.

From these statistics, we can see that the classifier has an overall accuracy of 88% being able to identify each of the plaques as plaques, but unfortunately only 33% of the tangles were classified as tangles. We can also see from the Confusion matrix that the other 67% of the tangles were classified as noise - an example of this can be seen in Figure 26 where a large tangle in the centre was classified as noise.

By analysing the labelled features, it came to understanding that after normalising for size the tangles and the majority of the noise looked very similar and could easily confuse the classifier. This size is only artificial after being stretched to normalise the size although this could also be an issue with how the features have been labelled.

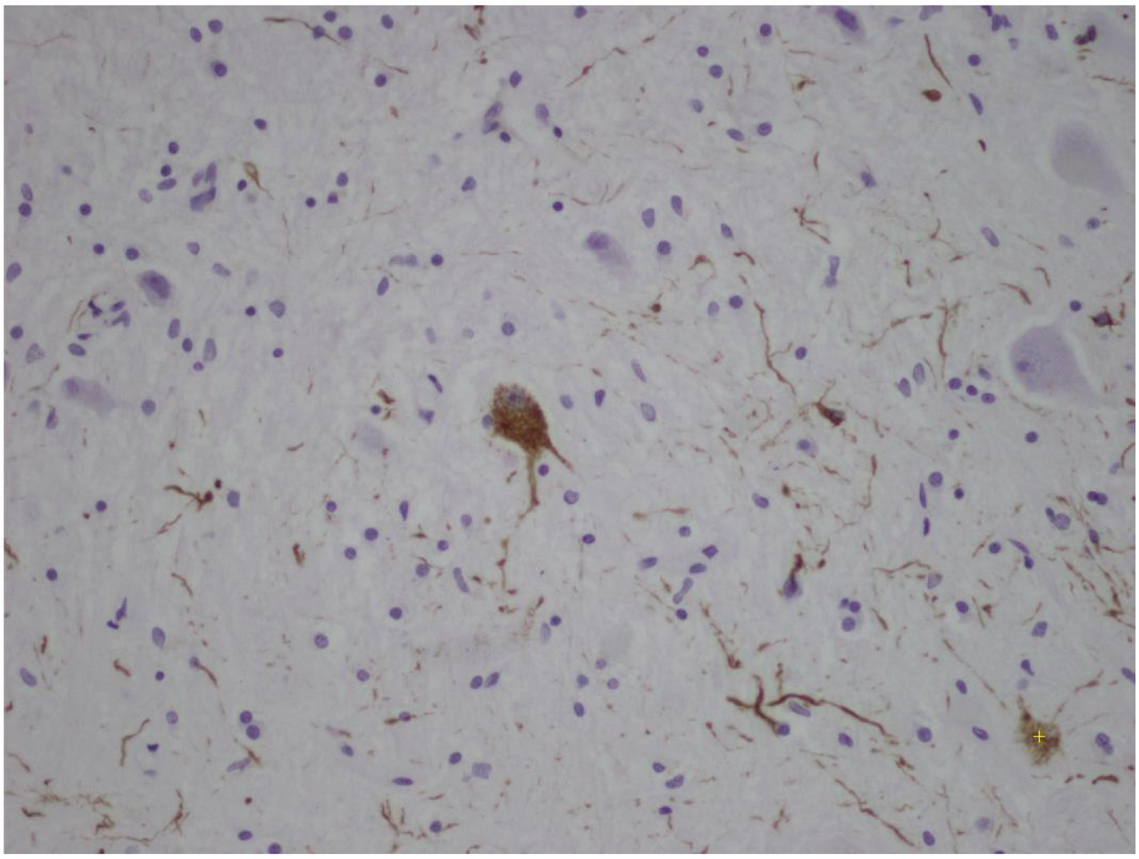

Figure 26. An example result when running the system with a large plaque being misclassified as noise.

This however could also be an issue with the number of images used to train the classifier. The more data that is used to train a classifier, the more accurate the classifier will be. As there were only 11 images available for testing, only 20 images of each classification were then extracted to train the classifier. This may not be a suitable number of images to train the classifier, in order to achieve the accuracy level which is required. Other systems that use Visual Bag of Words and SVM classifiers use large image sets to train and evaluate the classifiers such as the system developed by Vedaldi and Zisserman's (2011) where they used just over 4500 images to train their classifier.

