

Developing a Computer Aided Diagnosis System to Identify Pulmonary Nodules in Computed Tomography Scans

Stuart Clark - C1353032

Supervised by Paul Rosin

Moderated by Xianfang Sun

Final Year Project, BSc Computer Science

School of Computer Science and Informatics, Cardiff University

May 5, 2017

Abstract

A pulmonary nodule is a small somewhat round growth on the lungs. Approximately 40% of pulmonary nodules found turn out to be malignant. Studies have shown that using computer aided diagnosis systems can result in earlier detection of pulmonary nodules, saving lives. This report details the design, iterative implementation and evaluation of a prototype computer aided diagnosis system used to detect pulmonary nodules in computed tomography scans. The system produced at the end of the project had an overall system sensitivity 0.316 and 3782.416 FPs/per stack. This poor performance was largely due to the use of poorly performing segmentation techniques, however suggestions are made as to how these techniques could be improved. The systems greatest merit was in the classification stage where a TP rate of 0.932 and a FP rate of 0.099 was achieved. The classifier used to obtain these results was an implementation of a Support Vector Machine trained using 382 nodule cross sections and tested using 117 nodule cross sections. 49 features were used for classification including what is thought to be a novel adaptation of Local Ternary Patterns. Preliminary tests showed this feature to be the best feature used by the system.

Acknowledgements

I would like to take this opportunity to express my whole hearted gratitude to the people who have supported me over the duration of this project, the years that built up to it and the years to come. A massive thank you to:

- Paul Rosin for lending me his invaluable guidance and expertise throughout the project. I have no doubt the without his assistance the project would have been far less of a success.
- All my family for their continued support, I would be nowhere without them.
- My house mates past and present for keeping me fed and sane during the periods when I have been extremely busy during my time at university.

Contents

1	Introduction	7
2	Background	12
2.1	Pulmonary Nodule Detection	12
2.2	Computed Tomography Scans	15
2.3	Existing Computer Aided Diagnosis Systems	17
2.3.1	Noise Removal	17
2.3.2	Case Study	18
2.4	Dataset Selection	23
3	Approach	25
3.1	Requirements	25
3.2	Development Strategy	26
3.3	Tools and Libraries	26
3.4	Initial Design	28
4	Implementation	31
4.1	Version 0.1	32
4.1.1	Acquisition	32
4.1.2	GUI	34
4.1.3	Configuration	35
4.1.4	Subset Selection	36
4.1.5	Segmentation	37
4.1.6	ROI Classification	42
4.1.7	Classifier Training and Testing	43
4.1.8	Evaluation	44
4.2	Version 0.2	45
4.2.1	Segmentation	45
4.2.2	Classifier Training and Testing	49
4.2.3	Evaluation	49
4.3	Version 0.3	51
4.4	Version 0.4	52
4.4.1	Segmentation	52
4.4.2	Classifier Training and Testing	57
4.4.3	Evaluation	57
4.5	Version 0.5	58
4.5.1	Features	60
4.5.2	Classifier Testing	64
4.5.3	Evaluation	65
5	Future Work	67
6	Conclusions	69

7	Reflection on Learning	71
8	Glossary	72
9	Abbreviations	72
10	Appendices	73
10.1	Appendix 1 - V0.1 Segmentation	73
10.2	Appendix 2	76
10.3	Appendix 3	78
10.4	Appendix 4	80
10.5	Appendix 5	81
10.6	Appendix 6	82

List of Figures

1	Survival rates for men women and adults diagnosed with lung cancer (Cancer Research UK 2014b).	7
2	One year net survival rate for lung cancer patients (Cancer Research UK 2014b).	8
3	Estimated cost of treating individuals with cancer at different stages.	9
4	The model of CT scanner that was used to create the images used in this project.	10
5	A basic diagram identifying the key tissues that make up the human lungs (Lung Cancer Alliance 2017).	12
6	An annotated example of a CT cross section.	13
7	An example of a SPN.	14
8	An example of a JPN.	14
9	A CT scanner with the X-ray source and detector shown in three positions (Gibbs 2013).	16
10	Top level block diagram of CAD system developed by Tan et al. (2011)	20
11	An Entity Relationship Diagram showing the initial design for the structure of the database.	30
12	A flow chart showing the top level design for the pipeline used to train the classifier.	31
13	A class diagram showing the main classes used in Lungs.	31
14	The distribution of slice thicknesses for stacks created using the Sensation 16 CT scanner.	38
15	Frequency of voxel intensities comparison.	41
16	Cumulative frequency of voxel intensities comparison.	42
17	Distribution of minimum nodule voxel intensities.	46
18	A one dimensional representation of the model used for SPNs.	46
19	A histogram showing the distribution of voxel intensities for the unmatched nodules.	50
20	A histogram showing the distribution of voxel intensities for the unmatched nodules using 16 bit imagery.	51

21	A closer look at Figure 20	52
22	A visual representation of the DOG pyramid implemented (Lowe 2004) . . .	53
23	A visual example of the main steps in Algorithm 9.	56
24	Distribution of match scores frequencies for nodules matched to a ground truth by at least one voxel.	58
25	Distribution of match scores for nodules matched to a ground truth by at least one voxel.	59
26	An example match score of 0.25. The ROI is filled green and the nodule outlined in red.	60
27	The distribution estimate nodule areas.	61
28	Example LTP neighbourhood for a voxel found at the edge of an ROI.	63
29	An example slice annotated using the classifier. Green regions indicate ROIs classified as nodules, orange regions indicate ROIs classified as non-nodules and the red contour shows a nodule identified by the ground truth	66
30	An unprocessed slice.	76
31	The results of filtering the unprocessed slice. The parameters chosen for this filter where not the optimum ones but instead were selected to exaggerate the effect of the filter.	76
32	The result of thresholding the filtered slice.	76
33	The result of applying opening to the thresholded slice.	76
34	The result of extracting the connected components from the opened slice and rejecting the one with the largest area.	77
35	The largest ROI returned from by the solitary nodule segmentation stage. A juxtapleural nodule is highlighted by the red square.	78
36	The convex hulls of the plural cavities, and a false positive (the smallest region).	78
37	The eroded hulls.	78
38	The mask created by inverting the eroded plural hulls.	78
39	The results of subtracting the mask from the largest ROI. A juxtapleural nodule is highlighted by the red square	79
40	The original slice.	81
41	The largest ROI returned from solitary nodule segmentation	81
42	The internal cavities of the largest ROI	81
43	The results of applying the mask to the original image	81

List of Tables

1	The effect of CAD on false positive rate and sensativity (Sahiner, Berkman et al. 2009).	10
2	Example CT numbers for various tissues, values taken from Webster (1988) .	16
3	The PSNR & MSE values for the different filters (Vijaya and Suhasini 2014). .	18
4	The effectiveness of existing CAD systems (NI = Not Informed) (Firmino et al. 2014).	19
5	Table of features used by Tan et al. (2011)	22
6	The results of running the aggregation in Listing 5	37

7	V0.2 Segmentation Results.	50
8	V0.5 Segmentation Results.	58
9	Result of testing and training the classifier with the two JPN segmentation techniques.	60
10	Result of testing and training different Weka classifiers using undersampling.	64
11	Result of testing and training different Weka classifiers when sampling each nodule 9 times.	65
12	Analysis of how well requirements were met.	70
13	Analysis of how well requirements were met continued.	71
14	Result of testing and training different Weka classifiers using undersampling.	82

1 Introduction

Lung cancer is the most common form of cancer across the globe. In 2012 alone there were approximately 1,825,000 new cases diagnosed (Cancer Research UK 2014a). Lung cancer is a highly debilitating disease and is often fatal. Data provided to Cancer Research UK by the London School of Hygiene and Tropical Medicine in 2014 provided survival rates for patients in England and Wales. The patients were aged 15-99 and were diagnosed with Non-Small Cell Lung Cancer between 2010-2011. This is the most common form of lung cancer. Figure 1 shows the aforementioned data plotted on a graph. In the figure all of the patients described above are included as Adults and either Men or Women.

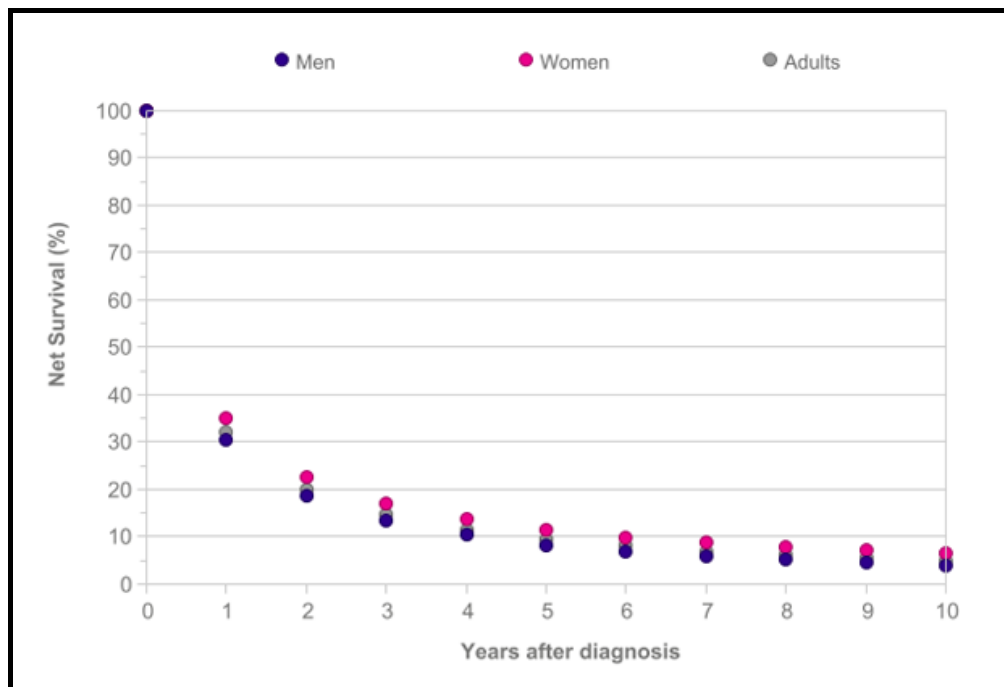


Figure 1: Survival rates for men women and adults diagnosed with lung cancer (Cancer Research UK 2014b).

Survival rates of ≥ 5 years were predicted using an excess hazard statistical model (Cancer Research UK 2014b). The figure shows that survival rates are very poor and patients diagnosed with lung cancer rarely survive more than a few years. Another issue surrounding the disease aside from this tragic loss of life is the financial cost to individuals, health services and even whole societies affected by it. The American Cancer Society (2010) state that the worldwide economic impact of premature death and disability due to lung cancer was \$188 billion in 2008. Furthermore, this figure does not include direct medical costs. Each lung cancer patient costs the NHS appropriately £9,071 annually. For this figure patients are defined as “people who have previously been diagnosed with cancer and who are still alive. These may be newly diagnosed individuals, individuals with a stable disease being followed-up regularly, or individuals considered to be cured” (National Cancer Research Institute 2012). It would be a fair assumption that for other countries that are not fortunate enough to have a public health service that a similar cost would fall upon the shoulders of

the affected individuals and their family.

A key factor that effects the survival rate of patients diagnosed with lung cancer is the stage that the cancer is in when it is detected. The following list gives a brief description of the main stages using information provided by Cancer Research (Cancer Research UK 2014b):

- Stage I - The cancer is small and contained inside the lung and has not spread to lymph nodes.
- Stage II - The cancer is smaller than 7cm and may have spread to lymph nodes, or it is bigger than 7cm but has not spread to lymph nodes. Alternatively it may have spread into surrounding tissues, but if it has, it had not spread to the lymph nodes.
- Stage III - The cancer can be any size. It may have spread to the lymph nodes and it may also have grown into other parts of the lung or nearby structures in the chest.
- Stage IV - The cancer has spread to both the lungs, or it has spread to a distant part of the body such as the liver or bones.

A term used multiple times above is “lymph nodes”. Lymph nodes are part of the body’s immune system. If lung cancer has spread to the lymph nodes it is a strong indicator that it is growing quickly and is more likely to spread to other parts of the body (American Cancer Society 2015).

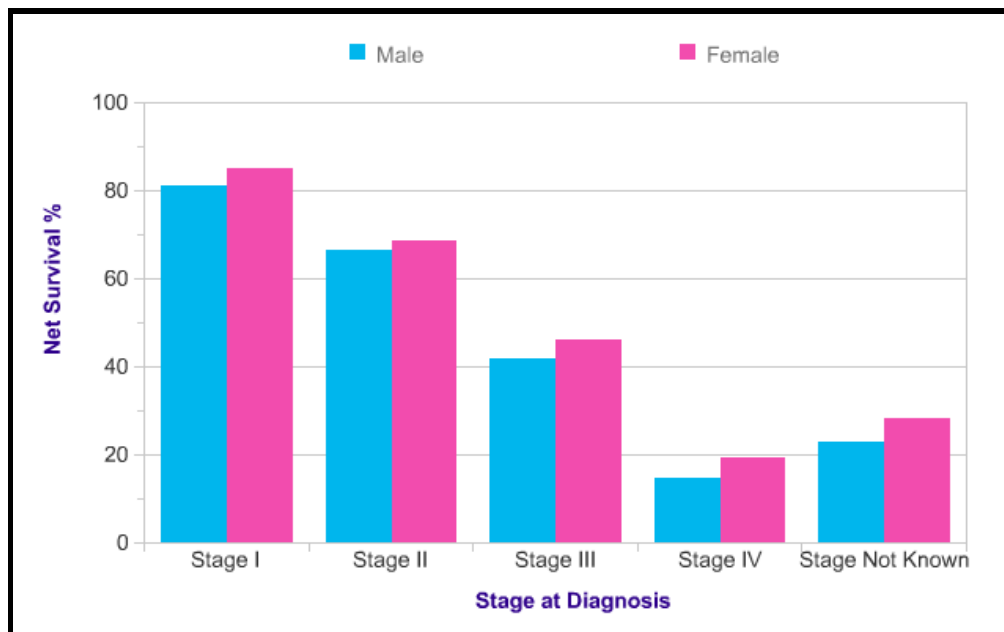


Figure 2: One year net survival rate for lung cancer patients (Cancer Research UK 2014b).

Figure 2 shows the net rate at which patients survive at least one year after being diagnosed with lung cancer at different stages. The figure clearly shows that the chances of survival are decreased the later the stage of the cancer at the point of diagnosis. Earlier diagnosis of

lung cancer does not only improve survival rates but also saves individuals and health services a great deal of money. For example, in 2014 the estimated treatment costs to the NHS per individual for lung cancer at stage 1 was £7,952 where as at stage 4 it was £13,078 (Incisive Health 2014). Figure 3 shows how the cost of treatment varies over all the stages using data from Incisive Health (2014) . Although the increase in cost when compared to stage 1 is less dramatic for stage 2 and 3 than stage 4 it is still significant due the the magnitude of patients being treated.

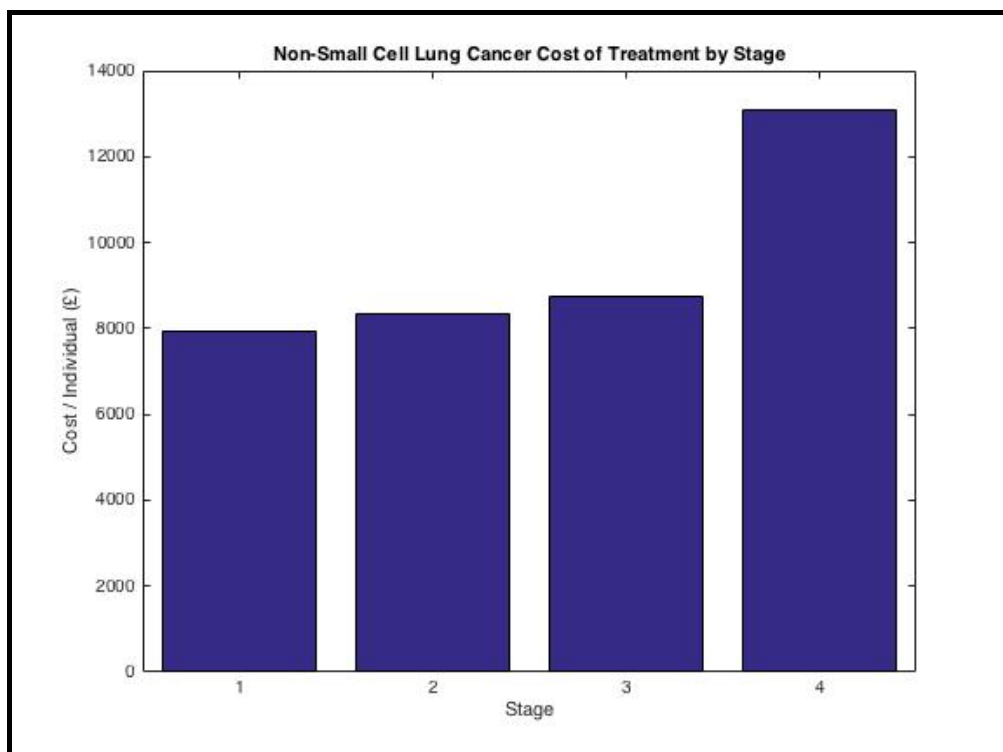


Figure 3: Estimated cost of treating individuals with cancer at different stages.

One of the primary methods used for detecting lung cancer is Computed Tomography (CT). Figure 4 shows an example of a CT scanner. In order to obtain images from the scanner a patient lies on the bed which is then moved incrementally through the ring. For each step that the bed moves a cross section of the patient is created. More detail about this process is provided in Section 2.2. Once the cross sections for a patient have been obtained they are traditionally examined by a radiologist who will attempt to identify any issues that have been revealed by the scan.

One of the main abnormalities that a radiologist will be looking for, as an indicator that a patient may have lung cancer, is the presence of pulmonary nodule(s). A pulmonary nodule (PN) is a small somewhat round growth on the lungs. Approximately 40% of PNs found turn out to be malignant (cancerous) as opposed to benign (non-cancerous) (University of Rochester 2017). As such it is highly important that no PNs are missed when reviewing the images obtained from a CT scan. This is as missing PNs can result in cancer remaining



Figure 4: The model of CT scanner that was used to create the images used in this project.

undetected until it has progressed to a later stage. One way to ensure that as few PNs are missed as possible is to use Computer Aided Diagnosis (CAD) systems to assist radiologist in the process of reviewing CT Scans. A study by Sahiner, Berkman et al. (2009) was conducted in order to assess the effectiveness of one such CAD system. The study involved six subjects, four fellowship-trained cardiothoracic radiologists¹ and two cardiothoracic radiologists who were in the final months of their fellowship year. The radiologists were instructed to conduct two reviews of 85 CT scans first without and then with the aid of CAD. Prior to this all of the CT scans were reviewed by a different group of expert thoracic radiologists in order to provide a ground truth used to measure the success of the readings made by the test subjects. To ensure that the results were not skewed when the subjects were reviewing the same set of CT scans a second time, the subjects only reviewed the regions of interest identified by the CAD system. Any PNs missed by the CAD system that were identified by the radiologist without CAD were automatically counted as a true positive (TP).

	FP rate (average # of FPs per scan)	Average sensitivity at nodule diameter threshold			
		3 mm	4 mm	5 mm	6 mm
without-CAD	0.667 (0.153–1.259)	0.559 (0.369–0.676)	0.670 (0.507–0.782)	0.773 (0.635–0.847)	0.836 (0.741–0.889)
with-CAD	0.778 (0.200–1.412)	0.665 (0.544–0.755)	0.758 (0.662–0.838)	0.825 (0.753–0.894)	0.901 (0.815–0.963)
% increase with CAD	16.8%	18.9%	13.1%	6.9%	7.7%
p-value	0.003	0.001	0.003	0.023	0.019

Table 1: The effect of CAD on false positive rate and sensativity (Sahiner, Berkman et al. 2009).

Table 2 shows the results obtained from the study. The table shows that the use of CAD improves the sensitivity of the readings especially for smaller PNs. Further more all of the p-values for the results were < 0.05 which is a good indicator that they were statistically relevant. A negative effect of using the CAD was that the false positive (FP) rate was increased. As "doctors approach every pulmonary nodule as cancerous until they can prove

¹A fellowship-trained cardiothoracic radiologists is a radiologists who has undertaking a post doctorate qualification and has specialised in the examination of the organs within the thorax.

otherwise" (University of Rochester 2017), increases in the number of FPs often means patients are subjected to further invasive testing such as biopsies. This wastes both the time and money of health services and individuals. Furthermore, it also inflicts unnecessary worry and risk on patients. As such, it is important that CAD systems return as few false positives as possible. None the less the benefits of increased sensitivity outweigh the drawbacks of increased false positives for patients who are exhibiting symptoms of lung cancer. The aim of the project documented in the following sections was to design, implement and evaluate a prototype CAD system (similar to the one used in the case study given above) capable of assisting a radiologist in the detection of PNs in CT scans of the chest.

2 Background

The first stage in the project was to obtain a strong understanding of both the medical and technical background surrounding the creation of a CAD system for nodule detection.

2.1 Pulmonary Nodule Detection

In order to complete the project it was important to have a basic understanding of the structure of the lungs and how PNs differ from the tissues expected to be present in a healthy pair of lungs. Figure 5 should be referred to throughout this section as a point of reference for the tissues being discussed.

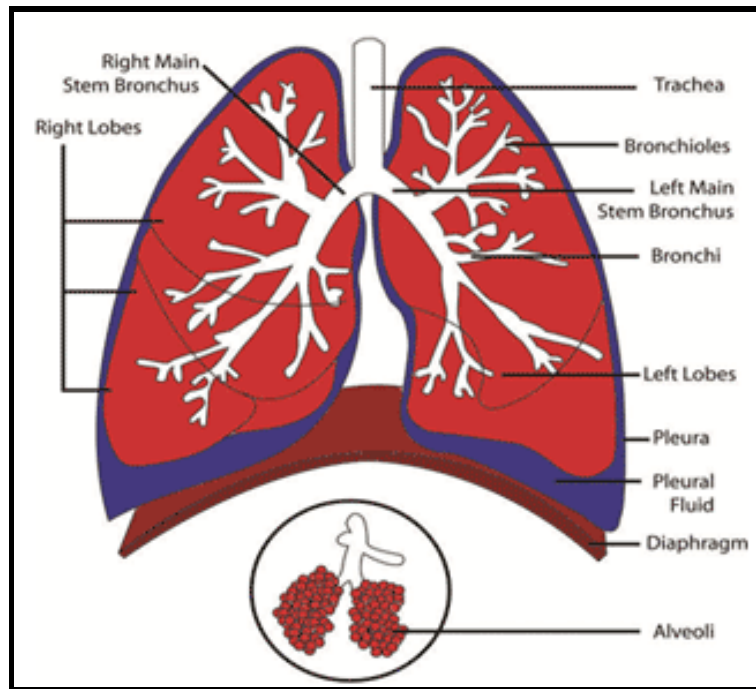


Figure 5: A basic diagram identifying the key tissues that make up the human lungs (Lung Cancer Alliance 2017).

One of the first steps taken during the project was to arrange a meeting with Dr Donald McLintock M.B., Ch.B. MRCP, a long serving general practitioner for the NHS with experience treating patients with lung cancer. During the meeting example CT scans were reviewed and Dr McLintock provided incite into how they should be interpreted. One of the most important concepts taken from the meeting was that tissues with high density appear lighter in CT imagery, when compared to those with a low density. This can be seen in Figure 6 which shows an annotated example of a CT cross-section or “slice”. For example, a vertebra of the spine (annotated by the yellow square) is made of dense bone and as such appears bright white in the slice. Conversely, at the top of the slice there is a large black region indicating low density. This is as this region represents the air above the patients body. The other squares in the figure highlight examples of blood vessels discussed with Dr McLintock. Blood vessels are visible in CT scans as they are more dense than their

surrounding lobes. As such they appear as light regions within the lobes. The red square shows the left main stem bronchus at the point where it meets the trachea. By referring to Figure 5 this should indicate to the reader the approximate location of this slice in the patients lungs. The green square highlights examples of less major blood vessels (bronchi / bronchioles) which extend appropriately orthogonally to the viewing vector i.e. the vector from the patients head to their feet (or vice versa). The blue square highlights similar blood vessels which extend appropriately parallel to the viewing vector. The blue arrow in Figure 6 points to the plural membrane of the left lung. This is the boundary between the left lobes (the dark area) and the pleura (the light area). The area enclosed by the plural membrane is often refereed to as the “plural cavity” in this report.

