Development of a 3D Mouse Atlas Tool for Improved Non-Invasive Imaging of Orthotopic Mouse Models of Pancreatic Cancer

Joseph Dalton Brook M.Sc. B.Sc. (Hons)

Primary Supervisor: Dr Jane Sosabowski

Secondary Supervisor: Dr Kairbaan Hodivala-Dilke

Invicro Supervisor: Dr Jacob Hesterman

Submitted in partial fulfilment of the requirements of the Degree of Doctor of Philosophy

Statement of Originality

I, Joseph Dalton Brook, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

Signature:

Date:26/09/2019

Details of collaboration and publications:

Jacob Hesterman is a Founding Partner & Head of R&D at Invicro LLC (Boston, USA).

Specific Learning Differences Cover Note

Student ID: 160002821

Guidelines for markers assessing coursework and examinations of students diagnosed with Specific Learning Differences (SpLDs).

As far as the learning outcomes for the module allow, examiners are asked to mark exam/essay scripts sympathetically, ignoring the types of errors that students with SpLDs make and to focus on content and the student's understanding of the subject.

SpLDs may affect student performance in written work in the following ways:

- Spelling, grammar and punctuation may be less accurate than expected
- Organisation of ideas may be confused, affecting the overall structure of written work, despite inlcluding appropriate and interesting content
- Proof reading may be weak with some errors undetected, particularly homophones and homonyms which can avoid spell checkers

Under examination conditions, these difficulties are likely to be exacerbated, particularly towards the end of scripts.

When marking, please ensure the following guidance is adhered to:

- Scan the text for content to gain an overview of the work, to avoid making a judgement based on the structure alone or any minor spelling and grammar errors
- Do not comment solely on spelling and grammar issues, SpLD students are usually aware of this already. If you feel this is a major problem which has affected the sense of the writing, then please encourage the student to use their study skills support from DDS to help with this.

- Include positive and constructive comments amongst the feedback so that students can acknowledge their strengths but work with specialist study skills tutors on developing new strategies for areas of difficulty.
- Use clear English providing specific examples of what is right or wrong, to ensure that the student knows what to improve upon for future work

Colleagues in Schools and Institutes are asked to encourage students with specific learning differences to access the support provided by the Disability and Dyslexia Service if they have any concerns about a students work.

For more information regarding marking guidelines see the Institutional Marking Practices for Dyslexic Students on the DDS webpage.

Disability and Dyslexia Service: http://www.dds.qmul.ac.uk/

Disability and Dyslexia Service

Room 3.06, Francis Bancroft Building

Queen Mary University of London

Mile End Road

London

E1 4NS

T: +44 (0) 20 7882 2756

F: +44 (0) 20 7882 5223

E: dds@qmul.ac.uk

W: www.dds.qmul.ac.uk

Abstract

Pancreatic cancer is the 10th most common cancer in the UK with 10,000 people a year being diagnosed. This form of cancer also has one of the lowest survival rates, with only 5% of patient surviving for 5 years (1). There has not been significant progress in the treatment of pancreatic cancer for the last 30 years (1). Recognition of this historic lack of progress has led to an increase in research effort and funding aimed at developing novel treatments for pancreatic cancer. This in turn has had an inflationary effect on the numbers of animals being used to study the effects of these treatments. Genetically engineered mouse models (GEMMs) are currently thought to be most appropriate for these types of studies as the manner in which the mice develop pancreatic tumours is much closer to that seen in the clinic. One such GEMM is the K-rasLSL.G12D/+;p53R172H/+;PdxCre (KPC) model (2) in which the mouse is born with normal pancreas and then develops PanIN lesions (one of the main lesions linked to pancreatic ductal adenocarcinoma (PDAC) (2)) at an accelerated rate. The KPC model is immune competent and because the tumours develop orthotopically in the pancreas, they have a relevant microenvironment and stromal makeup, suitable for testing of new therapeutic approaches.

Unlike the human pancreas which is regular in shape, the mouse pancreas is a soft and spongy organ that has its dimensions defined to a large extent by the position of the organs that surround it, such as the kidney, stomach and spleen (3). This changes as pancreatic tumours develop, with the elasticity of the pancreas decreasing as the tissue becomes more desmoplastic. Because the tumours are deep within the body, disease burden is difficult to assess except by sacrificing groups of animals or by using non-invasive imaging. Collecting data by sacrificing groups of animals at different timepoints results in use of very high numbers per study. This is in addition to the fact that in the KPC model (similar to other GEMMs), fewer than 25% have the desired genetic makeup, meaning that 3-4 animals are destroyed for every one that is put into study (2). Therefore, in order to reduce the numbers of animals used in

pancreatic research, a non-invasive imaging tool that allows accurate assessment of pancreatic tumour burden longitudinally over time has been developed. Magnetic resonance imaging (MRI) has been used as it is not operator dependent (allowing it to be used by non-experts) and does not use ionising radiation which is a potential confounding factor when monitoring tumour development. The tool has been developed for use with a low field instrument (1T) which ensures its universal applicability as it will perform even better when used with magnets of field strength higher than 1T.

This work has been carried out starting from an existing 3D computational mouse atlas and developing a mathematical model that can automatically detect and segment mouse pancreas as well as pancreatic tumours in MRI images. This has been achieved using multiple image analysis techniques including thresholding, texture analysis, object detection, edge detection, multi-atlas segmentation, and machine learning. Through these techniques, unnecessary information is removed from the image, the area of analysis is reduced, the pancreas is isolated (and then classified healthy or unhealthy), and - if unhealthy - the pancreas is evaluated to identify tumour location and volume. This semi-automated approach aims to aid researchers by reducing image analysis time (especially for non-expert users) and increasing both objectivity and statistical accuracy. It facilitates the use of MRI as a method of longitudinally tracking tumour development and measuring response to therapy in the same animal, thus reducing biological variability and leading to a reduction in group size. The MR images of mice and pancreatic tumours used in this work were obtained through studies already being conducted in order to reduce the number of animals used without having to compromise on the validity of results.

Contents

List of Figures

1 Introduction

1.1 Pancreatic Cancer

Pancreatic cancer is the 10th most common cancer in the UK, affecting approximately 10,000 people per annum with a 5 year survival rate of only 5% (1). This poor prognosis of patients with pancreatic cancer has persisted for the last 40 years (1) with Cancer Research UK regarding it as a cancer of unmet need. In pre-clinical research there has been a recent shift to using genetically engineered mouse-models to study pancreatic cancer (4). For example, the KPC mouse-model has an intact immune system and the animals are born with a healthy pancreas that develops Pan-IN lesions that develop into PDAC tumours (2, 5, 6), the most common form of pancreatic tumours seen clinically (7). In order to detect spontaneous tumour development within KPC mice scientists often rely upon palpating the animal or ultrasound (8). However, palpation is unreliable as it is hard to discern if an object is a tumour, a cyst, or food within the digestive system. Although ultrasound can easily distinguish between tumour and cyst, it can also be problematic in that it has a small field of view and is very vulnerable to user-interpretation (9). I propose the use of MR-imaging combined with automated segmentation as a means of combating the mentioned issues. With regard to addressing the unreliability of tumour identification presented by palpation, MR-imaging provides high-detail internal images that can show clear differences between food in the intestine, cysts, and tumours. The disadvantages associated with ultrasound are also combatted by MR-imaging in that it provides a large field of view with high-sensitivity. However, there are also challenges associated with MR-imaging, specifically the imaging data is complicated and hard to analyse (especially early on in spontaneous tumour development where it can be difficult to distinguish tumour from normal pancreas), which means that results are open to user-interpretation.

High field instruments (e.g. with field strength of 9.4 T) have much better image quality but the user complexity, cost and siting requirements of these are prohibitive for non-specialist units

that simply require screening of tumour burden. Low field (e.g. 1T) bench-top MRI instruments are easy to use, affordable and have none of the magnet cooling costs or high field siting requirements. However, the trade-off is that low field instruments have lower resolution which increases the scan time and difficulty of image analysis.

Therefore, in the context of the growth of research carried out in spontaneous orthotopic pancreatic tumour mouse models, a need exists to develop software that provides automatic segmentation of a reasonable resolution (0.15-0.25mm) MR images with minimal user-input. Such a tool would also be able to be used with higher field systems. This project seeks to use a combination of pre-segmented manual and atlas data and machine-learning algorithms to be able to identify and segment a number of organs and tumour. This includes training an existing whole body atlas to segment the liver, spleen, kidneys, stomach, hepatic portal vein and gall bladder while also creating a new machine learning tool capable of segmenting pancreas and distinguishing pancreatic tumour from normal tissue. This automatic organ/tissue segmentation should be achievable with minimal user-input and minimal specialist training required on the part of the user. Evaluation of the sensitivity and specificity of the tool, as well as exact tumour locations and size should show agreement with analysis carried out by a panel of experts in the field (a preclinical veterinarian and two experts in preclinical image analysis) with over 35 years of image-analysis expertise combined. The core impacts sought from this study are as follows: the use of low field MRI combined with a 3D mouse atlas machine learning tool allows users with minimal training to obtain accurate and statistically-significant tumour measurements from smaller numbers of animals than previously achievable , in keeping with the rationale behind the 3Rs.

1.2 Preclinical Research

1.2.1 Animal models for pancreatic cancer

The mouse is a good subject of study when considering researching possible treatments for pancreatic cancer. Mice can be bred to be almost genetically identical through inbreeding. Their

reproduction rate is relatively quick, so long term effects can be followed within reasonable time frames. Additionally, the pathology and anatomy of mice is well understood. However, the mouse pancreas is not as well defined or regularly shaped as the human pancreas. Instead, it is defined geometrically by the organs surrounding it due to its soft structure (2, 3). Therefore, consistent images of the mouse pancreas are difficult to produce and analyse, which has led many scientists to disregard Magnetic Resonance Imaging (MRI) as a method of studying mouse pancreas, relying on ultrasound or dissection instead.

A very common method of inducing orthotopic pancreatic tumours is through injection of tumour cells directly into the pancreas of the mouse. When the procedure is performed by an experienced worker, it is reproducible, but can suffer the drawback of possible contamination of other areas of the mouse with tumour cells. Also, since the injections are mainly performed in the tail of the pancreas, this is the area where the tumours will most likely form (and not necessarily where a spontaneous tumour would form). The tumours develop relatively rapidly which is not always ideal in the context of the tumour microenvironment. Additionally, this is a surgical procedure which causes discomfort to the mouse. However, mice that are genetically modified to spontaneously develop pancreatic cancers do this in a similar way to that observed in the clinical situation and over long periods of time. Nevertheless, both orthotopic methods are more useful for the study of tumour development, local pathology and anatomy than subcutaneously xenografted mice, in which a tumour is implanted in the flank of the mouse and not in the pancreas (although the latter method can be useful for the study of drug effects on the tumour) (10).

Genetically engineered mouse models (GEMMs) are considered to be the most clinically relevant model. This is due to the development and progression of pancreatic cancer - as well as the tumour micro-environment in the context of an intact immune system - being much closer to that seen in humans. One commonly used GEMM in the study of pancreatic cancer is the KrasLSL.G12D/+;p53R172H/+;PdxCre model (KPC) along with PDXCre+;KrasG12D--;p53mut--

(KP) (11). KPC mice are born with a healthy pancreas but develop Pancreatic Intraepithelial Neoplasia 1-A (PanIN) lesions, a precursor to pancreatic ductal adenocarcinoma (3). Similar to the human disease, these tumours develop in the pancreas spontaneously over long periods of time. C57BL/6 (WT) animals were also used as a true control in order to be able to detect changes between WT and KP mice. However, one of the drawbacks of this KPC model is that less than 25% of the litter will have the desired genetic makeup, which means that 3-4 mice are destroyed for every one that can be used in a study. Furthermore, unlike the human pancreas which is very regular in shape, the mouse pancreas is a soft and spongy organ that has its dimensions defined primarily by the organs that surround it, such as the kidney, stomach and spleen (3). In the KPC mouse, this changes as pancreatic tumours develop, with the elasticity of the pancreas changing as the tumour becomes more desmoplastic. Since the tumours develop deep within the body, tumour burden is difficult to assess by palpation. In studies which measure response to therapy over time, killing groups of animals (typically >10) at different time points is commonplace, meaning that a large number of mice must be used in every study. Use of longitudinal non-invasive imaging, whereby the same group of animals is imaged at multiple timepoints, reduces both biological variability and the number of mice used. Accurate assessment of disease burden also has the potential to improve animal welfare.

The animals are housed in Individually Ventilated Cages (IVCs) in groups whenever possible and kept in rooms with a 12-hour day and night light cycle. A large percentage of the animals imaged in this study were also used in other studies to reduce the amount of animals needing to be bred.

Multiple academic groups are working with the KPC model in a variety of different studies, from tumour development, response to therapies (12) (11, 13, 14), to formation of the stroma in and around the tumour (15, 16).

1.3 Medical Imaging

Medical imaging is a vital tool used in the diagnosis and management of the vast majority of cancer patients (17). There are many different medical imaging techniques, each of which visualises a different physical or physiological property of the body. The major clinical imaging modalities are covered briefly below.

X-ray images, both planar (X-ray) and tomographic (CT), are images of the distribution of the linear attenuation coefficient at an average X-ray energy within the human body. This is a property which is largely a function of tissue density, at least for soft tissues, but is also a function of tissue composition. Thus, bone has a higher attenuation coefficient than can be explained simply by increased density because it contains significant amounts of relatively high atomic number elements (17). X-ray images are largely images of anatomy. Nuclear imaging produces images (PET: positron emission tomography and SPECT: single photon emission computed tomography) of the distribution of a molecule which has been labelled with a γ-ray-emitting isotope (termed a radiotracer). The radiotracer is distributed according to physiological function so the image is primarily a functional image, although it can in some cases produce recognisably anatomical images. Ultrasound imaging (US) produces images related to changes in the acoustic impedance of tissues, again mainly anatomical in nature. Magnetic resonance imaging (MRI) produces images of proton density, largely a function of the water content of tissue, and images of relaxation times which depend on the environment of the protons. These are mainly anatomical. Each of the imaging techniques is unique, in the sense that a unique physical property is imaged. X-ray images are quick to obtain and give good anatomical detail (skeletal and to a lesser extent soft tissue), however (in the case of CT) the use of ionising radiation can result in significant radiation doses. Nuclear imaging gives good functional images but limited anatomical imaging. It also uses ionising radiation as it requires the synthesis and administration of radiopharmaceuticals, and there is an associated radioactive dose to the subject. Ultrasound offers good anatomical soft tissue detail and resolution but has a small field

of view, is limited by the depth of the feature within the subject and is very operator dependant. MRI offers good soft tissue detail, high resolution and uses non-ionising radiation with no effect on the subject (17)

Currently, analysis and reporting of medical images is mainly done manually, carried out by medical physicists and radiologists who are experts in the field. However, computational models to help the radiologist validate their choices and optimise their workflows are coming into use. This is commonly referred to as Computer Aided Diagnosis (CAD) such as lung nodes, skin pigmentation and mammograms. Computer aided image analysis has been used for many years in many different fields, from the reading of bar codes, the segmentation of satellite images, and machine vision used for facial recognition (18-21).

Image analysis in medicine has powered not only diagnosis, but also guided patient management and delivery of therapy for many decades. In this project, semi-automated image analysis is used to demonstrate the feasibility of using pre-clinical MRI as a rapid and user-independent method for tumour detection and tracking in mice, with aim of facilitating the monitoring of pancreatic tumour development and response to therapy.

1.3.1 Clinical Abdominal Segmentation

Some novel methods of automatic abdominal segmentation have been developed in the clinic. This has allowed for the segmentation of all organs in the abdomen. In the method proposed by Shen *et al* (2016) (22) subcutaneous adipose tissue (SAT) regions were assessed by a fully automatic algorithm, using morphological operations and a multi-atlas-based segmentation method. However, although a high confidence was found in liver segmentation, calculated through Dice Similarity Coefficient (DSC, a measure of spatial overlap between two sample sets), in pancreas segmentation, only a 0.672 confidence when compared to expert manual segmentation was found (22). Another group used Support Vector Machines (SVMs) and graph-based Convolution Neural Networks (CNNs) in combination to build an automatic segmentation model (23). Their model works better than previous models, but still with a segmentation

confidence of 0.761. Both methods, SVMs and CNNs, had reasonable success with the combination of one or two different systems being used in the clinic (22, 23).

1.3.2 Radiomics

The field of Radiomics was born from CAD, in which automated image analysis is performed of image features. The term 'Radiomics' was first defined in a conference abstract by Lambin et al (2011):

"Comprehensive quantification of disease phenotypes by applying a large number of quantitative image features representing lesion heterogeneity and correlating with omics and clinical data" (24)

Lambin then went on to publish the paper "Radiomics'' in 2012 which defined the field as we recognise it today (25). The field was built off the basis of Genomics and Big Data where an increase in computational power had allowed for sequencing of thousands of genomes and the need to analyse this so-called 'Big Data'. Radiomics approaches imaging data in the same way by using features of each voxel to produce big imaging data.

Prior to the definition of 'radiomics', multiple papers had shown the ability to classify gene expressions using non-invasive imaging. (26, 27)

Non-invasive imaging provides an alternative to biopsies and histological data. The benefits of imaging include avoidance of a painful invasive technique, reduced chance of infection to the subject, and the ability to account for the heterogeneity of tumours. However, radiologists rely on visual inspection and it has been shown that when it comes to tumour classification, 31-37% of cases have discordant interpretations (28-30). Some of the issues that face radiologists are: Technical Errors (Scanning Factors, Scanning Parameters, Imaging Protocols), Active Errors (Perceptual Factors, Window Settings, Blind Spots, Reading Environment and Ergonomics, Satisfaction of Search, i.e. after finding one tumour not looking for more.) and Interpretative Factors (Study Indication, Effects of Therapies and Other Interventions) to name a few (31, 32).

Radiomics provides a quantitative approach to image-segmentation in which the features identified can be related to anatomical and genomic expressions. There were two papers that heralded the inception of the field of Radiomics just prior to Lambin's 2011 paper. The first of these papers was written in 2007 by Segal et al (26) in which they identified 28 imaging traits that were capable of reconstructing 78% of global gene expression profiles. The second was written in 2008 by Diehn et al (27) in which they identified a "infiltrative" imaging phenotype that was able to predict the patient outcome. The combined effect of these two papers and others was the study of image features, resulting in Lambin's inspiration for the field of Radiomics.

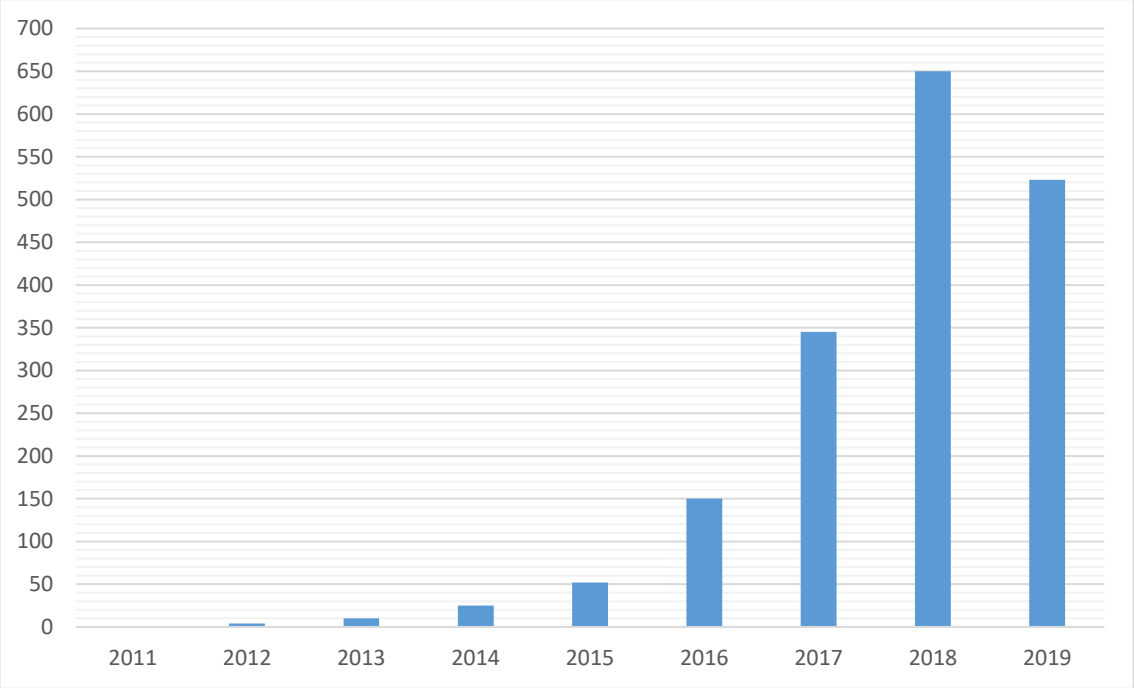


Figure 1-1: The number of papers published using the term radiomics between 2011 and August 2019

The field of radiomics spans from 2011 to the present day (2019) and has faced multiple challenges, many of which have been outlined by Kumar *et al.* (32). These include the requirement of a large multidisciplinary team not only including biologists, chemists, and physicists but also computer scientists, mathematicians, bioinformaticians, and statisticians.

Another challenge faced is the availability of data where institutes such as universities and hospitals will have large cohorts of data but are unable to share them with other institutes. This problem is further compounded by the effect of different institutes having different imaging standards meaning that even if data can be shared it is often not comparable. Another major challenge that radiomics has had to overcome is the validation of its methodologies and proof of concepts with reliable correlation to the histological data collected. This issue is one of the most complex as the histological sample is only a very small area of the tumour and co-registering the histological slice with the image slice is not a trivial matter (33). Feature extraction is another method that requires standardisation especially if data from multiple institutes and reproducibility are to be considered. The statistical analysis performed on the features should also be standardised as imaging data in general can be very misleading and the statistics, if improperly handled, even more so. The radiomics field has brought together numerous texture and feature analysis methods used in multiple disciplines, from medical imaging to monitoring of sea ice (34-41).

1.4 Current Methods of Detection of pancreatic tumours in preclinical models.

1.4.1 Dissection

Dissection of a mouse is the most basic way to study its anatomy and is the only way to get immunohistochemical data. This has many drawbacks: individual mice cannot be studied longitudinally which introduces biological variability; if the mice are difficult to breed, as with the KPC model, it can be very costly; and it can be difficult to decide when to dissect a mouse as tumour development may not be linear and so many more mice must be dissected to gain statistical significance. Because of this, non-invasive techniques to study tumours have been developed.

1.4.2 Ultrasound

Ultrasound is a common method of detection and tracking of mouse pancreatic tumours (13). Ultrasound offers a high accuracy as well as anatomical and functional imaging. Protocol time in the hands of an experienced user is also relatively short when compared to MRI and in the current UK-wide university model of cost recovery from users in the form of an hourly rate, throughput has a significant impact on imaging costs. Ultrasound has a relatively high depth of penetration, although certain features (air, bone) can obscure the image.

In terms of instrument cost, a preclinical ultrasound for oncology applications can range from £150,000-£250,000 and preclinical MRI £225,000-£1,500,000. An MRI will often need a specially designed room with a Faraday cage built into the walls, floor, and ceiling unless it is a low field MRI. In contrast an ultrasound is often on wheels and can be moved relatively easily. However, with the further development of low field MRIs, “desktop” MRIs which do not require the infrastructure are becoming increasingly popular. Low-Field MRI is also becoming more readily available in preclinical imaging units as it is currently offered as an anatomical imaging modality combined with PET and SPECT instruments from a number of manufacturers as an alternative to PET/CT or SPECT/CT (42-45).