6.2. Objectives

6.2.1. Main Objectives

6.2.1.1. To produce a Pre-Processing solution that will remove noise from the input images without modifying its content.

Pre-Processing only needed to be applied to the images after the Colour Slicing process because small specs of noise remained present in the binary image. Morphological transforms were used as a means to reduce the amount of noise within the image. This process was mostly a success as the majority of the test images had all noise removed as seen in Figure 13(a) and other nosier images had the amount of noise greatly reduced as shown in Figure 13(b). To avoid modifying the content within the image, the system then used bounding boxes to position of each remaining feature then extracting the image from the original source image. This meant that it did not matter if the binary feature lost its shape as it was to be extracted from the unmodified image as seen in Figure 23.

6.2.1.2. To segment the image by colour, extracting all of the stained cells in the image.

By using Colour Slicing, the cells with stained tau proteins were able to be extracted from the input images. Once the colour ranges for the plaques and tangles were defined, binary thresholding was then applied to the image to extract the shapes of the plaques, tangles and other small traces of noise. The results of this process can be seen in Figure 10(b) and 11(b). Figure 10(b) which shows how efficient the process can be when processing image with low amounts of noise where Figure 11 shows the effect that noise can have when Colour Slicing. To remove this noise from within the images, a de-nosing method was applied to the binary images.

6.2.1.3. To perform Feature Detection on the image to identify its features.

To build the Visual Bag of Words, a Feature Detection algorithm needs to be performed to detect and then extract the descriptors from all the images. This has been able to be accomplished by using MATLAB's implementation of the SURF algorithm. The SURF algorithm was used in the artefact instead of the SIFT algorithm because SIFT has been patented making it difficult to locate a reliable implementation of the SIFT for MATLAB. The SURF algorithm works reliably in the artefact, being able to extract over 25,000 features from the ten images from the generated image set seen in 6.1.1. Fabricated Data.

6.2.1.4. To classify the identified features as either plaques or tangles.

A full evaluation on the classifier has not been able to be performed because of the lack of a correctly labelled image data. Thus, a rough evaluation with estimated labels had to be performed as well as an evaluation with fabricated images. From the estimated data, the system was able to identify plaques and tangles with an average accuracy of 88% but with a high number of false negatives. This may be an issue with how the training data was labelled but without a full evaluation, the result is unclear. Although by using the fabricated images, the system was able to be trained to classify three categories of shapes with an 89% accuracy with an even number of false positives and false negatives. This shows that the system is fully capable of classifying robust shapes in a noisy environment with a high level of accuracy and precision.

6.2.2. Optional Objectives

During the development of the project, radical changes were made to software development methodology used to develop the artefact. Initially the Incremental methodology was to be used but because of complications with development, the Evolutionary methodology was instead chosen. The development then went through a number of evolutions to reach its current position; this took more time than was originally expected and tasks involving development overran. This meant to continue to meet the specified milestones the project's lower priority tasks were circumvented. This unfortunately led to the OpenCV implementation and the GUI not to be developed as more time was spent creating the system. These objectives were given lower priority as they do not help reach the aim of the project and instead help strengthen the aim. By skipping these tasks, it did however mean that the project was able to meet the milestones, progress as according to the project schedule and develop a solution that fits the aim.

6.3. Aim

The overall aim of the project was to “To automate the process of identifying senile plaques and neurofibrillary tangles from high resolution scans of the Hippocampus, in order to aid in the diagnosis of Alzheimer’s disease.” This aim has been achieved by developing a system that can identify plaques and tangles from within a scan of the Hippocampus. The system is able to accomplish this by using machine learning by training a classifier with a labelled image set built of plaques, tangles and noise that can be found within the scans. The system then uses the classifier to classify any features found within the any input image. Tests on the classifier showed an accuracy of 88% when trained with a small set of images labelled by a non-medical professional. This leads to the belief that if the system was trained with a larger data set which was labelled by a medical professional, the system would be much more efficient.