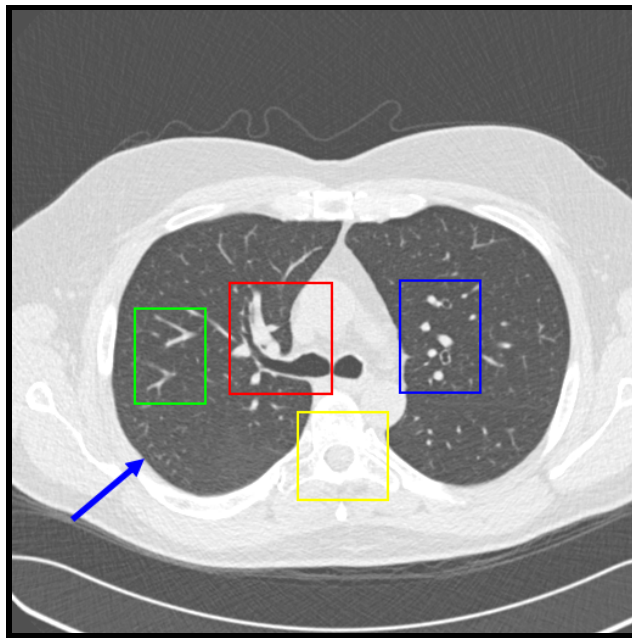


Figure 6: An annotated example of a CT cross section.

There were two main types of PN that were focussed upon during the project, Solitary PNs and juxta-pleural PNs. a Solitary Pulmonary Nodule (SPN) is "an isolated, single lesion in a round or oval shape with a diameter of ≤ 3 cm in lung parenchyma, surrounded entirely by gas-containing lung tissue" (Xu, Chunhua et al. 2017). Essentially this means a PN which is found within a lobe and is not connected to either a blood vessel or the plural membrane. Figure 7 shows an example of an SPN highlighted by a red square. A Juxta-pleural Pulmonary Nodule (JPN) is a PN that is attached to the plural membrane and protrudes into one of the lobes (Jirapatnakul, Artit et al. 2011) as shown highlighted by the red square in Figure 8. Another class of PN occasionally refereed to in the report is a Juxtavascular nodule. These as the name suggests are PNs that are attached to a blood vessel.

Dr McLintock suggested the following properties which could be used to distinguish PNs from blood vessels:

- They are often either lighter or darker than the surrounding blood vessels.

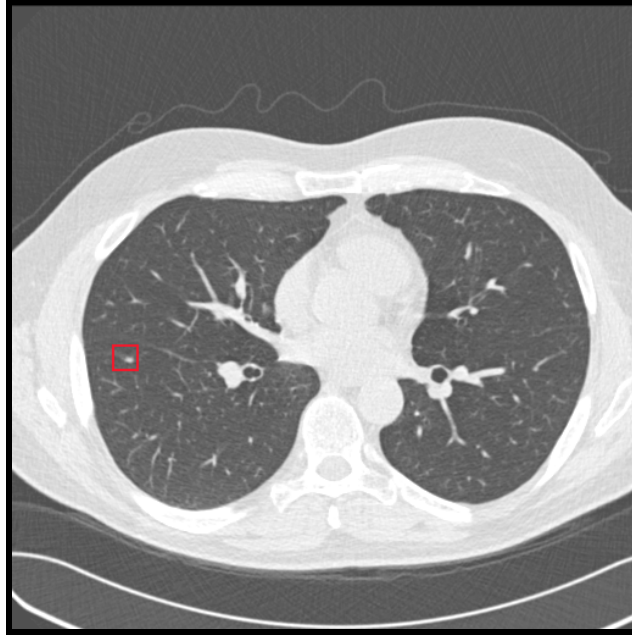


Figure 7: An example of a SPN.

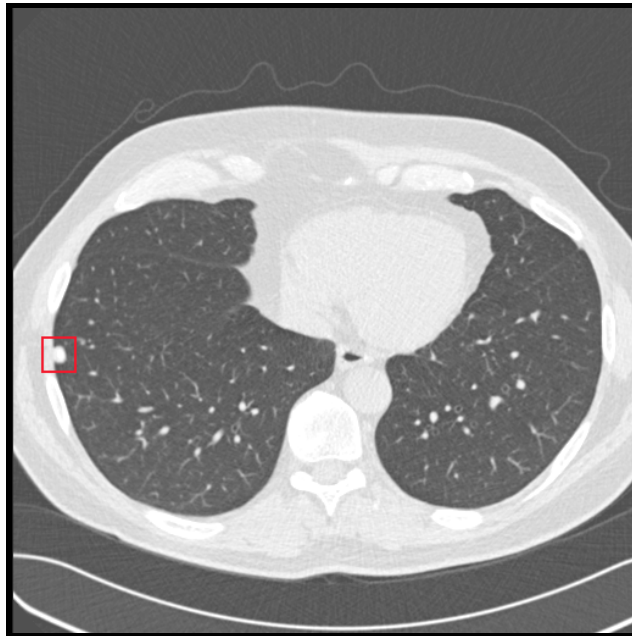


Figure 8: An example of a JPN.

- They are often speculated i.e. covered with needle like protrusions into the surrounding lobe.
- They often have a non-uniform texture.
- They do not track across the image when iterating through an ordered set of slices (a “stack”).

- They do not exhibit a defined tree like structure similar to that of the blood vessels highlighted by the green square in Figure 6.

2.2 Computed Tomography Scans

In order to process the images obtained from CT scans it was first important to obtain a strong understanding of what was being represented by the pixel values in them. CT scanners use X-rays to scan the bodies of patients. X-rays are a highly penetrating form of electromagnetic radiation with a far shorter wavelength than that of visible light. The wavelength of X-rays ranges from approximately 10^{-8}m to 10^{-11}m (Columbia University Press 2000). As X-rays are highly penetrating they can pass through solid objects and be used to create images which reveal the internal structure of an object such as the human body. When X-rays pass through the body there are four types of attenuation that can occur as photons interact with the atoms within it: Raliegth scatter, Compton scatter, photoelectric absorption and pair production. Pair production does not occur at energy levels used for X-ray imagery and as such is not covered in any detail in this report. Toennies (2012) provides detail on each of the other types of attenuation. Raliegth scatter occurs when an atom absorbs a photon and the entire atom becomes excited. The atom instantaneously releases a new photon with slightly less energy than the one absorbed. The new photon is usually scattered in a direction different than that of the one the original photon was travelling in. Compton scatter is where an X-ray photon displaces a valance electron from an atom. The photon loses as much energy as is required to displace the electron and is scattered in a different direction to the one it was initially travelling in. Both Raliegth and Compton scatter have the effect of creating noise in X-ray imagery. Photoelectric absorption is the most important type of attenuation in X-ray imagery. This is as it is the form of attenuation that is most responsible for the images produced. Photoelectric absorption occurs when a photon displaces one of the electrons from the inner shell of an atom. This requires all of the energy in the photon. The shell with the missing electron fills the space using an electron from an outer shell. This process is repeated until the outermost shell is reached. Each time an electron changes shell a photon is released. These photons have a much lower energy than than the original photon and as such are absorbed by the surrounding atoms. The probability of photoelectric absorption increases with atomic number of the atom the photon is incident with. Hence, denser materials are more prone to photoelectric absorption. Overall the probability of a photon reaching the detector unscattered can be modelled as a Poission process, this is useful information for removing noise from slices during pre-processing.

As shown in Figure 9 CT scanners obtain images by having a patient lie on a bed between an X-ray source and X-ray detector. Using the properties of photoelectric absorption discussed above, projection images can be created for small slices of the patients body. The source and detector are rotated in synchrony through 180° creating a projection image at each step. When all the projection images for the slice have been captured the bed is moved and the process is repeated to capture the next slice. The number of slices that are created depends on many factors such as the size of the objects that are being examined in the scan. Each set of projections is converted into a cross section containing voxels. Each voxel holds the value for attenuation per unit volume in the imaged patient. This value is known as an

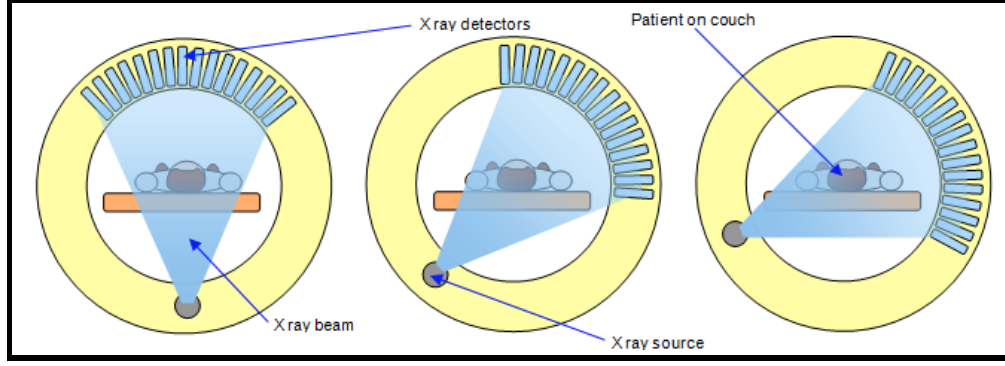


Figure 9: A CT scanner with the X-ray source and detector shown in three positions (Gibbs 2013).

Tissue	CT number (HU)
Bone	1000+
Haemorrhage	60 to 110
Liver	50 to 80
Muscle	44 to 59
Blood	42 to 58
Grey matter	32 to 44
White matter	24 to 36
Heart	24
Cerebrospinal fluid	0 to 22
Water	0
Fat	-100 to -20
Lung	-300
Air	-1000

Table 2: Example CT numbers for various tissues, values taken from Webster (1988) .

attenuation coefficient (μ). Attenuation coefficients are obtained using a highly complicated algorithm based around the Central Slice Theorem and the use of filtered back projection. The algorithm is explained in detail by Toennies (2012) but is beyond the scope of this report. In order to make the results of CT scans independent of parameters such as the wave length of the X-rays being used. In each of the voxels the attenuation coefficient is converted into a CT number measured in Hounsfield units (HU). Hounsfield units are defined relative to water and can be calculate using the following equation (Dougherty 2009):

$$\text{CT number} = 1000 \times \frac{\mu - \mu_{H_2O}}{\mu_{H_2O}} \quad (1)$$

2.3 Existing Computer Aided Diagnosis Systems

As discussed in Section 1 there are multiple large incentives for the creation of CAD systems used to detect PNs in CT scans. As such it is no surprise that there has been a great deal of research conducted in the area. All of the CAD systems discussed in Firmino et al. (2014) used the same generic pipeline with the following stages:

- Acquisition - The importing of images and associated data into the system. Images and data can be obtained from either through partnerships with hospitals or from publicly available databased. The choice of dataset used for this project is discussed in detail in section 2.4.
- Pre-processing - The application of filters and enhancement techniques to the slices in order to improve the accuracy of the detection algorithms used.
- Segmentation - The separation of regions of interest believed to be candidate nodules from the remainder of the slices. The regions of interest obtained in this stage should aim to represent the shape and position of the candidate nodules as accurately as possible. The segmentation stage aims to reduce computational complexity through minimising the number of pixels that are processed by later stages of the system.
- Classification - The process of determining if each region of interest (ROI) obtained in the previous stage is in fact a nodule. This is achieved through the computation of features for each of the ROIs. These features are then used to train a classifier against some ground truth imported into the system during the acquisition stage. The trained classifier can then be used to classify previously unseen ROIs.

2.3.1 Noise Removal

One of the key stages in pre-processing CT slices is noise removal. This is as noisy images result in poor segmentation and increased computational complexity. As stated in section 2.2 the production of noise in CT imagery is often modelled using as Poission process. This being the case the obvious choice of method for removing noise from slices would be to apply a Gaussian Filter. This is as a Gaussian function is a good approximation of a Poission distribution for a large number of events such as the effect of scattering on the vast number of photons required for CT imagery (Toennies 2012). Further more, a Gaussian Filter is efficient as it is a separable filter. This means that it can be implemented such that it has a computational complexity of $O(n)$, where n is the number of pixels in the image being filtered (Rosin 2017). However, in a comparison of 7 different filters Vijaya and Suhasini (2014) use empirical evidence to demonstrate that a Bilateral Filter was the most appropriate for noise removal in CT imagery. In an experiment using 418 images the filters were applied to each of the images and compared to the original in order to obtain the Mean Square Error (MSE) and the Peak Signal-to-Noise Ratio (PSNR). An effective noise removal technique should minimise MSE and maximise PSNR. As can be seen in table 3 the Bilateral filter had the highest PSNR and the lowest MSE of all of the filters tested.

Bilateral filtering works by combining domain and range filtering and is defined by Tomasi (1998) using the following equations:

$$\mathbf{h}(\mathbf{x}) = k^{-1}(\mathbf{x}) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathbf{f}(\xi) c(\xi, \mathbf{x}) s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x})) d\xi \quad (2)$$

with the normalisation:

$$k(\mathbf{x}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} c(\xi, \mathbf{x}) s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x})) d\xi \quad (3)$$

Where:

- $\mathbf{f}(\mathbf{x})$ = the value(s) for point \mathbf{x} in a single or multi-band input image.
- $\mathbf{h}(\mathbf{x})$ = the value(s) for point \mathbf{x} in the output image (this image will have the same number of bands as the input image).
- $c(\xi, \mathbf{x})$ = a measure of geometric closeness between point \mathbf{x} and near by point ξ .
- $s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x}))$ = a measure of photometric similarity between point \mathbf{x} and near by point ξ .

The Bilateral Filter is non-separable and as such incurs a computational complexity $O(n^2)$. However, the Bilateral Filter was still selected for use in this project over the more efficient Gaussian filter as for a CAD system accuracy holds precedence over efficiency.

Filters Name	PSNR Value	MSE Value
Adaptive Median Filter	39.4309	29.33109
Alpha Trimmed Mean Filter	71.86155	0.016638
Gaussian Filter	46.70281	17.08423
Gabor Filter	16.65061	5472.75
High Pass Filter	23.34334	3475.875
Laplacian Filter	21.34006	5573.9
Bilateral Filter	88.22711	0.000379

Table 3: The PSNR & MSE values for the different filters (Vijaya and Suhasini 2014).

2.3.2 Case Study

In their review paper Firmino et al. (2014) collated results used to determine the effectiveness of numerous CAD systems, see Table 4. As shown in the table CAD systems are capable of achieving sensitivities in the range of 70%-97% and false positive (FP) rates of between 2-25.3 per stack. The CAD system studied in the greatest depth was developed by Tan et al. (2011) . The primary decisive factor when choosing this systems was the number

of nodules used to test it. This is as although other systems achieved higher sensitivity and specificity they were tested using fewer nodules. Hence these values were likely to be less accurate.

Methods	Year	Sensitivity	FP	N ^o of nodules	Size	Response time	Type of nodules
Xu et al. [70]	1997	70%	1,7 per image	122	4 - 27mm	20s	NI
Armato et al. [48]	1999	70%	9,6 per case	187	3,1 - 27,8mm	NI	Solitary and juxtaleural
Lee et al. [45]	2001	72%	25,3 per case	98	< 10mm	187 min	NI
Suzuki et al. [71]	2003	80,3%	4,8 per case	121	4 - 27mm	1,4s	Juxtavascular, hilum, ground-glass opacity and juxtaleural
Murphy et al. [40]	2007	84%	8,2 per case	268	2 - 14mm	NI	Pleural and non-pleural
Ye et al. [23]	2009	90,2%	8,2 per case	220	2 - 20mm	2,5 min	Juxtavascular, isolated, ground-glass opacity and juxtaleural
Messay, Hardie and Rogers [21]	2010	82,66%	3 per case	143	3 - 30mm	2,3 min	Juxtavascular, solitary, ground-glass opacity and juxtaleural
Liu et al. [20]	2010	97%	4,3 per case	32	NI	NI	Solitary
Kumar et al. [75]	2011	86%	2,17 per case	538	NI	NI	NI
Tan et al. [76]	2011	87,5%	4 per case	574	3 - 30mm	NI	Isolated, juxtavascular, and juxtaleural
Hong, Li and Yang [22]	2012	89,47%	11,9 per case	44	NI	NI	Solitary
Cascio et al. [65]	2012	97%	6,1 per case	148	≥ 3mm	1,5 min	Internal and juxtaleural
Orozco et al. [63]	2012	96,15%	2 per case	50	NI	NI	NI
Teramoto and Fujita [24]	2013	80%	4,2 per case	103	5 - 20mm	30s	Juxtavascular, isolated, ground-glass opacity and juxtaleural

Table 4: The effectiveness of existing CAD systems (NI = Not Informed) (Firmino et al. 2014).

Tan et al. devised a system with a sensitivity of 87.5% and 4 FPs per case when tested using 574 nodules. Their system was tested and trained using publicly available images provided by the Lung Image Database Consortium (LIDC) (Armato et al. 2011). A basic outline of the system is given in Figure 10. As the resolution, slice thickness and the distance between each slice varies between stacks in the LIDC dataset, the first step performed by the system was to normalise them. This was achieved through Isotropic resampling in order to normalise voxel dimensions to 1mm³ and Trilinear interpolation to obtain grey levels at a sub-voxel precision.

Next a 3D mask of the lungs was created to ensure that nodule detection only took place within the area of the lungs. This was achieved through the application of a binary threshold using a value of -550HU. This value was obtained through optimisation against the training set. After thresholding, voxels below -550HU were retained as the foreground. The mask was then obtained by discarding regions adjacent to the very top and bottom of the images and selecting the remaining segment with the largest volume. This ordinarily returned a single ROI, as when viewed in three dimensions the lungs are connected by the primary bronchi. However, occasionally where a tumour was present, two ROIs were obtained. In this case, if the second largest ROI contained greater than half the number of voxels in the largest ROI, the ROIs were combined to create the mask. Finally a 3D morphological closing operation was applied to the mask in order to include missing structures in the lungs such as JPNs.

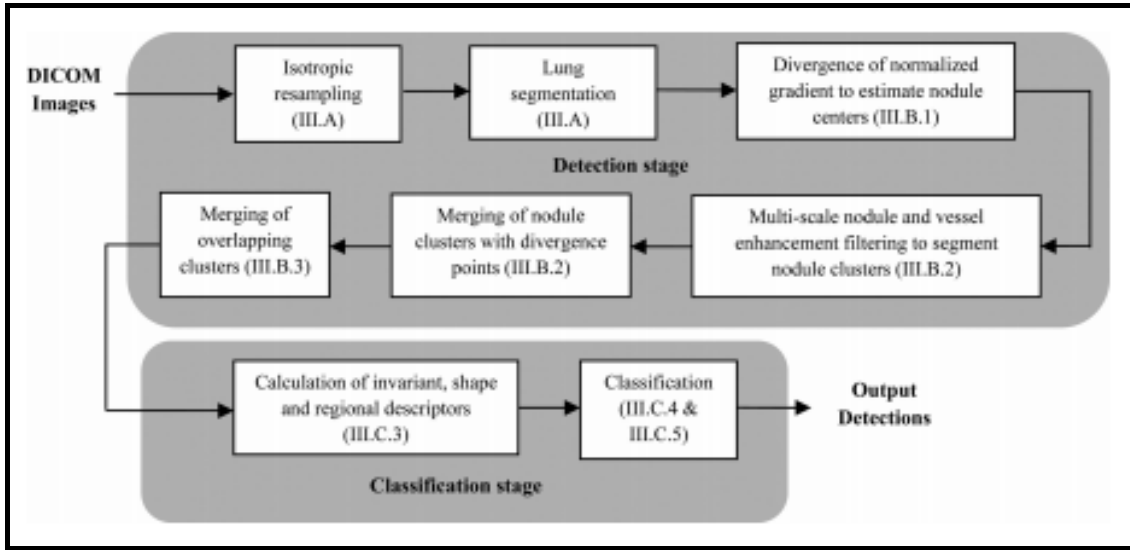


Figure 10: Top level block diagram of CAD system developed by Tan et al. (2011) .

Once the ROI for the lungs had been identified the next stage was to identify nodule candidates. This was accomplished by first detecting seed points that could be used to aid the segmentation of the nodules. The seed points were obtained by computing the maxima of the divergence of normalised gradient (DNG) at 6 different scales. DNG is a measure of mean curvature in 3D and is calculated using the following equation:

$$k = \text{div}(\mathbf{w}), \text{ where } \mathbf{w} = \frac{\nabla \mathbf{L}}{\|\nabla \mathbf{L}\|}, \text{ and } \mathbf{L} = \text{the image intensity} \quad (4)$$

By locating the maxima the centres of the candidate nodules were found. These maxima were then thresholded so that voxels with values bellow two empirically obtained thresholds (100 for solitary and juxtavascular nodules and 25 for JPNs) were rejected. Next in order to obtain sets of voxels for the center points different segmentation techniques were used for each type of nodule. For SPNs first the resampled images were thresholded at -600HU. Next the selective nodule and vessel enhancement filters proposed in (Li et al. 2003) were applied to the images and a threshold of 6 was used to obtain clusters of voxels. Those clusters which correspond to the center points detected in the previous stage and had a volume in

the range of 9mm^3 - 500mm^3 were selected as candidate nodules. Candidate juxtavascular nodules were obtained by applying a grey level threshold of 150 to vessel enhanced images in order to obtain clusters of voxels. Those clusters of voxels which corresponded to previously detected center points were rejected. The remaining center points which were within 2 voxels from a cluster were used as seeds in a region growing algorithm. This algorithm uses a 3×3 neighbourhood. Neighbouring voxels were added to the candidate nodule if the nodule enhanced value for the voxel did not differ by >10 from the most recent voxel added. After region growing a threshold of 6 was applied to the enhanced values of the candidate nodules. Finally those nodules with a volume of less than 9mm^3 were rejected. Candidate JPNs were detected using a third technique. Firstly a threshold of -400HU was applied to the resampled images. The enhancement filters were then applied and a threshold of 4 was used on the enhanced image to obtain the voxel clusters. As with the SPNs, only clusters which correspond to detected center points were excepted as candidate nodules. However, for the JPNs only center points which lay within 4 pixels of the plural membrane were valid. This region was obtained by applying 2D erosion to the mask of the lungs previously created. The remaining center points which did not correspond to clusters were used in a region growing algorithm similar to the one for juxtavascular candidates. However, a final threshold of 4 was applied rather than 6. Finally any juxtapleural candidates that had a volume $<2\text{mm}^3$ were rejected. Once all the candidate nodules had been extracted the final stage of the segmentation process was to cluster the candidates. This was done in order to remove duplicate candidates that were detected using multiple segmentation techniques. This was achieved in two steps. Firstly, any candidates obtained through region growing that occurred outside the mask of the lungs were rejected. Next a logical OR operation was applied to all the remaining candidates to combine them where appropriate. When all these stages were complete the remaining set of nodule candidates were ready to have features computed. This segmentation method achieved a sensitivity of 91.1% and produced 479FP/scan.

As shown in Table 5 Tan et al. used a total of 45 features to train their classifier. Many of these features were transformation and/or translation invariant. It is important that such features were used when classifying nodules to ensure the trained classifier is robust to variance in size, orientation, location and intensity values etc. In order to obtain such features a 3D gauge co-ordinate system defined by Salden, Florack, and Haar Romeny (1991) was used. Some of the features selected were computed using spherical kernels rather than all of the segmented voxels. These features were required as the thresholding applied during segmentation led to the poor segmentation of some nodule candidates. Demir and Yener (2005) highlights five main types of features that are often used in cancer focused CAD systems:

- Morphological features - These features are used to describe the shape and size of candidate legions. Features 1-7 in Table 5 provide examples of morphological features.
- Textural features - These features describe changes in intensity throughout candidate legions. There were no textual features used in Table 5. However, Orozco et al. (2012) used the Fast Fourier Transform and Discrete Cosine Transform to compute average grey level and third moment features in the frequency domain.

No.	Feature	Notes
1	Volume (number of voxels)	
2	$\min_dim = \min_i(dim_i)$	dim_i = diameter corresponding to the principal axis i of the minimum volume-enclosing ellipsoid, (Kachiyan algorithm (Ref. 60 and 61))
3	$\max_dim = \max_i(dim_i)$	
4	Compactness1, $volume / \prod_{i=1}^3 (dim_i)$	–
5	Compactness2, $volume / \max_dim^3$	–
6	Elongation factor, \max_dim / \min_dim	–
7	Bounding ellipsoid feature	0 for 2D ellipsoid; 1 for 3D ellipsoid
8	Distance of nodule candidate centroid to lung wall	–
9	Distance of nodule candidate centroid to the center of the 2D image slice on which the nodule candidate centroid is located	–
10–21	Mean (average) of L_{int} and L_{vv}	On segmented voxels, and spherical kernels of radius = 1 and 3 pixels at scale, $s = 1$ and 2
22–24	Mean (average) of nodule filter values	On segmented voxels, and spherical kernels of radius = 1 and 3
25–27	Mean (average) of vessel filter values	On segmented voxels, and spherical kernels of radius = 1 and 3
28–30	Mean (average) of divergence $K(x)$ values	On segmented voxels, and spherical kernels of radius = 1 and 3
31–45	Grey-value features: mean, median, maximum, minimum, and standard deviation	On segmented voxels, and spherical kernels of radius = 1 and 3

Table 5: Table of features used by Tan et al. (2011) .