In terms of a direct comparison of usability, all things being equal re. accessibility, MRI presents as the more convenient. While ultrasound is often faster for an experienced ultrasonographer and is able to provide a higher resolution than MRI, MRI offers a larger field of view as well as greater depth of penetration, this method also provides more usable results and reduced user variability. In terms of practical convenience, ultrasound requires the animal to be shaved prior to the procedure which adds to the time required per animal, as well as having a welfare impact on the animal.

There are 6 categories by which we can measure the quality of MRI and ultrasound: image resolution, time taken, depth of penetration, field of view, required operator proficiency level and animal welfare. In ultrasound, the image resolution is very high (0.01-1mm resolution (46)),

it is a relatively fast procedure, and the depth of penetration is good (1-10cm depending on resolution (46), however the field of view is relatively small and heavily dependent on the probe used, the shaving required prior to the procedure is distressing to the animal. The data is easily influenced by user technique/proficiency and interpretation which leads to variability in data quality between users (47). These limiting factors can be addressed and sometimes eliminated when using other non-invasive imaging modalities. In MRI, the welfare impact is low as no shaving is necessary. In contrast to ultrasound also, MRI offers a relatively large field of view and an unlimited depth of penetration. The image resolution is also relatively high (0.25mm isotropic on a 30 minute T1 scan on the Bruker Icon 1T) – although not quite as high as ultrasound – with the only potential downside being the time required, as the higher the resolution required the longer time each scan will take. See Table 1 for a comparison of these factors.

Table 1: Comparison of low field MRI, high field MRI and Ultrasound

	Low Field MRI	High Field MRI	Ultrasound
Resolution	0.1-0.25mm	0.01-0.25mm	0.001-1mm
Soft tissue contrast	+++	+++	+++
Field of View	+++	+++	+
Depth of penetration	Infinite	Infinite	1-10cm
Throughput (volumetric imaging time plus animal prep)	30-45 minutes	10-20 minutes	25-45 minutes
Welfare advantage	+++	+++	++
Ease of use	+++	+	+
Affordability	+++	+	+++

1.5 MRI

MRI is a good imaging modality to consider for this application. It is an anatomical imaging modality, like Computed Tomography (CT), but is superior for soft tissue imaging and (unlike CT) produces the images using non-ionising radiation (47). Large areas of the mouse can be imaged in a single scan, allowing better understanding of the effects of the tumours on the full organ

system, as well as giving anatomical references that will aid definition of pancreas in the mouse (a challenging organ to define). MRI also allows high resolution imaging and features as small as 0.2mm^3 are distinguishable on a low field MRI (48). There is no need to shave the mouse or have it undergo any other pre-scan procedures apart from anaesthesia.

1.5.1 Basics

Magnetic Resonance Imaging (MRI) is a non-invasive imaging technique which, using a strong magnetic field and radiofrequency (RF) pulses, is able to visualise the inner structures of an object. MRI is a powerful tool in the study of tumour development and responses to therapy as the good soft tissue contrast can provide a large amount of anatomical and pathological information (49).

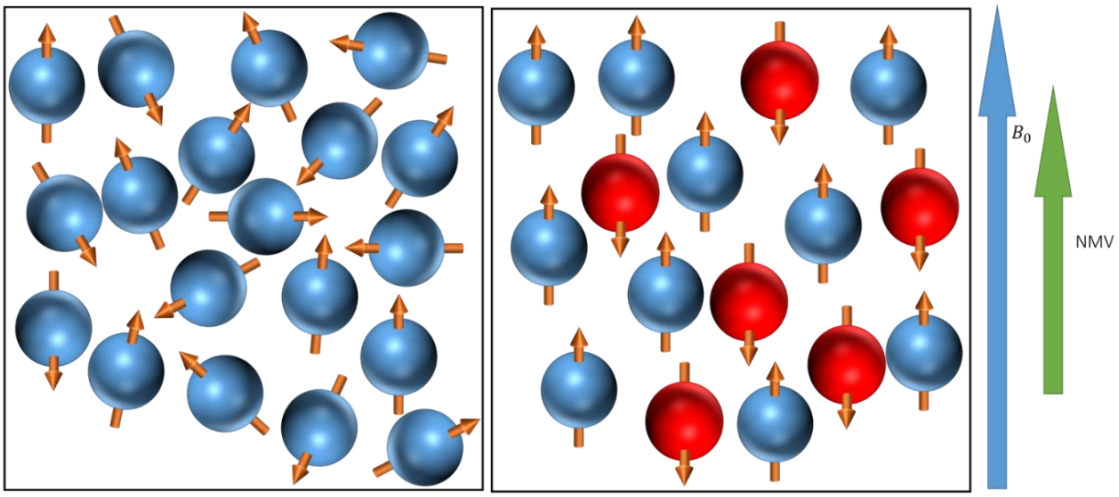


Figure 1-2: Alignment of hydrogen protons. (a) No external magnetic field applied and the protons are aligned randomly. (b) the magnetic field has been applied to the protons and they have aligned either spin-up or spin-down. More have aligned spin-up so the net magnetisation vector, NMV, is aligned with the external magnetic field. Adapted from (50).

MRI works by aligning the magnetic spin of hydrogen atoms within a large magnet. Because a hydrogen atom has one proton and no neutrons it has the property of a magnetic (dipole) moment i.e.: it will align with an external magnetic field (B_0) and has two poles. A very small number of atoms will align inverse to the magnetic field (spin-down) but the majority will align

with the magnetic field (spin-up), as this is the lower energy state. This causes the net magnetic field to be aligned with the magnetic field of the MRI, called net magnetisation vector (NMV), see Figure 1-2. The atoms also have spin (wobble or precession) of the charged nuclei around B_0 from the magnetic field's influence.

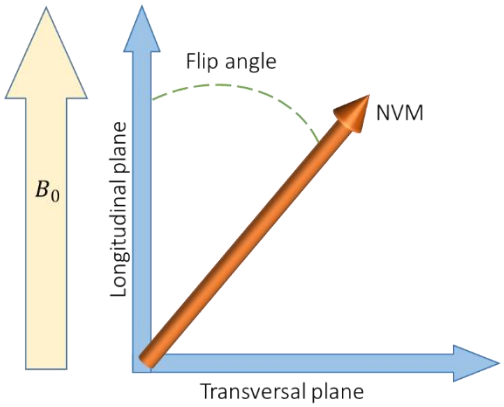


Figure 1-3: The net magnetisation vector (NMV) moves out of alignment with the magnetic field when excited at resonance by the RF pulse. The angle out of alignment is called the flip angle. The transversal plane is perpendicular to the longitudinal plane, so a flip angle of 90^O moves the NMV onto the transversal plane. Adapted from (50).

1.5.1.1 The Larmor Equation

The Larmor (precession) frequency, ω_0, refers to the rate of precession of the protons around the external magnetic field and is related to the strength of the magnetic field, B_0, and the gyromagnetic ratio, y, which is the ratio of the nucleus magnetic moment and angular moment.

$$\omega_0 = yB_0 \qquad 1\text{-}1$$

For example, the Larmor frequency of hydrogen in a 1 T MRI is 42.57 MHz, while the Larmor frequency of a sodium atom in a 1 T MRI is 11.26 MHz (51).

1.5.1.2 Resonance

When an object is exposed to an oscillating perturbation that has a similar frequency to its own natural frequency, resonance occurs. The energy required for resonance of hydrogen at the precessional frequency corresponds to the radio frequency (RF) band of the electromagnetic

spectrum and, depending on the field strength of the magnetic field, must be the exact energy defined by the Larmor equation. This resonance frequency will not resonate with other nuclei as they have a different Larmor frequency since their gyromagnetic ratios are different.

This causes the NMV to slip out of alignment with the external magnetic field as some protons are excited to a higher energy state. The degree to which the NMV is out of alignment is called the flip angle (see Figure 1-3) and can be altered by the amplitude and duration of the RF pulse. Furthermore, this resonance frequency causes the magnetic moment of the hydrogen nuclei to be in phase with each other.

1.5.2 The MR Signal

After the RF pulse has been applied, the coherent magnetisation precesses at the Larmor frequency in the transversal plane. This means that if a conductive loop is placed in a moving magnetic field a voltage is induced across it (52). A receiver coil performs this function, with the frequency of the resulting? magnetic resonance signal being the same as the Larmor frequency, and the magnitude defined by the amount of magnetisation in the transversal plane.

When the RF pulse is switched off the NMV is again influenced by the external magnetic field and so returns to that state. In order to do this the hydrogen must lose the energy caused by the RF pulse, this is called relaxation. At the same time, the magnetic moments of the hydrogens dephase.

Spin lattice relaxation (T1 recovery) is the effect seen when the nuclei release their energy into environment or lattice. This causes the magnetic moments of nuclei to recover back to the external field. The rate of recovery is called the T1 relaxation time, which refers to the time it takes for 63% of the longitudinal magnetisation to recover into the tissue (see Figure 1-4).

Spin-spin decay (T2 relaxation) is the interaction of the magnetic fields of neighbouring nuclei and results in the decay of coherent transverse magnetisation (see Figure 1-4). This is also an

exponential process, so that the T2 relaxation time of a tissue is its time constant of decay and the time taken until 37% of the transverse magnetisation, tumbling, remains.

A pulse sequence is a combination of RF pulses, signals, and periods of recovery. The repetition time (TR) is the delay from the application of one pulse to the next and determines the amount of longitudinal relaxation. The echo time (TE) is the delay from one pulse to the peak of the signal induced in the coil.

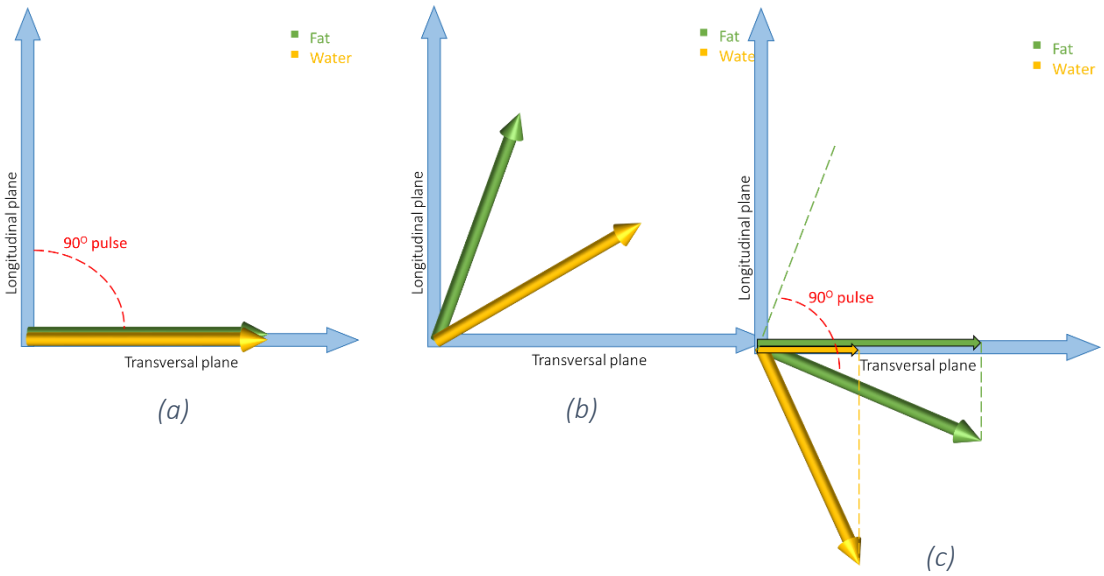


Figure 1-4: The relaxation rates of different tissues can be used to increase the contrast in the image. (a) shows the alignment of the hydrogen atoms in water (yellow) and fat (green) after a 90^{O} pulse is applied. (b) shows the alignment after some time has passed. Fat is progressing back to the longitudinal plane faster than water. (c) shows the alignments after another 90^{O} pulse is applied. At this point the fat has a larger component vector in the transverse plane and so if the signal is collected now fat will have a higher signal. As time continues and both tissues progress back to the longitudinal plane the transversal vector of water will get larger than that of fat and so will become the stronger signal. This is defines the TR length that can be used for high contrast. Adapted from (50).

1.5.3 Imaging Contrast

Intrinsic image contrast parameters (parameters that cannot be changed) in MRI are critical and consist of T1 recovery time, T2 decay time, and proton density. T1 recovery and T2 decay are defined in section 1.5.2. Proton density refers to the number of mobile hydrogen nuclei per unit

volume of the tissue. Higher proton density means there are more nuclei to contribute to the signal. T1 and T2 relaxation rely on three main characteristics:

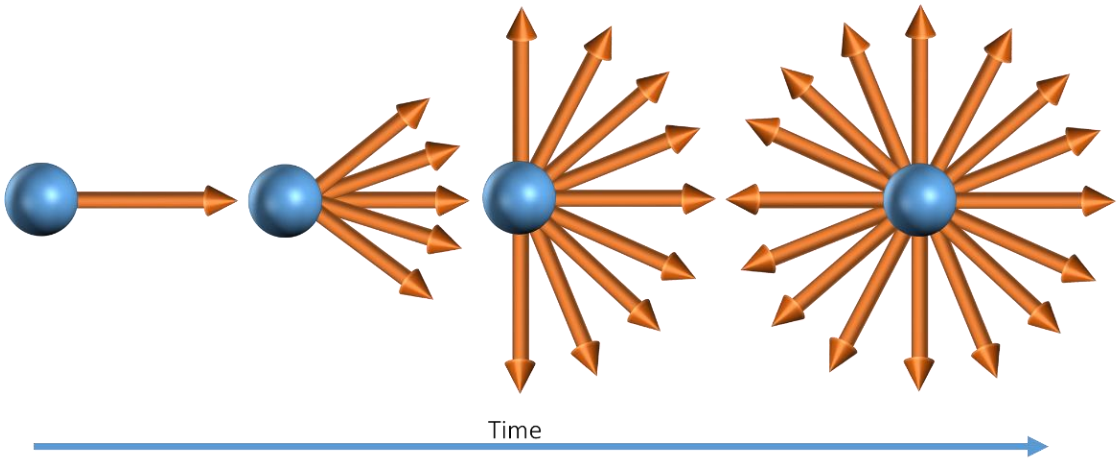


Figure 1-5: Dephasing of the hydrogen vectors over time in the transversal plane. Adapted from (50).

- The inherent energy of the tissue: if the inherent energy of the tissue is low, this means it can absorb a great amount of energy during the hydrogen relaxation. If the inherent energy is high then it cannot absorb as much energy. This is of particular importance for the T1 relaxation process.
- The tissue density: in a tissue with closely packed molecules the interactions between magnetic fields is more efficient, with less dense tissues having less efficient interaction. This is important for T2 relaxation and spin-spin decay.
- The relationship between the molecular tumbling rate and the Larmor frequency: if the molecular tumbling rate and the Larmor frequency are similar then interaction between the hydrogen molecules and the molecular lattice are more efficient than if there were a large discrepancy between the tumbling rate and Larmor frequency.

Different tissues have different relaxation times due to these intrinsic contrast features. This difference can be related to T1 relaxation and T2 decay.

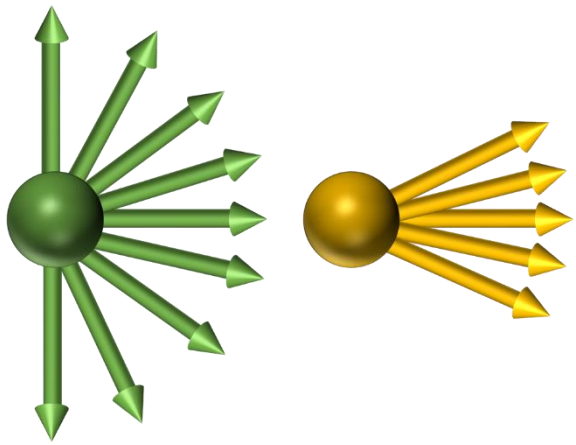


Figure 1-6: Fat (green) dephase more quickly in the transversal plane than water (yellow) this means that after a short delay the vector in the transversal plane of fat is smaller than of water as the water is still more in phase. This causes the water to give off a stronger signal in the image. Over time they both completely dephase. Adapted from (50).

The molecules in fat are large, since they are formed of hydrogen, carbon, and oxygen atoms and are packed closely together, this causes the tumbling rate to be relatively slow, the result being a relatively slow relaxation time (17).

Fat has a low inherent energy and so can easily absorb energy into the lattice. This, coupled with a relatively slow relaxation time, means that the NMV of fat realigns with B_0 rapidly, longitudinal magnetisation, causing the T1 time for fat to be short.

Fat molecules are close together causing a high chance of spin-spin interaction. This causes a rapid dephasing and loss of transverse magnetism, resulting in a fast T2 time (see Figure 1-6).

The molecules in water are spaced relatively far from each other and are relatively small, as they are formed of two hydrogen and one oxygen atom. This results in them having a relatively fast tumbling speed (17).

Water has a high inherent energy and so cannot absorb much energy into the lattice. Furthermore, the tumbling rate of water is not as close to the Larmor frequency, the result being a longer T1 recovery. Therefore, the NMV of water takes longer to align back to B_0.

The molecules in water interactions are spaced further apart than those seen in fat. This causes the dephasing and a slower loss of transverse magnetism, resulting in a longer T2 time.

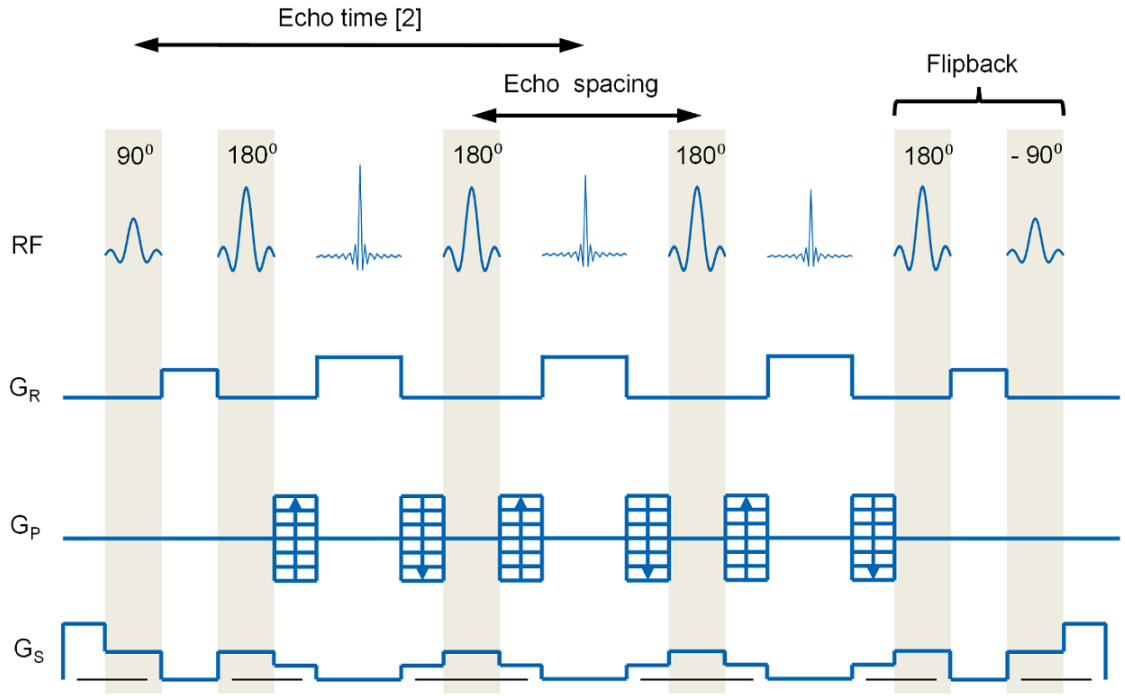


Figure 1-7: A spin echo RARE pulse sequence with 4 phase encoding steps. This is able to acquire 4 times the data a standard pulse sequence could attain and so reduces scan time by 25% (51). RF = radio frequency, G_R = gradient readout, G_P = gradient phase, and G_S = gradient slice. Image taken from the Bruker ICON user documentation. Adapted from (50).

1.5.3.1 T1 Weighted Images

As fat and water recover from a RF pulse, with a 90^o flip angle, the fat recovers to the longitudinal plane quicker. This means that if a second 90^o pulse is applied to the system before the water has recovered then fat will be aligned back onto the transversal plane. However, water will be aligned past the transversal plane. This means that the signal from the fat will be stronger relative to the water as its vector has a larger transversal component. This will cause fat to show up bright and water dark in the image. This is a T1 weighted image (see Figure 1-4).

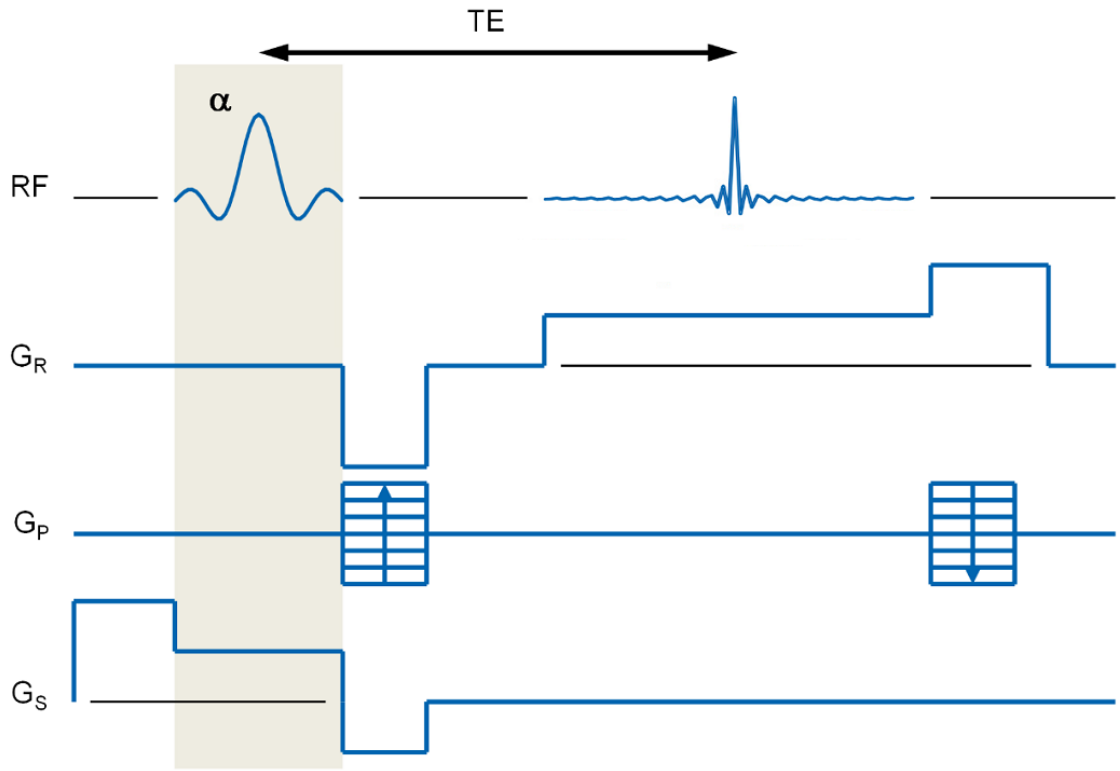


Figure 1-8: A gradient echo FLASH pulse sequence (51). RF = radio frequency, GR = gradient readout, GP = gradient phase, and GS = gradient slice. Image taken from the Bruker ICON user documentation. Adapted from (50).

1.5.3.2 T2 Weighted Images

Fat and water recover from a 90^o RF pulse, however fat has a fast T2 recovery time meaning that the vector component of fat is smaller than that of water after the pulse is applied. This causes fat to appear dark and water to appear bright on the image (see Figure 1-6). This is a T2 weighted image.