The evaluation also found that the classifier’s prediction of tangles had a false negative rating of 33%. It upholds that for every 3 tangles, 1 would be misclassified as noise. This can be a major issue in diagnosis as false negatives could mean that someone with Alzheimer’s disease may be diagnosed to not have it. Given the scans are taken from post mortem brains, there is no issue with treatment but it could be an issue with research or family of the deceased.

6.4. Future Work

Given that the project was conducted within a strict time scale, several tasks which would have helped strengthen the aim were not able to be conducted. The one of these tasks was to evaluate the system using a larger set of labelled testing images. By conducting this full evaluation, a much more complete indication of how the system classifies the features could have been given. This would have also created a more accurate Confusion matrix which would give a better understanding on how the system worked.

The system's implementation could in the future be developed in a platform other than MATLAB such as OpenCV.

This was originally an optional objective for the project which was not able to be met because of complications with development. However an OpenCV implementation of the system would perform better because features, such as parallel processing, are available in OpenCV. OpenCV would also not be restricted by the licencing which MATLAB is restricted by; this would mean that anyone would be able to use, modify and update the system without having to also own a MATLAB licence.

Another optional objective that was not able to be completed was the development of a GUI for the system so that medical professionals are able to use the system with ease. This would encourage the medical professionals that would be using the system to use the system to its maximum potential. This would mean more users would be able to train and use the system, improving its accuracy for identifying plaques and tangles.

Other improvements, which could have made to the system, could include improved Pre-Processing to the input images such as histogram equalization. This will make sure that all the images which are inputted into the system and confirmed to be of a certain standard throughout the system. The system could also improve its Colour Slicing process by first converting the image to a HSV colour model to make colour representation more meaningful, which can then be used to extract certain ranges of colour.

7. Reflective Analysis

The following Reflective analysis was conducted by utilizing Boud's Triangular Representation - a model used to construct a reflective analysis.

The first step when starting this project was to conduct research to understand how to approach the problem that had been put in place. During the research for the project, a large range of potential algorithms were identified that could have been used within the artefact. For each algorithm, a review of literature was conducted to identify the potential best suited algorithm to work within the system. This was done by looking at other Computer Vision studies that implemented the algorithms in question and seeing their results. This method did help find some very useful algorithm but because of how the search was conducted few Medical Imaging studies that set out to accomplish a similar aim were reviewed. From these studies, different approaches to how a full system could implemented could have been reviewed including the algorithms they used, methodologies and tools. This shows that a more intensive literature review should have been conducted looking especially at studies with similar goals.

Once background research has been conducted, the aims and objectives of the project are then decided upon based on the research. For this project, a singular all-encompassing aim was used along with several objectives that are used to reach this aim. Also present were optional objectives which would aid in strengthening the aim. These optional objectives were however too ambitious for the project and were unable to be completed because of their complexity and the limited timescale of the project. For example, the optional objective to build a graphical user interface for the system would have been a task that would have used a lot of time because of the necessary planning and testing which would have been involved in its development. To avoid this the project's the aims and objectives should have been more heavily based on the background research to see if certain tasks is feasible and have been conducted as part of other studies.

The software methodology that had been chosen to be used in the initial project plan was the Incremental methodology. As the development of the artefact progressed, it became clear that the incremental methodology was not able to adjust to change as quickly as required. This led to the Evolutionary methodology being used for development instead, as it allows for rapid prototyping of the artefact. Implementing a new software methodology during development caused a number of issues with the planning for the project as the plans were based around the incremental nature of the project. This shows that an in depth understanding of the nature of the project is crucial when planning the development methodology of an artefact.

In the project, planning a project schedule was created that specified the timescales of all the specified tasks that need to be completed to meet the aim and objectives of the project. In the project schedule, each of the tasks was assigned a priority specifying how important it is to reaching the aim of the project. These priorities indicated which tasks were able to be skipped and still develop a full system. Even with the change of software methodology and a number of issues slowing down progress of the project, the priorities on the schedule helped keep to milestones as only the tasks necessary for the completion of the project were completed. This shows that by assigning priorities to tasks, it can help a project stay flexible in an uncertain development environment.