- Fractal features - These features are used to describe how self similar the structure of a candidate legion is. Again there are no examples of fractal features shown in Table 5. However, Demir and Yener (2005) states that “the most common feature is the fractal dimension. The fractal dimension d is defined as $d = \log(N)/\log(p)$, where N is the number of self similar pieces at the magnification scale of p ”.
- Topological features - These features are used to describe the spatial distribution of legion candidates. Features 8 and 9 in Table 5 provide examples of topological features.
- Intensity features - These features describe characteristics of the intensity values present in candidate legions. Features 22–27 and 31–45 to in Table 5 provide examples of intensity features. Another commonly used method is to use histogram bins as features.

Once the features have been computed the nodule candidates must be classified as either nodules or non-nodules. Tan et al. achieved this by examining the Euclidean distance between the candidate nodule centroid and the centroids provided by the ground truth. For nodules provided by the ground truth with a volume of less than 600mm^3 , if the distance was less than 5mm the candidate was considered to be a nodule. For nodules greater than this a threshold of 9mm was used. Nodule candidates that did not meet these criteria were classified as non-nodules. Once the nodule candidates were classified they could be divided into testing and training sets. The training set used by Tan et al. contained 574 nodules and 111332 non-nodules. For most classification algorithms it is important to have balanced numbers of each class in the training set. Tan et al. achieved this by clustering non-nodules in feature space and using appropriately even numbers of non-nodules from each cluster when training the classifier. Using this method the number of non-nodules used for training was reduced to 3382. Tan et al. tested their system using three classifiers: a Support Vector Machine (SVM), a standard feed-forward fixed-topology Artificial Neural Network (ANN)

and the novel classifier FS-NEAT proposed in (Tan et al. 2009). Tan et al. concluded that the most appropriate classifier for the system was the ANN which yielded a sensitivity of 91.1% and 479FP/scan.

2.4 Dataset Selection

How effective a classifier is greatly depends on the dataset upon which it was trained. As such it was highly important to select an appropriate dataset for the project. As mentioned in Section 2.3 there are two main sources for datasets containing CT imagery: private datasets obtained through partnership with a hospital and publicly available datasets. For this project it was decided that a publicly available dataset should be used. This decision was made for two main reasons. The first was ease of access. Obtaining a private dataset from a hospital would likely be very difficult and time consuming due to patient confidentiality regulations, where as a public dataset can simply be downloaded at will. The second advantage of using a public dataset was that the CAD system created could be more easily compared against other systems that had been tested using the same dataset. In order to effectively train and test the classifier used by the system, it was important to select a dataset of sufficient size and with a reliable ground truth identifying the locations of nodules in the CT imagery. There were two publicly available databases that were considered for use in the project. The first was the ELCAP database (ELCAP 2003), created in collaboration by the International Early Lung Cancer Action Program and the Vision and Image Analysis Group. This database consisted of 50 stacks with supporting csv files indicating the centroids of the nodules present. The supporting documentation for this dataset was very limited and only indicated that the annotations were created by a radiologist. The second dataset was the LIDC set used that was used in (Tan et al. 2011). This dataset contains 1,018 stacks and is a popular choice in the academic community having been cited 260 times (Google Scholar 2017). The dataset included annotations which not only provided a centroid for nodules, but for nodules $\geq 3\text{mm}$ in diameter a full outline of the nodule was provided. Furthermore, the dataset was extremely well documented and great detail was provided on how it was created. Armato et al. (2011) describe how the annotation process was divided into two phases. In the initial “blind read phase” four radiologists independently reviewed stacks and used a piece of software to annotate any nodules they identified. Next came the “unblinded read phase”. During this phase each radiologist reviewed the same stacks, however this time they were annotated with the markings created by themselves and the three other radiologists. Upon seeing the markings of the other radiologists, each radiologist was offered the opportunity to add, remove or edit their own annotations. These annotations were then stored in XML files so they could be easily used as a ground truth in a CAD system. This method was used to attempt to identify as many nodules as possible without demanding that all radiologists must agree with one another. Out of the two candidate datasets the LIDC dataset was selected for the project. This was as it provided superior documentation, a larger number of stacks and detail about the origin of the ground truth provided. Further more as it was a popular choice for other CAD systems comparisons drawn between observed results could be made in greater confidence.

A key piece of information required before implementation could begin was how to link each

of the annotations in the XML files to its respective slice. The slices provided in the LIDC data were encoded as DICOM files. DICOM “is the standard for the communication and management of medical imaging information and related data” (DICOM 2017). The files are not only used to encode images but also contain meta-data stored in the file header. This meta-data provides information about the patient and how the images were obtained. Part of the documentation provided with the dataset explained that `imageSopUID` field in the file header was used to identify a unique slice in the dataset. Each annotation represented in the XML also had an `imageSopUID` field which linked it to its respective image. Another useful identifier used throughout the dataset was the `seriesInstanceUID`. This was used to identify which stack each of the slices in the dataset belonged to.

3 Approach

In order for the project to be success it was important that sufficient time was spent on design, the selection of appropriate tools and resource acquisition. This section provides detail relating to each of these topics.

3.1 Requirements

The following requirements outline the features and properties that were expected of the CAD system developed.

The system should be capable of:

- importing and querying meta-data provided by CT scans.
- segmenting ROIs in CT slices.
- comparing ROIs to the ground truth.
- training a classifier using features computed for ROIs identified.
- using a trained classifier to identify PNs in previously unseen CT scans or state that there are no PNs.
- highlighting PNs in a suitable GUI so that it can be used as an aid for radiologists.
- having additional features added to the classifier easily.
- storing the results of features calculated for the training set so that they do not need to be recomputed each time the classifier is trained.
- unit testing functionality where possible and appropriate.
- evaluating the effectiveness of the classifier.
- easily configuring parameters the user may wish to change from one instance of the system to another.
- only using a small subset of images during development. This will help to speed up development as code can be tested on just a few images before being running again with the full dataset.
- optimising values for parameters which are not easily determinable using appropriate algorithms.
- be capable of processing 1000s of images in a reasonable amount of time. This should be achieved through the use of low computational complexity algorithms and multi-threading.

3.2 Development Strategy

The limiting factor for the project was the extremely short time frame in which the system had to be implemented and tested. This guided much of the decision making that was made both before and during implementation. To ensure that a functioning prototype was delivered at the end of the project the decision was made that an Agile development methodology should be applied throughout. As such, multiple versions of the system would be developed. The first of these versions was to be a Minimum Viable Product (MVP) which was incrementally improved in each of the following versions. In order to decide how the system could be improved from version to version experiments were conducted to ascertain how aspects of the system could be improved. Another methodology applied to the project was Test Driven Development. Unit tests were created for code whenever possible. Creating unit tests not only instils confidence in an implementation but also helps to flag issues in existing code created by adding new code. Identifying and rectifying issues early is advantageous as it avoids them cascading into large amounts of new code that depends on the the problematic code.

3.3 Tools and Libraries

In order to aid rapid development it was important that appropriate tools and libraries were used where ever possible. This section discusses why some of the most important ones were chosen. Arguably one of the most important decisions that had to be made before implementation could begin was the choice of the main image processing / computer vision library that would be used. Two main options were considered for the project: MATLAB alongside supporting toolboxes and OpenCV. MATLAB is an excellent tool for prototyping software as it comes packaged with a wide array of pre-implemented functionality. Two supporting packages which would have been particularly useful for the project were the Image Processing Toolbox (MathWorks 2017b) and the Computer Vision System Toolbox (MathWorks 2017a). The Image Processing Toolbox provides a suite of image processing functions and data visulisation tools. The Computer Vision System Toolbox provides functionality such as feature extraction and machine learning frameworks. A major draw back of MATLAB is that MATLAB code is slow to execute (as it is an interpreted language) and there is no explicit multi-threading. As such, the language is not ideal for processing vast numbers of images. Another drawback of the language is as it is loosely typed refactoring can be very time consuming. This is not ideal for highly iterative development. (OpenCV 2017) is an open source, cross-platform computer vision library written in C/C++. At the time of writing the library is on it's third major release, boasts more than 2500 optimized algorithms and has been used by well renowned companies such as Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda and Toyota. The library provides interfaces that allow programmers to develop in C++, C, Python, Java and MATLAB. This was advantageous as Java was the language the author had the most experience using, and as such could develop the system fastest in. Furthermore, in contrast to MATLAB Java 8 has a very simple framework that supports explicate multi-threading. It is also strongly typed, allowing for fast refactoring with the aid of an appropriate Integrated Development Environment (IDE). A downside of using Java is it too is slow to execute. This is as Java programs run on a virtual machine. However,

a great deal of the computationally heavy processes in the CAD system could be executed using OpenCV functions written in C/C++ and both of these languages are notoriously fast. Another powerful feature of Java is the remote debugger. This feature is particularly useful for identifying the causes of bugs that only occur when code is running on a server. For these reasons the CAD system created in this project was developed using OpenCV and Java 8.

Another important choice was that database management system (DBMS) used to store and query data created by the system. The DBMS selected for the project was MongoDB (MongoDB 2017a). MongoDB is a NoSQL (non-relational) database. Some terms used in MongoDB that should be defined are “collection” and “document”. In MongoDB a collection is analogous to a table and a document is analogous to a row in one such table. MongoDB typically stores all related information in single documents which are similar to JSON objects. As such documents can be very easily mapped to and from objects created in object orientated languages, such as Java. This is a task that is far more challenging when using a relational database. Another feature of MongoDB which made it an appropriate choice for the project was that it requires no rigid schema to be defined, as is common in SQL databases. This is ideal for the iterative development methodology that was selected for the project as it allows fields to be added to and removed from documents very quickly. MongoDB provide a Java driver that can be used to interact with a database from Java code (MongoDB 2017b). However, the driver requires that all Java objects are converted into Maps before they are inserted into a database and converted from Maps to Java objects when they are returned by a query. This requires a reasonable amount boiler plate code to be written and updated with every schema change. As such it was decided that the third party mapping library Morphia should be used (Morphia 2017). Morphia allows for Java objects to be saved to MongoDB in a matter of lines. It also provides a framework which can be used to perform complex queries and aggregations.

As many versions of the system were created it was important to use version control software to manage them. The software selected for this task was Git (2017). For each version of the CAD system a Git tag was created. A tag is used to record a specific point in the history of the project. Each of these tags had an associated message which was added as a reminder of the changes that had been made since the previous version. The git repository was backed up online using Bitbucket (2017). Bitbucket’s issue tracker was also used to manage the project. This tool allows for known issues to be logged and assigned a priority which could then be used to decide which features and bug fixes should be implemented in the next iteration of development.

Another library selected for use in the project was Weka (University of Waikato 2017). Weka is a machine learning library for Java, that comes with approximately 40 configurable, pre-implemented classification algorithms. These algorithms can be swapped between with great ease allowing the system to be tested using a wide range of them. The author also had experience using the library which would help during implementation. Another reason Weka was chosen was that it comes packaged with a tool set that can be used for tasks such as featuresselection and visualisation of feature space.

Image processing is in general a computationally complex process. As such it was predicted that a server would be needed in order to train the system. This is as 1000s of images would need to be processed. Fortunately Cardiff University were able to provide a Virtual Private Server (VPS) which could be used for the project. The VPS was managed through the OpenStack dashboard (OpenStack 2017) and was created with: 16GB of RAM, 8 Virtual CPUs and a total of 315GB of disk space. The additional RAM and CPUs meant that more threads could be introduced when processing images and the large disk space was useful for storing the large dataset, and data created by the system. As the system would be frequently deployed to the server, so that new code could be tested using the full dataset, it was important to be able to build the project quickly. As such two additional development tools were used Apache Ant (Apache 2017a) and Apache Ivy (Apache 2017c). Ant is a tool that facilitates the development and execution of build scripts. Ivy is used to manage dependencies for a project and can easily be used as part of an Ant build script. Together these tools were used to: download any libraries required from the Maven Central Repository (Maven 2017), compile the project code, pack it into a JAR file and present the built project as a single directory which can be uploaded to a server and run. Once the build script had been created all this could be achieved in a single simple command. Not only does this save time it also encourages the testing of code against the full dataset more regularly. This helps to identify bugs early which as previously stated this is important for rapid development.

3.4 Initial Design

The design stage is a crucial stage for any project. Although the final product is often far from what was envisioned by the initial design, having the design is a great way of maintaining a top down view of how a system will work during implementation. During the design phase the system was given the name “Lungs” and will be referred to as such from this point in the report. Figure 11 shows the initial design used for the Lungs database. Each of the entities in figure can be mapped to a single collection in the database. Fields that have been marked with an asterisk indicate that they should have unique values for each document stored in the collection. As a non-relational database was used for the project the entities in the diagram were not normalised for instance both **CTSlice** and **CTStack** have share a **model** field with the same possible set of values but no **Model** entity was created. This was so that all the information about a **CTSlice** or **CTStack** could be returned in a single query. The following list outlines the purpose of each collection:

- **CTSlice** - Each document in this collection was used to hold information about a single slice. Some key fields used by **CTSlice** documents were **manufacture** and **model** these fields were used to identify the CT scanner which was used to create each slice. Another key field was **imageNumber**, this field was used to indicate the order that slices appear in their respective stacks.
- **CTStack** - Each document in this collection was used to hold all the information about the slices contained in a single stack. The **slices** field was used to hold a sorted list of **CTSlices**. It should be noted that the actual documents were saved in this list, not

references to them. This vastly reduced query times when whole stacks were required at once and meant that the slices only needed to be sorted once.

- **GroundTruth** - Each document in this collection was used to hold information about a single annotation on a slice created by a radiologist. The key fields in this collection were `groupId` and `type`. The `groupId` field was used to link **GroundTruths** that appeared on different slices but were for the same nodule. The `type` field was used to indicate which class of **GroundTruth** each document belonged to. The two classes used were `SMALL_NODULE` and `BIG_NODULE`. `SMALL_NODULEs` were **GroundTruths** for nodules with a diameter of $<3\text{mm}$. These **GroundTruths** had no `edgePoints` and `region` fields as only the centroids were marked by the radiologists.
- **ROI** - Each document in this collection was used to hold information about a single nodule candidate identified in the segmentation stage of the system. Some key fields were: `classification`, used to identify if the **ROI** was a nodule or non-nodule; `set` used to indicate whether the **ROI** belonged to the testing or training set and `groundTruth`, used to identify the **GroundTruth** that the **ROI** had been matched too if it was a nodule.

Figure 12 shows the initial design for the pipeline used to train the classifier. As can be seen from the flow chart the first stage was to import the dataset into the database. The XML and DICOM files could be imported in parallel as they were stored independently of one another in the database. Conversely the **CTSlices** could not be aggregated into **CTStacks** until all of the slices had been imported or some stack could be missing slices. Once the data had been imported the following stages processed it so that it could be used to train a classifier. Stages 4-6 could have been computed all at once. This would reduce the number of update operation performed on the **ROIs** however it was elected that the steps should be separated so that they could each be run individually. The advantage of this method was that each step could be run individually to avoid unnecessary processing time. For example, if it was decided that a parameter used to classify the **ROIs** should be changed then only stage 6 would need to be rerun. If the stages were combined then all of the processing performed in stages 4 and 5 would also need to be performed. This was an important feature to design into **Lungs** as the system processed large amounts of data.

Figure 13 shows a class diagram representing the main classes that were implemented for **Lungs**. It should be noted that one of the classes in the diagram **Lungs** has been named after the system. This is as it is the class that is the end-user uses in order to detect nodules in unseen CT imagery. Some helper classes and models have been omitted from the diagram in the interest of readability. Including these would have hindered readability as they are use in the majority of classes that have included in the diagram. Examples of the omitted classes were all of the models used by the system and classes to help access the database and configuration file. The concrete classes that implemented **Feature** were also omitted as they are numerous and are discussed in detail in Section 4. Many of the classes that are shown in Figure 13 are also discussed in Section 4 however for a more detailed insight the documentation and code submitted alongside this report should be examined. The final system developed in this project mostly conformed to the design detailed in this sections

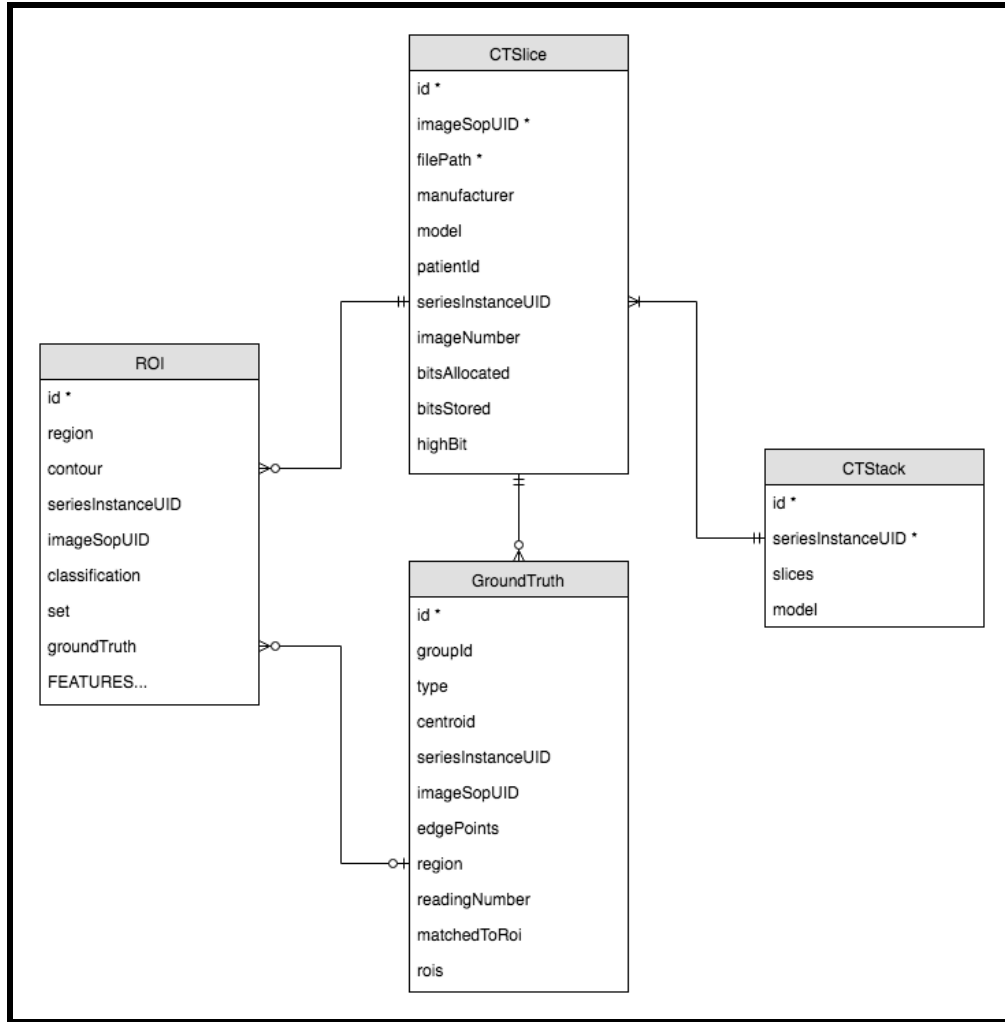


Figure 11: An Entity Relationship Diagram showing the initial design for the structure of the database.

however, any changes that were made throughout the iterative implementation process are described in the following section.

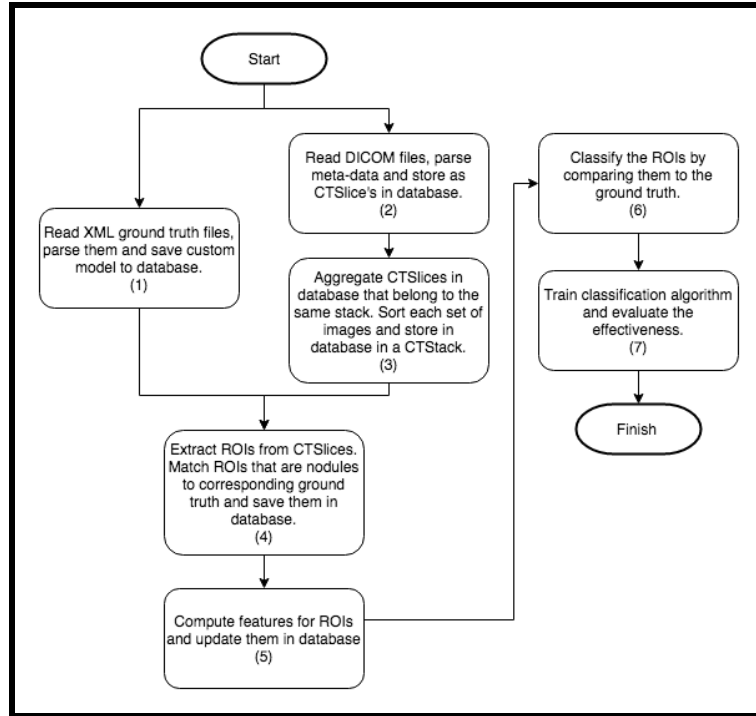


Figure 12: A flow chart showing the top level design for the pipeline used to train the classifier.

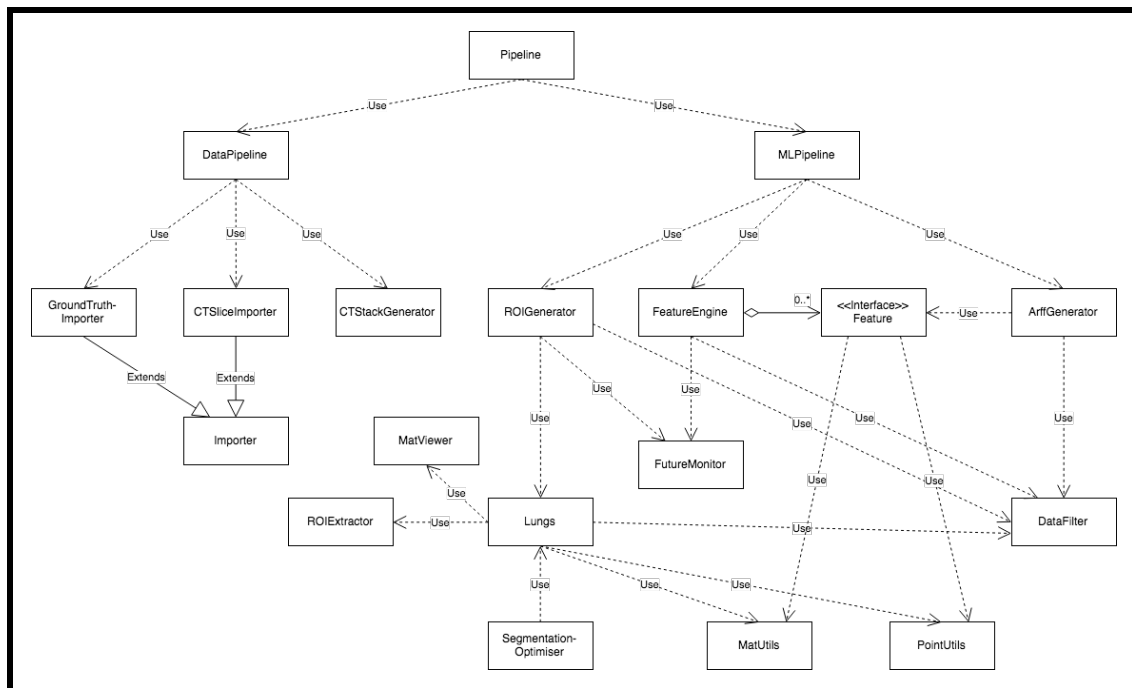


Figure 13: A class diagram showing the main classes used in Lungs.

4 Implementation

This section provides detail on the iterative implementation of Lungs that took place during the project. The code for each of the versions created has been submitted along side this

report, however the dataset had not. This is as the full data set comes to ≈ 126 GB. Appendix 10.1 contains a tree that shows the structure of the source code for the final version of Lungs. The general structure of the project did not change greatly between version. As such this tree should be refereed in order to view the source code for classes discussed in this section.

4.1 Version 0.1

As previously stated the aim for the first version of Lungs was to create a MVP. As such the version needed to be as simple as possible. This subsection details the how the MVP was implemented and the methods and assumptions used to simplify the system. One key simplification of Lungs when compared to the system developed by Tan et al. (2011) was it only operated in 2D. This reduced the complexity of the algorithms developed and allowed more of the functionality implemented by OpenCV to be used. However, the system was developed with the possibility of adding a 3rd dimension at a later date in mind.

4.1.1 Acquisition

The first task during implementation was to import the data required for the system into the database. In order to keep the code organised all the classes used to achieve this were placed in the same package `data`. `Importer` was an abstract class created to provide a generic framework that could be used by subclasses to import data from files.

Algorithm 1 Importer

```

1: procedure RUN
2:   path  $\leftarrow$  the to the file or directory that will be imported
3:   collection  $\leftarrow$  the collection the file(s) will be imported into
4:   collection.drop()
5:   collection.dropIndexes()
6:   importModels(path)
7:   collection.createIndexes()

```

Algorithm 1 shows how `Importer` was used. `importModels()` was an abstract method that was implemented by all subclasses of `Importer`. This allowed importing to be handled in the specific way required by the subclass. The indexes on the collection were dropped before `importModels()` was called and created after, as this reduces the time required to insert the new documents into the collection. In order to reduce query times indexes were added to all the fields that were queried by any part of the system.