1.5.4 Pulse Sequences

1.5.4.1.1 *Spin Echo Pulse Sequences*

A spin echo pulse uses one 90^o excitation pulse followed by at least one 180^o rephasing pulse to generate a spin echo. With one echo generated you can create a T1 weighted image with a short TE and TR time. To generate a T2 weighted image two pulses are applied using a long TE and TR.

Instead of using the same phase encoding for each echo, Rapid Acquisition with Relaxation Enhancement (RARE) sequences operate by differing the phase encoding for each echo and so can populate multiple lines of an image on each pass (53), see Figure 1-7Figure 1-8. This results in a much faster technique for studying a tissue with a long TE, i.e. a T2 weighted scan. RARE sequences are also called 'fast' or 'turbo spin echo'.

Fast low angle shot (FLASH) sequences use a low flip angle (under 90^o) and subsequent gradients to produce the gradient echo (54). The RF spoiling option allows for T1-dependent contrast see Figure 1-8.

1.5.5 Field Strength

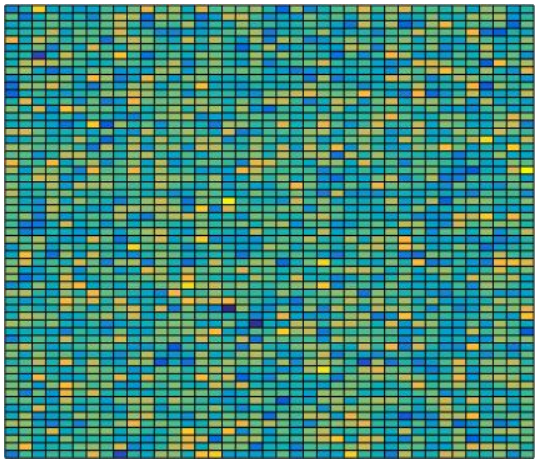
One of the major developments in preclinical MR imaging has been the creation of low field strength instruments. For example, the system used in this project is a 1T MRI which is low field strength compared to a possible 9.4 T MRI commonly used in specialist preclinical MRI labs (55). In general, a higher field MRI gives better signal to noise ratio and a higher resolution image. However, they are more complicated to use, take up much more space, require a lot more shielding to contain the magnetic field and cost more to run. So even though higher field machines are widely used in preclinical imaging, low field systems have become very popular especially as part of a multimodality preclinical systems such as PET/MRI and SPECT/MRI. Furthermore, users can easily be trained on low field systems, which means that it does not require an MRI specialist to run all the scans, saving time and money. However, the low field systems are not as versatile in terms of the range of protocols possible with high field machines. Nevertheless, a computational model developed to be used with low field MRI should work also on a high field MRI. In addition, it is possible to use the low field MRI instrument for quick and accurate screening as well as for longitudinal response to therapy studies.

1.6 MRI Data Conversion and reorientation

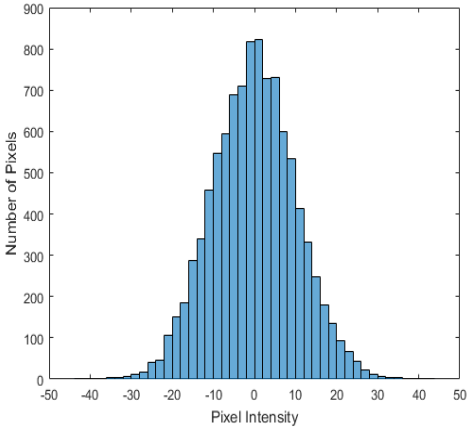
Once the MRI scan has been carried out and reconstructed, the images can be further processed in order to be used in an application such as the computational tool being created here. In order to do this, data from the MRI scanner can be converted from DICOM format to '.RAW' format in order to carry out various manipulations such as reducing file size, making meta-data analysis simpler, and to increase the efficiency of integration of the data into other programs such as MATLAB. Similarly any tool that uses the data will require it to be inputted in the same orientation, and this is best handled using a global rule that checks and reorients each scan before the analysis is performed.

1.7 Bias Field Correction

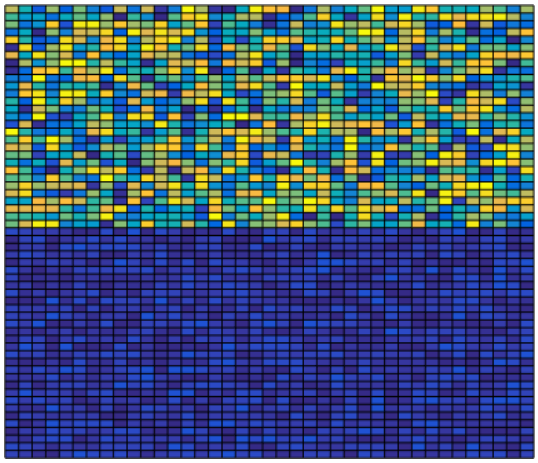
In MRI scanning a bias field is an artefact which adds randomness to the scan via a smooth, low frequency signal (56). This can cause significant problems with texture and feature analysis as this randomness can cause pseudo pattern formation. This is a major problem for machine learning for which multiple counter-methods have been proposed (57). Bias Field Correction allows the user to compensate for changes in the magnetic field due to environmental factors (e.g. ambient temperature), subject factors (e.g. size of the animal) and user discretion (e.g. where the animal is positioned on the bed). Bias field correction is particularly relevant in longitudinal studies and inter-subject studies. The effects of bias field correction are hard to detect with the human eye, but these effects are nevertheless important. The method used here is built into VivoQuant image analysis software (Invicro, USA) and uses an iterative Otsu method to identify and remove these signals (58).



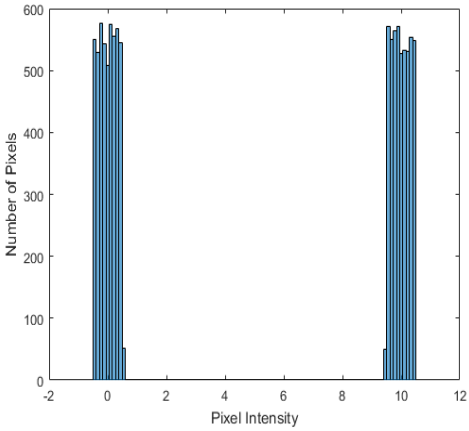
(a)



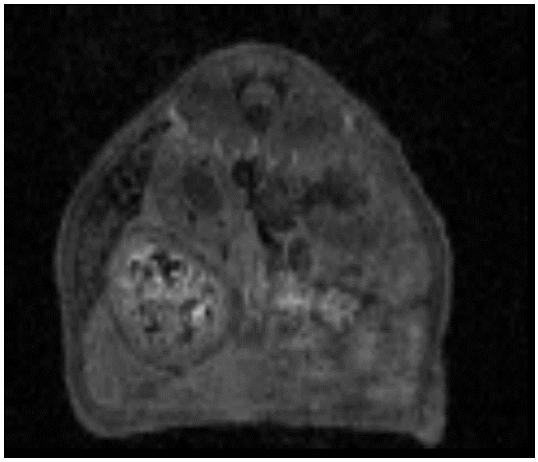
(d)



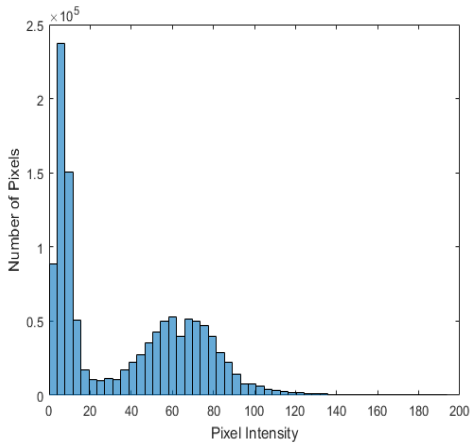
(b)



(e)



(c)



(f)

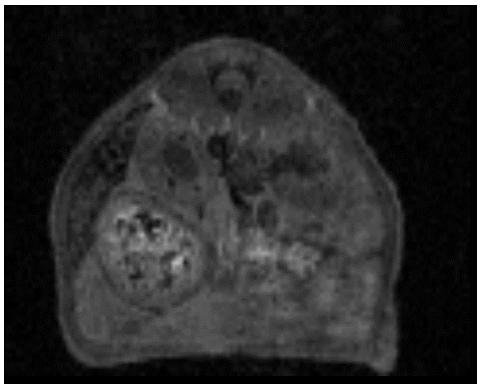
Figure 1-9: (a) shows an image with normally distributed random pixel values. (b) shows an image where both halves have normally distributed random pixels values but the top half has a mean of 10 and the bottom a mean of 1.(c) shows a single slice from a MR image. (d) is a histogram of (a); this case would be very difficult to identify individual objects. (e) is a bimodal

histogram of (b), in which it would be easy to distinguish two objects. (f) shows the histogram (c), the background is distinguishable by the peak below 10, but to try to split the other peak would be very difficult. Also, there is no guarantee that the first peak does not contain some important information from the image.

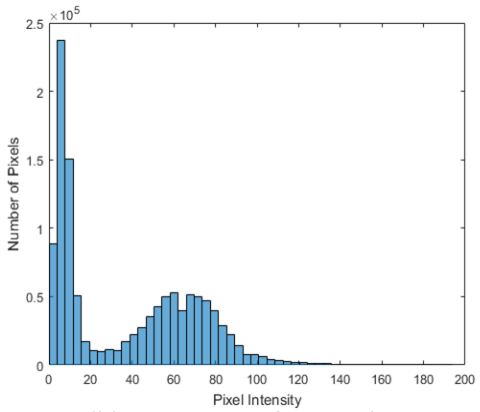
1.8 Thresholding

Segmentation through thresholding takes advantage of the difference in grey levels in an image and is particularly good at segmenting an object from background. It is fast and computationally simple. The most common way to use thresholding as a detection method is to study the histogram. If the grey level range of the object's intensity is known, then only the values of the histogram corresponding to that area need to be found. For example, in blood cell segmentation, where particular grey levels define the cytoplasm, background and cell kernel (59). A simple example would be that of an object with separate grey values, the result is a bimodal histogram (see Figure 1-9, (b) and (e)) easily showing the grey levels of the object. However, consider an image with the left side white and the right-side black, it would have the same histogram as a black image with 50% of the pixels coloured white at random. Thresholding cannot definitively link an area on the histogram to an area on the image, as it only finds the groups of signal intensity. One threshold can be applied to the entire image (global thresholding) to remove the object from the background if the object and background are well separated in intensity. One of the problems with MR imaging the grey levels of the different tissue types are not always well separated in the histogram.

It is uncommon in MR imaging for there to be clear boundaries between tissues as the grey levels of different tissues are not well defined and noise in the image. If segmentation with more than just one threshold is needed, then band thresholding can be used in which the image is segmented into multiple regions defined by the grey level. This method is often used in MRI segmentation of the brain to separate the grey matter, white matter, and cerebrospinal fluid (CSF) (60, 61). This method works well in the brain, where the three tissue types have well-defined separate proton densities and therefore different signal intensities, see Figure 1-11.

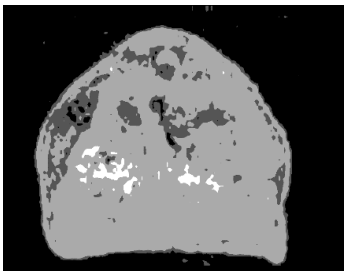


(a) Original image

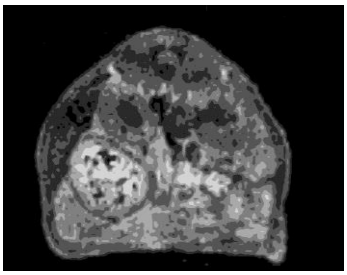


(b) Histogram of original image

(c) Global thresholding



(d) Band threshold with 4 bands

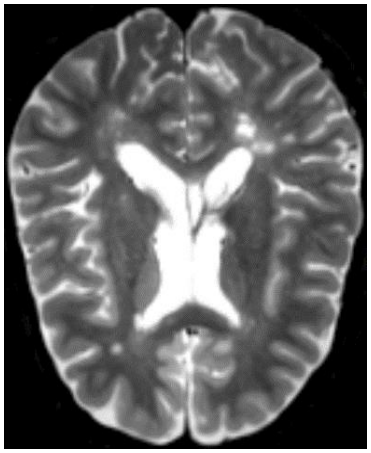


(e) Band thresholding with 7 bands

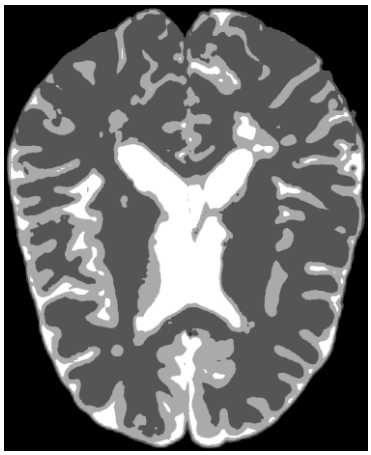
Figure 1-10: (a) shows MR image of a mouse. (b) shows the histogram of (a). Individual structures are much harder to distinguish than the following example, see Figure 1-11. (c) shows global thresholding in which the background of the image can be segmented out, but note some of the internal pixels have also been segmented with the background. (d) has four bands rather than the two used for global thresholding. Use of 4 bands his has not achieved useful thresholding in the image and has led to some of the background noise to be included, as seen at the top of the image. However, fewer of the internal pixels have been segmented with the background. (e) has more bands and although it has more detail, may still not be useful. It may be easier to segment some of the organs seen but use as a simple background removal method might be the most useful application.

1.9 Otsu's Method

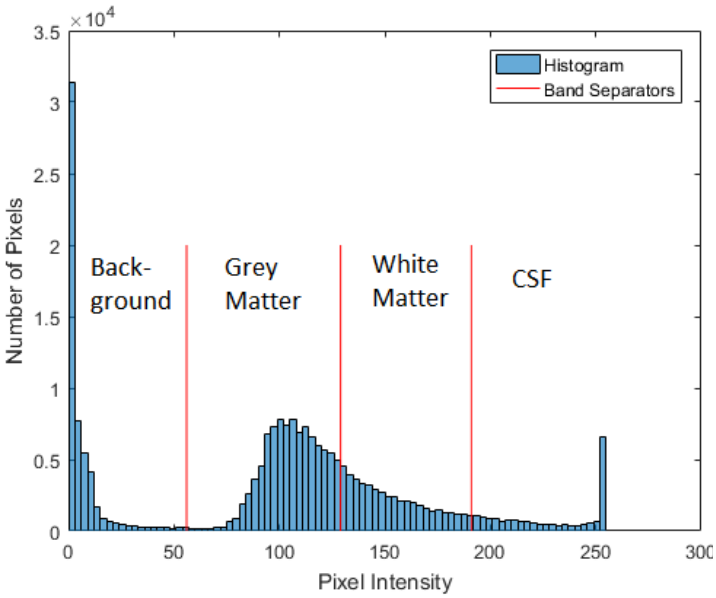
A simple threshold would be to choose the lowest point of a valley seen in a histogram; this would be effective for a bimodal histogram as seen in Figure 1-9. This is often not the case, as the histogram may be too complicated and have uneven peaks or a lot of noise. Where previous methods used differentiation to look for local minimums, Otsu's method uses integration to find the global minimums (62, 63). This means that Otsu's method could be used for unsupervised computation with no prior knowledge of grey levels of the object.



(a)



(b)



(c)

Figure 1-11: Band thresholding applied to brain MRI. (a) Single slice taken from an MRI of a human brain. (b) Band thresholding applied to a to segment of 4 bands. (c) Shows the histogram of (a) along with the bands that are applied to obtain (b).

To implement Otsu's method, an image is taken. Then for L, the number of grey levels (i.e. $[1,2,\dots,L]$) where the number pixels at each level is n_i. This gives the histogram of the image which is then normalised and taken as a probability density:

$$p_i = \frac{n_i}{N}, p_i \geq 0, \sum_{i=1}^{L} p_i = 1, for\ N = n_1 + n_2 + \cdots + n_L, \qquad 1\text{-}2$$

Then to calculate one threshold and two bands the probability density would be split into two groups:

$$G_1 = [1,2,\dots,k], G_2 = [k+1, k+2, \dots, L], \qquad 1\text{-}3$$

The probabilities and means of the groups can then be calculated, respectively:

$$\omega_1 = Pr(G_1) = \sum_{i=1}^{k} p_i = \omega(k), \omega_2 = 1 - \omega(k), \quad 1\text{-}4$$

$$\mu_1 = \sum_{i=1}^{k} i\ Pr\ (i|G_1) = \sum_{i=1}^{k} i \frac{p_i}{\omega_1} = \frac{\mu(k)}{\omega(k)}, \quad 1\text{-}5$$

$$\mu_2 = \frac{\mu_T - \mu(k)}{1-\omega(k)}, for\ \mu_T = \mu(L) = \sum_{i=1}^{L} iP_i$$

where

$$\omega(k) = \sum_{i=1}^{k} P_i\ , \mu(k) = \sum_{i=1}^{k} iP_i, \quad 1\text{-}6$$

are the zero[th] and first order cumulative moments of the probability density up to the k[th] level respectively. This gives the simple relationships which are used to verify the threshold choice:

$$\omega_1\mu_1 + \omega_2\mu_2 = \mu_T, \omega_1 + \omega_2 = 1 \quad 1\text{-}7$$

The group variances can be calculated by:

$$\sigma_1 = \sum_{i=1}^{k} (i - \mu_1)^2 \Pr(i|G_1) = \sum_{i=1}^{k} \frac{(i-\mu_1)^2 p_i}{\omega_1},$$

$$\sigma_2 = \sum_{i=k+1}^{L} (i - \mu_2)^2\ Pr\ (i|G_2) = \sum_{i=k+1}^{L} \frac{(i-\mu_2)^2 p_i}{\omega_2}, \quad 1\text{-}8$$

Therefore, the "goodness" of a threshold, which in this case refers to the effectiveness of thresholding separate objects, can be calculated by the following discriminant criterion measures, defined by (63).

$$\lambda = \frac{\sigma_B^2}{\sigma_W^2}, \kappa = \frac{\sigma_T^2}{\sigma_W^2}, \eta = \frac{\sigma_B^2}{\sigma_T^2}, \quad 1\text{-}9$$

where the variances are defined as:

$$\text{within-group variance } = \sigma_W^2 = \omega_1\sigma_1^2 + \omega_2\sigma_2^2 \quad 1\text{-}10$$

$$\text{between-group variance } = \sigma_B^2 = \omega_1(\mu_1 - \mu_T)^2 + \omega_2(\mu_2 - \mu_T)^2 = \omega_1\omega_2(\mu_2 - \mu_1)^2 \quad 1\text{-}11$$

$$\text{total variance } = \sigma_T^2 = \sum_{i=1}^{L} (i - \mu_T)^2 p_i \quad \text{1-12}$$

It is then an optimisation task to maximise λ, κ and η. But note that σ_W^2 is a second-order cumulative moment, so it is more complex than the others. Therefore, since η does not include σ_W^2, it is the simplest to maximise with respect to k. Therefore, maximising η is a measure of "goodness", which can be maximised by a brute force method:

$$\eta(k) = \frac{\sigma_B^2(k)}{\sigma_T^2} \quad \text{1-13}$$

Therefore, in order to maximise

$$\sigma_B^2(k) = \frac{[\mu_T \omega(k) - \mu(k)]^2}{\omega(k)[1 - \omega(k)]} \quad \text{1-14}$$

Which means that the optimal threshold k^* is

$$\sigma_B^2(k^*) = \max_{1 \le k \le L} \sigma_B^2(k) \quad \text{1-15}$$

This method thus gives a simple way to find the "good" thresholds. Once these groups have been found, each pixel within each group is assigned to the same value. This then forms the output and can be used for further processing, such as removing all data located in a certain band.

It can be expanded to have more groups by looking at

$$G_1 = [1,2,\dots,k_1],\ G_2 = [k_1 + 1, k_1 + 2, \dots, k_2], \\ G_3 = [k_2 + 1, k_2 + 2, \dots, k_3], G_4 = [k_3 + 1, k_3 + 2, \dots, L] \quad \text{1-16}$$

and then

$$\sigma_B^2(k_1^*, k_2^*, k_3^*, k_4^*) = \max_{1 \le k_1 < k_2 < k_3 < k_4 \le L} \sigma_B^2(k_1, k_2, k_3, k_4) \quad \text{1-17}$$

However, the more groups considered the less credible the thresholds become. Meaning that whilst this method works well for a brain MRI with defined groups, the same number of groups in an abdomen MRI would not be as effective. Adding bands does not necessarily make it operate more efficiently, as seen in Figure 1-11.

Once the threshold image is created, each band can be looked at individually for texture and features. This can then be applied back to the original image to evaluate the effectiveness of the thresholding or be used for further processing.

1.9.1 Adaptive Thresholding

Adaptive thresholding is another method that has been considered. In adaptive thresholding, instead of taking the entire image and calculating the histogram, each voxel is taken individually. A predefined neighbourhood around that voxel is considered and the histogram computed. This histogram is then banded with Otsu's method and the central voxel is given a value accordingly, see Figure 1-12. The main problem with adaptive thresholding is that objects will not provide a consistent signal over the area of the image, i.e. fat in one area of the image may have a different signal value than fat in another area due to the surroundings. This can cause misleading thresholds when a small window is used, as can be seen in Figure 1-12.

1.9.2 2D vs 3D Thresholding

Applying Otsu's method to a 3D image is no different from to that of a 2D image, since the histogram of the full image is considered. Applying adaptive thresholding techniques to a 3D image requires a change in the neighbourhood type to a 3D neighbourhood. After this point the thresholding method is identical (64).