To reduce the chance of risks and in order to plan risk mitigation, a risk matrix was constructed that detailed the risks that had the potential of occurred throughout the project. This basic form of risk identification meant that all the risks which could occur during its course were acknowledged. The risk matrix was created at the beginning of the project before the Evolutionary methodology was implemented. When the Evolutionary methodology was implemented, it meant that large parts of the risk matrix became invalid as the issues that had the potential to happen during the course of the project changed as the direction of the project changed. This shows the importance of selecting the correct methodology at the beginning of the project to avoid any issues with having risks without planned mitigations. It also shows how risk avoidance could have been handled better by updating the risk matrix each time a new evolution was designed.

The first evolution of the system used the Circular Hough Transform but it was found to not work as a suitable method of classification, and was replaced in favour of the Generalised Hough Transform. Given that the Generalised Hough Transform is not included in any of MATLAB’s toolboxes, a third part implementation would need to be sourced. An implementation was found but it did not account for any differences in scale from the template and the image. This therefore meant that the Generalised Hough Transform was not able to be implemented into the system because of the limitations from the third party implementation. This shows that the reliability on third party implementations should be tested before committing to using them within system in order to avoid these situations.

The second evolution used more recent, less established algorithms and because of this there was no standard implementation for them and an implementation of these algorithms had to be found. Implementations for the algorithms that worked in MATLAB were found but this meant that the system had to rely on third party implementations. This had proven unsuccessful in the previous evolution and once again was found unsuccessful after a series of trials to get the implementation working, for example installing Ubuntu, downgrading the GCC compiler, etc. This only wasted time which could have been used to advance the project and only proved the previous point of testing the third party implementations before relying on them. This also shows that when creating a standard system, established methods should be used that have been to work.

In the third evolution, the system used a form of supervised machine learning where a SVM classifier was trained with a Visual Bag of Words from labelled image sets. The way that the system works is very different from how the two previous evolutions worked, demonstrating that the project had been progressing in the wrong direction for a large portion of its development. This could have been avoided if more meetings with the project supervisor were have been organised to make sure that the project was progressing in a direction that would have resulted in a working solution that met the aim of the project.

During the course of the project, there were issues getting the test images labelled. This meant that when the classifier was trained, the identification of the plaques and tangles were estimated. This lead to inaccuracies as features may be trained as the incorrect classification. Without the labelled data, the artefact was not able to be fully evaluated and instead a rough evaluation was conducted. At the start of a project, a check should have been conducted to confirm that all data, that would be needed to develop the artefact, would be available in order to avoid these issues.

8. References

Ahmed, K. (2013) Advantages and Disadvantages of Using MATLAB code for Solving Differential Equations in Engineering Application. *International Journal of Engineering*, 7(1) 25-31.

Allinson, N. (2015) *Pathology Images.* [email] Sent to Jacob Carse, 14 October.

Alzheimer's Association (2015) *Diagnosis of Alzheimer's Disease and Dementia* [online] Chicago: Alzheimer's Association. Available from http://www.alz.org/alzheimers_disease_diagnosis.asp [Accessed on 18 October 2015].

Ambler, S. (2014) *Agile Best Practice: Prioritized Requirements.* [online] Available from http://agilemodeling.com/essays/prioritizedRequirements.htm [Accessed on 14 March 2016].

Bagon, S., Brostovski, O., Galun, M. and Irani, M. (2010) Detecting and Sketching the Common. In: *Computer Vision and Pattern Recognition*, San Francisco 13-18 June, Los Alamitos, USA: The Institute of Electrical and Electronics Engineers, 33-40.

Bagon, S., Brostovsky, O., Galun, M. and Irani M. (2010) *Matlab implementing the sketching part of Shai Bagon, Or Brostovsky, Meirav Galun and Michal Irani's Detecting and Sketching the Common.* [online] Available from http://www.wisdom.weizmann.ac.il/~bagon/matlab_code/SketchCommonCVPR10_v1.1.tar.gz [Accessed on 5 April 2016].

Bagon, S., Brostovsky, O., Galun, M. and Irani, M. (2010) Detecting and Sketching the Common. [online] Available from http://www.wisdom.weizmann.ac.il/~vision/SketchTheCommon/ [Accessed 4 April 2016].

Ballard, D. (1981) Generalizing the Hough Transform to Detect Arbitrary Shapes. *Pattern Recognition*, 13(2) 111-122.