`GroundTruthImporter` was a subclass of `Importer` that was used to import the annotations created by the radiologists. In order to do this the XML files needed to be parsed into Java objects. Fortunately the LIDC had provided a schema file on the website that the dataset was downloaded from (The Cancer Imaging Archive 2014). This could be used in conjunction with the Java XML Binding Compiler and the JAXB API (JAXB 2017) to generate the Java classes that the XML files were parsed into. All of these classes were stored in their own package `model.lidc`. The objects obtained from parsing the XML were

then parsed again but this time into `GroundTruth` instances which were save to the database.

Another subclass of `Importer` was `CTSliceImporter`. This class was used to import the meta-data and file paths of the slices provided in the dataset. This was achived by recursively searching the image directory for DICOM files. The meta-data stored in the header of these files was then extracted into a single string using a class provided by the ImageJ library (ImageJ 2017). The following listing shows a truncated example of the meta-data for a single slice:

Listing 1: Truncated DICOM meta-data.

0008,0060	Modality: CT
0008,0070	Manufacturer: GE MEDICAL SYSTEMS
0008,1030	Study Description: CT THORAX W/CONTRAST
0008,103E	Series Description: Recon 2: CHEST
0008,1090	Manufacturer's Model Name: LightSpeed16

In each line of meta-data the characters preceding the colon represent the key and the characters succeeding it the value. The values required for each `CTSlice` object were extracted from the meta-data using regular expressions (Regex) and string handling techniques. These objects were then saved to the database. It was later realised that it may have been more efficient to insert the key-value pairs into a `HashMap` which could then be queried, rather than using Regex. This was as using Regex meant that all the meta-data had to be searched once for each key-value pair required. However, the suggested method was never tested as `CTSliceImporter` was rarely executed so improving efficiency was not a priority. Once `CTSliceImporter` had been run `CTStackGenerator` used Algorithm 2 to aggregate the `CT-Slices` and store them as `CTStacks`.

`DataPipeline` was the top level class used to run all the other classes that have been discussed in this subsection. `DataPipeline` used multi-threading to run `GroundTruthImporter` and `CTSliceImporter` in paralleled followed by `CTStackGenerator`. This implementation could have been more efficient if `GroundTruthImporter`, `CTSliceImporter` and `CTStackGenerator` were multi-threaded internally and run one after another. This is as the implementation used had a maximum of two threads running concurrently. As such it could only ever utilise two CPU cores. If the creation and saving of the models imported/generated by these classes was multi-threaded additional threads could be created to maximise CPU core usage. The ideal number of threads would likely exceed the number of CPUs as while one thread was saving an object another that was not performing any IO would be able to make use of the CPU time. However this improvement was never implemented for the same reason as the previous one discussed in this subsection.

In order to process the slices the DICOM files needed to be read into `Mat` objects. These are objects by OpenCV to represent and manipulate images. This was accomplished using code from the ImageJ library to read the files into DICOM objects. Then the method of `MatUtils` shown in Listing 2 was used to convert the DICOM objects into 8bit `Mats`. The resampling of the DICOM files from their original 16bit depth to to 8bits was handled by the call to

`dicom.getBufferedImage().`

Algorithm 2 CTStackGenerator

```
1: procedure AGGREGATE
2:   stackCollection  $\leftarrow$  the CTStack collection
3:   sliceCollection  $\leftarrow$  the CTSlice collection
4:   field  $\leftarrow$  “seriesInstanceUID”
5:
6:   stackCollection.drop()
7:   stackCollection.dropIndexes()
8:   for each id in stackCollection.findDistinctValues(field) do
9:     slices  $\leftarrow$  sliceCollection.allWhere(field).equals(id)
10:    slices.sort()
11:    stack  $\leftarrow$  new CTStack()
12:    stack.setSlices(slices)
13:    stack.setSeriesInstanceUID(id)
14:    stackCollection.save(stack)
15:  stackCollection.createIndexes()
```

Listing 2: MatUtils.fromDICOM(..)

```
/**
 * @param dicom
 * @return a {@link Mat} read from {@code dicom}.
 */
public static Mat fromDICOM(DICOM dicom) {
    BufferedImage bi = dicom.getBufferedImage();
    Mat mat = new Mat(bi.getHeight(), bi.getWidth(), CvType.CV_8UC1);
    byte[] data = ((DataBufferByte) bi.getRaster().getDataBuffer())
        .getData();
    mat.put(0, 0, data);
    return mat;
}
```

4.1.2 GUI

The next stage of implementation was to create a very simple GUI that could be used to view **Mats**. The class that contained the code for the GUI was **MatViewer**. **MatViewer** allowed two list of **Mats** to be displayed. A common use case for this was one list that contained all the **Mats** for a stack with annotations and another that contained the same **Mats** without the annotations. The lists of **Mats** provided were then converted into **BufferedImages** which were stored in circular lists. The images could then be presented to the user one at a time using a swing **JFrame**. Using keyboard controls the user could then iterate through the

images and swap between the annotated un-annotated version. **MatViewer** was created this early on in the project as it was an invaluable tool for development and debugging. However, it also served as a prototype for the GUI which would be used by radiologists to review the nodules detected by the system.

4.1.3 Configuration

As Lungs was a complex system there were many parameters that required configuration. This was amplified by the fact that the system was not only being used locally but also on a server. For example, the path to the directory containing the ground truth files was different for the two hosts. In order to avoid editing code every time an instance of the project was deployed to the server, a configuration file (**application.conf**) was used to set the parameters. **application.conf** was read using the Apache Commons Configuration library (Apache 2017b). The class **ConfigHelper** was also introduced in order to validate and parse the key-value pairs in the configuration file. The class was also used to provide system wide access to the values. In order to make the configuration system maintainable classes such as **Mode**, shown in Listing 3, were created. These classes provided **public static final** variables that mapped to the keys and values used in **application.conf**. Classes like **Mode** allowed keys and values in **application.conf** to be edited without having to adjust the code in every place they were used.

Listing 3: Mode.java

```
/**
 * The key and values for the system mode configuration
 * variable
 *
 * @author Stuart Clark
 */
public class Mode {

    public static final String KEY = "mode";

    public enum Value {
        PROD, DEV, TEST
    }

    private Mode() {
        // Hide the constructor
    }

}
```

The **mode** key in **application.conf** was used to set the system mode. This in conjunction with **DataFilter** was used to control what data was used by the system. **TEST** mode was used whilst unit tests were ran. When in **TEST** mode a new database was created upon

running the unit tests. This was done in order to avoid corrupting data created in the other modes. **DEV** mode was used during development or when the system was to be tested using a very small subset of the data (one stack for training and one stack for testing). Using a small subset meant that code could be run often during development. **PROD** mode was used when the code was run on the server against the full subset of the LIDC dataset selected, see Section 4.1.4.

4.1.4 Subset Selection

As the LIDC data set was so vast it was decided that initially only a subset of it would be used. One of the motivations for this decision was it reduced the time required to process the images and train the classifier allowing for faster testing of **PROD** mode. In an effort to reduce variance in the slices use by the system, the subset selected contained only stacks created using the same model of CT scanner. The aggregation in Listing 5 (Appendix 10.4) was used to count the number of slices created by each model of CT scanner and determine how many bits were used to represent the highest values in the slices. Table 6 shows the results obtained by running the aggregation. A subset with a uniform highest bit across all slices was chosen in an effort to minimise the amount of preprocessing required. Out of these subsets the **Sensation 16** subset was chosen. One of the reasons this subset was chosen was all the images it contained were of the same resolution so would not need rescaling. Another was that it the largest and as such would provide ample data to train the classifier with.

Once the subset had been decided upon **DataFilter** was used to append conditions to queries so that only results contained in the subset were returned. **DataFilter** was also used to separate the subset into training and testing sets, such that they contained 80% and 20% of the stacks respectively. This method was chosen primarily as it was simple to implement. Stacks were used to create the split, rather than slices, to ensure the training and testing sets were distinct.

CT Scanner Model	Number of Slices	High Bit(s)
Emotion Duo	278	11
Definition	827	11
Brilliance16	1683	11
Brilliance 64	1750	11
Brilliance 40	3416	11
LightSpeed Power	3819	15
Emotion 6	3907	11
LightSpeed Plus	5066	15
Aquilion	7506	15
LightSpeed VCT	9719	15
LightSpeed QX	12594	15
Sensation 64	14240	11
Brilliance 16P	15465	11
Sensation 16	33357	11
LightSpeed Pro 16	34126	15, 11
LightSpeed16	35899	11, 15
LightSpeed Ultra	43973	11, 15

Table 6: The results of running the aggregation in Listing 5

4.1.5 Segmentation

The first stage of the segmentation process implemented by Tan et al. was to resample the CT imagery that was being segmented. As Lungs operated in 2D the distance between slices would not affect it. However, variations in slice thickness would result in voxels containing tissues of the same density to have differing intensity values. Figure 14 shows the distribution of slice thicknesses in the slices used. Despite the variation in slice thicknesses no effort was made to normalise the voxels. This was as it was proposed that the variance in intensity may have been removed (or at least lessened) in the resampling the took place during the acquisition stage. Furthermore, as the density of nodules is variable, the segmentation and classification techniques used would have to be robust to variance in intensity levels. As such the normalisation of voxels could be implemented in a later version if it’s absence appeared to be causing problems.

ROIGenerator was created in order to manage the concurrent extraction of **ROIs** from slices in the training set. These **ROIs** were then saved to the database. For this version only **SPNs** were intended to be segmented. Algorithm 3 provides a basic outline of how each of the slices had it’s **ROIs** extracted. The first stage of the algorithm was to apply a Bilateral filter to the image. This filter was chosen because as discussed in Section 2.3.1 empirical evidence suggested it was the best filter to use to remove noise from CT imagery. Next the filtered image had a binary threshold applied in order to segment the lighter nodules from the dark background. After thresholding a morphological opening operation was ap-

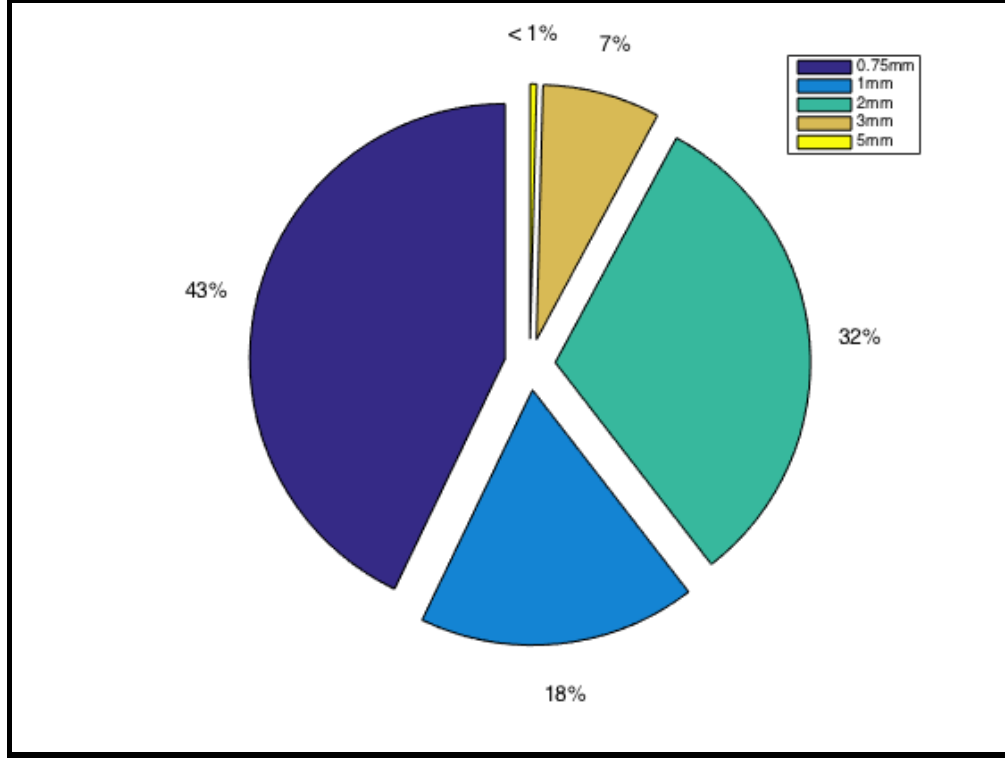


Figure 14: The distribution of slice thicknesses for stacks created using the Sensation 16 CT scanner.

plied to the thresholded image. This was done to remove noise from the thresholded image. Failing to remove this noise would have resulted in a huge number of ROIs being extracted and massively increased the computational complexity of training the classifier. Once the noise had been removed the connected component algorithm was used to extract each of the connected regions in the foreground of the thresholded image into an ROI object. The ROI with the greatest area was then rejected. This was done as the ROI with the largest area was always the area surrounding the lungs. Due to the large number of voxels contained within it, computing features for this ROI was expensive hence it was dropped. Appendix 10.2 contains a set of figures that show the effects of each of the stages of the algorithm.

The following list states the parameters that required values for this segmentation technique. These parameters were omitted from Algorithm 3 for simplicity.

- Size of kernel used for the Bilateral filter.
- The sigma colour for the Bilateral filter. The larger this value the larger the range of intensities found within the kernel that would be mixed together.
- The sigma space for the Bilateral filter. The larger this value the greater the influence of pixels further from the center of the kernel.
- The threshold value used.

Algorithm 3 Solitary Nodule Segmentation

```
1: procedure SEGMENT(slice)
2:   filtered  $\leftarrow$  bilateralFilter(slice)
3:   thesholded  $\leftarrow$  threshold(filtered)
4:   opened  $\leftarrow$  morphologicalOpening(thesholded)
5:   rois  $\leftarrow$  connectedComponents(opened)
6:
7:   largest  $\leftarrow$  null
8:   maxArea  $\leftarrow$  0
9:   for each roi in rois do
10:    if roi.size() > maxSize then
11:      largest  $\leftarrow$  roi
12:      maxArea  $\leftarrow$  roi.area()
13:   rois.remove(largest)
14:
15:   return rois
```

- The shaped of the kernel used for the opening operation.
- The width of the kernel used for the opening operation.
- The height of the kernel used for the opening operation.

The values for these parameters were not easy to determine as there were many permutations of values which could be used. Furthermore, changing the values used at each stage in the algorithm would impact the optimum values for all the stages after. As such a Genetic Algorithm (GA) was used to set the parameters. A GA was selected above other combinatorial optimisation algorithms as it was easy to implement and allowed for the parameters that were being optimised to be changed with ease. This was as no neighbourhood functions had to be devised for them. The GA was implemented in `SegmentationOptimiser` with the aid of the Jenetics library (Jenetics 2017). This library was chosen as it was well documented, required little additional code to be written and handled all the multi-threading. Each member of the GA’s population was used to run the segmentation algorithm with the parameter values present in its genotype. The effectiveness of the parameters used was then evaluated using the fitness function defined by Algorithm 4. The fitness function worked by calculating the mean match score for all of the nodules. All possible fitness values fell within the range $[0 - 1]$. In order to obtain a mean value the total number of nodules in the training set needed to be known. As the LIDC ground truth was composed of the annotations of 4 separate radiologists it contained many duplicates. It would have been possible to remove these duplicates using clustering techniques. However, in the interest of simplicity, it was elected that annotations created by the radiologist who made who identified the most nodules should be used. This was as it was better for the system to produce False Positives (FPs) rather than False Negatives (FNs). This method of filtering the ground truth was applied across the whole system. The function `match(...)` used the formula defined in Equation 5 to calculate how well each nodule had been matched to an ROI, see Section 4.1.6. An alternative fitness

function could have been to maximise the number of nodules that had a match score above a certain threshold. However, the fitness function in Algorithm 4 was chosen so that the nodules that were included in the segmentation were segmented as accurately as possible. This would result in more representative features being computed later in the the pipeline. Furthermore, as in (Tan et al. 2011) additional segmentation techniques could be introduced to segment the nodules that were missed.

The GA was run against the full training set using a population size of 200. Ideally a larger population size would have been used. However, due to the computational complexity of the segmentation technique and fitness function used this was not feasible. The GA was assumed to have found the optimum parameters when three generations had been evaluated with no increase in the maximum fitness for the population. The following list shows the parameters obtained:

- Bilateral Filter Kernel Size = 3×3
- Sigma Colour = 1
- Sigma Space = 7
- Threshold = 73
- Opening Kernel Shape = Elliptical
- Opening Kernel Width = 4
- Opening Kernel Height = 1

Algorithm 4 SegmentationOptimiser

```

1: procedure FITNESS
2:   slices  $\leftarrow$  a list of all the slices in the training set that contain nodules
3:   numNodules  $\leftarrow$  total number of nodules for the slices in the training set ground truth
4:   fitness  $\leftarrow$  0.0
5:
6:   for each slice in slices do
7:     for each groundTruth in slice.getGroundTruths() do
8:       bestScore  $\leftarrow$  0.0
9:       for each roi in slice.getRois() do
10:        score  $\leftarrow$  match(roi, groundTruth)
11:        if bestScore < score then
12:          bestScore  $\leftarrow$  score
13:        fitness  $\leftarrow$  fitness + bestScore
14:   fitness  $\leftarrow$  fitness/numNodules
15:
16:   return fitness

```

These parameters achieved an average match score of 0.22133346318817104. In order to sanity check the threshold used, histograms were created for the nodules and slices in the training set. These were computed using `NoduleHistograms` and `SliceHistograms` respectively. The nodule histogram bins were then subtracted from the respective bins in the slice histograms to create a “non-nodule-slice” histogram. The values in the bins were then converted into frequencies and plotted to create Figure 15.

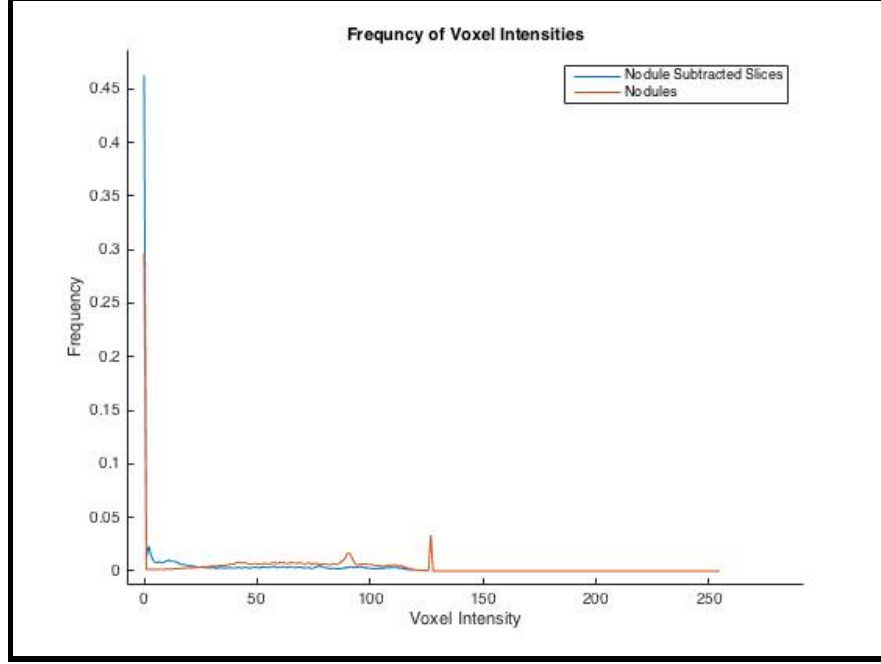


Figure 15: Frequency of voxel intensities comparison.

Figure 15 showed that the nodules had a disproportionate frequency of intensities in the range of approximately 85 – 95 when compared to rest of the surrounding slices with. Another observation of the figure was that nodules had a high frequency of voxels with an intensity of 0. As the non-nodule-slices exhibited this trait also it was assumed that these voxels had been unintentionally included in the nodules during the annotation process. Figure 16 was used to compare the cumulative frequency of the voxel intensities in the nodules and non-nodule-slices. In accordance with the assumption stated, the nodule voxels with intensity values of 0 were omitted from this figure. Using the values shown by the annotations in Figure 16 probabilities could be calculated.

$$p(S) = p(\text{non-nodule-slice voxles present after thresholding}) = 1 - 0.8273 = 0.1727$$

$$p(N) = p(\text{nodule voxles present after thresholding}) = 1 - 0.5254 = 0.4746$$

$p(S)$ was a good indicator that the segmentation technique would not return too many FPs. This is as only ~ 0.1727 of the pixels that were FPs would be returned from thresholding. Having as few FPs as possible was important as it reduced computational complexity in the later stages of the pipeline. Furthermore, fewer FPs would reduce the number of non-nodules that the classifier would have to identify. $p(N)$ showed that ~ 0.4746 of the nodule pixels

would be returned from thresholding indicating that the sensitivity of the segmentation technique would be reasonable. These probabilities were only approximations as they were computed using the unfiltered intensity values and did not factor in any stages of segmentation other than the thresholding. When the segmentation was run with the parameters stated there were 24602699 ROIs generated for the data set.

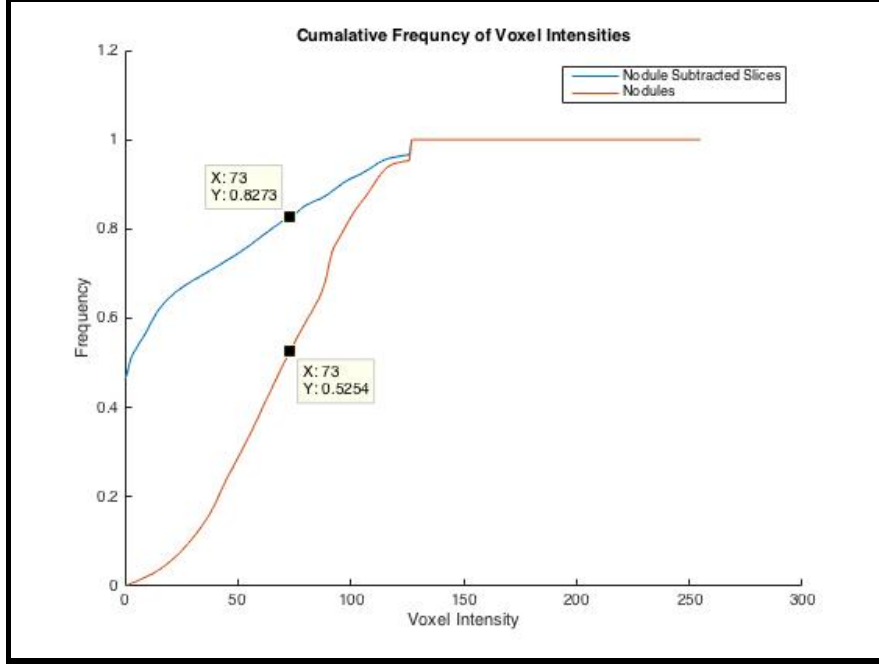


Figure 16: Cumulative frequency of voxel intensities comparison.

4.1.6 ROI Classification

Once the ROIs had been extracted it was necessary to classify them as either nodules or non-nodules so that they could be used to train the classifier. Tan et al. (2011) measured the affinity of candidate nodules and nodules identified by the ground truth by measuring the Euclidean distance between centroids. A drawback of this method is that it does not measure how well nodules have been segmented. To overcome this issue Equation 5 was used. This method was suggested by project supervisor Paul Rosin.

$$\begin{aligned}
 R &= \{\text{all points in the ROI}\} \\
 G &= \{\text{all points in the GroundTruth}\} \\
 \text{match}(R, G) &= \text{match score for } R \text{ and } G \\
 \text{match}(R, G) &= \frac{\| R \cap G \|}{\| R \cup G \|}
 \end{aligned} \tag{5}$$

Similarly to the system developed by Tan et al. it was decided that nodules provided by the ground truth with a diameter less than 3mm should be ignored. This was as the annotations provided only gave the centroid for the nodules. As such a technique such as region growing

would need to be used in order to obtain the regions for the nodules. Algorithm 5 was used to classify the ROIs. Each ROI was compared to all the **GroundTruths** that appeared in the slice it belonged to. The comparisons were made using Equation 5 which returned match scores in the range $[0, 1]$. A match score of 0 indicated no match at all, where as a match score of 1 indicated a perfect voxel for voxel match between the ROI and **GroundTruth**. If the highest match score for a ROI was greater than the “match threshold” the ROI was classified as a nodule, otherwise it was classified as a non-nodule. It was important that each ROI was only compared to the **GroundTruths** that appeared in the same slice as the ROI in order to reduce computational complexity. Algorithm 5 was used both in **ROIGenerator** and **ROIClassifier**. In **ROIGenerator** it was used to classify newly generated ROIs before they were saved, where as **ROIClassifier** used it to reclassify ROIs using a new match threshold without having to generate them again.