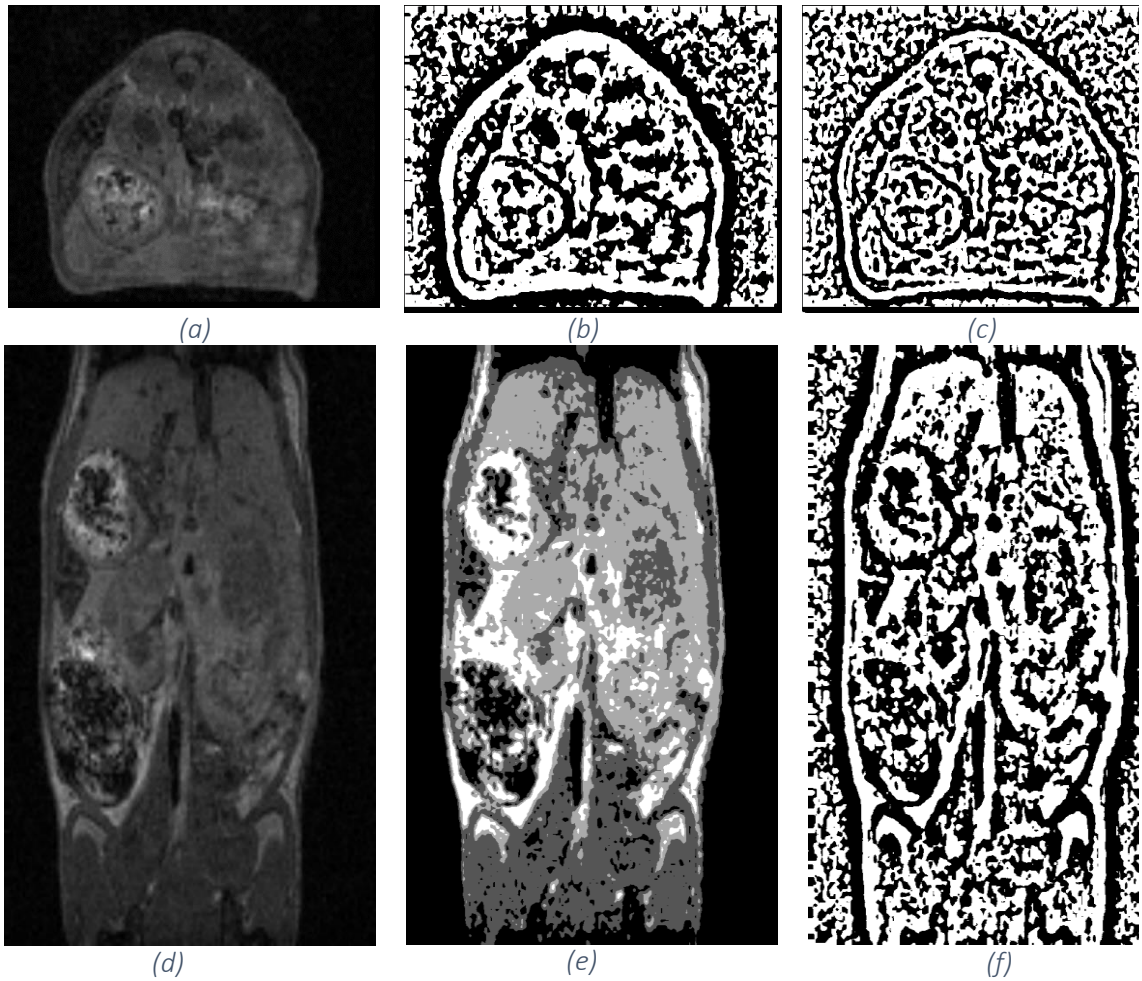


Figure 1-12: (a) shows an MRI image. (b) and (c) show adaptive thresholding with different parameters. These are optimised through Otsu's method for small areas within the image. (b) has a neighbourhood of 51 pixels, (c) has a neighbourhood of 101 pixels. A problem with this method shown is by the noise in the background. The backgrounds have stochastic noise which is small when compared to signals from the mouse, but large when no other signal is seen. (d) shows the coronal slice of the same MR image. (e) has band thresholding with 3 bands. (f) is adaptive thresholding with a neighbourhood of 51 pixels. Both (e) and (f) can segment out the background of the image. However, very few of the segments other than background have a high confidence in either image.

1.9.3 After Otsu

Otsu's method is one of the foremost thresholding techniques as it is computationally simple and reliable. It does however rely on an exhaustive search to maximise the thresholds, which can become very computationally heavy on large data sets. Because of this there has been research carried out into reducing the computational time while maintaining reasonable thresholding (65). This indicates that if Otsu's method is unable to threshold the data, then one of the methods based on Out's method (63) would not be any more successful.

1.10 Region Based Methods

Region-based segmentation methods examine pixels in an image and form disjointed regions by merging neighbouring pixels with predefined properties (66). Two of the most commonly-used region-based methods are region growing and watershed segmentation.

1.10.1 Region Growing

Region growing is a common region-based segmentation method (67), which starts with at least one “seed” voxel that belongs to the structure of interest. Neighbours of the seed are checked and those satisfying the similarity criteria are added to the region (68). This process iterates until no more pixels can be added to the region. The primary disadvantage of region growing method is the partial volume effect, which limits the accuracy of MR image segmentation. The partial volume effect blurs the intensity distinction between different tissue classes at the border of the two tissues types, because one voxel may represent more than one tissue type. This means that borders that are not well defined (or are too small) may merge together, causing the region to grow as one area across both tissue types.

1.10.2 Watershed Segmentation

Watershed segmentation starts seed points from the lowest intensity voxels, then grows regions from them (69). Whenever one region meets another region, a watershed line is drawn on the boundary. Once all the pixels have been assigned a region, the watershed lines make up the segmentation. This technique has been used successfully to automatically segment out regions of tumour in brain (70-74). One disadvantage is that watershed segmentation methods usually suffer from over-segmentation leading to overly accurate segmentation and false boundary creation, which has led Bieniecki (2004) to develop alternative strategies in order to overcome this (69).

1.11 Multi-Atlas Segmentation

Atlas models have a predefined segmentation that they use in order to segment any new scans. The first atlas-based algorithm was introduced to register (or superimpose) different brain images on to each other in order to compare them (75). Subsequently, atlas-based segmentation approaches have been widely used for guiding brain tissue segmentation. In general, this method of segmentation includes three steps: firstly, an affine registration (i.e. a deformation that preserves points, straight lines and planes) brings the atlas and the image into global correspondence; secondly, a template for the organs is provided through the seeding of synthetic organs which are normally defined through the input of a database of pre-segmented scans into the atlas; thirdly, the synthetic organs are mapped onto the image by deforming the seeded atlas using optical flow principles, such as the detection of edges and surface matching (76).

Atlases can also provide probabilistic information about tissue models. Dempster *et al* (1977) employed a probabilistic tissue model and used an Expectation Maximization (EM) method (77) to segment by modifying a brain atlas (78, 79), to improve on current atlas techniques. Since existing atlases are usually constructed by equally averaging pre-segmented images in a population, these processing methods reduce local inter-subject structural variability and lead to lower segmentation guidance capability, increasing the accuracy of atlas systems (76).

Building a precise atlas is the key to atlas-based methods as the efficacy and practicability of these methods is very dependent on the atlas itself. In the case of pancreatic tumours, the spatial positions of the surrounding organs can be changed due to the invasive tumour, adding error to the atlas and making the segmentation method worse.

Multi-atlas segmentation is a process which allows a library of scans with pre-segmented object to be used in automatic segmentation of a novel scan. A novel scan is first compared to all scans within the library and the N scans that match the novel scan most closely with the least amount

of registration functions are used to build probability maps of the requested object. Once the probability clouds have been generated, all voxels with a probability higher than a set threshold are classified as the object. This is then repeated for each object to be obtained (80), see Figure 1-13.

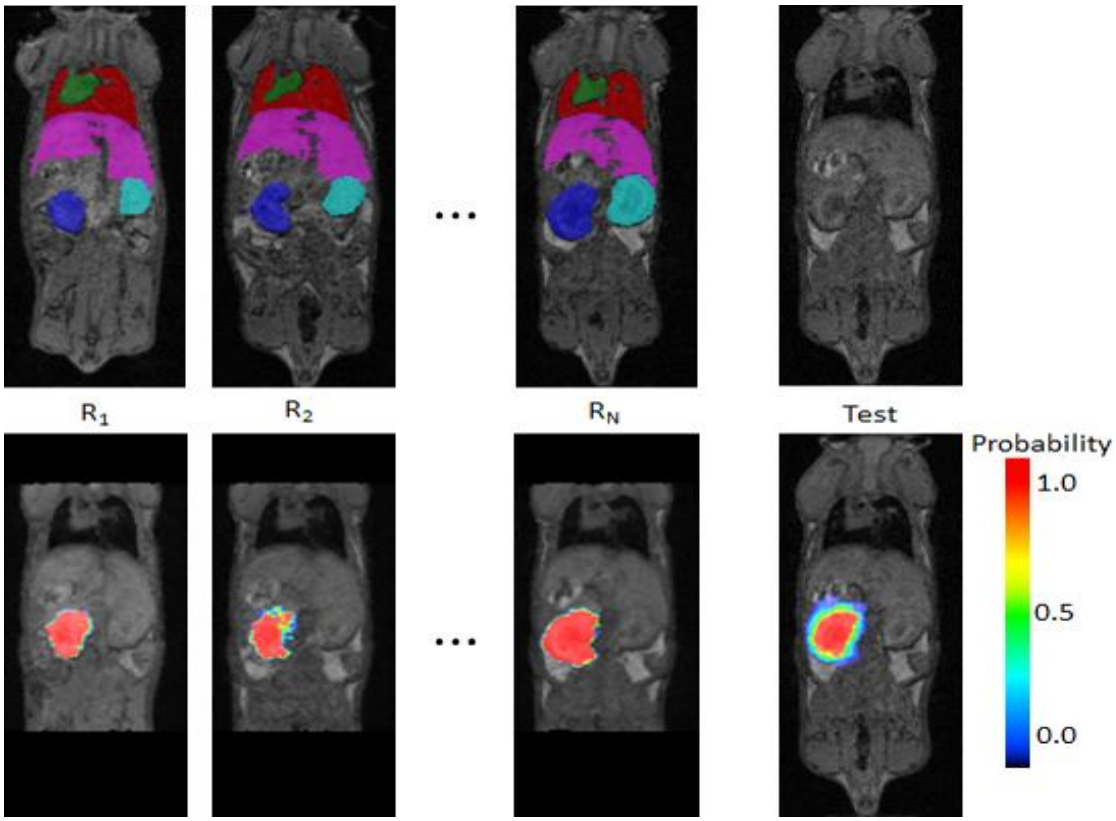


Figure 1-13: An example of Invicro's Whole Body Atlas. A library of scans, R_N, is used to evaluate a novel scan, $Test$, and produce the probability of organ location (81).

1.12 Bounding Area

The bounding area is the area that is classified by the machine. This area must be defined large enough that no objected that should be classified are missed but small enough to reduce computational time.

1.13 Normalisation

In order for two MRI scans to be compared they must undergo normalisation (82). This is due to the fact that an MRI scan does not have any units and is only a measure of intensity, meaning

that the scale can change depending on the amount of signal in the field of view and the value of a tissue will change dependant on what other tissues are in the field of view.

1.14 Object detection (Channelized Hotelling Observer)

Humans are very good observers - for example, visualising facial features in clouds. Computers can also be good at this process, when object shape or location are known. The method of *Channelized Hotelling Observer (CHO)* can be used to find an object of expected shape or position within a novel image (83). The CHO can see the object with high accuracy. This is demonstrated in the receiver-operating characteristic (ROC) curve, which compares the sensitivity (proportion of positives correctly identified) and 1 - specificity (proportion of negatives correctly identified) to show the accuracy of the method, and calculate AUC. This method, however, requires quite a large amount of prior knowledge, i.e. either the location or the shape of sought object. Therefore, further analysis is required before this can be implemented.

CHO is a form of side-scrolling object detection. It can be used in two separate methods. Firstly, if the location of the object is known, the CHO is able to segment the object with zero prior knowledge of its shape. Secondly, if the shape of an object is known then the CHO is able to search through the image for objects that match the shape. This method is often used for the detection of brain tumours as the structure of the brain is relatively stable and tumours can often have a relatively defined shape (84).

1.15 Edge Detection

Edge detection is a process in which changes in the image intensity are detected, typically with a minimum run-length of voxels to exclude single-voxel variation.

Canny Edge Detection is a well-established and robust method (85). Canny edge detection can be seen as 5 steps:

1. Apply Gaussian filter to smooth the image in order to remove the noise. For a Gaussian filter kernel size of $(2k+1) * (2k+1)$, this can be defined as

$$H_{ij} = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(i-(k+1))^2 + \left(j-(k+1)\right)^2}{2\sigma^2}\right) \quad 1\text{-}18$$

$$1 \leq i, j \leq (2k+1)$$

2. Find the intensity gradients of the image. The first derivative of the image in the horizontal (G_x) and vertical (G_y) direction and is used to calculate

$$G = \sqrt{G_x^2 + G_y^2} \quad 1\text{-}19$$

$$\Theta = \arctan\left(G_y, G_x\right) \quad 1\text{-}20$$

3. Apply non-maximum suppression to get rid of spurious response to edge detection, edge thinning. Each voxel intensity is compared to the intensity of the voxel in the negative and positive gradient direction. If the current voxel does not have the highest intensity of these neighbours it is supressed to 0, else it is preserved. This method is applied in each direction.
4. Apply double threshold to determine potential edges. To account for noise in the image two thresholds are defined, high and low. Any gradient intensity greater than the high threshold is classified as a strong gradient. Any gradient intensity less than the low threshold is suppressed to 0. Any other gradient intensities are classified as weak gradients.
5. Track edge by hysteresis. Finally, any weak gradient is supressed if one of it's 8 neighbour pixels (defined along the surfaces) is not a strong gradient.

Edge detection as a stand-alone classification method is not appropriate for pancreatic tumour segmentation as the tumour is typically heterogeneous, similar to the intestines that often neighbour the tumour.

1.16 Gradient Analysis

Gradient analysis finds the gradient magnitude and direction of an image. It does this by using a 3D Sobel method. A neighbourhood around each pixel is selected and the Sobel operators are applied. A 2D example:

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A, G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A \quad \text{1-21}$$

This allows for an approximation of the derivative in the x, y, and z directions. From this, the gradient magnitude can be calculated and give the x, y, and z vectors for each of the voxels. This can be used as a form of edge detection and therefore if used within a region of interest can give information on the homogeneity of the region, see Figure 1-14. The gradient can also be calculated using a navigational system and give the azimuth, elevation and magnitude.

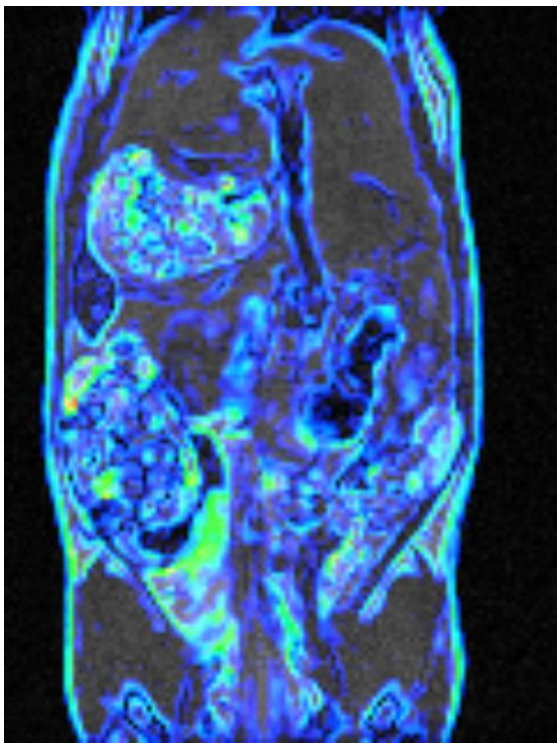


Figure 1-14: T1 FLASH image of a KPC mouse with the gradient information overlaid. The Gradient data shows the edges of the animal and structural information within the animal.

Using these two systems, the magnitude of the gradient and the direction of flow can be calculated. Finally, multiscale gradient analysis can be used, in which case the gradient at different diffusion levels is calculated to be able to define larger more intense gradient changes.

1.17 Fractional Anisotropy

Fractional Anisotropy is a method that is used to study the flow of neurons and similar structures in diffusion tensor imaging (DTI), an emergent form of MR Imaging (86). A scalar value between 1 and 0 is calculated with 0 having an unrestricted float (i.e. free floating) and 1 having flow in only along one axis.

This method has proved to be exceptionally powerful on DTI images (86) but would not provide any information if applied to a T1 flash image. Fractional Anisotropy however, could be applied to the gradient flow images to more clearly identify areas of high heterogeneity.

The classical calculation for fractional anisotropy is:

$$FA = \sqrt{\frac{3}{2}}\frac{\sqrt{\left(\lambda_1 - \hat{\lambda}\right)^2 + \left(\lambda_2 - \hat{\lambda}\right)^2 + \left(\lambda_3 - \hat{\lambda}\right)^2}}{\sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}} \quad 1\text{-}22$$

Where the Eigen vectors ($\lambda_1, \lambda_2, \lambda_3$) of the diffusion tensor image are used. This can then be analysed to get the equation:

$$FA = \sqrt{\frac{1}{2}\left(3 - \frac{1}{trace(R^2)}\right)} \quad 1\text{-}23$$

Where R is the normalised diffusion tensor:

$$R = \frac{D}{trace(D)} \quad 1\text{-}24$$

For D the diffusion tensor.

1.18 Grey Level Co-Occurrence Matrix

A grey-level co-occurrence matrix (GLCM) is a matrix that is defined over an image to be the distribution of co-occurring voxel values at a given offset (39). The offset (Δx,Δy) is a position

operator that can be applied to any voxel in the image, e.g. (1,1) could indicate "one right, one down":

$$\text{Image} = \begin{bmatrix} 0 & 1 & 1 \\ 2 & 1 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \text{1-25}$$

An image with p different voxel values will produce a p*p co-occurrence matrix for the given offset. The $(i,j)^{th}$ value of the co-occurrence matrix gives the number of times in the image that the i^{th} and j^{th} voxel values occur in the relation given by the offset. The co-occurrence matrix C for an image with p different voxel values sized $n * m$ is defined explicitly as:

$$C_{\Delta x,\Delta y}(i,j) = \sum_{x=1}^{n}\sum_{y=1}^{m}\begin{cases}1, & \text{if } I(x,y) = i \text{ and } I(x+\Delta x, y+\Delta y) = j \\ 0, & \text{otherwise}\end{cases} \quad \text{1-26}$$

Once the GLCM is calculated, different metrics (such as contrast, correlation, inverse difference, and maximum probability), can be applied to it to look for patterns or connections in the texture of an image.

1.19 Grey Level Run Length Matrix

The grey level run length matrix (GLRLM) takes one central voxel and a window around it. This area is then analysed to find the frequency of occurring voxel intensities at specific orientations. The result is that for every voxel in the image there is one matrix for each orientation (41). If for instance four orientations are considered, 0, 45, 90, 135 degree angles (Θ) the image before being analysed by the GLRLM is converted to have 32 grey levels only. This means that the GLRLM will have a size of 32 by 32. N_g is the number of discreet intensity values (32). N_r is the number of discreet run lengths. N_p is the number of voxels in the window. Therefore, the number of runs in the window along angle Θ is:

$$N_r(\Theta) - \sum_{i=1}^{N_g}\sum_{j=1}^{N_r} P(i,j|\Theta) \quad \text{1-27}$$

$$1 \leq N_r(\Theta) \leq N_p$$

Where $P(i,j|\Theta)$ is the run length at each angle. And

$$p(i,j|\Theta) = \frac{P(i,j|\Theta)}{N_r(\Theta)} \quad 1\text{-}28$$

Multiple features can then be calculated on each voxel's 4 GLRLMs.

1.20 Machine Learning

Machine learning provides an effective way to automate image analysis as the machine can learn complex relationships or patterns (87). Machine learning algorithms can be categorised according to the different principles upon which they are based, some of which are covered below.

1.20.1 Supervised Learning

In supervised learning, each sample consists of two parts: the first is input observations or features and the second is output observations or labels (88). Usually the input observations are causes and the output observations are effects. The purpose of supervised learning is to deduce a functional relationship from training data that generalises well to testing data. The form of the relationship is a set of equations and numerical coefficients or weights, e.g. a classification algorithm is a representative method of supervised learning that could be used to identify the clusters within data but would require prior knowledge (89).

1.20.2 Unsupervised Learning

In unsupervised learning, there is only one set of observations and there is no label information for each sample (90). Usually, features are produced by a set of unobserved or latent variables. The main purpose of unsupervised learning is to discover relationships between samples or to reveal the latent variables behind the observations. A clustering algorithm is a representative method of unsupervised learning, but this type would require no prior knowledge (91).

1.20.3 Semi-supervised Learning

Semi-supervised learning combines supervised and unsupervised learning. It utilises both labelled data and unlabelled data during the training process (92). Semi-supervised learning

algorithms were developed mainly because the labelling of data is time consuming and therefore expensive (93). Li *et al* (2001) has used semi-supervised learning to develop a stochastic graph labelling technique to search medical image databases (94).

1.20.4 K-means

K-means clustering is a method of vector quantitation. It aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. The problem is computationally difficult; however, there are efficient heuristic algorithms that are commonly employed in image analysis and converge quickly to a local optimum (95).

1.20.5 Fuzzy C-means Clustering Models

Fuzzy C-means clustering (FCM) is a method which divides one group of data into two or more clusters. This method (96) is frequently used in pattern recognition. The algorithm works by assigning membership to each data point corresponding to each cluster centre based on distance between the cluster and the data point. The nearer the data point is to the cluster centre, the more possible is its membership within it. Some advantages of FCM algorithm include giving the best result for overlapped data sets and giving comparatively better results than a **k**-means algorithm. A data point can be assigned to multiple cluster centres, unlike **k**-means where the data point must exclusively belong to one cluster centre. Since FCM is an iterative algorithm, it is a computationally heavy method.

1.20.6 Support Vector Machine Models

Support vector machine (SVM) is treated as a parametrically kernel based method, defined as

$$K(x_i, y_j) = e^{\frac{-1}{2\sigma^2}(x_i - x_j)^2} \qquad 1\text{-}29$$

a pattern analysis technique, to deal with supervised classification problems (97). SVM has been widely used for tumour segmentation (98-101).

A brain tumour segmentation method exploring one-class SVM has been proposed (98). This method has the ability to learn the nonlinear distribution of the image data without prior knowledge, via the automatic process of SVM parameter training and an implicit learning kernel. Zhou et al (2005) found that it achieved better segmentation results for the extraction of brain tumours, compared to the fuzzy clustering method, with more tumour identified correctly (98). Some researchers have numerous MRI techniques, including diffusion tensor imaging, to create voxel-wise intensity-based feature vectors, which they classify by SVM (99, 101). This method could not only segment the healthy tissues, but also segment sub-compartments of healthy and tumour regions. A multi-kernel based SVM integrated with a feature selection and a fusion process was proposed by Zhang *et al* (2009) (102) to segment the brain tumour from multi-sequence MR images (102). Compared with traditional single kernel SVM, the results of this method showed a decrease of the total error and improvement of accuracy. A fully automatic method for brain tissue segmentation was proposed by Cai *et al* (2011) (23, 103) , which combined SVM classification using multispectral intensities and textures with subsequent hierarchical regularisation based on conditional random field (CRF) methods, a statistical modelling method that considers neighbouring pixels during classification. Cai's method used a hierarchical approach to add robustness and speed by allowing different levels of regularisation at different stages and had good results. In conclusion, SVM has demonstrated great potential and usefulness in tumour segmentation for MR images. This can be extended from linear classification to non-linear, using the "kernel trick" which needs only inner product between examples, instead of mapping examples in a space. This allows for mapping into higher dimensional problems and allows multiple features to be considered.

A further step from SVM is support vector clustering, where not all the example data is labelled and the model attempts to find natural clustering of the data. This becomes useful as the data libraries become exceedingly large.

1.20.6.1 Generalised Linear Model

The generalised linear model (GLM) is a flexible generalisation of ordinary linear regression that allows for response variables that have error distribution models other than a normal distribution (104). The GLM generalises linear regression by allowing the linear model to be related to the response variable via a link function and by allowing the magnitude of the variance of each measurement to be a function of its predicted value.

1.20.7 K-Nearest Neighbour

K-nearest neighbour (KNN) is a machine-learning technique which uses Euclidean distance to calculate the feature vectors of each object and assign similar clusters as distinct classes (105). Training a KNN is relatively computationally simple and quick. Each object in the training library has the feature vectors calculated and the classes are defined from the meta-data. The result is that when a novel scan needs to be classified the feature vectors are simply calculated and the class which the majority of features is contained in is the class assigned to the novel scan. This method, although computationally cheap, has issues when one class is much larger (magnitudes greater) than the other classes.

1.20.8 Random Forests

A Random Forest is a multitude of decision trees which are assigned random features (106). A decision tree is a simple method of binary choice at each level, in this case the choice of the object's feature-class. If a decision tree is grown too deep, it will start to discern insignificant or erratic patterns. To combat this instability, multiple decision trees can be grown in parallel to a restricted depth, this prevents the model from identifying irregular patterns. However, due to the binary decision making, if trees are grown in this manner and provided all the features to choose from, they will often choose the same features. To account for this issue, each branch of each decision tree is provided with only a subset of features, randomly assigned. An issue arises if all trees are trained on the same data as the trees will be highly correlated to one another. To avoid this issue, boot-strapping is deployed in which the training set is broken up and used

separately. This decreases variants of the model without increasing bias (107). After training a random forest on a library of objects a novel scan's features are classified by all decision trees and the majority vote for which class the object should belong to determines which class the object should be assigned to. This classification can also more directly be outputted as the probability of an object to be in each class.