Bay, H., Ess, A., Tuytelaars, T. and Van Gool, L. (2008) SURF: Speeded up Robust Features. *Computer Vision and Image Understanding. 110(3) 346-359.*

Blondel, M. (2010) *Support Vector Machines in Python.* [online] Available from http://www.mblondel.org/journal/2010/09/19/support-vector-machines-in-python/ [Accessed on 17 March 2016].

Braak, H. and Braak, E. (1995) Staging of Alzheimer's disease-related neurofibrillary changes. *Neurobiology of Aging*, 16(3) 271-278.

Braidy, N., Poljak, A., Marjo, C., Rutlidge, H., Rich, A., Jayasena, T., Inestrosa, T. and Sachdec, P. (2014) Metal and Complementary Molecular Bioimaging in Alzheimer's Disease. Frontiers in Aging Neuroscience, 6 138.

Brooks, F. (1986) No Silver Bullet Essence and Accidents of Software Engineering. Information Processing, 86 10-19.

Canonical (2016) *Ubuntu.* [software] Version 14.04.4. Canonical. Available from http://www.ubuntu.com/desktop [Accessed on 5 April 2016].

Capelle, O., Haffner, P. and Vapnik, V. (1999) Support Vector Machines for Image Classification. *IEEE Transactions on Neural Networks,* 10(5) 1055-1064.

Chatfield, K., Gulshan, V., Philbin, J. and Zisserman, A. (2009) *Self-Similarity Descriptor Code.* [online] Oxford: University of Oxford. Available from: http://www.robots.ox.ac.uk/~vgg/software/SelfSimilarity/ [Accessed on 4 April 2016].

Chaudhuri, S., Chatterjee, S., Katz, N., Nelson, M. and Goldbaum, M. (1989) Detection of Blood Vessels in Retinal Images Using Two-Dimensional Matched Filters. IEEE Transactions on Medical Imaging, 8(3) 263-269.

Cox, L. (2008) What's wrong with risk matrices?. *Risk Analysis, 28(2) 497-512.*

Cras, P., Kawai, M., Lowery, D., Gonzalez-DeWhitt, P., Greenberg, B. and Perry, G. (1991) Senile plaque neurites in Alzheimer disease accumulate amyloid precursor protein. *Proceedings of the National Academy of Sciences*, 88(17) 7552-7556.

Dance, C., Wilamowski, J., Fan, L., Bray, C. and Csurka, G. (2004) Visual Categorization with Bags of Keywords. In: *ECCV International Workshop on Statistical Learning in Computer Vision,* Prague 15 May, Meylan, France: Xerox Research Center Europe.

Draug, C. (2015) *image.* Version 2.4.1 Available from http://octave.sourceforge.net/image/index.html [Accessed on 10 March 2016].

El-Naqa, I., Yang, Y., Wernick, M., Galatsanos, N. and Nishikawa, M. (2002) A support Vector Machine Approach for Detection of Microcalcifications. *IEEE Transactions on Medical Imaging 21(12) 1552-1563.*

Eppel, S. (2015) *Generalized Hough Transform with Rotation.* [software] Version 1.0. Available from http://www.mathworks.com/matlabcentral/fileexchange/50330-generalized-hough-transform-with-rotation [Accessed 31 March 2016].

Etzioni, D., Liu, J., Maggard, M. and Ko, C. (2003) The Aging Population and Its Impact on the Surgery Workforce. Annals of Surgery, 238(2), 170-177.

Fermin, C., Gerber, M. and Torre-Buenot, J. (1992) Colour Thresholding and Objective Quantification in Bioimaging. Journal of Microscopy, 167 85-95.

GitHub (2016) GitHub [online] San Francisco: GitHub. Available from https://GitHub.com/ [Accessed on 10 March 2016].

Gouillart, E. and Varoquaux, G. (2011) Image Manipulation and Processing using Numpy and Scipy. [online] Umeå University. Available from http://www.tp.umu.se/~nylen/fnm/pylect/advanced/image_processing/index.html [Accessed on 16 March 2016].

Granados, J. (2012) *Using Matlab was a mistake.* [blog entry] 7 April. Available from https://joelgranados.com/2012/04/07/using-matlab-was-a-mistake/ [Accessed on 8 April 2016].