Algorithm 5 ROI Classification

```

1: procedure CLASSIFYROIs(threshold)
2:   slices  $\leftarrow$  all the slices that had nodules according to the ground truth
3:
4:   for each slice in slices do
5:     groudTruths = slice.getGroundTruths()
6:     rois = slice.getRois()
7:
8:     for each roi in rois do
9:       Classify(roi, groudTruths, threshold)
10:    save(rois)
11:
12: procedure CLASSIFY(roi, groudTruths, threshold)
13:   bestScore  $\leftarrow$  0.0
14:   for each gt in groudTruths do
15:     score  $\leftarrow$  match(roi, gt)
16:     if bestScore < score then
17:       bestScore  $\leftarrow$  score
18:
19:   if bestScore > threshold then
20:     roi.setClassification(NODULE)
21:   else
22:     roi.setClassification(NON_NODULE)

```

4.1.7 Classifier Training and Testing

In order to train and test the classifier it was necessary to compute features for the ROIs. **FeatureEngine** was the classed used to accomplish this. To reduce the amount of IO required **FeatureEngine** was implemented so that each slice would only be loaded into memory once. Objects used to compute features were stored in a list and computed sequentially for

each ROI. This was an inefficient way of computing the features as it meant that the voxels in the ROIs were iterated over multiple times. However, this method was favoured as it vastly simplified the code and allowed for features to be added or removed with great ease. **FeatureEngine** used multi-threading to reduce the amount of time required to compute the features. Each ROI had its own thread which computed the features for the ROI then updated the database. The maximum number of threads that were executed concurrently was equal to the number of CPU cores available (8 in the case of the server). This limit was imposed as using more than 8 threads required too much memory. The features used in this version were: Mean Intensity, Area, Perimeter, Min Circle Radius, Fitted Ellipse Angle, Fitted Ellipse Area, Fitted Ellipse Width, Fitted Ellipse Height, Coarse Hist , Medium Hist , Fine Hist . A discussion of how features were implemented and why they were chosen is provided in Section 4.5.1

The classifier was trained and tested using **ArffGenerator**. This class was used to create two ARFF files, one for training and one for testing. These files were used to store the feature values for the instances used to train and test the classifier. As stated in Section 2.3.2 it is important to train the classifier using roughly equal numbers of instances for each class, in this case nodules and non-nodules. In this version both the testing and training files contained equal numbers of instances for each class. The classification algorithm chosen for use during development was Weka's J48 classifier. This algorithm was chosen as it uses a decision tree. This was advantageous as using tools provided by Weka decision trees can be viewed giving a useful insight into how the classifier is working.

4.1.8 Evaluation

In order to evaluate the success of the version and decide what improvements could be made **MLPipeline** was run. This class was responsible for running all code required to train and test the classifier. 1580/24602699 ROIs obtained by the segmentation technique could be matched to a nodule by at least one voxel. From these results it was known that multiple ROIs were being matched to the same nodule. This was as according to the ground truth used there were 1381 nodules in the images processed. Furthermore, JPNs were not yet expected to be included in the segmentation. The mean match score for these ROIs was 0.33229. Figure 24 shows the distribution of match scores amongst the matched ROIs. From the figure it could be seen that some nodules had been segmented with a high degree of accuracy however, there was definitely room for improvement.

The classifier was trained using a match threshold of 0.0, in order to maximise the number of instances used for testing and training. Out of the nodules obtained 1085 were used for training and 495 were used for testing. Testing revealed the classifier achieved a total accuracy of 85.6566%, a TP rate of 0.764 and a FP rate of 0.051. However, these figures were unlikely to be representative of the classifier's true performance as in reality there would be far more non-nodules than were present in the testing set.

4.2 Version 0.2

Overall version 0.1 of Lungs achieved better results than expected. This is as many of the implementation choices made were based purely of getting a working system as fast as possible. The main aims for the Version 0.2 were to: improve the segmentation technique for SPNs, add an additional segmentation technique for JPNs, add additional features to improve classifier accuracy and implement classifier training in such a way that the statics obtained was were more representative of the classifier’s performance.

4.2.1 Segmentation

In order to try to improve the segmentation of SPNs it was decided that a region growing algorithm should be used. Region growing was an attractive method as it offered the potential for a more intensity robust segmentation technique. The region growing algorithm selected was the the marker based Watershed algorithm implemented in OpenCV (OpenCV 2015). This algorithm works by selecting markers which are placed on an image to define seed regions. These regions are then incrementally expanded. Neighbouring pixels with a low gradient are expanded into more readily and vice versa. As the regions expand they begin to collide. The pixels where the regions meet are marked as the boundaries between them and are no longer used to further expand the regions. The algorithm stops when all the pixels have been assigned a region.

In order for the segmentation technique in version 0.1 to perfectly segment all the nodules they would all have had to have the same minimum intensity vale, as shown in Figure 17 this was not the case. Using a region growing algorithm meant that each nodule would be considered individually and as such it would be possible to account for the variations in the nodules. Tan et al. (2011) modelled SPNs as “a sphere with decreasing intensity along the radial axis against a darker background”. A similar model was adopted for Lungs differing in that, as the system operated in 2D, nodules were modelled as circles. This assumption was supported by the fact that malignant nodules grow over time so are likely to be denser in the middle than at the edge. It also conformed to what has been observed visually in the slices. The solid blue line in Figure 18 shows a 1D representation of the model used. The area between the red markers represents a SPN and the area outside it’s surrounding tissue. Algorithm 6 shows how the seed regions were obtained by Lungs. The intuition behind this method is best described with reference to Figure 18. The area above the dashed red line represents a seed region identified using $fgThreshold(filtered)$. The area bellow the dashed magenta line represents a seed region identified using $bgThreshold(filtered)$. The area between the lines is where the regions were expected to expanded into using the watershed algorithm. As the gradient between voxels in the nodule is less than the gradient between voxels at the edge and the surroundings tissue, the foreground region would expand to fully segment the nodule before the background was region was expanded.

Version 0.2 also included the first attempt to segment JPNs present in the slices. Algorithm 7 shows the method used to achieve this and complementary images can be found in Appendix 10.3. The main assumption made by this algorithm was that the tissues the surrounding

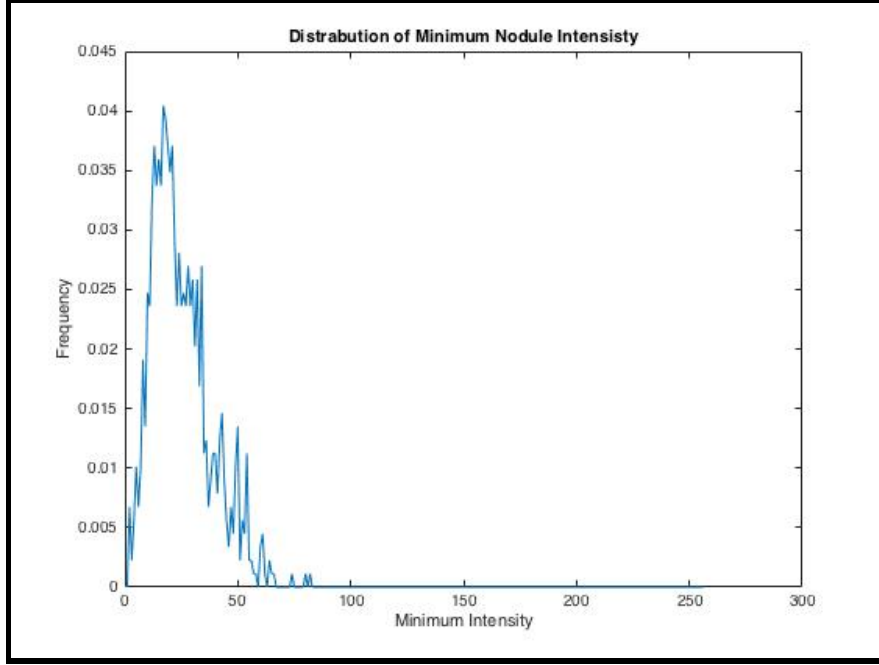


Figure 17: Distribution of minimum nodule voxel intensities.

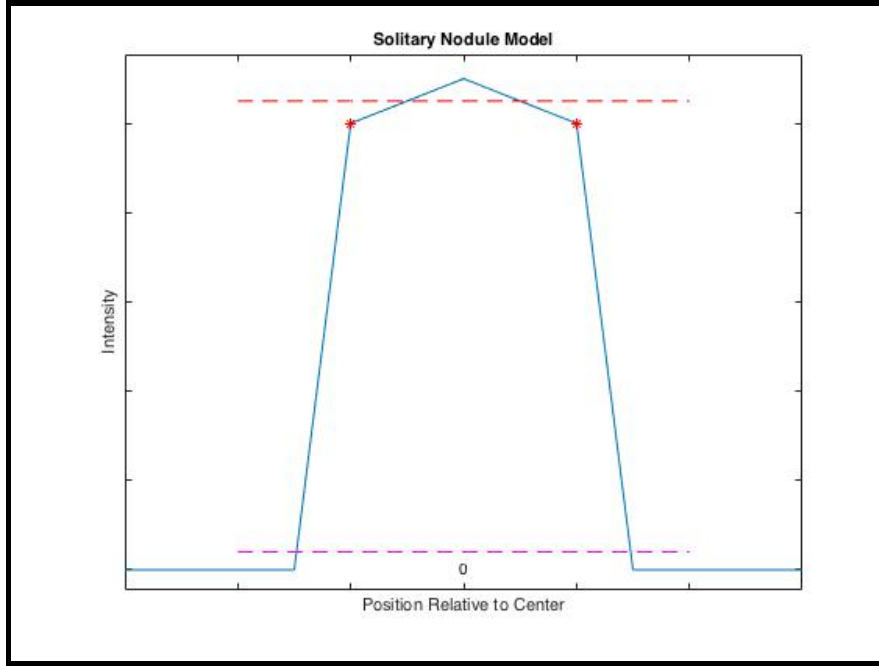


Figure 18: A one dimensional representation of the model used for SPNs.

the lungs were always segmented into a single ROI, and as such the plural cavities could be obtained from the ROI. As shown in Algorithm 6 once all the ROIs had been obtained ROIs with a radius > 35.2279 or < 1.0 pixels were rejected. These values were obtained by measuring the radius of nodules in the training set. This had the effect of rejecting the

background region obtained using the watershed algorithm and other FPs.

Similarly to version 0.1 a GA was used to optimise the various parameters for the segmentation technique. The following list states the parameters and the values obtained through optimisation.

- Bilateral Filter Kernel Size = 5×5
- Sigma Colour = 3
- Sigma Space = 3
- Foreground threshold (used by *fgThreshold(.)*) = 65
- Background threshold (used by *bgThreshold(.)*) = 104
- Size of erosion structure = 6×6

These results were surprising as the foreground threshold was less than background threshold. This meant that when the background was subtracted from the foreground there was no unknown region. This in effect meant that a binary threshold of 65 was being used to segment the ROIs and no region growing was occurring.

Algorithm 6 Nodule Segmentation

```
1:  $MAX\_R \leftarrow 35.2279$ 
2:  $MIN\_R \leftarrow 1.0$ 
3:
4: procedure SEGMENT(slice)
5:    $filtered \leftarrow bilateralFilter(slice)$ 
6:
7:    $sureForeground \leftarrow fgThreshold(filtered)$ 
8:    $sureBackground \leftarrow bgThreshold(filtered)$ 
9:   // Note this is element by element matrix subtraction
10:   $unknown \leftarrow sureBackground - sureForeground$ 
11:
12:  // Labels values of 0 identify the unknown region, labels >0 identify the seed regions
13:  // Init all labels as 1 so they default to background region
14:   $labels \leftarrow matOfOnes(size(slice))$ 
15:  // Add the foreground regions
16:   $id \leftarrow 2$ 
17:  for each region in connectedComponents(sureForeground) do
18:     $labels.setAllInRegion(region, id)$ 
19:     $id \leftarrow id + 1$ 
20:  // Add the unknown regions
21:  for each region in connectedComponents(unknown) do
22:     $labels.setAllInRegion(region, 0)$ 
23:
24:   $rois \leftarrow watershed(filtered, labels)$ 
25:
26:  // Find the largest ROI
27:   $largest \leftarrow null$ 
28:   $maxArea \leftarrow 0$ 
29:  for each roi in rois do
30:    if  $roi.size() > maxSize$  then
31:       $largest \leftarrow roi$ 
32:       $maxArea \leftarrow roi.area()$ 
33:
34:  // See Algorithm 7
35:   $rois.add(Juxtapleural(largest, slice))$ 
36:
37:  // Remove ROIs that are too big or too small
38:  for each roi in rois do
39:     $radius \leftarrow minimumCircle(roi).radius()$ 
40:    if  $radius < MIN\_R$  OR  $radius > MAX\_R$  then
41:       $rois.remove(roi)$ 
42:
43:  return rois
```

Algorithm 7 Juxtapleural Nodule Segmentation

```
1: procedure JUXTAPLEURAL(largestRoi, source)
2:   // Get the convex hull of cavities in largestRoi
3:   contours  $\leftarrow$  largestRoi.internalContours()
4:   hulls  $\leftarrow$  convexHulls(contours)
5:   hullsImg  $\leftarrow$  imageOfZeros(source.size())
6:   hullsImg.paintFilled(hulls)
7:
8:   // Create the mask (morphological erosion using circular kernel)
9:   eroded  $\leftarrow$  erode(hullsImg)
10:  mask  $\leftarrow$  invert(eroded)
11:
12:  // Apply the mask to largestRoi
13:  roiImg  $\leftarrow$  imageOfZeros(source.size())
14:  roiImg.paintFilled(largestRoi)
15:  // Note this is element by element subtraction of two matrices
16:  masked  $\leftarrow$  roiImg  $-$  mask
17:
18:  rois  $\leftarrow$  connectedComponents(masked)
19:
20:  return rois
```

4.2.2 Classifier Training and Testing

The features used to train and test the classifier for this version were: Juxtapleural, Mean Intensity, Area, Perimeter, Min Circle Radius, Circularity, Fitted Ellipse Angle, Fitted Ellipse Area, Fitted Ellipse Width, Fitted Ellipse Height, Elongation, Coarse Hist, Medium Hist and Fine Hist. The only other change made from version 0.1 was that the classifier was tested using all the instances in the testing set. In order to do this the instances had to be loaded incrementally from the ARFF file as there was too many to fit them into memory.

4.2.3 Evaluation

Despite the parameters used for the watershed algorithm preventing any region growing version 0.2 yielded improved match scores for the SPNs successfully segmented, see Figure 24. This was most likely due to the fact FPs were removed based on radius measurements rather than applying morphological opening. Table 7 shows the results of the segmentation with and without JPN segmentation turned on. These results showed an improvement on version 0.1 as the number of ROIs matched to at least one voxel of a nodule fell below the number of nodules expected. This indicated that there was likely to be fewer duplicates than in version 0.1. Furthermore, with JPN segmentation turned on 187 additional ROIs were matched to nodules without dramatically affecting the mean match score.

Despite the improvement in the segmentation technique it appeared that many nodules where

Juxtaplural On	ROIs matched ≥ 1 voxel	Total ROIs	Mean match score
No	808	8523284	0.4088
Yes	995	10741007	0.40467

Table 7: V0.2 Segmentation Results.

still being missed. For example when the match threshold was set to 0.5 only 781 out of 1381 nodules were detected. This indicated that many nodules were being missed or poorly segmented during segmentation. In order to ascertain the characteristics of the nodules that were being missed **MissedNodules** was created. This class was used to create images and histograms of the nodules that had not been matched to any **ROI**. In these images the vast majority of the nodules appeared completely black. Figure19 shows the distribution of voxel intensities for the unmatched nodules.

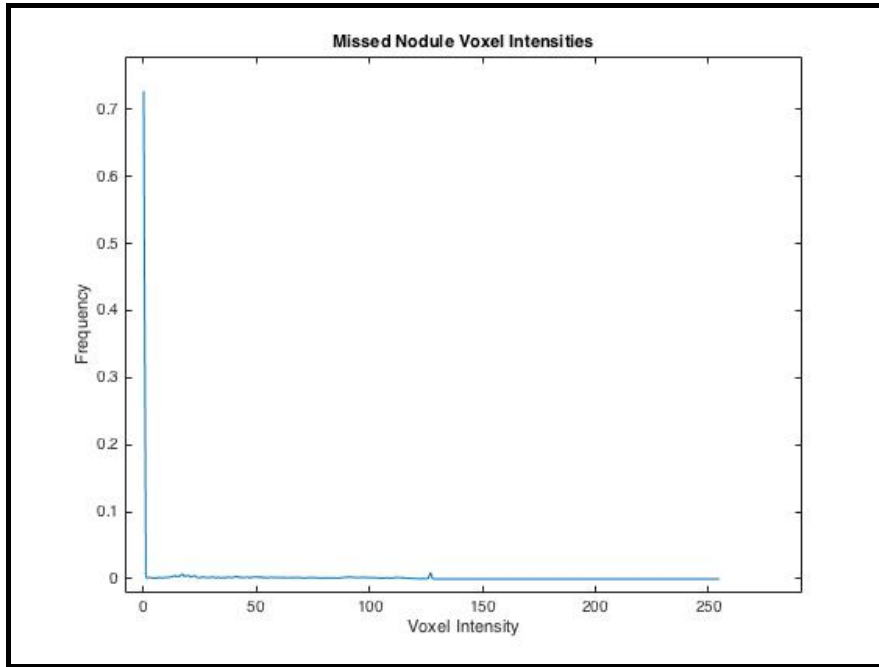


Figure 19: A histogram showing the distribution of voxel intensities for the unmatched nodules.

As can be seen from Figure 19 the vast majority of the voxels that were not detected had an intensity of 0. As such these nodules would be very hard to detect against the dark background of the plural cavity.

For this version the classifier was again trained using a match threshold of 0.0. 750 nodules and 750 non-nodules were used for training. Where as testing used 245 nodules and 1346799 non-nodules. Testing revealed the classifier achieved a total accuracy of 60.1378% and a FP rate of 0.399. This decrease and increase respectively was expected due to the improved

testing method implemented. The TP rate had increased to 0.837 this was a good indicator that the improved segmentation and additional features had aided classification.

4.3 Version 0.3

The main focus of this version was to move the whole system over to using 16bit slices as opposed to the 8bit slices that had been used up to this point. It was hoped by increasing the bit depth that the unmatched nodules could more easily be detected. In order to obtain the 16 bit images the PixelMed Java DICOM Toolkit (PixelMed 2016) was used. This library was chosen over the previous one as it offered a far simpler API that could be used to control the bit depth that the images were imported at. Once the code to read the DICOM images in to `Mats` had been implemented the next task was to make them displayable with annotations. This required converting the `Mats` into `BufferedImage`s. In order to do this a non-standard colour space had to be defined which supported 16bits per channel. The implementation for this can be seen in `MatUtils.toBufferedImage(...)`.

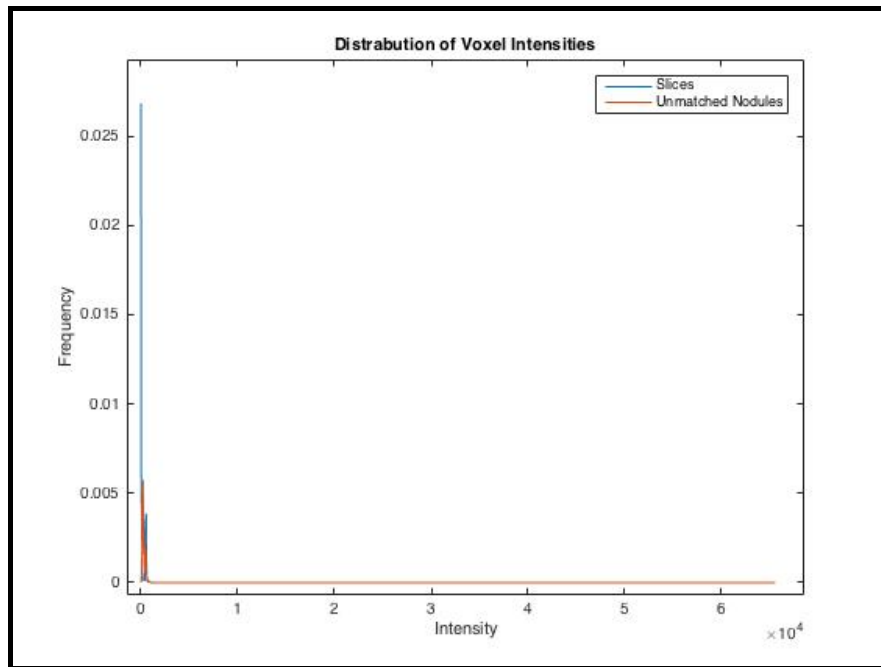


Figure 20: A histogram showing the distribution of voxel intensities for the unmatched nodules using 16 bit imagery.

To see if the unmatched nodules could be detected using a 16bit system more histograms were created. As can be seen from Figure 21 there is characteristic peak for the unmatched nodules when using 16bit imagery. This showed promise that increasing the bit depth used over the system would improve detection rates. However, much of OpenCVs pre-implemented functionality does not currently support 16bit images, for instance the watershed algorithm. As the time allotted for the project was waning, it was elected to spend it improving the 8bit system rather than moving over to another library or implementing the missing functionality from scratch.

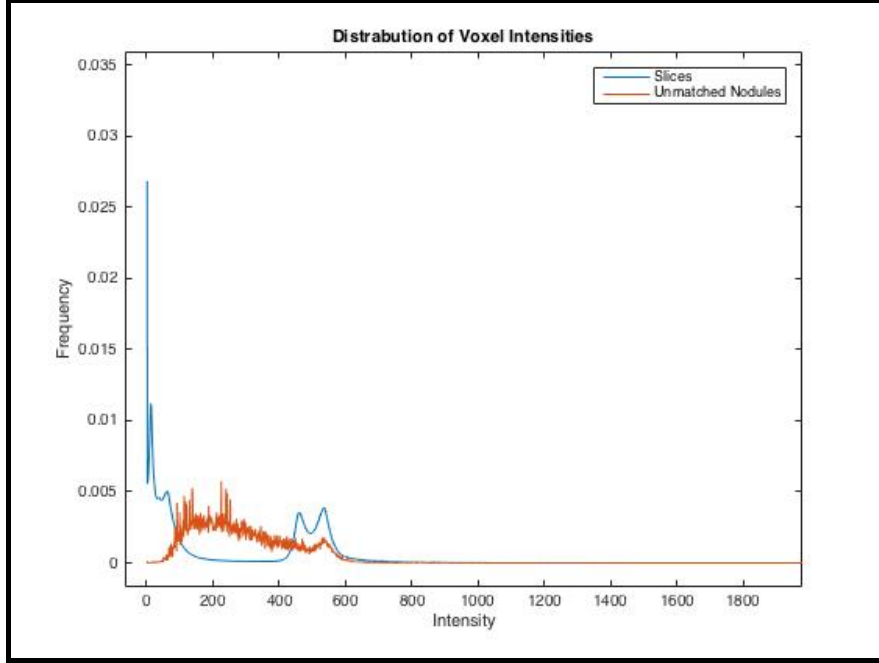


Figure 21: A closer look at Figure 20

4.4 Version 0.4

The main aims for version 0.4 were to: reduce the number of FPs returned by segmentation, explore another juxtaplural segmentation technique and add additional features. It was also decided that the segmentation optimiser should be forced to return appropriate values for the watershed algorithm. This was to ensure that parameters obtained in version 0.2 had not just occurred as a quirk of optimisation.

4.4.1 Segmentation

The primary method for reducing the number of FP obtained through the segmentation of SPNs was to use a mask. Similarly to Tan et al. (2011) the mask was obtained using the largest ROI returned from segmentation. Once the largest ROI had been found the areas that filled it's internal contours were used as the mask. Any ROIs that had one or more pixel outside this mask were rejected. Appendix 10.5 provides images that show visually how the masking worked.

The alternate JPN segmentation technique proposed was based around a blob detector very similar the initial stages used in David Lowe's SIFT (Lowe 2004). This blob detector developed was implemented from scratch so that it could be optimised for the task at hand. A blob detector was a valid option for detecting JPNs as from observations of the dataset it could be seen they were in most cases blob like protrusions into the lungs. The initial stages of the segmentation technique were essentially the same as used in SIFT. First a Difference of Gaussian (DOG) pyramid was created using a wide range of sigma values, see Figure 22. This was implemented in `DOGPyramid` using the same method and parameters

described in (Lowe 2004). In SIFT once the DOG pyramid has been constructed the local extrema are found using a 3×3 neighbourhood and returned as key-points. For the blob detector implemented (**BlobDetector**) only the maxima were found. This is as the nodules that were to be detected were composed of high intensity voxels. In SIFT the extrema are found to sub pixel precision through interpolation using the Taylor expansion. This was not the case for **BlobDetector**. For the sake of simplicity nearest neighbour interpolation was used. This was deemed likely to be sufficiently accurate as the key-points returned were not required to be at the exact center of the nodules, see Algorithm 8. To reduce the number of key points that were returned three checks were used. These were made before the scale space neighbourhood was examined. Performing the checks at this stage greatly improved performance. The first check was to ensure that the key-points lay within the convex hull of the plural cavity but not within the plural cavity. Algorithm 8 shows how the mask used to achieve this was obtained. The second check used a threshold to reject key-points with a DOG value below a given threshold and the third check used a threshold to reject key-points with too large a gradient. The remaining key-points were converted into blobs using the sigma coordinate for each key-point as the radius. Once the blobs had been returned from **BlobDetector** an additional algorithm was used to better segment the nodule candidates contained within them, see Algorithm 9. This algorithm was implemented in **BlobToROI**.