1.21 Post Processing

Post processing allows for the analysis of data after the machine learning output. Often the direct output from machine learning is quite course and is not what would be observed, post processing allows for the smoothing and further analysis of the machine learning output.

1.21.1 Level Sets

Level sets are a partial differential equation technique in which the user defines a proposed area around an object. The technique then attempts to accurately segment the object. The partial differential equations are made up of 2 constraints, one internal and one external. The internal constraint attempts to keep the object smooth within degree parameters. The external constraint uses a form of edge detection in order to attempt to fit the classification to the image exactly. These two constraints work against each other to try to define the object (33).

1.21.2 Open/Close

Opening is a morphological process of dilation of the erosion of a binary object using a given structure. It is a computationally simple process. Opening removes bright areas and small unconnected objects from an image. For a structure:

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \qquad 1\text{-}30$$

For each voxel with a given class, the surrounding voxels, defined by the size of A,are assigned to the same class if they are in the location of 1 defined by A, with the starting voxel in the centre. This process can have different dimensions of A and can be applied multiple times.

Application multiple times instead of using a larger matrix for A decrease the loss of structural shape (85).

Closing is the morphological process of erosion of the dilation of a binary object using a given structure. It is again a computationally simple process. Closing removes gaps in an image and connects previously unconnected components. Closing is applied using the same technique as opening but with the classes reversed (85).

1.21.3 Geometric Mean

The geometric mean is a morphological smoothing process which takes each voxel and calculates the mean of all 24 neighbouring voxels in the original image and assigns the centre voxel to this mean value in a new image. This method is computationally simple and softens sharp edges (85).

1.21.4 Gaussian Smoothing

Gaussian smoothing is a convolution operator that is used to remove noise and detail from an image. This makes it a useful post processing technique as it is able to smooth a classification, though caution should be used as it can remove large amounts of the class (85).

A n dimensional Gaussian function is defined as:

$$f(x) = e^{-x^T Ax + s^T x} \qquad 1\text{-}31$$

Where $x = \{x_1, \dots x_n\}$ is the column coordinates of n, A is a $n \times n$ positive definite, symmetric matrix and $s = \{s_1, \dots s_n\}$ is the shift vector.

1.21.5 Connected Components

Connected components is a structural process that identifies all distinct components in an image where a connection can be defined along the surface, edge, or vertex of the voxel.

For a given voxel the number of connecting can be evaluated compared to a minimum number, set by the user, and if it is below this threshold the class of the voxel is changed (85).

1.22 Aims

There are 3 core issues associated with the current methods. 1) Owing to the inaccuracy of palpation and operator dependency of ultrasound, when detecting tumour size and disease burden in the study of pancreatic cancer in KPCs, more mice are required per group per study to reach statistical significance. The result of this is more animals must be bred and subsequently culled. 2) On account of the difficulty associated with detection and measuring tumours within a KPC using palpation and ultrasound, many studies recruit mice with larger tumours than the researchers necessarily require for study. The larger-than-necessary tumours lead to studies not being as clinically relevant as they possibly could be, as well as potentially causing more animal suffering. 3) In a study group, scientists commonly dissect animals at each pre-determined time point to measure tumour-burden and then assume uniformity of tumour burden throughout the rest of the animals in the study (108). However, due to the spontaneous nature of the KPC mouse-model this assumption is not always statistically significant. The current method is flawed on account of the existing model coming from uniformly injected mouse models, yet is being applied to mice with spontaneously formed tumours where tumour size and location is highly variable. The effect of this is study groups must be much larger to account for the number of dissections that must be performed owing to the statistical variability inherent in the KPC model. The result is that more mice must be bred and subsequently culled.

1.22.1 An Automatic Computational Model

The aim of this project is to build a computational tool that will automatically segment out the pancreas and any pancreatic tumours from an MRI scan. This 3D computational atlas for mouse models of pancreatic cancer (3D-CAMMP) will provide a reliable and reproducible image analysis method for volume and surface area quantification of the pancreas and pancreatic tumours, reducing effects of user discretion and thus improving the accuracy of the data.

1.22.2 The 3Rs in Animal Research: Replace, Reduce, Refine

Animal research in the UK is regulated by the Home Office under the Animals (Scientific Procedures) Act 1986 and it is a statutory requirement of the project licence holders that the principles of the 3Rs are adhered to: Replacement, Reduction and Refinement. Replacement refers to methods which avoid or replace the use of animals, e.g. developing *in silico* or *in vitro* techniques. Reduction refers to methods that minimise the number of animals used per experiment. Refinement refers to methods that reduce animal suffering and improve welfare. (109). The 3R's areas in which this project aims to have an impact are reduction and refinement.

1.22.2.1 Reduction

As discussed, one of the primary issues with current methods is the large number of mice required per study group to achieve statistical significance, owing to the inaccuracy and/or operator dependency of existing measurement methods. The aim is to combine MR-Imaging and machine learning to provide more accurate and reproducible tumour measurements, allowing researchers to recruit animals to study groups when they have more similar tumour burden. This should in turn lead to lower variability within the group, increasing statistical power and allowing a reduction in group size. Furthermore, the use of MRI and 3D-CAMMP in a longitudinal study allows for direct comparison of tumour growth / treatment curves (using the animal as its own control/baseline). The result of this reduction in biological variability is that fewer animals are required per group to achieve statistical significance.

1.22.2.2 Refinement

As discussed, another issue associated with current methods is the amount of suffering an animal must endure owing to larger-than-necessary disease burdens. The combination of MRI and 3D-CAMMP refines this by allowing scientists to accurately detect smaller tumours that not only are more clinically relevant but also allows experiments to start earlier when tumour burden is lower, and end sooner thus potentially reducing suffering. Reduction in suffering of

individual animals is an important goal as the trade-off in using fewer animals overall is that those animals will undergo more procedures in total and all will reach the endpoint of the experiment (rather than some being culled at early timepoints for dissection). Therefore, the aim must be to move to less severe endpoints in the fewer animals used, which can be achieved through regular imaging to monitor tumour burden and animal welfare.

In terms of the trade-offs of this approach, if palpation is the method being replaced, animals must undergo additional procedures using anaesthesia. However, due to higher disease burden in palpated animals and lack of information as to disease progression, these animals are generally culled based on clinical signs. Ideally, imaged animals would be culled based on them reaching a defined disease burden before experiencing clinical signs. If volumetric ultrasound is the method being replaced, the benefits of a more accurate MRI based method are that they would not need to undergo hair removal and that more accurate determination of disease burden would allow reduction in group size.

All of these benefits would affect the pancreatic cancer research landscape by improving the standardisation of data and have the important 3Rs effect of reducing the number of animals required for each group in a study. In the UK, BCI along with 3 other major UK Institutes have sizeable colonies of pancreatic GEMM's. The estimated overall number of animals used in scientific studies in these 4 institutes in 2018 was about 1410 per annum. Since on average only 20-25 % of these have the right genotype and phenotype, this translates to 7050 animals bred.

Since the model was first published in 2005, it has increasingly appeared in publications 2011 (2), 2012 (3), 2013 (11), 2014 (13), 2015 (24), 2016 (31), 2017 (41), 2018(40) and this trend is expected to continue. About 18 centres worldwide (4 in the UK) are major users of this specific model, but there are many similar GEMMs of pancreatic cancer and their use is rapidly expanding. The atlas tool could be developed to be applied to all of these models.

In KPC studies it is common to use 10-12 animals per group (110-113). This is doubled if there is a control or comparative treatment. In order to reduce these numbers, we need to extract the full potential of longitudinal imaging studies, by developing tools that provide robust quantitative data at differing time-points. Up to a 50 % reduction in the numbers of animals used could be made for every study converted from euthanasia plus dissection to a longitudinal study. In addition, being able to size-match tumours between mice with more confidence would certainly improve results and may have the effect of being able to reduce group number. As some studies are already longitudinal, a calculation can be made based on 60 % reduction in KPC. Applied to the major UK users of the model, this would translate to 4,320 fewer KPCs bred p.a. (17,280 animals including litter mates who do not have the correct genotype). As is evident, this method would greatly decrease the number of animals needing to be culled per study, as well as the overall cost associated with breeding KPCs. Including breeding, maintenance, imaging, and staff it is estimated to cost approximately £700 to produce one KPC mouse ready to go on study. If a 60% reduction in KPCs were to be achieved 4,320 fewer animals p.a. would need to be bred and culled, amounting to in excess of £3 million in research costs.

1.22.3 The Strategy

The primary aim of this project is the development of a mathematical model that can automatically detect and segment the mouse pancreas as well as any pancreatic tumours. This will be addressed through a combination of multiple image analysis techniques including thresholding, texture analysis, object detection, edge and region segmentation, atlas segmentation, and machine learning. Through these techniques, unnecessary information will be removed, the area of analysis reduced, the pancreas isolated, classified as either healthy or unhealthy and unhealthy pancreas evaluated to identify tumour location and volume. The final pipeline of steps carried out (see Figure 2-1) to achieve this from acquiring the image through to providing the tumour volume is described in section 2.1 below.

1.22.4 Image Segmentation Ground Truth / Gold Standard

Ground truth will be used to measure 3D-CAMMP's performance, specifically the program's sensitivity, specificity, and exact location. Ground truth was defined by having each image in the library segmented independently by 3 image analysis experts, with 35 years of experience between them. A round-table meeting was subsequently held to discuss each image and decide on a final segmentation. This final segmentation is taken as ground truth with an average of 10 hours of expert time spent per image. A comparison of an image segmented by an image analysis expert and an inexperienced user is shown in Figure 1-15. This work demonstrates that with the help of 3D-CAMMP, inexperienced users with very little training are able to obtain results similar to that of an image analysis expert.

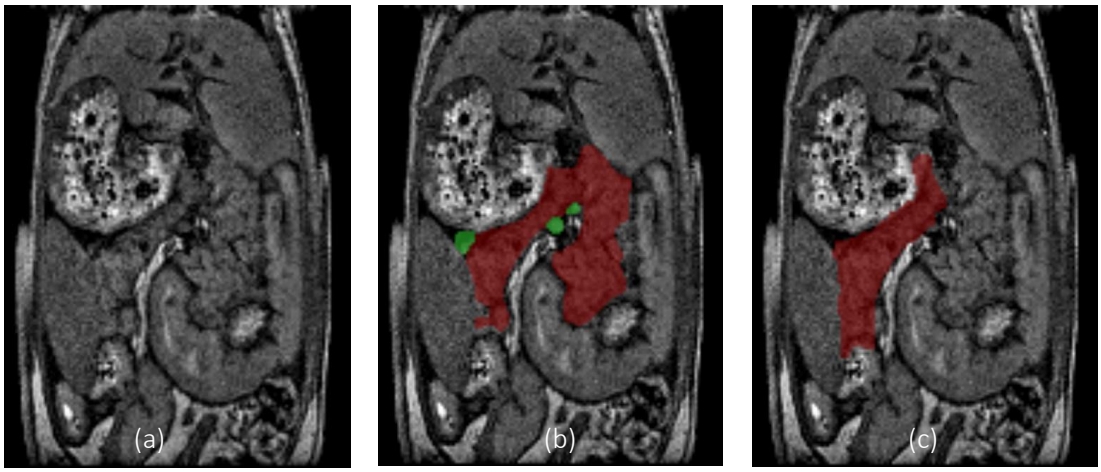


Figure 1-15: (a) shows an MRI scan of a KPC mouse at 127 days old. (b) shows the pancreas (green) and pancreatic tumour (red) manually segmented by an expert user in 45 minutes. (c) shows the segmentation of the pancreas (green) and pancreatic tumours (red) by an inexperienced user in 4 hours.

2 Methods

2.1 3D Computational Atlas for Mouse Models of Pancreatic Cancer

3D Computational Atlas for Mouse Models of Pancreatic Cancer (3D-CAMMP) is capable of taking a scan extracted from the Bruker ICON 1 T MRI and automatically segmenting any pancreatic tumours along with the pancreas, liver, stomach, spleen, hepatic portal vein, gallbladder and the left and right kidney. It will also output the volume of any pancreatic tumours. 3D-CAMMP's major steps are outlined in this section and is implemented through MATLAB and the script ThreeDCAMMP.*m, Appendix 1: 6.1,* that calls the functions defined in the following sections, see Figure 2-1.

2.1.1 Animal Handling

To perform the MRI examination, the mouse is placed into an induction chamber with isoflurane at 4% in oxygen at 1.5L/min. The animal is placed in a prone position on the MRI bed (Bruker BioSpin MRI, Germany). The bed is equipped with isoflurane anaesthesia, water-heating, and respiration monitoring. A solenoid whole body RF coil is placed over the animal (Aspect Imaging, Israel). The bed is inserted into the 1 T permanent magnet (Aspect Imaging, Israel) coupled to the ICON MRI console controlled by Paravision 6.0 software (Bruker). The respiration is controlled to 30-45 bpm (SA Instruments, USA). Two images are acquired on the MRI. Firstly, a localiser T1 FLASH scan with 3 slices is acquire for identification of animal position. Secondly, a T1 FLASH isotropic scan is performed. The data is then reconstructed using standard Bruker reconstruction.

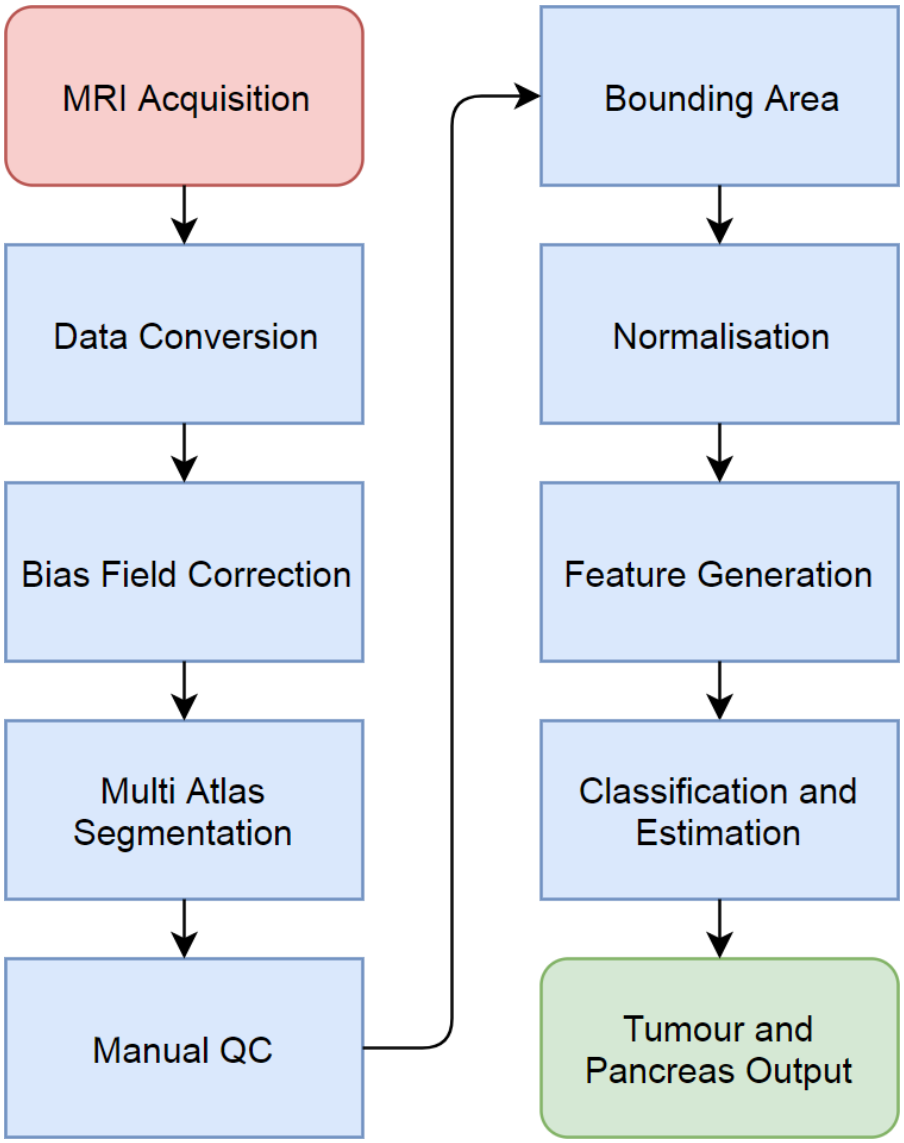


Figure 2-1: Flowchart of 3D-CAMMP for tumour and pancreas segmentation.

2.1.2 MRI Protocols

3D isotropic T1 FLASH scans are performed with parameters outlined in Table 2. This image protocol has a scan time of 21 minutes without respiratory gating and approximately 27 minutes with the gating.

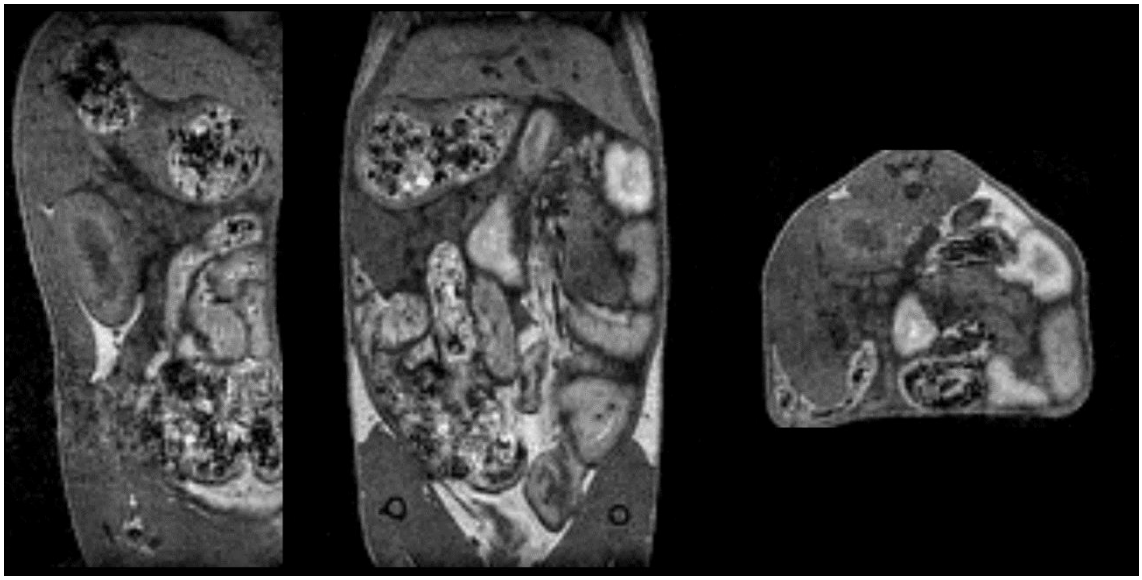


Figure 2-2: A T1 FLASH 3D isotropic scan of a KPC mouse with a pancreatic tumour. The images are in order sagittal, coronal and transverse plane. This image shows the anatomy of the animal well.

Table 2: 3D Isotropic T1 FLASH protocol parameters

Parameter	Value
TR	40ms
TE	6.6ms
Flip angle	40^{O}
Oversampling	3
Slices	1
Slice orientation	Coronal
Read orientation	Ro-Cd
Slice thickness	20mm
Image size	160x120x80
Flied of view	40x30x20mm
Dummy scans	13
Dummy duration	520ms
Segments =	1
Segmentation method	sequential
Dimension	3D
Resolution	0.25x0.25x0.25mm
Anti-aliasing	1x1x1
Slice gap mode	non-contiguous
Slice gap	0mm
Slice distance	20mm
Bandwidth	25kHz
Interpellation	Read 1
Phase	1
Slice	1
Partial-FT	Read 1

2.1.3 Data Conversion

The scan data, obtained through the protocol defined in section 2.1.2 is then converted from Bruker format to '.zraw' using the script *runBMC.vqs*, *Appendix 1: 6.2*, which stands for 'run Bruker MRI Conversion' and is a piece of code designed to run in VivoQuant (Invicro, USA). Firstly, the script looks in a designated folder for any sub folders that start with '201'. This is how the data folders are designated from the MRI after extracting the zip file (year, month, day, i.e. 20180703). Once it has identified the folders then for each folder it looks inside the sub folders, of which there will be one for each scan performed. Any scan with the label 'Localizer' is ignored. Each of the scans is then loaded into VivoQuant and the user is asked to define the animals age in days, the type of model (KPC, KP or WT) and if there are believed to be any tumours. The script asks if the data needs to be reoriented and any such flips are performed.

The script, *runBMC.vqs*, then performs a bias field correction using an iterative Otsu method with a down sample factor of 2, max iteration of 50x50x50, control points 2x2x3, a spline order of 3 and a convergence threshold of 0.5. Finally, the data is save out in '.zraw' format into an output folder. This script allows for quick application of multiple pre-processing techniques including bias filed correction to reduce the corruption effect from the bias field of the MRI and data conversion to make it easier for MATLAB to read. This is implemented through the MATLAB script *runBMC.m, Appendix 1: 6.2.*

2.1.4 Multi-atlas Segmentation

The multi-atlas segmentation tool developed by Invicro is implemented through MATLAB (2018). A reference list is generated from all images that have pass the QC stage of the pipeline. This allows for the model to constantly update its own library. A separate settings file is used and can be updated, for example increasing the number of scans that will be used (default 6), which regions of interest to consider (default is kidneys, spleen, stomach, liver, gall bladder and hepatic portal vein, referred to as Other Organs). The Whole Body Atlas (WBA) is then automatically implemented through the command window. The WBA then stores these organ segmentations

as regions of interest (ROI). The organ probability according to the WBA can also be stored and is used to generate Other Organs Probability Cloud, combining all the organs classified by the WBA. This is implemented through the MATLAB script *MAS_Tool.vqs, Appendix 1: 6.3.*

2.1.5 Move Whole Body Atlas Output Data

The ROIs generated by the WBA are located within a folder and are named 'average_BestN-*-affine-deform.rmha' where '*' is the name of the subject and its age. These ROIs were collected, moved and renamed to make it easier for the pipeline and any user checks. This is implemented through the MATLAB script *runArea, Appendix 1: 6.1.* The script identifies the input location, creates an output folder if one does not already exist, and identifies all the folders within the input folder. Then for each of these folders it finds the correct file and saves it to the output location under the new name of the subject and its age. In each of these folders is also the probability map for all the WBA segmented organs (kidneys, spleen, stomach, hepatic portal vein, gallbladder and liver). The files are similarly named 'average_BestN-*-affine-deform.mhd' where '*' is the name of the subject and its age. Therefore, at the same time as the ROIs are being moved and renamed, all the other organs probabilities are combined, moved and renamed in the same folder as the ROIs, this probability image is defined as 'Other Organ Probability Cloud'.

2.1.6 ROI and Other Combine

The Other Organ Probability Cloud is then combined with the confirmed ROIs from the manual QC to give an updated other organ probability cloud. The is implemented through the MATLAB script *runCombineOther*, *Appendix 1: 6.1*. For each scan's other probability, the corresponding ROIs is converted to a binary image and then dilated and smoothed. The max value for each voxel is then taken from the dilated ROIs or the other probability. Finally, the new probability cloud is saved out.