Grave, P. (2009) *The Undergraduate Research Handbook*. Basingstoke: Palgrave Macmillan.

Haris, K., Efstratiadis, S., Maglaveras, N., Gourassas, J. and Lauridas, G. (2001) Artery Skeleton Extraction Using Topographic and Connected Component Labelling. *Image Processing,* 2 339-342.

Hossain, E., Babar, M., Paik, H. and Verner, J. (2009) Risk Identification and Mitigation Processes for Using Scrum in Global Software Development: A conceptual Framework. In: *16th Asia-Pacific Software Engineering Conference.* Penang 1 - 3 December, Penang, Malaysia: IEEE, 457-464.

House of Commons Library Research (2010) *Key Issues for the New Parliament 2010*. RP1037. London: House of Commons Library. Available from http://researchbriefings.files.parliament.uk/documents/RP10-37/RP10-37.pdf [Accessed 18 October 2015].

Illingworth, J. and Kittler, J. (1987) The Adaptive Hough Transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5) 690-698.

Ishizuka, K., Kimura, T., Igata-yi, R., Katsuragi, S., Takamatsu, J. and Miyakawa, T. (1997) Identification of Monocyte Chrmoattractant Protein-1 in Senile Plaques and Reactive Microglia of Alzheimer's Disease. Psychiatry and Clinical Neurosciences, 51 135-138.

ISTQB (2016) *What is Incremental model – advantages, disadvantages and when to use it?* [online] Available from http://istqbexamcertification.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/ [Accessed on 15 March 2016].

Itseez (2016) *OpenCV*. [software] version 3.1. San Francisco, USA: Itseez. Available from opencv.org [Accessed on 15 March 2016].

Jamil, N., Sembok, T. and Bakar, Z. (2008) Noise Removal and Enhancement of Binary Images Using Morphological Operations. *Information Technology,* 4 1-6*.*

Jet Brains (2016) *CLion* [software] version 1.2. Prague, Czech Republic. Available from https://www.jetbrains.com/clion/ [Accessed on 15 March 2016].

Johnson, Eva. (2015) *Why Upgrade from Waterfall to Evolutionary Development.* [online] Intland Software. Available from http://intland.com/blog/why-upgrade-from-waterfall-to-evolutionary-development-evo/ [Accessed on 15 March].

Kassim, A., Tan, T. and Tan, K. (1999) A Comparative Study of Efficient Generalised Hough Transform Techniques. *Image and Vison Computing,* 17(10) 737 – 748.

Ke, Y., Sukthankar, R. and Hebert, M. (2007) Event Detection in Crowded Videos. In: IEEE 11th International Conference on Computer Vision, Rio de Janeiro 14-20 October, Rio de Janeiro, Brazil: The Institute of Electrical and Electronics Engineers, 1-8.

Kelly, A. (2013) *3 Styles of Agile: Iterative, Incremental, and Evolutionary.* [online] DZone. Available from https://dzone.com/articles/3-styles-agile-iterative [Accessed on 15 March 2016].

Khronos Group (2016) OpenCL [software] Version 2.1. Khronos Group. Available from https://www.khronos.org/opencl [Accessed on 20 March 2016].

Kiermire, A. (2011) *The Application of the Pareto Principle in Software Engineering.* [online] Available from http://www2.latech.edu/~box/ase/papers2011/Ankunda_termpaper.PDF [Accessed on 9 April 2016].

Krupinski, E. and Jiang, Y. (2008) Anniversary Paper: Evaluation of Medical Imaging Systems. *Medical Physics,* 35(2) 645 – 659.

Levi, G. (2013) *Bag of Words Models for Visual Categorization.* [online] Available from: http://gilscvblog.com/2013/08/23/bag-of-words-models-for-visual-categorization/ [Accessed on 3 April 2016].

Li, B. and Que, D. (2011) Medical Images Denoising Based on Total Variation Algorithm. In International Conference on Environment Science and Biotechnology, Maldives 25-27 November, Chemical, Biological, & Environmental Engineering, 227-234.