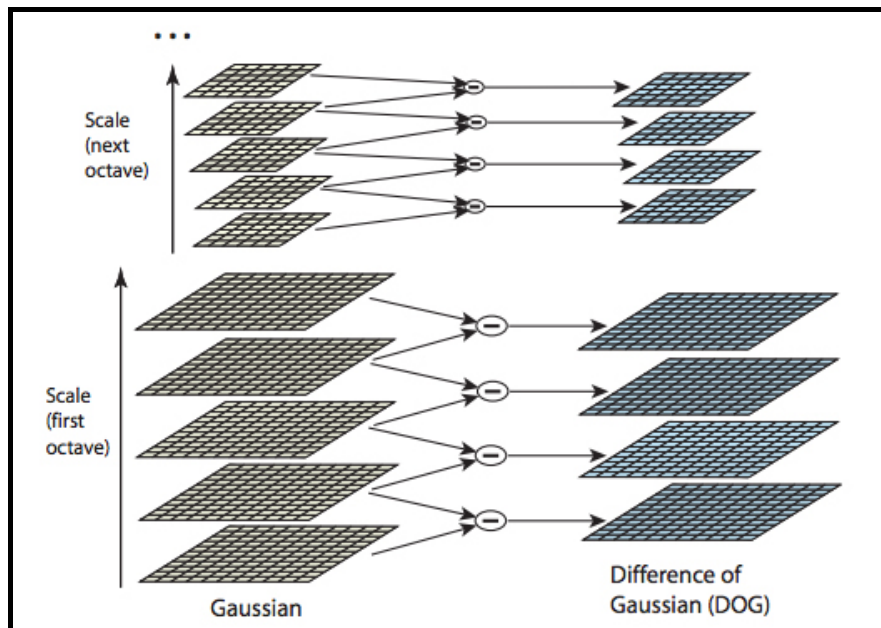


Figure 22: A visual representation of the DOG pyramid implemented (Lowe 2004)

Figure 23 provides a visual example of how each blob was segmented. The image in the top left shows sub-matrix created for a JPN. The image to the right of this shows the blob that was detected by **BlobDetector** for the JPN. The equation for the *subMatSize* shown in Algorithm 9 was used as it returned yielded a sub-matrix with twice the area of the blob, as shown in Equation 6. As such, for the sake of thresholding each sub-matrix was assumed to have a bi-modal distribution of intensities (a light area for the nodule and a dark area

Algorithm 8 Juxtapleural Nodule Segmentation

```
1: procedure JUXTAPLEURAL(largestRoi, slice)
2:   // Create the mask
3:   contours  $\leftarrow$  largestRoi.internalContours()
4:   hulls  $\leftarrow$  convexHulls(contours)
5:   hullsImg  $\leftarrow$  imageOfZeros(slice.size())
6:   hullsImg.paintFilled(hulls)
7:   inverted  $\leftarrow$  invert(hullsImg)
8:   roiImg  $\leftarrow$  imageOfZeros(slice.size())
9:   roiImg.paintFilled(largestRoi)
10:  // Note this is element by element subtraction of two matrices
11:  mask  $\leftarrow$  roiImg - inverted
12:
13:  rois  $\leftarrow$  emptyList()
14:  for each blob in BlobDetector.getBlobs(slice, mask) do
15:    rois.add(SegmentBlob(blob, slice))
16:
17:  return rois
```

for the surrounding tissues). This meant that Ostu thresholding could be applied to each sub-matrix in order to better segment the nodule. The image at the bottom left of Figure 23 shows regions that remained when the blob was subtracted from the thresholded image. The image to the right of this shows the region that connected the blob to the plural membrane. This region was then subtracted from the original thresholded image in order to obtain the improved ROI for the nodule (shown bottom right).

$$\begin{aligned} r &= \text{the radius of the blob} \\ b &= \pi r^2 = \text{the area of the blob} \\ l &= \text{the length of one side of the sub-matrix} \\ s &= l^2 = \text{the area of the sub-matrix} \\ s &= 2b \\ l^2 &= 2\pi r^2 \\ l &= \sqrt{2\pi r^2} \end{aligned} \tag{6}$$

BlobOpt was created in order to optimise the thresholds that were used to limit the number of key-points detected by **BlobDetector**. As these two parameters were independent of one another they could be optimised using two sequential binary searches. The first binary search was used to find the maximum DOG threshold that could be used whilst maintaining maximum nodule inclusion. Similarly, the second binary search was used to find the minimum gradient threshold that could be used whilst maintaining maximum nodule inclusion. Nodule inclusion was computed using the same method as **SegmentationOptimiser**, this was discussed in Section 4.1.5. The maximum nodule inclusion was found by setting the

Algorithm 9 Blob Segmentation

```
1: procedure SEGMENTBLOB(blob, slice)
2:    $r \leftarrow \text{blob.getRadius}()$ 
3:    $\text{subMatSize} \leftarrow \sqrt{2\pi r^2}$ 
4:   // Returns a sub subMatSize by subMatSize sub-matrix with the blob
5:   // at the center
6:    $\text{subMat} \leftarrow \text{createSubMat}(\text{slice}, \text{blob}, \text{subMatSize})$ 
7:
8:    $\text{thresholded} \leftarrow \text{otsuThreshold}(\text{subMat})$ 
9:
10:  // Create a list of all the ROIS that are connected to the blob
11:   $\text{rois} \leftarrow \text{connectedComponents}(\text{thresholded})$ 
12:  for each roi in rois do
13:    if NOT  $\text{connected}(\text{blob}, \text{roi})$  then
14:       $\text{rois.remove}(\text{roi})$ 
15:
16:  if  $\text{rois.size}() \neq 1$  then
17:    // When tested this error was only thrown 1.9% of cases
18:    throw ERROR
19:   $\text{blobROI} \leftarrow \text{rois.get}(0)$ 
20:
21:   $\text{thresholded.remove}(\text{blob})$ 
22:
23:  for each roi in  $\text{connectedComponents}(\text{thresholded})$  do
24:    if  $\text{touchingSideOfImg}(\text{roi}, \text{submat})$  then
25:       $\text{blobROI.removeIntersection}(\text{roi})$ 
26:
27:  return  $\text{blobROI}$ 
```

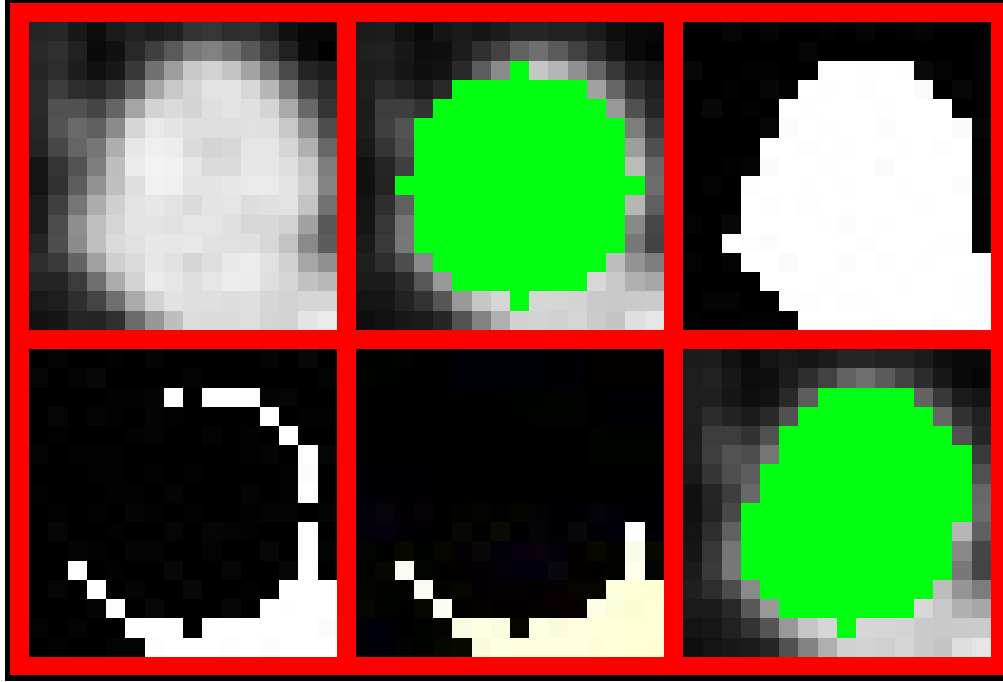


Figure 23: A visual example of the main steps in Algorithm 9.

DOG and gradient thresholds to values that would result in no key-points being filtered. Obtaining the maximum nodule inclusion was highly computationally expensive. This was because as no thresholding of key-points takes place a huge number of ROIs needed to be considered. This being the case and with the remaining time for the project rapidly depleting the thresholds were optimised using only a few stacks known to have nodules in. The values obtained were 27 for the DOG threshold and 51 for the gradient threshold. In order to try to account for variance in the nodules these values were decreased and increased by 20% respectively.

Forcing the segmentation optimiser to return appropriate values for the watershed algorithm required only a very small change. Instead of optimising a parameter that was used as the background threshold a new parameter was introduced with possible values in the range $[0 - 0.9]$. This parameter was named the “background fraction” and was multiplied by the foreground threshold in order to calculate the background threshold. This guaranteed that the background threshold would always be less than the foreground threshold. The parameters obtained from optimisation were as follows:

- Bilateral Filter Kernel Size = 4×4
- Sigma Colour = 8
- Sigma Space = 6
- Foreground threshold = 105
- Background fraction = 0.8987624741608482

4.4.2 Classifier Training and Testing

In this version the ability to oversample ² the nodule instances was introduced. Oversampling the nodules meant that more of the non-nodule instances in the training set could be used to train the classifier whilst maintaining the ratio of nodules to non-nodules. This would hopefully reduce the number of FPs returned during testing. The features that were used in this version were: Juxtapleural, Mean Intensity, Area, Perimeter, Min Circle Radius, Circularity, Hu Circularity, Convexity, Fitted Ellipse Angle, Fitted Ellipse Area, Fitted Ellipse Width, Fitted Ellipse Height, Elongation, Coarse Hist, Fine Hist, Coarse LTP and Fine LTP.

4.4.3 Evaluation

The value obtained from optimisation for the background fraction could not have been much closer to 0.9. This is a strong indicator that the values obtained in version 0.2 were not a quirk of optimisation. To try to gain a better understanding of why the algorithm did not appear to be working as expected the source code was investigated. This revealed a major oversight in the way it had been used in the system. The algorithm used a priority queue to ensure that pixels with the lowest intensities were examined first when attempting to expand regions. This meant if the nodule model presented in Figure 18 was correct, the background region would be expanded until it met the foreground, effectively resulting in a binary threshold. As such, it was proposed that if the slices were inverted before segmentation better results would have been obtained. However, as there was not enough time remaining in the project to run the optimiser again this theory was never tested.

Out of the 5222440 ROIs returned by SPN segmentation 738 were matched by at least one voxel to a nodule. This was 70 fewer than version 0.2. Furthermore a reduced mean match score of 0.34319 was obtained. Figure 24 shows a comparison of all of the SPN segmentation techniques used and confirms that version 0.2 was the best. As it was known that the segmentation technique used in this version was inferior to version 0.2 testing of JPN segmentation and classifier were deferred to version 0.5.

²To train the classifier with the same instances multiple times.

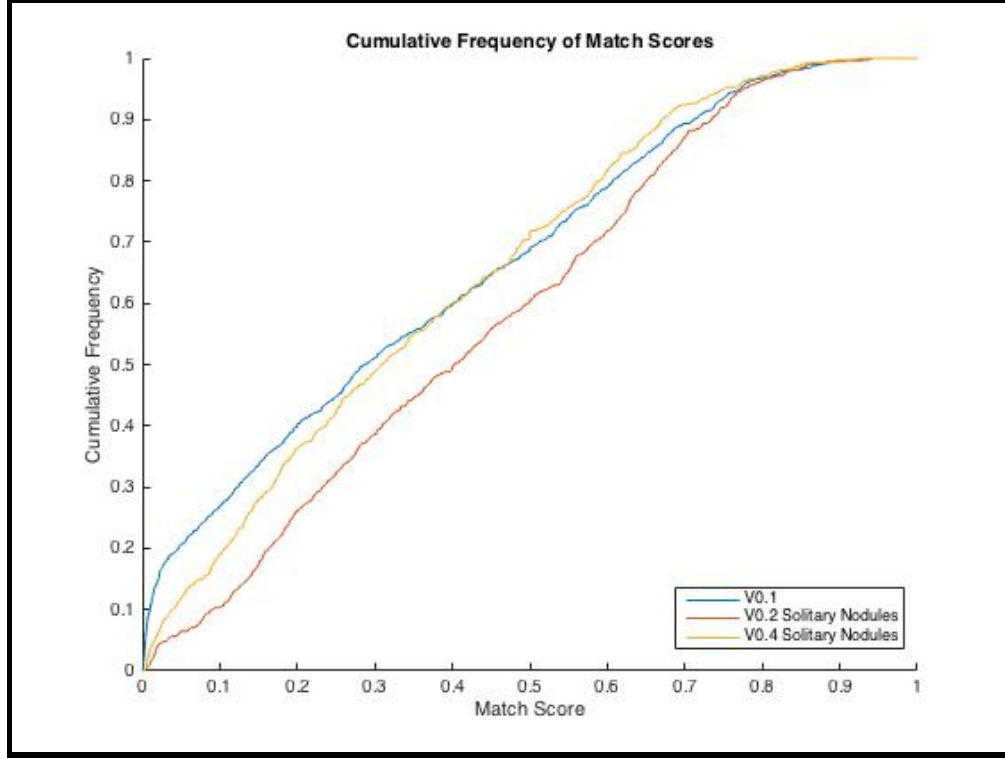


Figure 24: Distribution of match scores frequencies for nodules matched to a ground truth by at least one voxel.

4.5 Version 0.5

The aim of this version was to combine the most successful segmentation techniques developed so that comprehensive testing of the classification stage could take place. As it had been established that the most successful SPN segmentation technique was the one implemented in version 0.2. This technique and its parameters were used in version 0.5. The only change being that the mask described at that start of Section 4.4.1 was used. Both JPN segmentation techniques were tested along side the SPN techniques. As can be seen from Table 8 the erosion based technique used in version 0.2 obtained a better mean match score than the blob detector, however it also matched fewer nodules.

Ideally the classifier experiments conducted in Section 4.5.2 would have used a range of match thresholds. This would allow for observations of how it affected the performance of

JPN segmentation	ROIs Matched	Matched Rate	Total ROIs	Mean match score
Blob Detection	738	0.646	6484656	0.40157
Erosion	698	0.610	8117908	0.43291

Table 8: V0.5 Segmentation Results.

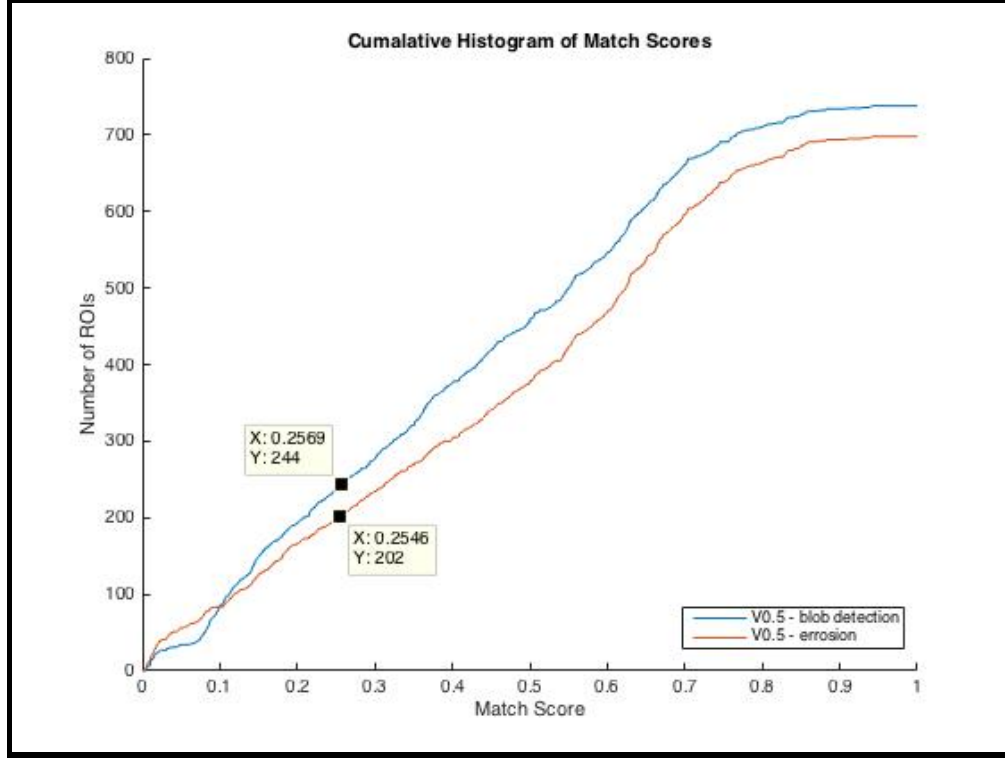


Figure 25: Distribution of match scores for nodules matched to a ground truth by at least one voxel.

the classifier to be made. As there was not enough time to do this, a single match score was chosen to perform the experiments. The match score was chosen by viewing images of many hundreds of matches at different match scores and selecting the minimum match score that the author deemed to be a reasonable match. These images were generated using **MatchExamples**. The minimum value was chosen as it would maximise the number of nodules used for testing and training. Figure 26 shows an example of a ROI with a match score of 0.25, the chosen match threshold for the experiments.

It was initially planned to use the JPN segmentation technique that obtained the largest number of matched ROIs above the match threshold for the classifier experiments. However, by subtracting the Y values in the annotations in Figure 25 from their respective values for total matched ROIs, it could be seen that the blob detection and erosion methods would have 494 and 496 remaining matched nodules. As these values were so close both techniques were used to train and test the classifier so it could be seen which technique yielded the best results when trained with the same slice features.

Table 9 shows the results of training and testing the classifier using the two JPN segmentation techniques. It was elected that the erosion based method should be used for the next phase of testing. This method was selected because although it returned 4 fewer TPs it also returned 65940 fewer FPs. Reducing the number of FPs was important at this stage. This is as if too many were presented to a radiologist using the system the degree of assistance that system

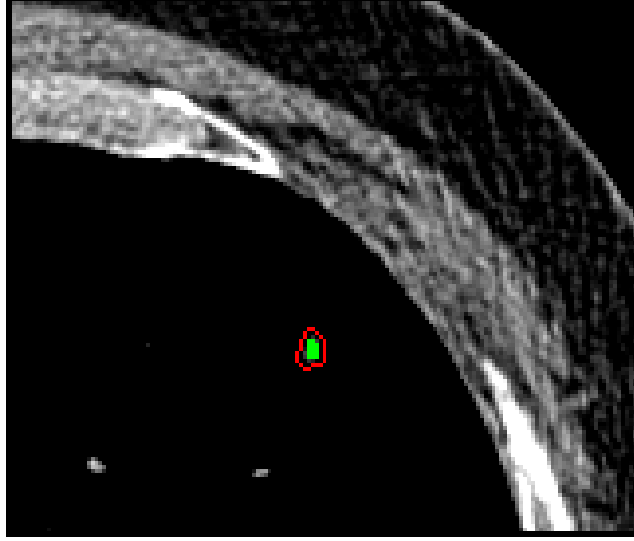


Figure 26: An example match score of 0.25. The ROI is filled green and the nodule outlined in red.

JPN segmentation	TPs	FPs
Blob Detection	103	161733
Erosion	99	95793

Table 9: Result of testing and training the classifier with the two JPN segmentation techniques.

could provide would be vastly diminished. Given more time it would definitely be worth fully optimising the blob detector as this could potentially reduce the number of FPs whilst maintaining the higher TP rate.

4.5.1 Features

This subsection provides information about how the features used to train the classifier were computed and why they were selected. Where equations are used to define how features were computed $P = \{\text{all pixels in an ROI}\}$ and $val(p) = \text{the intensity value for pixel } p$.

Juxtapleural - This feature was a simple boolean flag set to true on all ROIs that were obtained using a JPN segmentation and false on all ROIs that were obtained using a SPN segmentation. The feature was introduced as it was expected that the ROIs obtained using different segmentation methods would exhibit different characteristics. This feature could be used by classifier to easily distinguish between the two techniques used.

Mean Intensity - This feature was chosen as the density of the tissues found in the lungs is variable as such some tissues would have a higher mean intensity than others.

$$\text{MeanIntensity} = \frac{1}{\|P\|} \sum_{p \in P} \text{val}(p)$$

Area - Figure 27 was computed by modelling nodules as circles and using their radii to estimate the area. As can be seen from the figure some area values were more common than others. This is valuable information for the classifier.

$$\text{Area} = \sum_{p \in P} 1$$

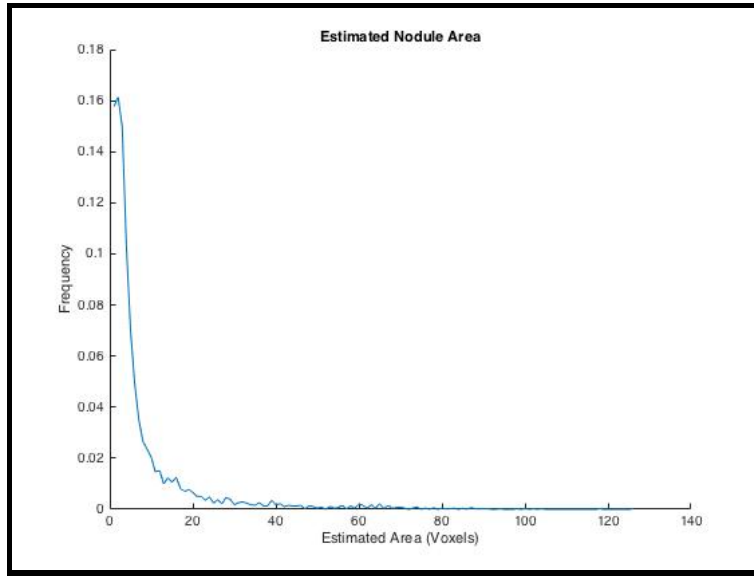


Figure 27: The distribution estimate nodule areas.

Circularity & Hu Circularity - As most nodules are roughly circular and most blood vessels had a tree like structure (when viewed in 3D) scale invariant measures of circularity were an obvious choice for features. **Circularity** was computed dividing the area of ROI by the area of the minimum fitting circle for the ROI. Where as **Hu Circularity** was calculated using moments, as defined in (Rosin et al. 2010). Both features were used as the values returned were effected differently by small imperfections in the circularity of ROIs.

Convexity - This feature was selected in an attempted distinguish the concave ROIs from blood vessels that exhibited a branching structure from the remaining convex ROIs. Values were obtained by dividing the area of an ROI by the area of its convex hull.

Elongation - This feature was selected as most nodules are approximately circular where as some blood vessels exhibited a long elongated shape. As such it was hoped that the feature could be used to distinguish between them. The values where computed as shown

in the equation below. Computing the values in this manner made them rotation and scale invariant.

w = width of minimum bounding box

h = height of minimum bounding box

$$\text{Elongation} = 1 - \frac{\min(w, h)}{\max(w, h)}$$

Coarse Hist & Fine Hist - These features were used to encode the frequency of intensity values found within ROIs. Similarly to **Mean Intensity** these features were hoped to encode the density of the tissue the ROI was composed of. Each bin in the histogram was used as a feature. The number of bins used in each histogram is discussed at the end of this section.

Coarse LTP & Fine LTP - These features were used to attempt to encode the texture of ROIs. This was valuable information for the classifier because as stated in Section 2.1 nodules often exhibit a non-uniform texture. It was hoped this texture would be distinctive and could be used to distinguish them from non-nodules. LTP stands for Local Ternary Pattern. This is a standard technique used to create texture features (Ren, Jiang, and Yuan 2013). However the features used by Lungs implemented what is though to be a novel adaptation of the algorithm in order to encode additional information about the voxels found at the edge of each ROI. The following method shows how the LTP value was computed for each voxel at the center of a 3×3 neighbourhood. The numbers in the black boxes shown in Figure 28 correspond to the values for i used in this method.

$val(i)$ = The intensity values for voxel i of the neighbourhood

$$compare(i) = \begin{cases} 0, val(i) < val(8) \\ 1, val(i) \geq val(8) \\ 2, val(i) = \text{VOID} \end{cases}$$

$$LTP = \sum_{i=0}^7 3^i \times compare(i)$$

Once all the LTP values were obtained they were placed into a frequency histogram and each bin was used as a feature. The number of bins used in each histogram is discussed at the end of this section. Voxel intensities marked as **VOID** are voxels which lie within the 3×3 neighbourhood but are not included in the ROI. It was proposed that this method would produce scale invariant features that encoded aspects of both the shape and texture of the ROI and distinguished the textures found on the edge of the ROI from those found within it. However, further investigation would be required to establish what information is truly encoded using the method.