2.1.7 Probability Clouds

Using the library of completed ROIs the probability of an organ's position can be created, similar to that of a multi-atlas segmentation method. By finding the centroids of the organs segmented by the WBA and then calculating the mean of these spatial positions a point, defined as the 'Centre of Organs', is calculated. The distance for each segmented organ from Centre of Organs is calculated and the mean sum of these distances is then used as a scaling factor for the transformation of the tumour ROI into a normalised space. The scaling factor is defined as:

$$\lambda = \frac{\varepsilon}{\tau} \qquad 2\text{-}1$$

Where λ is the scaling factor, ε is the mean value of all distances and τ is the scaling factor of the current scan. By doing this to all the scans in the completed library a probability map for the tumour location is created in normalised space.

When a novel scan is introduced the scaling factor is calculated in the same way but 1 over the scaling factor is applied to transform the normalised tumour probability cloud onto the novel scan. This is implemented through the MATLAB script.

These two methods are duplicated for the pancreas ROI to produce and apply a pancreas probability cloud in the same way.

2.1.8 Bounding Area

The bounding area is calculated by taking a novel scan with the tumour and pancreas probability clouds applied to it, as described in section 2.1.5. The area that is covered by these probability clouds is defined as the area of interest and is used as the bounding area for the rest of the pipeline. This is implemented through the MATLAB script *runArea, Appendix 1: 6.1.* The output directory is identified or created if it does not already exist, each ROI file in the input folder is identified. The 'mean magnitude', 'tumour cloud' and 'pancreas clouds' are loaded into the workspace. For each ROI file in the input folder, the ROIs are loaded into the workspace. If the mouse model is a KPC then the *pancreasKPC* cloud is used along with the *tumour cloud*, if not the

pancreasKP cloud is used and the tumour cloud is set to an empty matrix. The scan is then loaded in from the correct folder. The scan is reshaped into 1D and the Otsu's threshold is found. This threshold is applied to the original scan, with everything below the threshold removed to create a new scan, call it $scanD$. A dilation matrix is then defined and used to dilate $scanD$. Next the ROI file from the WBA is used. The centroid of each segmented organ is calculated and the mean of these spatial positions gives the 'Centre of Organs' for this animal. The mean value of all these distance is the used to calculate the scaling factor for the normalised cloud translation by using the equation and the 'mean magnitude'. The scan limits are also calculated to ensure that the translation is performed from the centre of the scan not the top left forward corner. The translation and warps can now be applied to the probability clouds to map them onto the novel scan. The area covered by the tumour and pancreas clouds is taken as the bounding area and the area is save as an ROI file.

2.1.9 Normalisation

Each scan is normalised using the standard score method (114) defined as:

$$X_n = \frac{X - \mu}{\sigma} \qquad 2\text{-}2$$

Where X is the scan, μ is the mean on the scan and σ is the standard deviation of the scan. This is implemented through the MATLAB script *runArea, Appendix 1: 6.1.* The values in the scan that are greater than the Otsu's threshold calculated are taken and the equation 2-2 is used on these values. The values are then transformed back into the correct shape of scan and the scan is saved out, now normalised.

2.1.10 Feature Generation

Features are generated for each voxel within the bounding area. For certain features, such as the grey level co-occurrence matrix, a neighbourhood is needed round the central voxel. The window size used for the neighbourhood is 11 voxels. All features were generated through *runFeatures, Appendix 1: 6.1.* Firstly, the input folder is identified, the names of the different files are

generated, i.e. scan, ROIs, probability clouds, and the output folder is defined. Next a dilation matrix is defined and the 'mean magnitude' is loaded into the workspace. For each scan the following features are generated with a window size of 11 voxels. The scan is loaded into the workspace and padded with the window size, this assures no voxel assessed will touch an edge of the imaged.

2.1.10.1 Grey Level Co-Occurrence Matrix

For each voxel in the bounding area a 2D GLCM is calculated (xy plane). The image is converted to have 32 distinct grey levels, a window size of 11 voxels and an offset of 1 left and 2 down is used. The output produced is a feature matrix A where each row is a voxel in the bounding area and each column in a different GLCM feature. The formula used to calculate the GLCM features can be found in *Appendix 1: 6.1.*

2.1.10.2 Grey Level Run Length Matrix

The image is converted to have 32 distinct grey levels and the 2D GLRLM is calculated for each voxel in the bounding area. A window size of 11 is used along with, multiple different offsets (0^{O}, 45^{O}, 90^{O} and 135^{O}) and the total count in each of the offsets is collected. Each of these offsets is used to calculate a set of GLRLM features. These features are merged with the feature matrix A. The definition of these features can be found in *Appendix 1: 6.1*.

2.1.10.3 Statistical Features

The statistical features are calculated on a 3D area around each voxel with window size of 11 voxels and are mean, mode, median, variance, standard deviation, kurtosis, skewness, and intensity. These are calculated for each voxel and merged with the feature matrix A. This is done through *runFeatures.m,*.

2.1.10.4 Gradient Features

The gradients of the image were calculated, including the magnitude, azimuth, elevation, x, y, and z component. These were used to calculate the fractional anisotropy and to calculate the gradient at different resolutions by using the equation

$$G_{xy}^{n} = \frac{\sqrt{G_x^2 + G_y^2}}{2^n - 1} \qquad 2\text{-}3$$

For $n = 1,2,\ldots,5$. These are calculated for each voxel and merged with the matrix A. This is done through *runFeatures, Appendix 1: 6.1.*

2.1.10.5 Spatial Features

Since the Centre of Organs is calculated in section 2.1.8 this can be compared to the location of every voxel considered. This is defined for the x coordinate by

$$L_x = \frac{C_x - V_x}{\lambda} \qquad 2\text{-}4$$

Where C_x is the Centre of Organs x component, V_X is the voxel's x component and λ is the scaling factor defined in equation 2-1. The calculation for the y and z component follow directly from equation 2-4.These are calculated for each voxel and merged with the matrix A. This is done through *runFeatures, Appendix 1: 6.1.*

2.1.10.6 Probability Features

At each voxel the value of the probability clouds (tumour, pancreas and other) is taken and used as a feature. These are calculated for each voxel and merged with the matrix A. This is done through *runFeatures.m, Appendix 1: 6.1.*

2.1.11 Machine Learning

The feature matrix A for each novel scan is passed into the pre trained random forest using runModel.m, *Appendix 1: 6.1*. The output of this random forest is three probability maps (tumour, pancreas and not tumour not pancreas).

Post Processing

The probability vectors for each class are passed from the machine learning model into the post processing script. Any value greater than or equal to 0.3 on the tumour probability vector is designated as tumour, this new vector is then resized to the proportions of the image data. Next this new image undergoes gaussian smoothing with a gaussian filter using a structure defined by:

Then the resulting image undergoes connective component clustering where a connection is defined as ‘any voxel with a neighbour touching any surface, edge, or vertex’. Any object that is formed of less than 16 voxels is removed.

The same process as above is applied to the pancreas probability vector with the same settings.

The other organs, as defined by the multi-atlas segmentation, are then applied to the tumour segmentation image over-writing any voxels that are defined as other organs. This new image is then combined with the pancreas segmentation, similarly with any overlapping voxels deleted from the pancreas segmentation.

This final segmentation image is then saved into the folder ‘classifications’. The normalised scan image is also saved into this same folder. The volume of the tumour and pancreas is calculated from the segmentation image and a CSV file is created with the tumour and pancreas volumes inserted in voxel and millimetres cubed. This is all done through the script *runModel, Appendix 1: 6.1*.

3 Results

3.1 MRI Protocols

The optimum MRI protocol needed to be identified that would provide the relevant data for the machine learning algorithms in a reasonable amount of time. A relatively long scan time was required to acquire the high feature detail that is required. A T1 FLASH scan (see Figure 3-1) provides more texture data of a subject than a T2 RARE scan (see Figure 3-1) and running the model on T1 data only instead of T1 and T2 data had no significant effect on the performance of 3D-CAMMP. However, this did almost halve the time taken for data acquisition for the study.

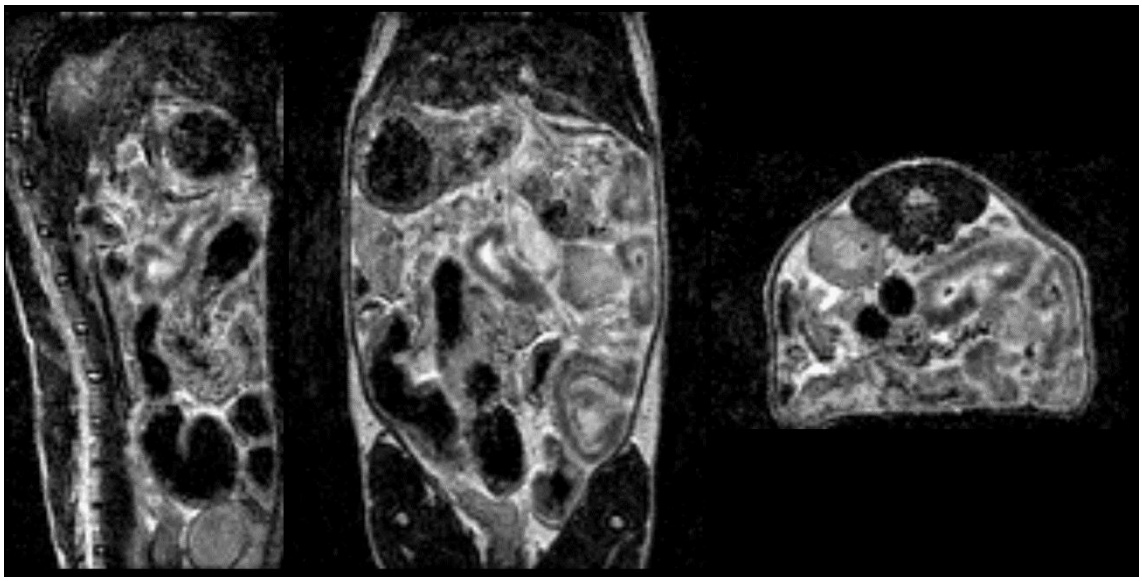


Figure 3-1 A T2 RARE 3D isotropic scan of a KPC mouse with a pancreatic tumour. The images are in order sagittal, coronal and transverse plane. This scan shows the pathology of the animal.

Parameter	Value
TR	1500ms
TE	84ms
Averages	1
Repetitions	1
Echo Spacing	16.8ms
Rare Factor	12
Slices	1
Slice Orientation	Coronal
Read Orientation	Ro-Cd
Slice Thickness	20mm
Image Size	160x120x80

Flied Of View	40x30x20mm
Extraction Angle	90°
Refocusing Angle	180°
Dummy Scans	1
Dummy Duration	1500ms
Dimension	3D
Resolution	0.25x0.25x0.25mm
Anti-Aliasing	1x1x1.1
Slice Gap Mode	non-contiguous
Slice Gap	0mm
Slice Distance	20mm
Bandwidth	25kHz
Interpellation	Read 1
Phase	1
Slice	1
Partial-FT	Read 1

3.2 Bias Field Correction

Bias field correction is an important step in any MRI image analysis that is often overlooked in pre-clinical work. Some of the effects of bias field correction are shown in Figure 3-3 where the liver is more homogeneous after the correction and Figure 3-4 with the changes in overall histogram of the image. The changes are slight and seem insignificant when one image is considered. However, when multiple images are compared, Bias Field Correction is required in order to be able to compare the scans accurately. As shown in Figure 3-2 if the variables of the bias field correction are selected too severely they can drastically change the image. For this reason the parameters; down sample factor of 2, max iteration of 50x50x50, control points 2x2x3, a spline order of 3 and a convergence threshold of 0.2 were chosen. The direct effects of bias field correction are relatively small. However, this small effect leads to large changes in 3D-CAMMPs overall performance as it cascades through all features used to calculate the tumour and pancreas location resulting in a reduction of the DSC of 0.06.

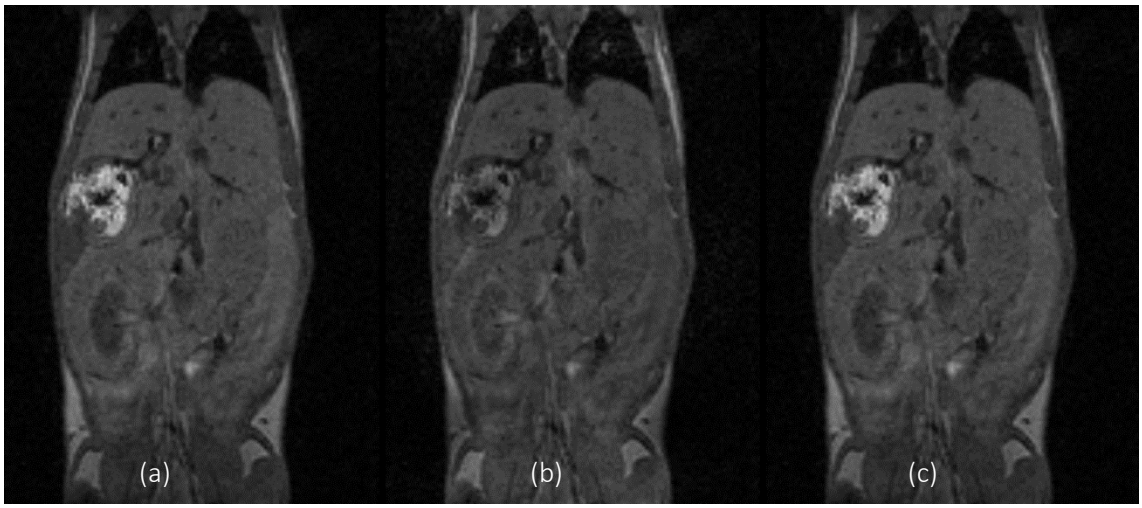


Figure 3-2: Three images showing (a) the original image , (b) an image with bias field correction variables set too strong and (c) the optimal bias field correction image settings . Note that (b) has lost contrast and has magnified some to the field instabilities, as shown in the fat around the legs. However, application of optimised correction in (c) avoids this while also reducing fluctuations within the liver, where the signal should be reasonably consistent.

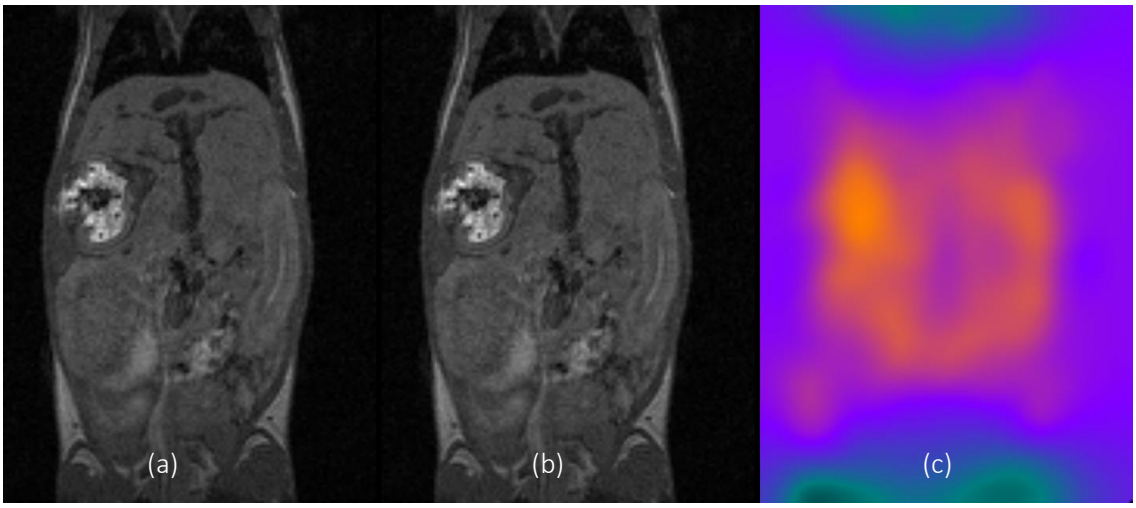


Figure 3-3: (a) Image of a mouse before bias field correction. (b) Image of a mouse after bias field correction.(c) The bias field that was corrected.

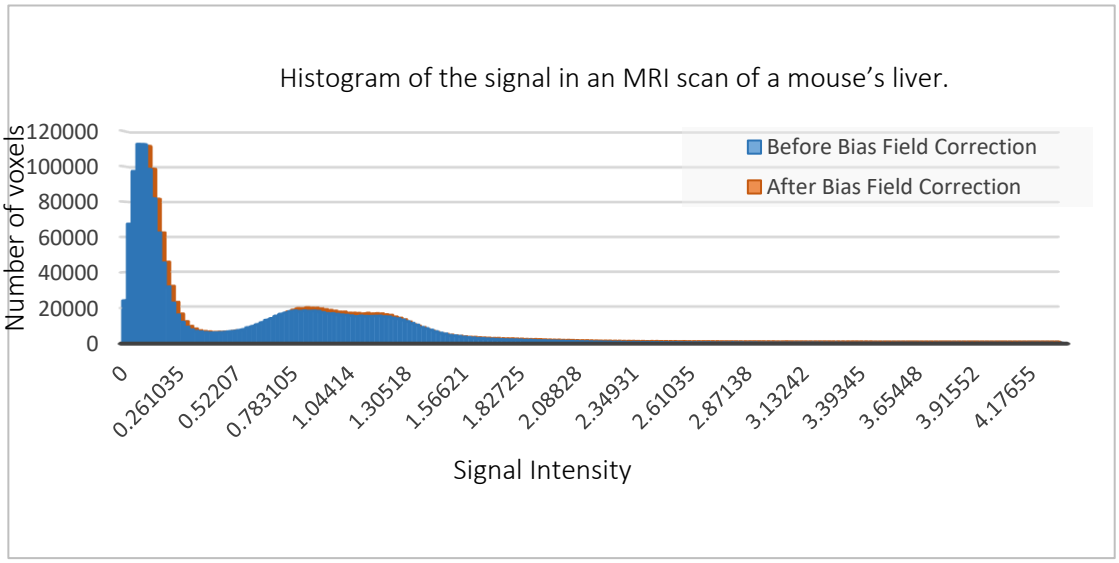


Figure 3-4: Histogram of the liver of a mouse before and after bias field correction. The slight shift in the histogram does not change the data dramatically but allows for images to be compared.

3.3 Multi-atlas Segmentation (Invicro's Whole Body Atlas)

3.3.1 Region Based Segmentation

Multiple region-based segmentation methods were tested as these have been shown to be a robust and relatively quick method of segmentation of large organs (e.g. liver and lungs) and areas in the brain (115) (116) (117) (80) (116). However due to the less defined nature of the abdomen, pancreas and pancreatic tumours with their deformable characteristics and variable spatial location, none of the approaches were successful. However, these approaches (such as Invicro's Whole Body Atlas (WBA)) can be used to segment the larger organs around the pancreas (and any tumour contained therein).

3.3.2 How many scans to compare. Time vs accuracy

A minimum of 12 scans is required to stably run the multi-atlas segmentation tool with reproducible results, as with fewer scans (116) (115) in the library the accuracy of the segmentation of large organs is decreased dramatically (DSC = 0.74 decreasing to 0.53 with a run time decrease from 9 minutes to 3 minutes). The addition of more scans to the multi-atlas segmentation tool reference library increases the accuracy and reproducibility of the

segmentation. However, an increase in the library size requires a linear increase in the computational time, therefore, a balance must be found between computational time and accuracy. A library of scans restricted to 50 produces the very high DSC of 0.91 but with a four-fold increase in computational time, see Figure 3-5. Researchers are encouraged to use between 20 and 40 scans in their library (81).

Once the library has been compared to the test scan, the best N scans from the reference library are used to calculate the segmentations. For 3D-CAMMP, an N of less than or equal to 3 provides a low computational time but a basic segmentation of the organs (DSC = 0.61, runtime = 10 minutes). An N of 9 or greater provides an accurate organ segmentation however the computational time required increases exponentially (DSC = 0.85, runtime = 11 minutes). Therefore an N of 6 provides an optimal balance between these two factors (DSC = 0.89, runtime = 12 minutes).

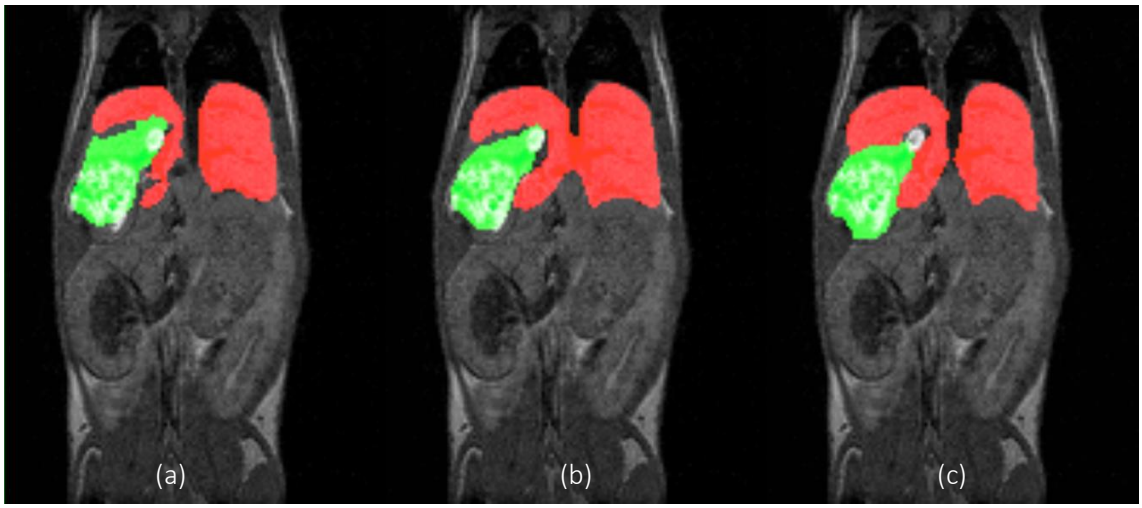


Figure 3-5: The comparison of the size of library used in the WBA. (a) 6 scans within the library, (b) 12 scans within the library and (c) 50 scans. The more library scans the more accurately the automated segmentation becomes. However, this must be weighed against the increased computational time.

3.3.3 Thresholds on each organ

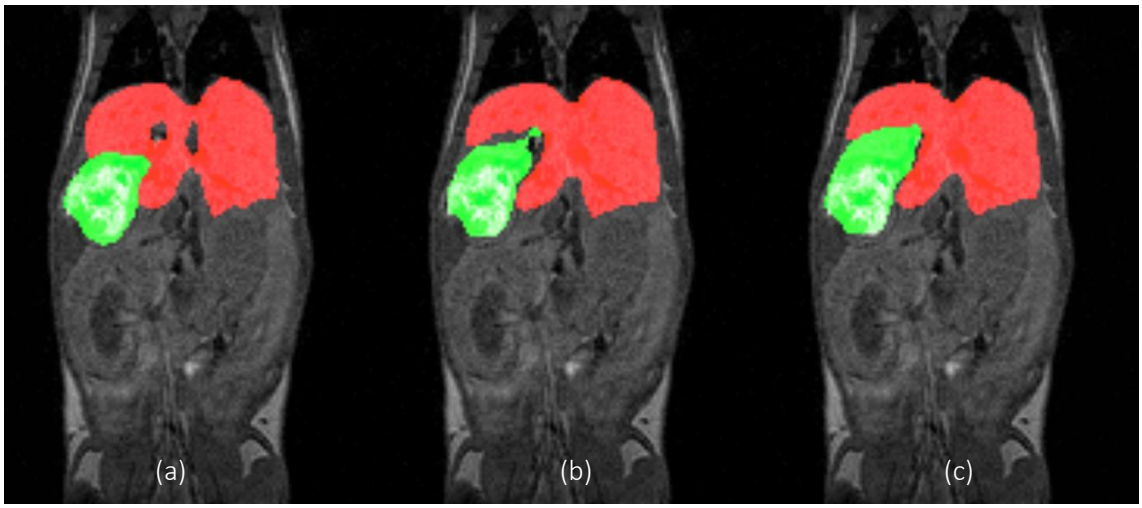


Figure 3-6: The number of scans combined by the WBA for segmentation. (a) the 3 closest scans, (b) the 6 closest and (c) the 9 closest. (a) and (c) under- and over-segment the organs respectively with (b) being the compromise.