Lowe, G. (1999) Object recognition from local scale-invariant features. In: *Proceedings of the International Conference on Computer Vision. Kerkyra 20-27 September , Washington D.C., USA: The Institute of Electrical and Electronic Engineers, 1150-1157.*

Lukachevich, P., Zalesky, B. and Ablameyko, S. (2011) Medical Image Registration Based on SURF Detector. *Pattern Recognition and Image Analysis,* 21(3) 519-521.

Maini, R and Aggarwal, H. (2009) Study and Comparison of Various Image Edge Detection Techniques. *International Journal of Image Processing* 3(1) 1-12.

MathWorks (2016) *Computer Vision Systems Toolbox.* [app] Version R2016a. Available from http://uk.mathworks.com/products/computer-vision/ [Accessed on 10 March 2016]

MathWorks (2016) File Exchange – MATLAB Central [online] Available from: http://uk.mathworks.com/matlabcentral/fileexchange/ [Accessed 31 March 2016].

Mathworks (2016) *Find Circle Using Circular Hough Transform.* [online] Available from http://uk.mathworks.com/help/images/ref/imfindcircles.html?refresh=true#outputarg_metric [Accessed 29 March 2016].

MathWorks (2016) *Image Processing Toolbox*. [app] version R2016a. Available from http://uk.mathworks.com/products/image/ [Accessed 10 March 2016].

MathWorks (2016) *Matlab.* [app] version R2016a. Available from http://uk.mathworks.com/products/matlab/ [Accessed 10 March 2016].

Microsoft (2016) *Microsoft Project*. [app] Version 2016. Microsoft. Available from https://products.office.com/en-gb/project/project-and-portfolio-management-software [Accessed 13 March 2016].

Microsoft (2016) *Visual C++ Compiler* [software] version 2015. Washington, USA: Microsoft. Available from https://www.microsoft.com/en-gb/download/details.aspx?id=41151 [Accessed on 15 March 2016].

Microsoft (2016) *Visual Studio.* [software] version Community 2015. Washington, United States: Microsoft. Available from https://www.visualstudio.com/ [Accessed on 15 March 2016].

Microsoft. (2016) Windows Forms. [online] Redmond: Microsoft. Available from https://msdn.microsoft.com/en-us/library/dd30h2yb(v=vs.110).aspx [Accessed 10 March 2016]

Misra, A. (2010) *Incremental Engineering of Computer Vision Systems.* PhD. University of New South Wales.

National Patient Safety Agency (2008) *A risk matrix for risk managers.* 0676. London: NHS. Available from http://www.nrls.npsa.nhs.uk/EasySiteWeb/getresource.axd?AssetID=60149& [Accessed 10 April 2016].

Nayak, J., Bhat, P., Acharya, R., Lim, C. and Kagathi, M. (2008) Automated Identification of Diabetic Retinopathy Stages Using Digital Fundus Images. *Journal of Medical Systems*, 32(2) 107-115.

Nvidia (2016) CUDA [software] Version 7.5. Nvidia. Available from http://www.nvidia.com/object/cuda_home_new.html [Accessed on 20 March 2016].

OpenCV (2016) *CUDA* [online] San Francisco, USA: Itseez. Available from http://opencv.org/platforms/cuda.html [Accessed on 15 March 2016].

Oracle (2016) *NetBeans IDE* [software] version 9.1. Redwood City, USA. Available from https://netbeans.org/ [Accessed on 15 March 2016].

Oracle (2016) *VirtualBox.* [software] Version 5.0. Oracle Available from https://www.virtualbox.org/ [Accessed on 5 April 2016].

Panchal, P., Panchal, S. and Shah, S. (2013) A Comparison of SIFT and SURF. *International Journal of Innovative Research in Computer and Communication Engineering,* 1(2) 323-327.

Patil, R. and Bhalchandra, A. (2012) Brain Tumour Extraction from MRI Images Using MATLAB. *International Journal of Electronics, Communication and Soft Computing Science and Engineering*, 2(1) 1-4

Philip, K., Dove, E., McPherson, D., Gorreiner, N., Stanford, W. and Chandran, K. (1994) The Fuzzy Hough Transform-Feature Extraction in Medical Images. *Transactions on Medical Imaging,* 13(2) 235 – 240.