Perimeter, Fitted Ellipse Angle, Fitted Ellipse Area, Fitted Ellipse Width, Fitted Ellipse Height, Min Circle Radius - These features were very basic features implemented in the hope that the classifier would use them to make inferences. For example a

measure of elongation could be created using Fitted Ellipse Width & Fitted Ellipse Height.

VOID 0	VOID 1	200 2
VOID 7	40 8	70 3
50 6	20 5	4 4

Figure 28: Example LTP neighbourhood for a voxel found at the edge of an ROI.

In order to maximise the amount of information that was conveyed to the classifier using histograms based features, Sturge’s rule was applied in an attempt to optimise the bin sizes used. Sturge’s rule is defined as follows (Legg et al. 2007):

$$\begin{aligned}
 w &= \text{the ideal bin width for the histogram} \\
 r &= \text{the range of values withing the dataset} \\
 n &= \text{the number of elements in the data set}
 \end{aligned}
 \tag{7}$$

$$w = \frac{r}{1 + \log_2(n)}$$

This equation was used to find the bin sizes both for the intensity and LTP histogram features. For the intensity histogram the value for r used was 256 this was as 8bit images were processed by the system and $2^8 = 256$. However, in hind sight this was a poor choice of value as the true range of values was smaller than this, see Figure 16. For the LTP histograms the range of values was calculated as such $r = 3^8$. A problem with using Sturge’s rule for histograms computed for ROIs was that the number of elements (n) varied from one ROI to another. As the histogram bins were used as features there needed to be the same number of bins for each ROI. For this reason two values of where used. For **Coarse Hist** and **Coarse LTP** the mean number of elements for all the ROIs was used, where as for **Fine Hist** and **Fine LTP** the max number of elements was used.

Using Weka’s **InfoGainAttributeEval** tool the entropy associated with each feature was calculated. This indicated that the best features were The LTP and intensity histograms, **Mean Intensity** and **Perimeter**. The table of results for the top 30 features can be seen in Appendix 10.6. This being said all the features were still used to test the classifier in the following section. This is as the combinations of features may have resulted in higher entropy then when calculated individually.

4.5.2 Classifier Testing

In order to attempt to maximise the accuracy of detection multiple classifiers were trained and tested using the same data. Initially all the classifiers were trained using undersampling. The training set used contained 382 nodules and 382 non-nodules and testing set contained 117 nodules and 915710 non-nodules. Table 14 shows the results for each of the classifiers used. The **MultilayerPerceptron** is an implementation of an ANN and **SMO** is an implementation of a SVM. These classifiers were chosen as they were similar to the ones used in (Tan et al. 2011). The remaining classifiers were chosen as they were methods of classification that had been discussed in lectures. All the classifiers tested performed reasonably well with total accuracy ranging from 85.423-90.087. A problem with all the classifiers was that there was a large number of false positives being returned. In order to attempt to combat this the same classifiers were tested using oversampling.

Weka Class	Accuracy %	TPs	TP rate	FPS	FP rate
J48	89.538	99	0.846	95793	0.105
MultilayerPerceptron	87.301	107	0.915	116290	0.127
SMO	90.087	109	0.932	90778	0.099
BayesNet	90.038	107	0.915	115133	0.126
RandomTree	85.423	102	0.872	133484	0.146
RandomForest	86.866	110	0.940	120282	0.131

Table 10: Result of testing and training different Weka classifiers using undersampling.

For the oversampled training 9 instances of each nodule were used to train the classifier. As such, 3438 nodules and 3438 non-nodules were used to train the classifier. The testing set remained the same. As can be seen in Table 14 oversampling did reduce the FP rate however this came at the price of diminished sensitivity. The only exception to this was the **SMO**. This classifier had increased TP and FP rates. This showed that oversampling was particularly ineffective for the **SMO**. The decreases in TP rates when using oversampling was likely to be caused by overtraining. As such better results may have been obtained if the number of times each nodule was sampled was reduced.

Weka Class	Accuracy %	TPs	TP rate	FPS	FP rate
J48	92.070	77	0.658	72585	0.079
MultilayerPerceptron	92.416	94	0.803	69437	0.076
SMO	88.401	102	0.872	106211	0.116
BayesNet	92.690	92	0.786	66922	0.073
RandomTree	92.962	63	0.538	64401	0.070
RandomForest	95.378	75	0.641	42287	0.046

Table 11: Result of testing and training different Weka classifiers when sampling each nodule 9 times.

4.5.3 Evaluation

Out of the classifiers and sampling techniques tested the **SMO** trained using undersampling was the chosen method for the final version of Lungs created during the project. This was as it had the second highest TP rate and the lowest FP rate. The **RandomForest** trained using undersampling had the highest TP rate however it returned an additional 29504 FPs for the training set. The reduction in FPs was chosen over the marginal increase in TPs so that the suggestions made by the system would be of more used to a radiologist using it. The following equation was used to calculate the overall sensitivity for the system:

$$\begin{aligned}
m &= \text{number of ROIs with match score} > 0.25 \\
n &= \text{total nodules in ground truth} \\
t &= \text{classifier TP rate} \\
\text{segmentaion sensativity} &= \frac{m}{n} \\
\text{system sensativity} &= t \times \text{segmentaion sensativity}
\end{aligned} \tag{8}$$

The values for the variables in Equation 8 were: $m = 496$, $n = 1381$ and $t = 0.932$ and as such the segmentation sensitivity was 0.359 and the total system sensitivity was 0.316. On average there were 3782.416 FPs/per stack. These figures only take into account nodules with a diameter $\geq 3\text{mm}$. In order to be able to compare these figures directly to the results shown in Table 4 the **ROIs** would need to be grouped in 3D regions and a consensus would need to be drawn between the **ROIs**. However, there was not time to implement this. From the system sensitivity and FPs/per stack calculated it could be inferred that the system was inferior to those given in the table.

From the tests conducted it appeared that the classifier was performing extremely well with a total accuracy of 90.087%. This indicated that a good choice of features had been made. However, to ensure this value was accurate a larger test set would need to be used. The main area that let the system down was the segmentation stage. This is as the segmentation technique had a poor TP rate and a high number of FPs/stack. This resulted in a low system sensitivity as although the classifier was working well not enough nodules were making it to that stage. Similarly, the huge number of FPs returned from segmentation meant that

even though 90.1% of them were being classified correctly a large number of FPs were being returned by the system.

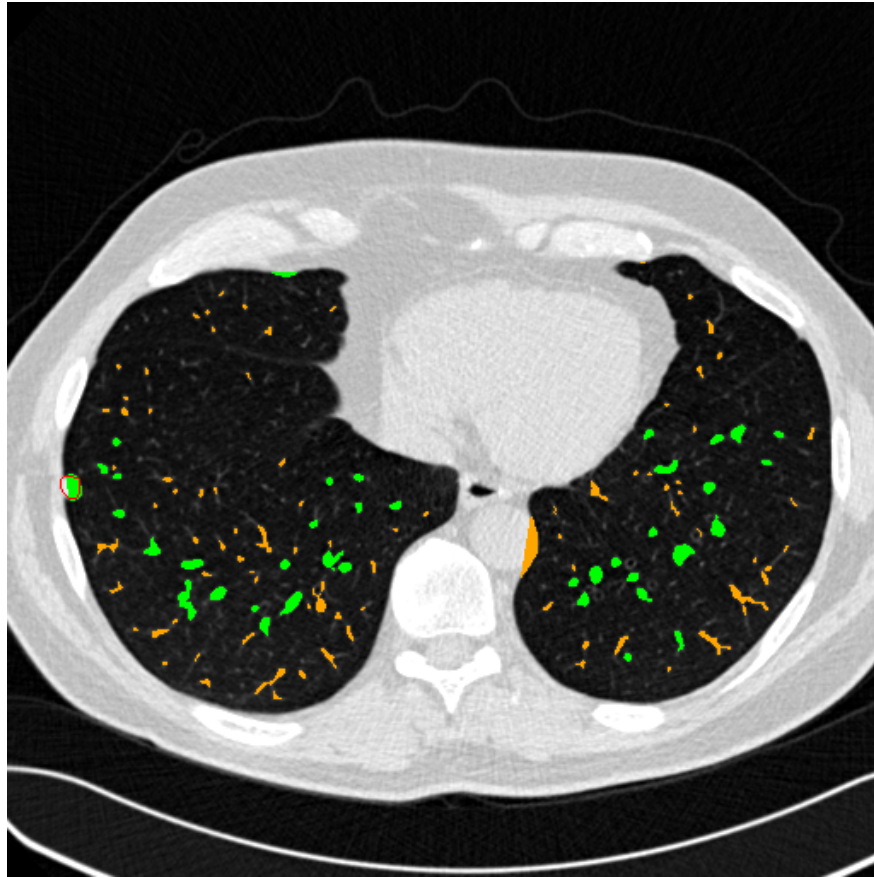


Figure 29: An example slice annotated using the classifier. Green regions indicate ROIs classified as nodules, orange regions indicate ROIs classified as non-nodules and the red contour shows a nodule identified by the ground truth

5 Future Work

Due to the short time frame allowed, at the end of the project there were many areas in which the system could potentially be improved. If more time was allowed one of the first changes made would be to increase the amount of information available to the system when segmenting and computing features. One way of doing this would be by using 16bit images, as stated in Section 4.3 this would be particularly useful for segmentation. The best way to implement this change would probably be to use a different library that had full support for 16bit images. Ideally this library would still be implemented in C/C++ to maintain performance. Another way of increasing the amount of information available would be to consider the 3rd dimension when segmenting and computing features. This would be particularly useful for helping the system distinguish between nodules and blood vessels that are extending parallel to the viewing vector, see Section 2.1. Furthermore, using 3D ROIs would reduce the number of non-nodules that needed to be identified by the classifier. This would likely be easier than reducing the number of FPs returned by the classifier than attempting to lower the FP rate further.

As stated in Section 4.4.3 2D SPN segmentation would likely be greatly improved by inverting the slices before the Watershed algorithm was used on them. Furthermore the blob detector JPN segmentation showed promise when tested and would likely out perform the erosion method if fully optimised. In order to attempt to try to reduce the number of FPs returned from segmentation an additional optimiser was created `SegOpt2`. This class takes the maximum nodule inclusion as an argument and tries to reduce the number of ROIs returned from segmentation while maintaining nodule inclusion. This class was never run against the full data set however it showed promise when tested on a small subset. As such it should be used to optimise the parameters in the next iteration of the project.

One way in which the training of the classifier could potentially be improved would be to synthesise additional nodules. This could be achieved through applying transformations to the existing nodules detected. Similarly to oversampling the nodules this would largely be of benefit as it would mean that more non-nodules could be used for training whilst maintaining the balance of nodules and non-nodules used. The advantage that this technique would have over oversampling is that it would reduce the likely hood of overfitting when training using large datasets. In all cases where undersampling was used the first n non-nodules returned from the database to train the classifier. This method is a naive approach as it is liable to introduce a sampling bias. To combat this a simple solution would be to randomise the order in which the sampled where returned from the database. An alternative solution would be to cluster the instances in sample space and take an even number of samples from each cluster. This method was used by Tan et al. (2011) and is a superior solution to the problem as if done correctly it would guarantee that the classifier was exposed to all the variation present in the instances. A way in which the testing of the classifier could be improved would be to use cross validation. The methods works by selecting some subset of the instances for testing and using the remained for training. Once training and testing is completed, the process is repeated using a different subset of the instances for testing. Using this method can help to eliminate a sample bias and also makes it possible to train the classifier using more data

while still testing it thoroughly.

Another improvement that could be made to the system would be to utilise more of the LIDC database. One way of doing this would be to take into consideration nodules with a diameter less than 3mm. These nodules were ignored through the project in the interest of simplicity however they make up 0.242 of the nodules in the LIDC database. Furthermore, not only are these nodules likely to be the hardest for a radiologist to detect visually their detection holds the most value. This is as small nodules are an indicator of cancer in the very early stages. As stated in Section 4.1.5, another way the LIDC ground truth could be further utilised would be to cluster the readings taken by the four radiologists and only used those annotations where more than one radiologists agree. This would improve confidence in the ground truth and lead to better training of the classifier. The final and most obvious way in which the database could be further utilised would be to train the classifier using more slices. This would be particularly useful as more instances of nodules and non-nodules could be used to train the classifier without having to use oversampling or synthesis. Incorporating more data would no doubt require the images to be normalised. However, even without the additional data this could be worth while as it may reduce the variance in nodule intensities, see Section 4.1.1. Another issue with including more data is that it would increase computational complexity. In order to combat this it would be wise to use distributed computing techniques.

6 Conclusions

At the end of the project the CAD system implemented was not competitive with the existing CAD systems described in Table 4. The main downfall of the system was the segmentation stage which had a sensitivity of just 35.900%. The SNP segmentation technique used in the final version was the one implemented in version 0.2, see Section 4.2.1. This technique was chosen to maximise the result obtained at the end of the project however there were known issues with it which if resolved would have likely improved results greatly, see Section 4.4.3. The JPN segmentation technique chosen for the final version was also the one implemented in version 0.2. This choice again was to maximise results by the end of the project. However, it was predicted that if the technique implemented in 4.4.1 had been fully optimised it would have out performed the method selected. Although the segmentation stage of the system performed quite poorly the classification stage yielded impressive results. The classifier had a total accuracy of 90.087%. This indicated that appropriate features had been selected. Of particular interest was the novel LTP feature described in Section 4.5.1. Initial testing showed this feature to be the most effective out of the features used to classify the pulmonary nodules. Within the time frame allowed for the project it would have been difficult to obtain better results than those delivered by the system implemented. This was due to the large amounts of data that needed to be processed. Merely creating the multi-threaded framework to process all this data was a complex task in it's self and took up a large portion of the development time. Furthermore the relatively small amount of computing power available for the project meant optimisation of parameters took a very long time. This restricted how well the system could be optimised as a small population had to be used for the GAs implemented and there was not always enough time to run the optimisation tasks. This had the greatest effect in that there was not enough time to re-optimize the parameters for the Watershed algorithm using inverted slices and as such this method was not tested. It was predicted that using this method would greatly improved the sensitivity of segmentation. A further indicator that the project was a success was that vast majority of the requirements defined in Section 3.1 were met, see Table 12. Overall it was the authors opinion that the project was a success. This is as in the short time allowed great steps had been made toward the development of a highly complex system which set out to accomplish a difficult task.

The system should be capable of:	Was it achieved?
importing and querying meta-data provided by CT scans.	The meta-data was successfully imported and index using <code>CTSliceImporter</code> and was queried by many parts of the system.
segmenting ROIs in CT slices.	Methods were developed for segmenting the <code>CTSlices</code> they performed poorly.
comparing ROIs to the ground truth.	This was achieved using <code>Matcher</code> which implemented the method of comparison shown in Equation 5.
training a classifier using features computed for ROIs identified.	This was successfully implemented and performed very well.
using a trained classifier to identify PNs in previously unseen CT scans or state that there are no PNs.	This was implemented in <code>Lungs</code> however the accuracy of the predictions made by the classifier was hindered by the segmentation technique.
highlighting PNs in a suitable GUI so that it can be used as an aid for radiologists.	This was successfully implemented in <code>Lungs</code> , see Figure 29. The orange regions have been added to show correctly classified non-nodules.
having additional features added to the classifier easily	Adding additional features was trivial. To add a new feature a single method of the <code>Feature</code> interface needed to be implemented by the class for the new feature. After that all that was required was for a new instance of this class to be added to the list in <code>FeatureEngine</code> .
storing the results of features calculated for the training set so that they do not need to be recomputed each time the classifier is trained.	This was implemented in <code>FeatureEngine</code> .
unit testing functionality where possible and appropriate.	A total of 24 unit test were created. These tests mostly had in excess of 80% line coverage and were used to test most of functionality of the system where the desired results were easily determinable. However to improve confidence in the system it would be wise to develop unit tests for the remaining functionality.
evaluating the effectiveness of the classifier.	The classifier was successfully tested and statistics were collected to evaluate it's effectiveness.
easily configuring parameters the user may wish to change from one instance of the system to another.	The configuration file <code>application.conf</code> allows the system to configure all such parameters.
only using a small subset of images during development. This will help to speed up development as code can be tested on just a few images before being running again with the full dataset.	This was achieved using <code>application.conf</code> and <code>DataFilter</code> .

Table 12: Analysis of how well requirements were met.

optimising values for parameters which are not easily determinable using appropriate algorithms.	Optimisation algorithms used however there was not time to run all of them against the full training set. The decision to use a GA may have been a poor choice in hindsight as it was a very computationally complex approach.
be capable of processing 1000s of images in a reasonable amount of time. This should be achieved through the use of low computational complexity algorithms and multi-threading.	The final version was capable of processing 227625 images, computing features training the classifier and testing it in approximately 6 hours.

Table 13: Analysis of how well requirements were met continued.

7 Reflection on Learning

The greatest lesson learnt from undertaking this project was the importance of writing the report as you go along. This was not the approach I took throughout the project electing instead to only taking notes during implementation and writing the report after. Whilst writing the report I felt my understanding of the problem at hand deepened enormously. This is as putting my thoughts in to writing helped me to organise and formalise them. Another major error in the manner the project was conducted was not enough time was spent acquiring a deep understanding of how the OpenCV implementation of the watershed algorithm was implemented. Had the source code been examined earlier there would have been time to test using it with an inverted image. Another lesson learnt was the importance of collecting data throughout the project. Although data was collected during implementation to assist decision making it was not stored and organised in way which meant it could be used in the report. Instead it was all collected at the end using Git to return to the previous versions. To run all the code required to collect the results took approximately two weeks with code running almost continuously. This resulted in a great deal of stress caused by the worry there would not be time to run it all which could have been avoided. These lessons were learnt the hard way however they will be invaluable for projects such as the research project I am undertaking with the university this summer.

On a different note a great deal was learnt about cancer and medical imagery whilst performing the background research. I had no previous experience working in the area and the project was a great insight into how computer vision can be applied in the field of medicine. Furthermore this is a field of work that I am highly interested in pursuing and the experience gained during this project will no doubt be invaluable.

Another skill I developed during the project was writing reports in \LaTeX . I only began using latex at the beginning of the project as I thought that it would be a useful tool for writing this report. Throughout the project I learnt how to make use of many additional packages and create references with great ease. For any future report of this scale I would definitely opt to use \LaTeX again as it has been an invaluable tool.

8 Glossary

- Pulmonary Nodule - A small somewhat round growth on the lungs.
- Slice - A single cross-section obtained using a CT scanner.
- Stack - A collection of slices obtained using a CT scanner.
- Lungs - The name given to the CAD system developed during the project.

9 Abbreviations

- CT - Computed Tomography
- PN - Pulmonary Nodule
- CAD - Computer Aided Diagnosis
- TP - True Positive
- FP - False Positive
- TN - True Negative
- FN - False Negative
- SPN - Solitary Pulmonary Nodule
- JPN - Juxtapleural Pulmonary Nodule
- ROI - Region Of Interest
- MSE - Mean Square Error
- PSNR - Peak Signal-to-Noise Ratio
- LIDC - Lung Image Database Consortium
- DNG - Divergence of Normalised Gradient
- DBMS - Database Management System
- IDE - Integrated Development Environment
- SVM - Support Vector Machine
- ANN - Artificial Neural Network
- MVP - Minimum Viable Product
- VPS - Virtual Private Server

- GA - Genetic Algorithm
- DOG - Difference of Gaussian
- LTP - Local Ternary Pattern

10 Appendices

10.1 Appendix 1 - V0.1 Segmentation

Listing 4: Source Tree

```
src
|-- config
|   |-- Annotation.java
|   |-- BlobOptimisation.java
|   |-- Misc.java
|   |-- Mode.java
|   |-- SamplingMethod.java
|   |-- SegOptimisation.java
|   `-- Segmentation.java
|-- core
|   |-- Lungs.java
|   `-- Pipeline.java
|-- data
|   |-- CTSliceImporter.java
|   |-- CTStackGenerator.java
|   |-- DataPipeline.java
|   |-- GroundTruthImporter.java
|   `-- Importer.java
|-- discover
|   |-- CsvWriter.java
|   |-- HistogramWriter.java
|   |-- MatchExamples.java
|   |-- MatchScores.java
|   |-- MissedNodules.java
|   |-- NoduleHistograms.java
|   |-- NoduleRadii.java
|   |-- ROIClassStats.java
|   `-- SliceHistograms.java
|-- ml
|   |-- ArffGenerator.java
|   |-- FeatureEngine.java
|   |-- InstancesBuilder.java
|   `-- MLPipeline.java
```

```

|-- ROIClassifier.java
|-- ROIGenerator.java
|-- TrainAndTest.java
`-- feature
    |-- AllHists.java
    |-- Area.java
    |-- BoundingBox.java
    |-- Circularity.java
    |-- Convexity.java
    |-- Feature.java
    |-- FitEllipse.java
    |-- HuCircularity.java
    |-- LTP.java
    |-- MeanIntensity.java
    |-- MinCircle.java
    `-- Perimeter.java
|-- model
    |-- CTSlice.java
    |-- CTStack.java
    |-- Circle.java
    |-- DOGPyramid.java
    |-- GroundTruth.java
    |-- Histogram.java
    |-- KeyPoint.java
    |-- MinMax.java
    |-- MinMaxXY.java
    |-- ObjectIdResult.java
    |-- ROI.java
    |-- ROIAreaStats.java
    |-- SigmaMat.java
    |-- StringResult.java
    `-- lidc
        |-- BlindedReadNodule.java
        |-- Characteristics.java
        |-- EdgeMap.java
        |-- LidcReadMessage.java
        |-- LobarLocation.java
        |-- Locus.java
        |-- NonNodule.java
        |-- ObjectFactory.java
        |-- ReadingSession.java
        |-- ResponseHeader.java
        |-- Roi.java
        `-- UnblindedReadNodule.java
|-- optimise

```

```

|-- BlobOpt.java
|-- LungsOptHelper.java
|-- Optimiser.java
|-- SegOpt1.java / SegmentationOptimier.java
`-- SegOpt2.java
|-- util
|-- CircularList.java
|-- ColourBGR.java
|-- ConfigHelper.java
|-- Counter.java
|-- DataFilter.java
|-- FutureMonitor.java
|-- LungsException.java
|-- MatUtils.java
|-- MatViewer.java
|-- MongoHelper.java
|-- MultiMap.java
|-- PointUtils.java
`-- TimeUtils.java
`-- vision
|-- BilateralFilter.java
|-- BlobDetector.java
|-- BlobToROI.java
|-- ConvexHull.java
|-- Matcher.java
|-- ROIE extractor.java
`-- Sobel.java

```

10.2 Appendix 2

V0.1 Solitary Nodule Segmentation



Figure 30: An unprocessed slice.

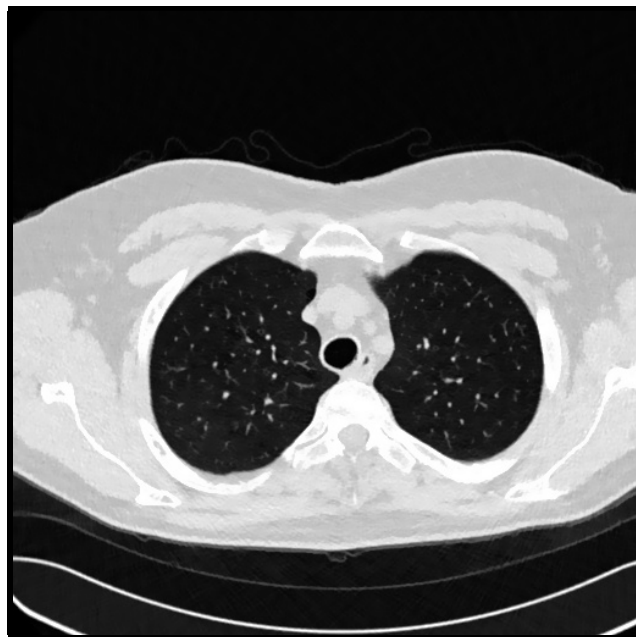


Figure 31: The results of filtering the unprocessed slice. The parameters chosen for this filter were not the optimum ones but instead were selected to exaggerate the effect of the filter.