A global probability threshold cannot be used in the multi-atlas segmentation tool when it is segmenting such a large variety of organs. Large and structurally stable organs such as the liver, kidneys, and stomach can be precisely segmented (DSC of 0.90 or higher) using 15 scans compared to ground truth with a threshold of 0.5 (all voxels with probability greater than or equal to 0.5 are classified). The spleen in the KPC mouse-model (on which 3DCAMMP is built) undergoes structural and morphological changes throughout development of the pancreatic tumours. However, the spleen is still able to be precisely segmented by the multi-atlas segmentation tool at a threshold of 0.5. Finally, the hepatic portal vein and the gallbladder are small features that have some variability, and as such, stricter thresholds of 0.2 and 0.15 respectively are required. This is summarised in Table 3. The effect of changing the threshold

limit on the accuracy of the classification is shown in Figure 3-7 and a specific example of the hepatic portal vein is shown in Figure 3-8

.

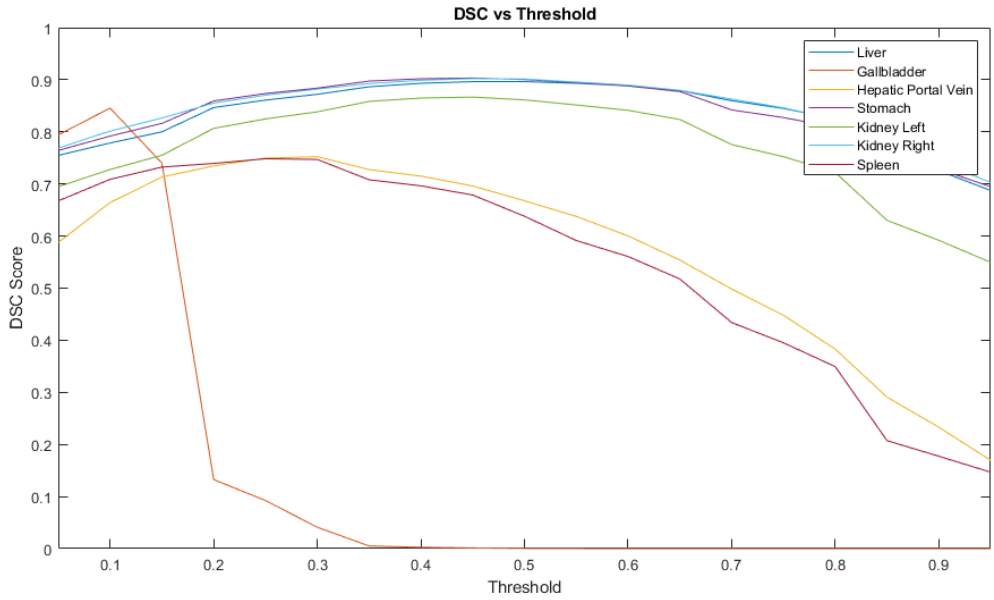


Figure 3-7: The Dice Similarity Coefficient at different thresholds for each organ segmented using the Multi-Atlas Segmentation Tool (Whole Body Atlas). The highest DSC value for each organ was taken as the threshold.

Table 3: Organ threshold used with Whole Body Atlas.

Organ	WBA Threshold
Kidneys	0.5
Liver	0.5
Stomach	0.5
Spleen	0.5
Hepatic Portal Vein	0.2
Gallbladder	0.15

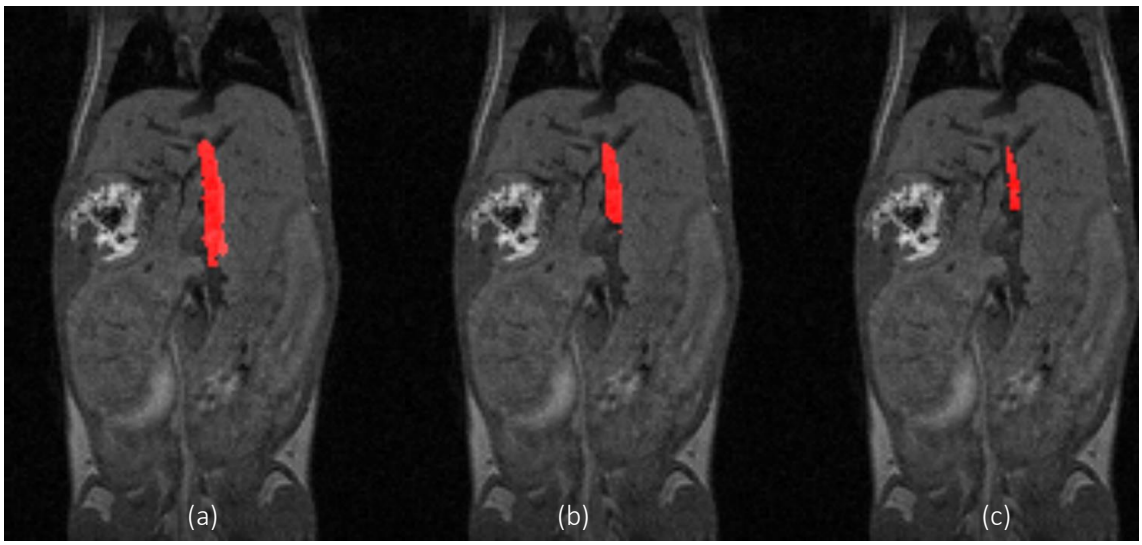


Figure 3-8: Effect of the different thresholds on organ segmentation. (a) A threshold of 0.1 causes bleeding of the ROI over the edge of the hepatic portal vein. (b) A threshold of 0.2 segments the majority of the hepatic portal vein without the ROI bleeding. (c) A threshold of 0.5 causes large areas to be missed.

3.3.4 Limitations of Multi-Atlas Segmentation Tools

Commercial Multi-Atlas Segmentation (MAS) tools rely on a predefined user-generated library of segmented images. The MAS tool identifies the library and the objects to be segmented in the novel scans, each novel scan is compared to every image in the library for large anatomical features with simple linear registration. The N most comparable library images to the novel scan are then used to build a probability of the location of the object. Each voxel in the novel scan is assigned a value between 0 and N, determined by the number of library scans that have an object located within the voxel. Each voxel in the novel scan is then divided by N to produce a probability. A pre-determined threshold consequently identifies the object. E.g. with a threshold of 0.5, any voxel with a probability greater than or equal to 0.5 is classed as the object.

Due to the high variability of the location of the pancreas and pancreatic tumours that change throughout the growth of the animal and the development of tumours, such a rigid method is unable to accurately segment the pancreas and any pancreatic tumours, as shown in Figure 3-9.

Therefore, a Multi-Atlas Segmentation tool can be used to be able to identify large and spatially fixed organs such as the brain, liver, lungs etc. but is not suitable for a variable organ like the pancreas or tumours and an other method must be used.

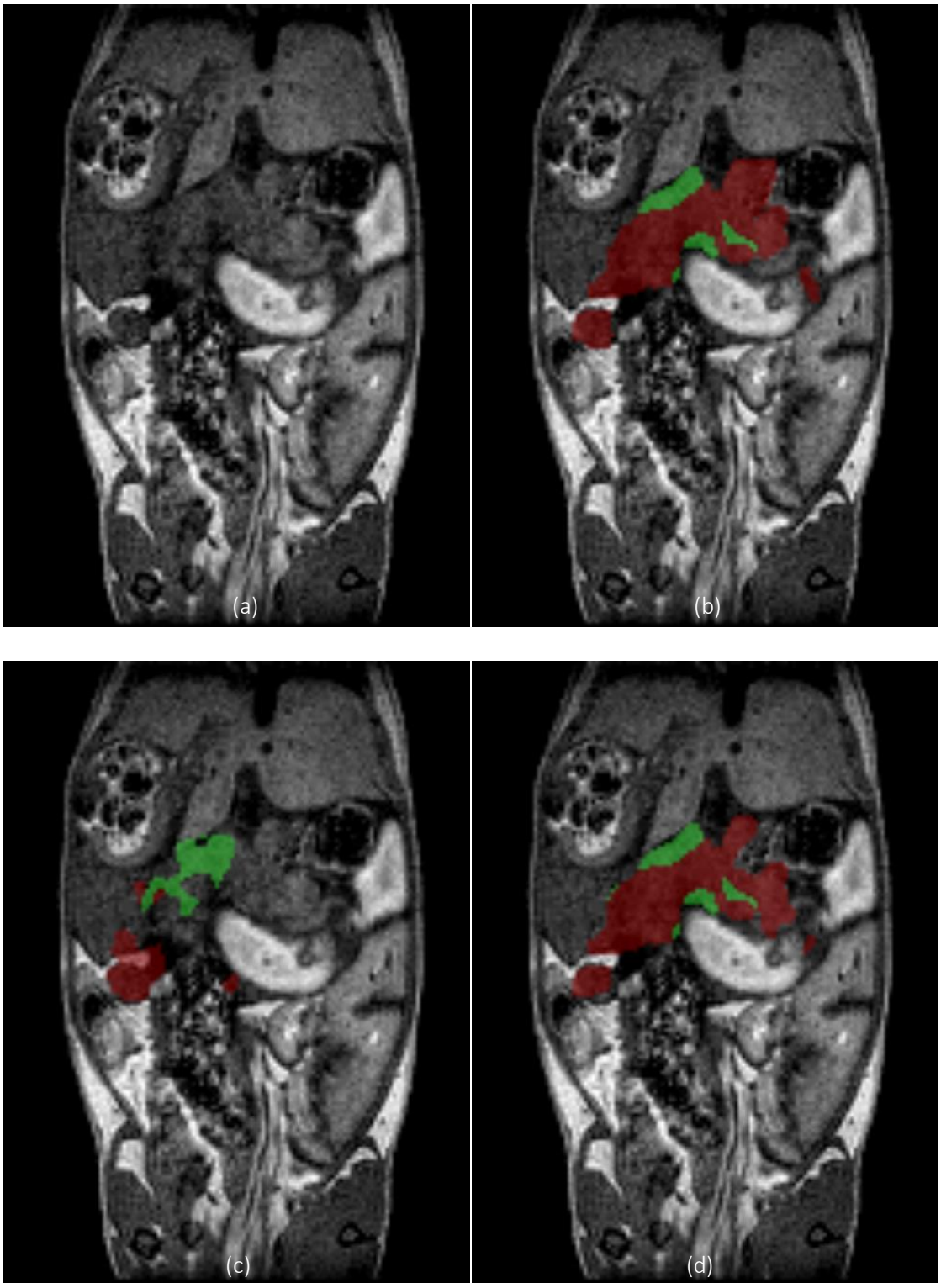


Figure 3-9 (a) shows an MRI scan of a KPC mouse at 102 days old, (b) shows the pancreas (green) and pancreatic tumour (red) manually segmented by an expert user and (c) shows the segmentation of the pancreas (green) and pancreatic tumours (red) by the Multi-Atlas Segmentation Tool using a library of 50 pre-segmented scans. In contrast, (d) shows 3D-CAMMP's segmentation of the pancreas and pancreatic tumours. As is evident when comparing. (c) and (d), 3D-CAMMP is able to match the accuracy of a manual segmentation performed by an expert user. 3D-CAMMP is able achieve a DSC of 0.78 when compared to expert segmentation, unlike the MAS tool, which only has a DSC of 0.24.

3.4 Pancreas and Tumour Cloud Bounding Area

A bounding box is defined by taking the maximum distance from the centre of the pancreas in each of the 6 directions from all scans in the library. This is then increased by one standard deviation in each direction to create a 6 distance measures for a bounding box. With the same scans the average distance between the centroids of each automatically segmented organ and the centroid of the pancreas can be calculated. For a novel scan the centroids of each automatically segmented organ are calculated and the centroid of the pancreas is estimated using the pre calculated measure. The bounding box is then applied using the 6 measures that have been calculated. The scan inside this box is then thresholded using Otsu's method to remove the area background to the scan. This box is then used for the analysis. This type of bounding box area for identification and classification of pancreatic tumours and the pancreas creates a large area for analysis, much of which is outside the subject or in areas that neither the pancreatic tumour nor the pancreas could develop. The tumour and pancreas cloud defines a smaller area removing voxels in which the pancreatic tumours and pancreas would not form. However, not all areas that the pancreatic tumours and pancreas could develop are identified. The normalised tumour-pancreas cloud is able to define the area in which pancreatic tumours and the pancreas are likely to develop as well as surrounding areas where there is a possibility of both developing, without defining large areas where neither would form. The three tested methods are shown in Figure 3-10.

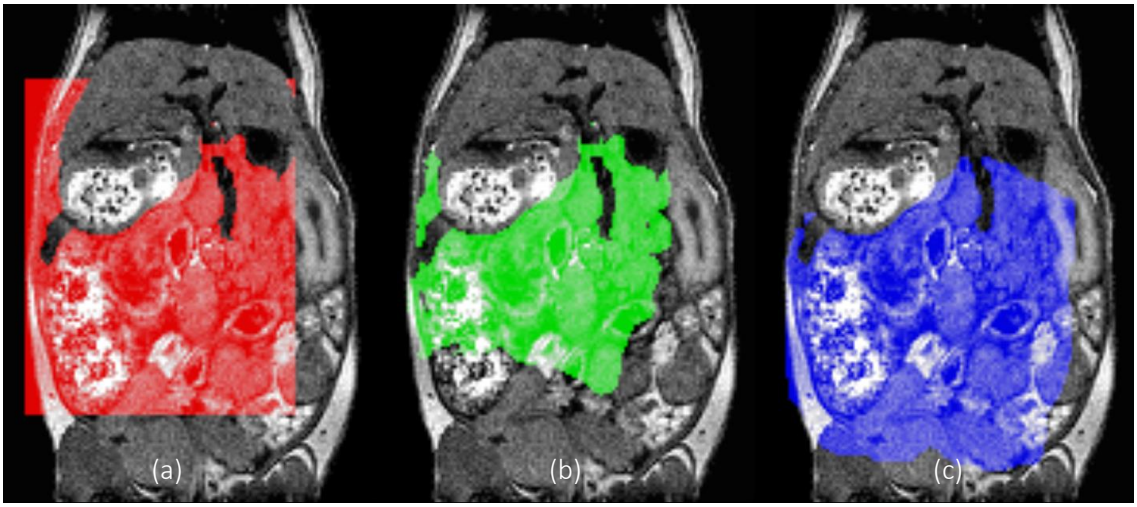


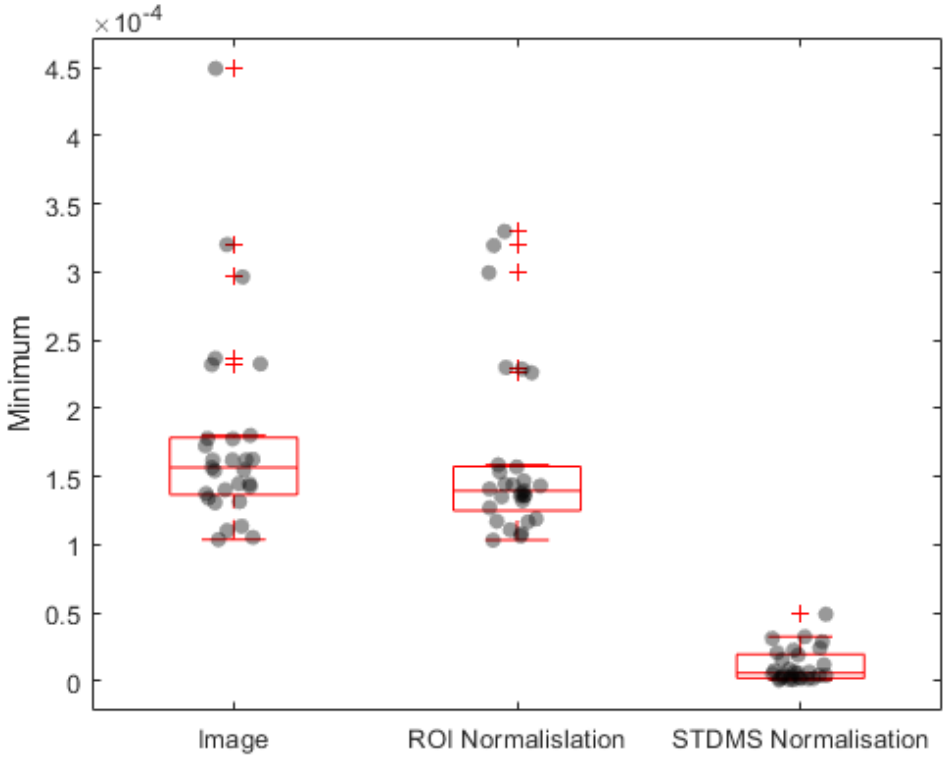
Figure 3-10: Three tested methods to find the area for the machine learning to analyse. (a) A bounding box created by taking the extreme values from all pancreata and tumours. The bounding box has an area of 6212mm^3 and encloses all the tumours and pancreata. (b) A cloud built from combing all pancreata and tumours. The cloud has an area of 2370mm^3, however, it misses 450mm^3 of the tumour/pancreas volume when compared to a large library of scans. (c) A normalised cloud built by taking each pancreas and tumour into a normalised space and then combining them. When a novel scan is passed to 3D-CAMMP the 'normalised' pancreata and tumours are transformed onto the novel scan using the 'centres of the organs'. The normalised cloud has an area of 5624mm^3 and encloses all of the tumours and pancreata.

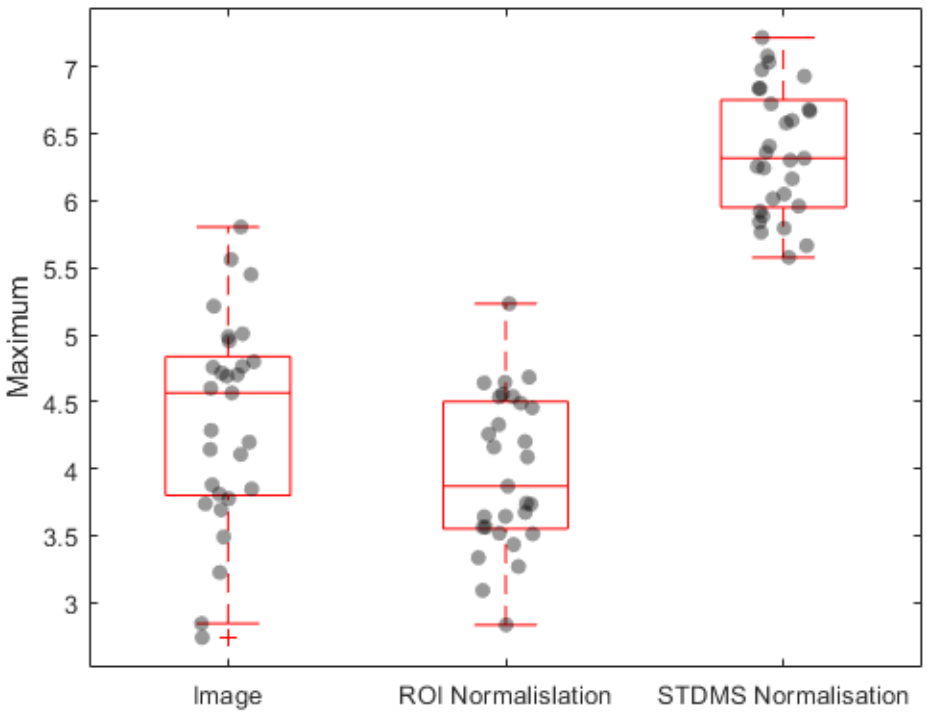
To reduce computational time, it was necessary to define an area for analysis by the machine learning tool. The area should have a high probability of containing the whole pancreas and pancreatic tumour, without incorporating areas that have no, or extremely low probability of containing either pancreas or pancreatic tumour. Therefore, three approaches were used (see Figure 3-10). Initially a bounding box area for identification and classification of both the pancreas and pancreatic tumours was used (see Figure 3-10 (a)). However, this creates a large area for analysis much of which is outside the subject, and also includes areas within the subject where neither the pancreatic tumour nor the pancreas could develop. This area was therefore refined to a smaller area (Figure 3-10 (b)) (a tumour and pancreas cloud) by removing voxels in which the pancreatic tumours and pancreas would not form. However, not all areas in which the pancreatic tumours and pancreas could develop are identified. Figure 3-10 (c) shows the result of normalising the scans before combining them. The normalised tumour-pancreas cloud

is able to define the area in which pancreatic tumours and the pancreas are likely to develop as well as surrounding areas where there is a possibility of both developing while excluding large areas where neither would form.

3.5 Development of Probability Clouds

There were three distinct approaches, two unsuccessful and one successful. In the first approach, all pre-segmented tumours were loaded onto a novel scan. This approach was unable to compensate for the position of the animal within the window or for variations in animal size. The second method tested used the library of pre-segmented scans and found the vertices of each object segmented. It then calculated the mean of the vertices for each object and related them to the vertices of the segmented tumour. This method then used that relationship to transform the tumour position to a normalised space. This would be performed on every scan in the library to produce a normalised tumour probability cloud. A novel scan would then have the vertices of each object segmented by the multi-atlas segmentation tool. Using the mean distance relationship, the normalised tumour probability cloud was then transformed on to the novel scan. This method did not perform well if there were variations in object size due to e.g. changes in the size of animal and inflammation of the spleen. The final method, however, was more successful. This approach used a library of pre-segmented scans. For each pre-segmented scan, the centre of mass of each segmented object was calculated and the mean of all of the objects' centre of mass was then calculated, which denoted 'centre of animal'. The distance from the centre of the animal to the centre of mass of the tumour was then used as the transformation function to transform the tumour segmentation into a normalised space. This was performed on all scans in the library to build the tumour probability cloud. When a novel scan was run through 3D-CAMMP the centre of mass and 'centre of animal' was calculated, the average transformation function of the library was used to transform the normalised tumour probability cloud into the space of the novel scan. This method is able to account for changes in animal size, position and disease burden and resulted in powerful and robust feature creation.