Rodriguez, A., Herrera, A. and Muller, H. (2015) Meaningful Bags of Words for Medical Image Classification and Retrieval. *Health Monitoring and Personalized Feedback using Multimedia Data,* 1 73-93.

Rudin, L., Osher, S. and Fatemi, E. (1992) Nonlinear total variation based noise removal algorithms. Physica D: Nonlinear Phenomena, 60 259-268.

Sharp, J. and Howard, K. (1996) *The Management of a Student Research Project*. 2nd edition. Aldershot: Gower.

Shechtman, E. and Irani, M. (2007) Matching Local Self-Similarities across Images and Videos. In: Computer Vision and Pattern Recognition, Minneapolis 17-22 June, Los Alamitos, USA: The Institute of Electrical and Electronics Engineers, 1-8.

Smereka, M. and Dulęba, I. (2008) Circular Object Detection Using A Modified Hough Transform. *International Journal of Applied Mathematics and Computer Science*, 18(1) 85-91.

Solaiman, B., Burdsall, B. and Roux, C. (1998) Hough Transform and Uncertainty Handling. Application to Circular Object Detection in Ultrasound Medical Images. In: International Conference on Image Processing, Chicago 4-7 October, Los Alamitos, USA: IEEE, 828-831.

SoundSoftware (2011) *Why use version control?* [online] London: Queen Mary University of London. Available from http://soundsoftware.ac.uk/why-version-control [Accessed 10 March 2016].

Stewart, S., MacIntyre, K., Capewell, S. and McMurry, J. (2003) Heart failure and the aging population: an increasing burden in the 21st century?. Heart, 89(1) 49-53.

Sundberg, T. (2015) When is evolutionary design a good way to implement software? [blog entry] 27 October. Thomas Sundberg. Available from https://thomassundberg.wordpress.com/2015/10/27/when-is-evolutionary-design-a-good-way-to-implement-software/ [Accessed on 9 April 2016].

Tang, H., Wu, E., Ma, Q., Gallagher, D., Perera, G. and Zhuang, T. (2000) MRI brain image segmentation by multi-resolution edge detection and region selection. Computerized Medical Imaging and Graphics, 24(6) 349-357.

U.S. Department of commerce, Economics and Statistics Administration (2014) An Aging Nation: The Older Population in the United States. P25-1140. Suitland: United States Census Bureau. Available from https://www.census.gov/prod/2014pubs/p25-1140.pdf [Accessed on 21 October 2015].

Vedaldi, A and Zisserman, A. (2011) *Image Classification Practical*. [online] Paris: École normale supérieure. Available from http://www.di.ens.fr/willow/events/cvml2011/materials/practical-classification/ [Accessed on 12 April 2016].

VLFeat (2013) *SIFT detector and descriptors.* [online] Available from http://www.vlfeat.org/overview/sift.html [Accessed on 3 April 2016].

VLFeat (2013) *VLFeat*. Version 0.9.20. Available from http://www.vlfeat.org/index.html [Accessed on 10 March 2016].

Vovk, U., Pernus, F. and Likar, B. (2007) A Review of Methods for Correction of Intensity Inhomogeneity in MRI. IEEE Transactions on Medical Imaging 26(3) 405-421.

Zhao, G., Yuan, J., Xu, J. and Wu, Y. (2011) Discovering the Thematic Object in Commercial Videos. *IEEE MultiMedia,* 31(3) 56-65.

Zhi, L., Zhang, S., Zhao, D., Zhao, H., Lin, S., Zhoa, D. and Zhao, H. (2009) Medical Image Retrieval Using SIFT Feature. In: CISP '09. 2nd International Congress on Image and Signal Processing, 2009. Tianjin 17-19 October, China: The Institute of Electrical and Electronics Engineers, 1-4.

9. Appendices

Appendix 1 – Project Schedule

Appendix 2 – Gantt Chart

Appendix 3 – Risk Matrix

Appendix 4 – Generated Test Images

Appendix 5 – Generated Test Results

Appendix 6 – Test Images

Appendix 7 – Estimated Labelled Image Set

Appendix 8 – Test Images Results

Appendix 9 – MATLAB Solution

All Appendices can be found on the CD attached to this Document.