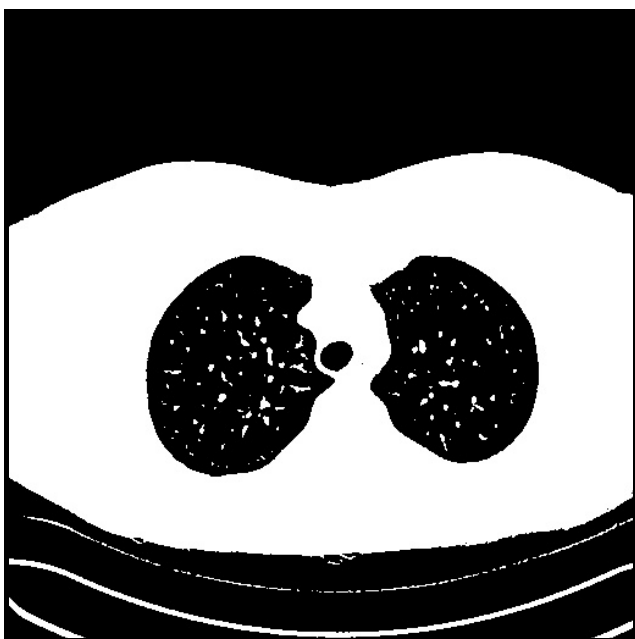


Figure 32: The result of thresholding the filtered slice.

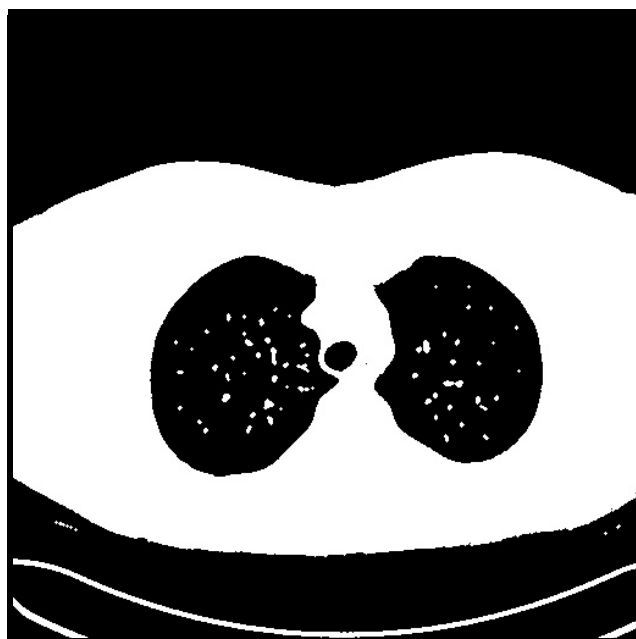


Figure 33: The result of applying opening to the thresholded slice.

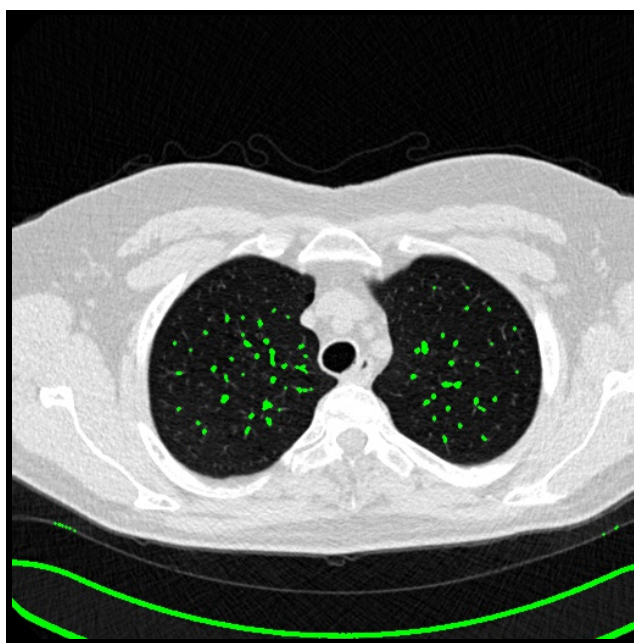


Figure 34: The result of extracting the connected components from the opened slice and rejecting the one with the largest area.

10.3 Appendix 3

Juxtapleural Segmentation Using Erosion



Figure 35: The largest ROI returned from by the solitary nodule segmentation stage. A juxtaleural nodule is highlighted by the red square.

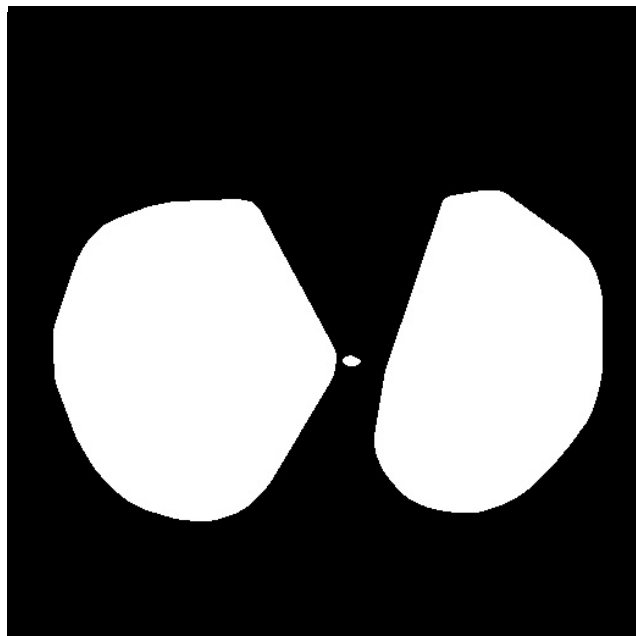


Figure 36: The convex hulls of the plural cavities, and a false positive (the smallest region).

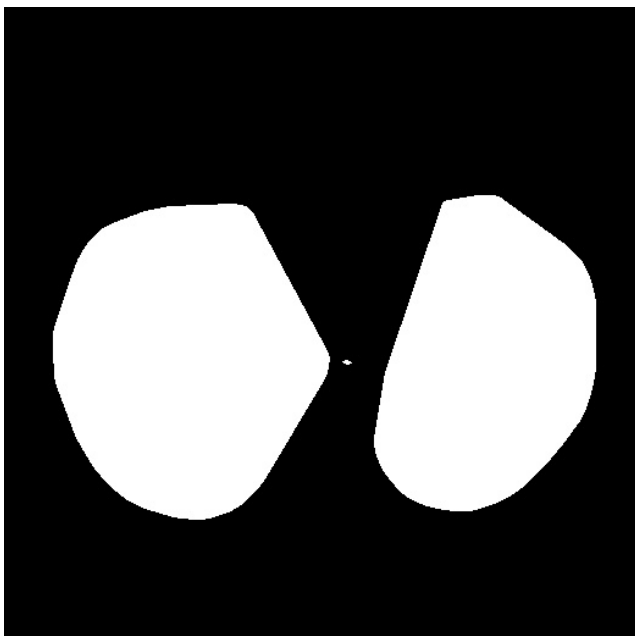


Figure 37: The eroded hulls.

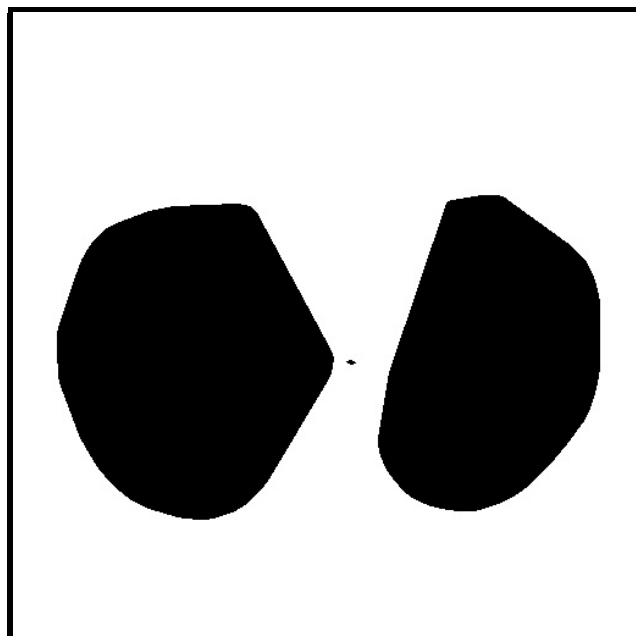


Figure 38: The mask created by inverting the eroded plural hulls.



Figure 39: The results of subtracting the mask from the largest ROI. A juxtapleural nodule is highlighted by the red square

10.4 Appendix 4

Listing 5: Subset statistics aggregation

```
db.CTSlice.aggregate(  
  [  
    {  
      "$group": {  
        "_id": "$model",  
        "count": {  
          "$sum": 1  
        },  
        "highBit": {  
          "$addToSet": "$highBit"  
        }  
      },  
    },  
    {  
      "$sort": {  
        "count": 1  
      }  
    }  
  ]  
)
```


10.5 Appendix 5

Solitary Nodule Masking

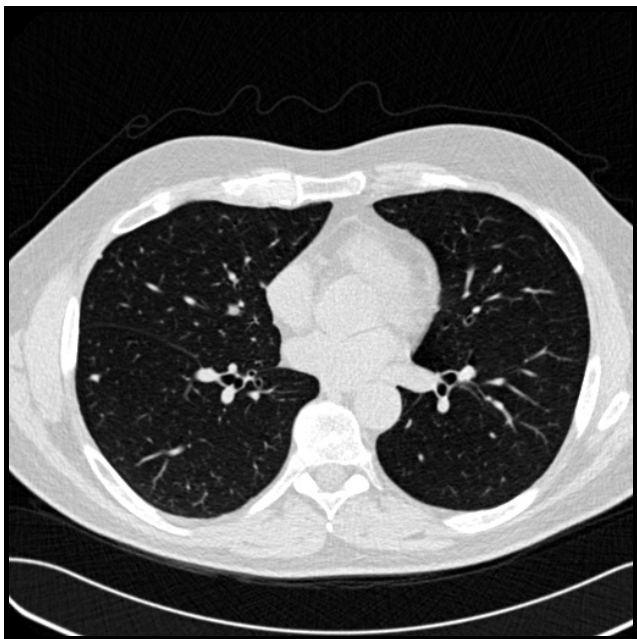


Figure 40: The original slice.

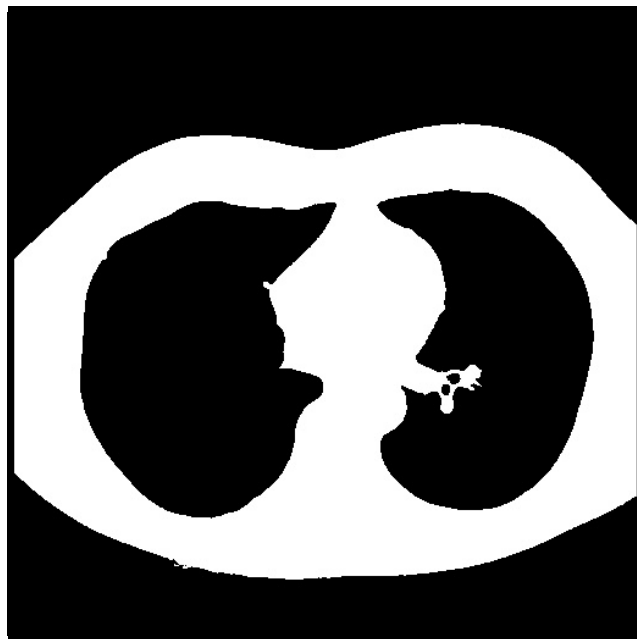


Figure 41: The largest ROI returned from solitary nodule segmentation

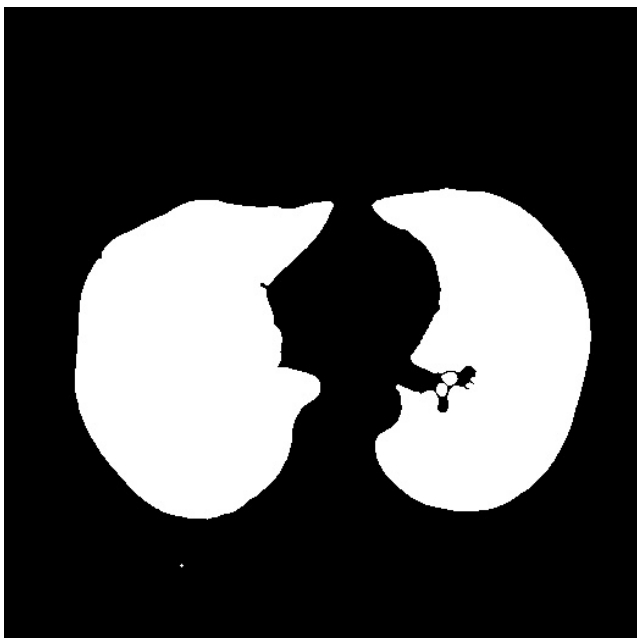


Figure 42: The internal cavities of the largest ROI

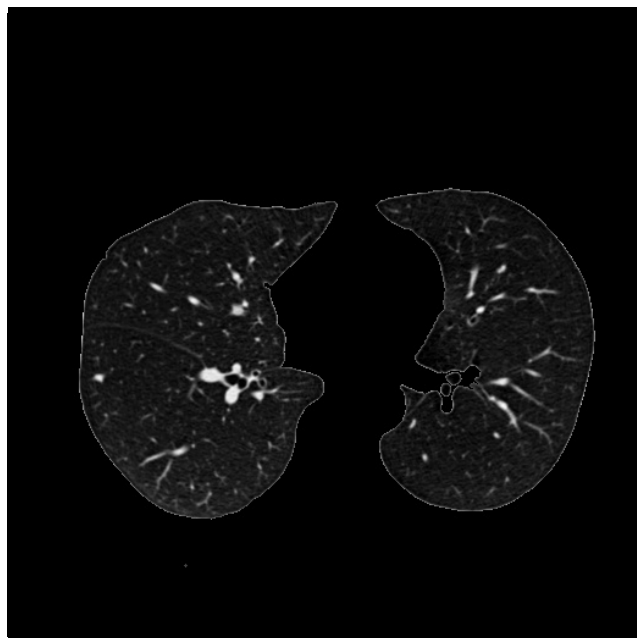


Figure 43: The results of applying the mask to the original image

10.6 Appendix 6

Feature Information Gain

N.B numbers after than name of histogram features indicate the index of the bin used for that feature.

Information Gain	Feature
0.5983	36 Coarse LTP 4
0.5666	3 Area
0.5179	49 Fine LTP 12
0.4887	48 Fine LTP 11
0.4868	2 Mean Intensity
0.4856	15 Coarse Hist 1
0.4729	4 Perimeter
0.4677	5 Min Circle Radius
0.4516	22 Fine Hist 3
0.4221	44 Fine LTP 7
0.4123	11 Fitted Ellipse Width
0.4107	47 Fine LTP 10
0.4019	45 Fine LTP 8
0.3964	40 Fine LTP 3
0.3913	10 Fitted Ellipse Area
0.3893	17 Coarse Hist 3
0.3726	35 Coarse LTP 3
0.3696	26 Fine Hist 7
0.3582	27 Fine Hist 8
0.3489	42 Fine LTP 5
0.3445	43 Fine LTP 6
0.3416	25 Fine Hist 6
0.336	32 Coarse LTP 0
0.3341	12 Fitted Ellipse Height
0.3247	37 Fine LTP 0
0.3187	46 Fine LTP 9
0.3056	28 Fine Hist 9
0.2986	34 Coarse LTP 2
0.2815	39 Fine LTP 2

Table 14: Result of testing and training different Weka classifiers using undersampling.

References

- American Cancer Society (2010). *THE GLOBAL ECONOMIC COST OF CANCER*. URL: http://phrma-docs.phrma.org/sites/default/files/pdf/08-17-2010_economic_impact_study.pdf (visited on 04/13/2017).
- (2015). *Lymph Nodes and Cancer*. URL: <https://www.cancer.org/cancer/cancer-basics/lymph-nodes-and-cancer.html> (visited on 04/13/2017).
- Apache (2017a). *Apache Ant - Welcome*. URL: <http://ant.apache.org/> (visited on 04/28/2017).
- (2017b). *Commons Configuration - Java Configuration API*. URL: <http://commons.apache.org/proper/commons-configuration/> (visited on 04/28/2017).
 - (2017c). *Home | Apache Ivy™*. URL: <http://ant.apache.org/ivy/> (visited on 04/28/2017).
- Armato et al. (2011). “The lung image database consortium (LIDC) and image database resource initiative (IDRI): a completed reference database of lung nodules on CT scans”. In: *Medical physics* 38.2, pp. 915–931.
- Bitbucket (2017). *Bitbucket | The Git solution for professional teams*. URL: <https://bitbucket.org/product> (visited on 04/25/2017).
- Cancer Research UK (2014a). *Lung cancer incidence statistics*. URL: <http://www.cancerresearchuk.org/health-professional/cancer-statistics/statistics-by-cancer-type/lung-cancer> (visited on 04/13/2017).
- (2014b). *Lung cancer survival statistics*. URL: <http://www.cancerresearchuk.org/health-professional/cancer-statistics/statistics-by-cancer-type/lung-cancer/survival> (visited on 04/13/2017).
- Columbia University Press (2000). *The Columbia Encyclopedia*. 6th ed. 0-7876-5015-3. Columbia University Press.
- Demir, Cigdem and Bülent Yener (2005). “Automated cancer diagnosis based on histopathological images: a systematic survey”. In: *Rensselaer Polytechnic Institute, Tech. Rep.*
- DICOM (2017). *Introduction and Overview*. URL: <http://dicom.nema.org/medical/dicom/current/output/pdf/part01.pdf> (visited on 04/26/2017).
- Dougherty, Geoff (2009). *Digital Image Processing for Medial Applications*. 1st ed. 978-0-521-86085-7. Cambridge.
- ELCAP (2003). *Public Lung Image Database*. URL: <http://www.via.cornell.edu/lungdb.html> (visited on 04/26/2017).
- Firmino et al. (2014). “Computer-aided detection system for lung cancer in computed tomography scans: Review and future prospects”. In: *BioMedical Engineering OnLine* 13.1, p. 41. DOI: 10.1186/1475-925X-13-41. URL: <http://dx.doi.org/10.1186/1475-925X-13-41>.
- Gibbs, Keith (2013). *Computerised axial tomography (CAT or CT)*. URL: http://www.schoolphysics.co.uk/age16-19/Medical%20physics/text/CT_scanning/index.html (visited on 04/17/2017).
- Git (2017). *Git*. URL: <https://git-scm.com/> (visited on 04/25/2017).
- Google Scholar (2017). *The lung image database consortium (LIDC) and image database resource initiative (IDRI): a completed reference database of lung nodules on CT scans - Google Scholar*. URL: <https://scholar.google.co.uk/scholar?q=The+lung+image+database+consortium+%28LIDC%29+and+image+database+resource+initiative+>

- %28IDRI%29%3A+a+completed+reference+database+of+lung+nodules+on+CT+scans&btnG=&hl=en&as_sdt=0%2C5 (visited on 04/26/2017).
- ImageJ (2017). *DICOM (ImageJ API)*. URL: <https://imagej.nih.gov/ij/developer/api/index.html> (visited on 04/27/2017).
- Incisive Health (2014). *Saving lives, averting costs*. URL: http://www.cancerresearchuk.org/sites/default/files/saving_lives_averting_costs.pdf (visited on 04/13/2017).
- JAXB (2017). *JAXB Reference Implementation — Project Kenai*. URL: <https://jaxb.java.net/> (visited on 04/27/2017).
- Jenetics (2017). *Jenetics: Java Genetic Algorithm Library*. URL: <http://jenetics.io/> (visited on 04/29/2017).
- Jirapatnakul, Artit et al. (2011). “Segmentation of Juxtapleural Pulmonary Nodules Using a Robust Surface Estimate”. In: *International Journal of Biomedical Imaging* 2011. DOI: 10.1155/2011/632195.
- Legg et al. (2007). “Improving accuracy and efficiency of registration by mutual information using Sturges’ histogram rule”. In: *Proc. Med. Image Understand. Anal.*, pp. 26–30.
- Li et al. (2003). “Selective enhancement filters for nodules, vessels, and airway walls in two- and three-dimensional CT scans”. In: *Medical Physics* 30, pp. 2040–2051.
- Lowe, David G. (2004). *Distinctive Image Features from Scale-Invariant Keypoints*. URL: <https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf> (visited on 04/24/2017).
- Lung Cancer Alliance (2017). *Overview - Lung Cancer Alliance*. URL: <http://lungcanceralliance.org/get-information/the-basics-about-lung-cancer.html> (visited on 04/16/2017).
- MathWorks (2017a). *Computer Vision System Toolbox - MATLAB & Simulink*. URL: <https://www.mathworks.com/products/computer-vision.html> (visited on 04/25/2017).
- (2017b). *Image Processing Toolbox - MATLAB - MATLAB*. URL: <https://uk.mathworks.com/products/image.html> (visited on 04/25/2017).
- Maven (2017). *Maven Repository: Search/Browse/Explore*. URL: <https://mvnrepository.com/> (visited on 04/28/2017).
- MongoDB (2017a). *MongoDB for GIANT Ideas | MongoDB*. URL: <https://www.mongodb.com/> (visited on 04/26/2017).
- (2017b). *Java MongoDB Driver*. URL: <https://docs.mongodb.com/ecosystem/drivers/java/> (visited on 04/26/2017).
- Morphia (2017). *Morphia*. URL: <http://mongodb.github.io/morphia/> (visited on 04/26/2017).
- National Cancer Research Institute (2012). *Lung cancer UK price tag eclipses the cost of any other cancer*. URL: <http://www.cancerresearchuk.org/about-us/cancer-news/press-release/2012-11-07-lung-cancer-uk-price-tag-eclipses-the-cost-of-any-other-cancer> (visited on 04/13/2017).
- OpenCV (2015). *OpenCV: Image Segmentation with Watershed Algorithm*. URL: http://docs.opencv.org/3.1.0/d3/db4/tutorial_py_watershed.html (visited on 04/29/2017).
- (2017). *OpenCV library*. URL: <http://opencv.org/> (visited on 04/25/2017).
- OpenStack (2017). *OpenStack Docs: Ocata*. URL: <https://docs.openstack.org/> (visited on 04/28/2017).
- Orozco et al. (2012). “Lung nodule classification in frequency domain using support vector machines”. In: *Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference on*. IEEE, pp. 870–875.

- PixelMed (2016). *PixelMed Publishing™ Java DICOM Toolkit*. URL: <http://www.pixelmed.com/dicomtoolkit.html> (visited on 05/01/2017).
- Ren, Jianfeng, Xudong Jiang, and Junsong Yuan (2013). “Relaxed local ternary pattern for face recognition.” In: *ICIP*, pp. 3680–3684.
- Rosin et al. (2010). “A Hu moment invariant as a shape circularity measure”. In: *Pattern Recognition* 43.1, pp. 47–57.
- Rosin, Paul (2017). “Computer Vision”. Lecutue slides available at https://learningcentral.cf.ac.uk/bbcswebdav/pid-4032303-dt-content-rid-6600869_2/courses/1617-CM3113/CM3113-slides.pdf.
- Sahiner, Berkman et al. (2009). “Effect of CAD on Radiologists’ Detection of Lung Nodules on Thoracic CT Scans: Analysis of an Observer Performance Study by Nodule Size”. In: *Academic Radiology* 16.12, pp. 1518–1530.
- Salden, Alfons H, LMJ Florack, and BM ter Haar Romeny (1991). “Differential geometric description of 3D scalar images”. In: *3D Computer Vision, Utrecht, The Netherlands, Technical Report*, pp. 91–23.
- Tan et al. (2009). “Automated feature selection in neuroevolution”. In: *Evolutionary Intelligence* 1.4, pp. 271–292. DOI: 10.1007/s12065-009-0018-z. URL: <http://dx.doi.org/10.1007/s12065-009-0018-z>.
- (2011). “A novel computer-aided lung nodule detection system for CT images”. In: *Medical Physics* 38.
- The Cancer Imaging Archive (2014). *LIDC-IDRI - The Cancer Imaging Archive (TCIA) Public Access - Cancer Imaging Archive Wiki*. URL: <https://wiki.cancerimagingarchive.net/display/Public/LIDC+IDRI#62619512ca5a48cea7103b0e9fbce3fd> (visited on 04/27/2017).
- Toennies, Klaus (2012). *Guide To Medical Image Analysis*. 1st ed. 978-1-4471-2750-5. Springer.
- Tomasi, C. (1998). “Bilateral Filtering for Gray and Color Images”. In: *IEEE International Conference on Computer Vision*.
- University of Rochester (2017). *Pulmonary Nodules*. URL: <https://www.urmc.rochester.edu/encyclopedia/content.aspx?contenttypeid=22&contentid=pulmonarynodules> (visited on 04/14/2017).
- University of Waikato (2017). *Weka 3 - Data Mining with Open Source Machine Learning Software in Java*. URL: <http://www.cs.waikato.ac.nz/ml/weka/> (visited on 04/29/2017).
- Vijaya, G. and A. Suhasini (2014). “An Adaptive Preprocessing of Lung CT Images with Various Filters for Better Enhancement”. In: *Academic Journal of Cancer Research* 7, pp. 179–184. DOI: 10.5829/idosi.ajcr.2014.7.3.84231.
- Webster, J (1988). *Encyclopedia of Medical Devices and Instrumentation*. 1st ed. Wiley.
- Xu, Chunhua et al. (2017). “Early Diagnosis of Solitary Pulmonary Nodules.” In: *Journal of Thoracic Disease* 5.6, pp. 830–840.