×10^-4
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
Minimum
Image
ROI Normalislation
STDMS Normalisation


7
6.5
6
5.5
5
4.5
4
3.5
3
Maximum
Image
ROI Normalislation
STDMS Normalisation

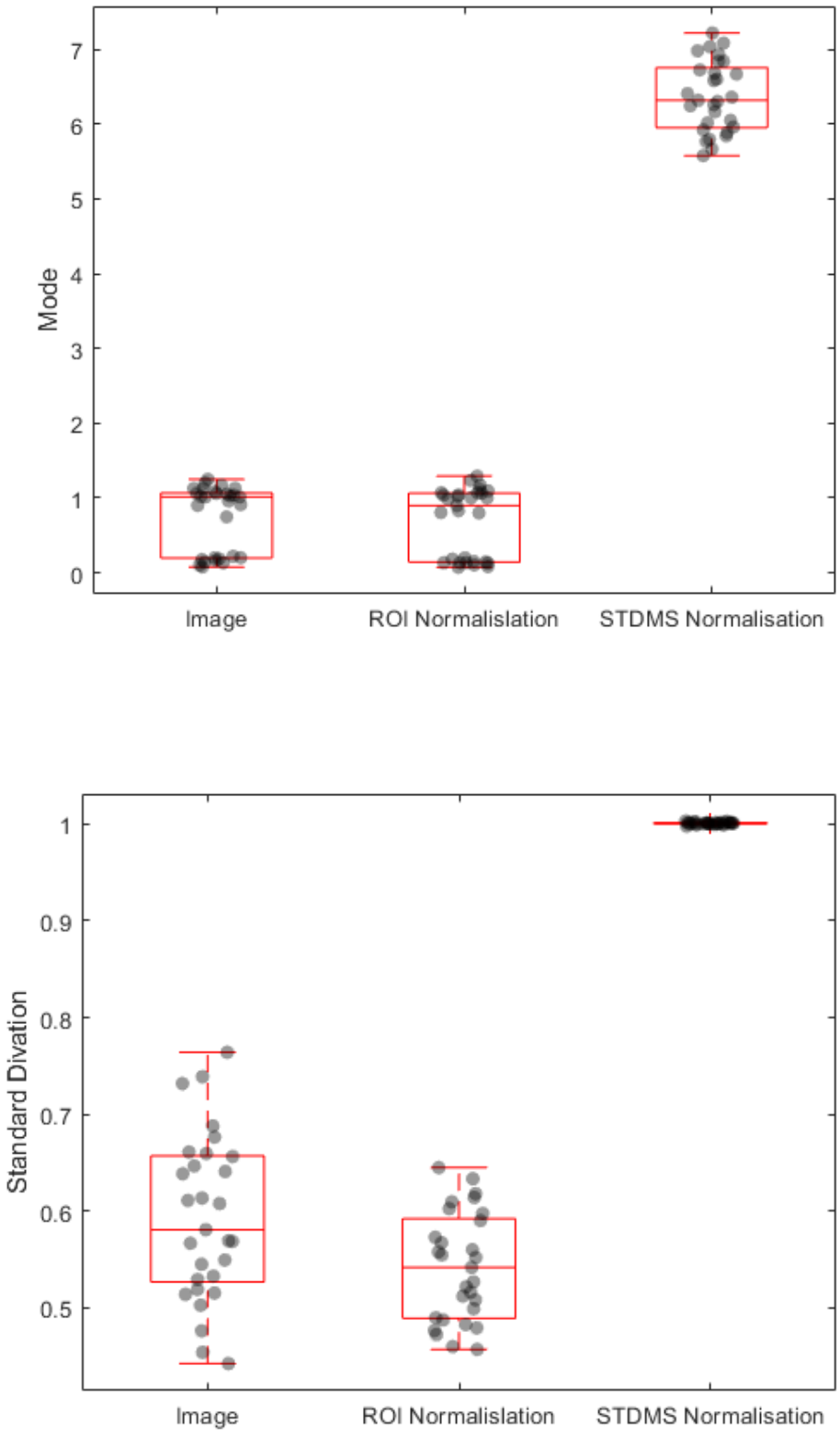


Figure 3-11: Image intensities of the left kidney before and after two types of normalisation. ROI Normalisation is done by creating a small ROI in the spinal muscle (an area that should not change dramatically over a mouse's life span). This entire scan is then divided by the mean of this ROI to normalise the image. STDMS Normalisation is Standard Deviation Mean Shift Normalisation, in which the normalisation takes the intensity of each voxel minus the mean of

the entire image all divided by the standard deviation of the entire image. The figure shows the effect of these normalisation techniques with STDMS normalisation bringing the separate image values much closer together.

3.6 Normalisation

It is impossible to compare MRI scans without normalisation due to the MRI measuring only signal intensity, and this can fluctuate between subjects and between scans.

In first method of normalisation tested a ROI is drawn within the spinal muscle of the scan. The mean intensity of this ROI is then set to 1 through transformation and this transformation is applied to the entire scan in order to shift the entire scans histogram.

Further testing of the model comparison showed a slight increase of model classification accuracy with the use of standard deviation shift mean normalisation (STDMS), a 0.15 increase in DSC, and a large increase in the robustness of the model. This is shown in Figure 3-11 for the image intensities of the left kidney and Figure 3-12 showing overall histograms of multiple scans.

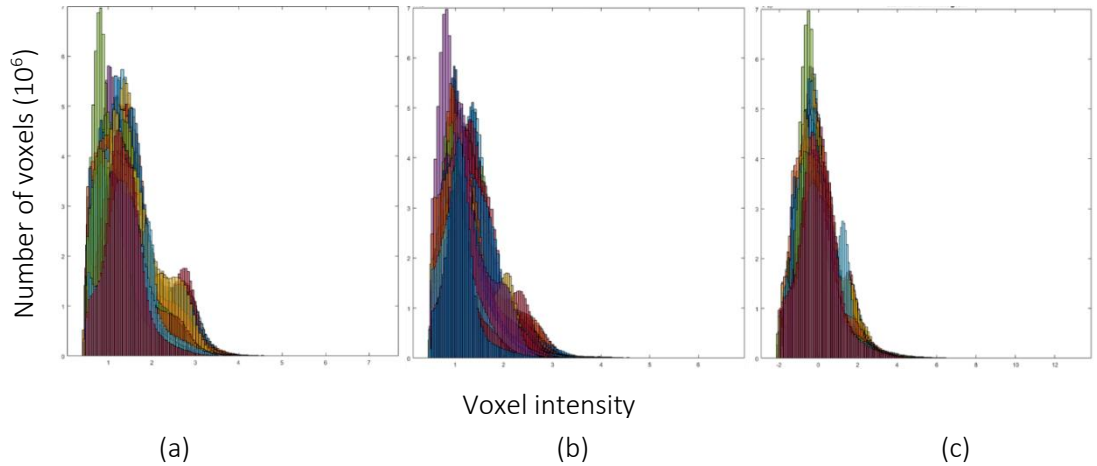


Figure 3-12: Histograms of scan intensity of multiple scan data overlaid. (a) scans with no normalisation. (b) scans with ROI Normalisation. (c) scans with STDMS normalisation. In (c) the histograms are much closer together and have a much more defined spread.

Table 4: Numerical data from Figure 3-11 is summarised here. The effect of normalisation on the standard deviation of the left kidney signal of multiple scans. The percentage reflects the increase in the clustering of intensities of the scan data. STDMS Normalisation consistently out performs ROI Normalisation.

1st degree feature	ROI Normalisation effect on standard deviation of multiple scans (% increase in clustering of intensities)	Standard Deviation Mean Shifted Normalisation effect on standard deviation of multiple scans (% increase in clustering of intensities)
Mode	3%	7%
Minimum	0%	100%
Maximum	25%	39%
Standard Deviation	34%	99%

The STDMS approach to normalisation was extremely successful and allowed the ability to compare MRI scans of subjects on different days as well as inter-subject comparisons. ROI normalisation did not provide a significant advantage over using non-normalised scans and required much more user interpretation and computational time. A full animal normalisation is shown in Figure 3-13.

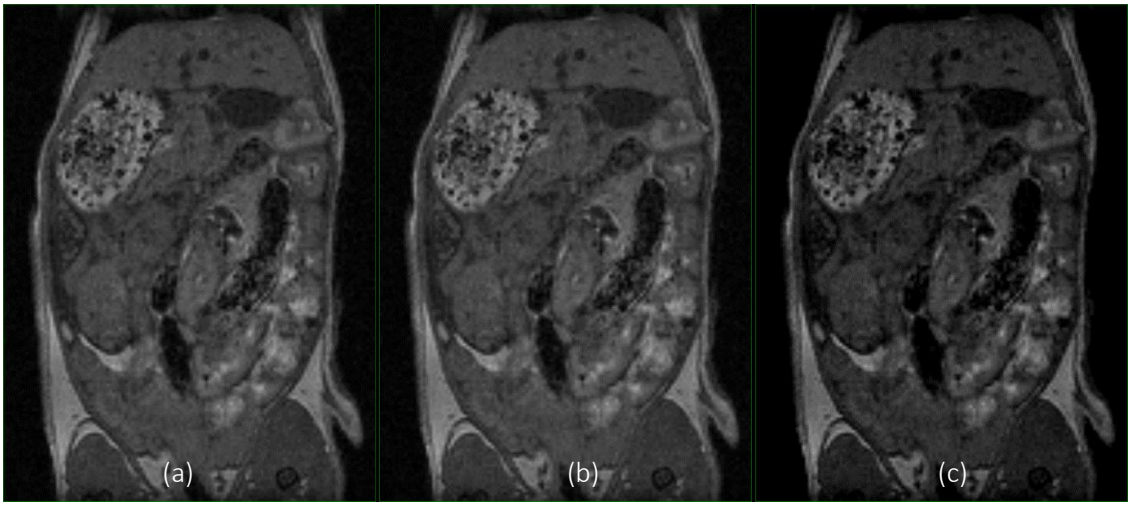


Figure 3-13: Scan before and after normalisation. (a) shows the original image. (b) shows the image after ROI normalisation. (c) shows the image after STDMS normalisation. It can be seen in (c) that the contrast ratio is much improved, there is less visible noise, but the textural features have all been preserved.

3.7 Object Detection

The CHO can see the object with high accuracy. This is demonstrated in the receiver-operating characteristic (ROC) curve, which compares the sensitivity (proportion of positives correctly

identified) and 1 - specificity (proportion of negatives correctly identified) to show the accuracy of the method, and calculate AUC (0.86 in this model). This method, however, requires quite a large amount of prior knowledge, i.e. either the location or the shape of sought object. Therefore, further analysis is required before this can be implemented.

CHO is a form of side-scrolling object detection. It can be used in two separate methods. Firstly, if the location of the object is known, the CHO is able to segment the object with zero prior knowledge of its shape. Secondly, if the shape of an object is known then the CHO is able to search through the image for objects that match the shape. This method is often used for the detection of brain tumours as the structure of the brain is relatively stable and tumours can often have a relatively defined shape (118).

3.8 Feature Generation

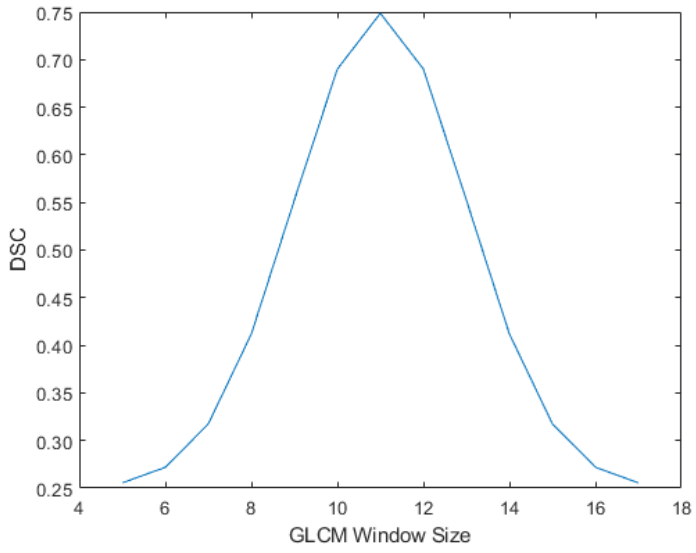


Figure 3-14: The Dice Similarity Coefficient of the output of 3D-CAMMP based upon the window size used for the GLCM feature generation.

76 features were generated for each voxel in each scan. This is the information that is used by the machine learning system to classify each voxel.

3.8.1 Grey Level Co-Occurrence Matrix

GLCMs proved to be invaluable features for the machine learning. Multiple offsets of the GLCMs were computed by changing the variable in *calculate_glcm_Run_IdxTest.m.* The different offsets tested were all combinations of $[a, b]$ where $a = -5, -4, -3, \dots 3, 4, 5$ and $b = 1,2,3,4,5$. The GLCMs window is also calculated in 3D, tested for different window sizes, 5, 11 and 16, and tested for different number of grey levels, 16, 32. As a 2D computation they are computationally cheap and have a high feature power. Not all GLCMs were ranked very highly, however the computational cost of producing them when the GLCM is already calculated is negligible.

3.8.1.1 Size of window

The use of a window size of 5 caused the GLCM features to become extremely susceptible to noise in the image. A window size of 17 caused the fine details in the textures to be smoothed out of the GLCM features. A window size of 11 proved to generate robust and sensitive GLCM features, see Figure 3-14.

3.8.1.2 2D vs 3D

The use of 3D GLCMs increased the DSC of 3D-CAMMP by 7%. However, it also increases the computational time exponentially from 300 seconds to > 1800 seconds.

3.8.2 Grey Level Run Length Matrix

3.8.2.1 Size of window

The use of a window size of 5 caused the GLRLM features to become extremely susceptible to noise in the image. A window size of 17 caused the fine details in the textures to be smoothed out of the GLRLM features. A window size of 11 proved to generate robust and sensitive GLRLM features

3.8.2.2 2D vs 3D

The use of 3D GLRLM increased the DSC of 3D-CAMMP by 4%. However, it also increases the computational time exponentially, and so was not implemented.

3.8.3 Gradient Features

Gradient analysis of this type is used in many different feature selection methods and provides a large amount of information such as the basis for a more robust but less sensitive edge detection. The multi-scale gradient levels were effective at segmentation of the tumour due to the ability to remove a large portion of the heterogeneity which occurs within the pancreatic tumour.

3.8.4 Fractional Anisotropy

Fractional Anisotropy proved to be an important feature in the machine learning step of 3D-CAMMP, providing features that are highly correlated with gradient analysis. However, Fractional Anisotropy was defined as a feature of the higher power than gradient analysis by the ensemble predictions of out of bag observations.

3.8.5 Spatial Features

The use of 'centre of mass' of the organs mean centre increased the DSC score of the model from 0.5001 to 0.6117 over the use of directed spatial location within the image. This shows the possibility of change of position within the animal as well as the animal location within the scanner.

3.9 Feature selection

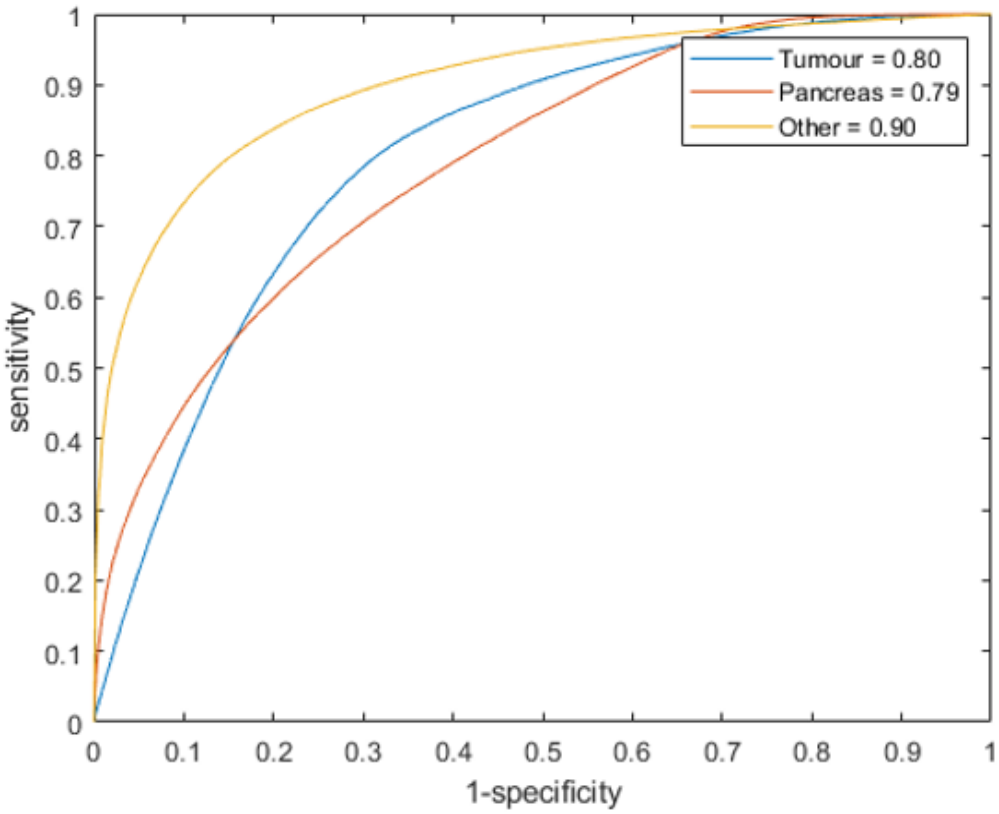
Not all features that were generated would be expected to be useful and the evaluation of the most relevant features was performed.

Multiple feature selection methods were tested in 3D-CAMMP, including forward incremental, backward incremental, principle component analysis.

A random selection of the training data was loaded into the workspace, 20,000 voxels of each class. These voxels were then fed into the script *featureSelection.m* which calculated the reliefF function of the features keeping only the 50 most relevant features. The features selected by the reliefF function were then fed to a k-means clustering algorithm to remove redundant features. 6 clusters were created, and this was used to choose 15 features. Spatial and probability cloud features were removed for the feature selection and added to the final feature selection. The selected features were then used to build machine learning models and compared against models with the full feature set. Principal component analysis was also performed on the feature set and then compared to the standard model output.

However, none of the methods tested resulted in enhanced performance of 3D-CAMMP and reduction in features had a negative effect on model performance that did not outweigh the reduced computational time. DSC decreased from 0.6117 to 0.5989 (ROC curves saw no change) when tested using MATLAB and multiple feature selection methods such as ReliefF, Recursive Feature elimination, Lasso Regression and Principle Component Analysis. Weka, a software designed for machine learning in data mining, was also used to confirm these results. This is believed to be because there is only a total of 76 features currently being used by 3D-CAMMP and this is a relatively small number in the field of radiomics where often hundreds if not thousands of features are generated. The reason so many features were not generated for 3D-CAMMP was due to computational time as 3D-CAMMP is designed to be run on an average work-station and so must be kept relatively simple. This is the same reason that GPU programming was not used. This is shown in Figure 3-15.

Standard model



Feature Reduction Model

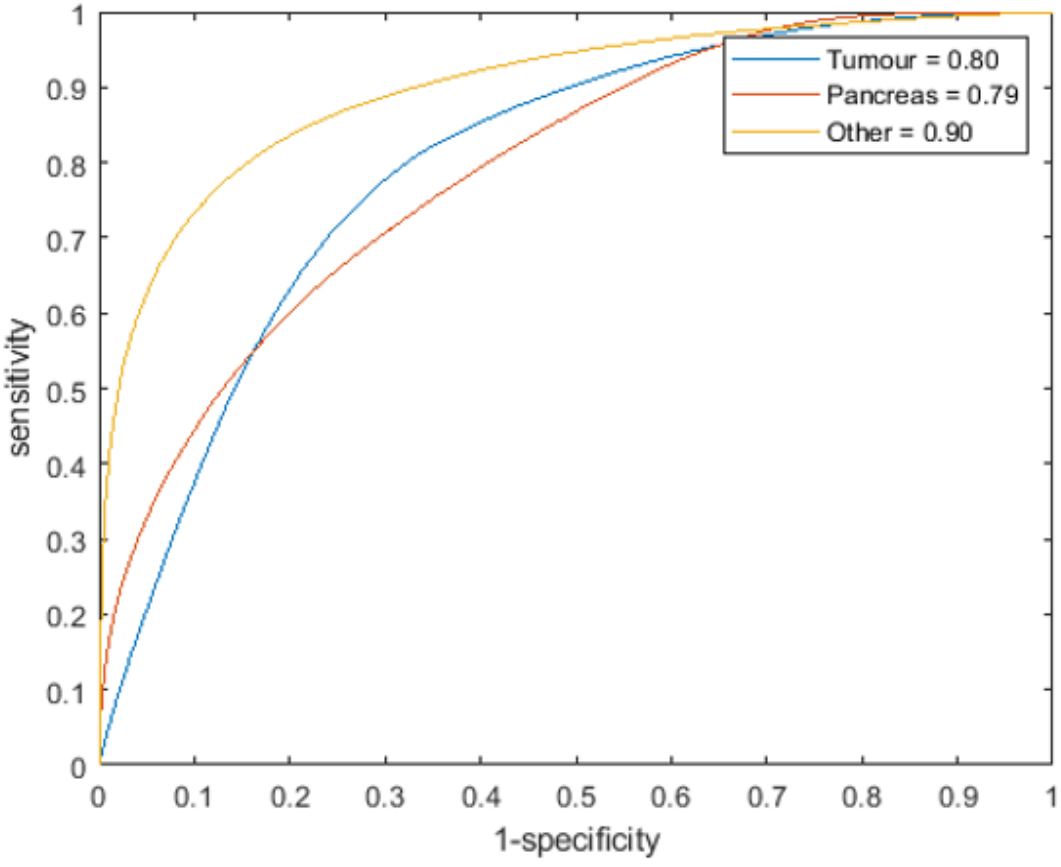


Figure 3-15: Feature selection was performed on the data and model. Multiple feature selection methods were tested which had no negative effect on the performance 3D-CAMMP and using only the selected features did not significantly increase run time of 3D-CAMMP.

3.10 Final Features List

The features generated by the GLCMs are given as follows (119). Let $p(i,j)$ be the $(i,j)^{th}$ entry of the GLCM and N_g be the number of discrete grey levels, then

Mean of row

$$\mu_x = \sum_i \sum_j i \cdot p(i,j) \tag{3-1}$$

Mean of column

$$\mu_y = \sum_i \sum_j j \cdot p(i,j) \tag{3-2}$$

Standard deviation of row

$$\sigma_x = \sum_i \sum_j \left(i - \mu_x\right)^2 \cdot p(i,j) \tag{3-3}$$

Standard deviation of column

$$\sigma_y = \sum_i \sum_j \left(j - \mu_y\right)^2 \cdot p(i,j) \tag{3-4}$$

Sum of rows

$$p_x = \sum_j p(i,j) \tag{3-5}$$

Sum of columns

$$p_y = \sum_i p(i,j) \tag{3-6}$$

$$u_x = \sum_i \sum_j ip(i,j) \tag{3-7}$$

$$u_y = \sum_i \sum_j jp(i,j) \quad 3\text{-}8$$

$$p_{x+y}(k) = \sum_i \sum_j p(i,j)\,, i+1 = k, k = 2,3,\dots 2N_g \quad 3\text{-}9$$

$$p_{x-y}(k) = \sum_i \sum_j p(i,j), |i-j| = k, k = 0,1,\dots,N_g - 1 \quad 3\text{-}10$$

Entropy of p_x

$$HX = -\sum_i p_x(i)\log_2(p_x) \quad 3\text{-}11$$

Entropy of p_y

$$HY = -\sum_j p_y(j)\log_2(p_y) \quad 3\text{-}12$$

Entropy of $p(i,j)$

$$H = -\sum_i \sum_j p(i,j)\log_2(p(i,j)) \quad 3\text{-}13$$

$$HXY1 = -\sum_i \sum_j p(i,j)\log(p_x p_y) \quad 3\text{-}14$$

$$HXY2 = -\sum_i \sum_j p_x(i)p_y(j)\log(p_x p_y) \quad 3\text{-}15$$

The GLCM features can therefore be define as follows:

Autocorrelation

$$f = \sum_i \sum_j ijp(i,j) \quad 3\text{-}16$$

Contrast

$$f = \sum_i \sum_j |i-j|^2 p(i,j) \tag{3-17}$$

Correlation (MATLAB)

$$f = \frac{\sum_i \sum_j (i-\mu_x)(j-\mu_y) p(i,j)}{\sigma_x \sigma_y}$$

Correlation

$$f = \frac{\sum_i ijp(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y} \tag{3-18}$$

Cluster Prominence

$$f = \sum_i \sum_j \left(i + j - \mu_x - \mu_y\right)^4 p(i,j) \tag{3-19}$$

Cluster Shade

$$f = \sum_i \sum_j \left(i + j - \mu_x - \mu_y\right)^3 p(i,j) \tag{3-20}$$

Dissimilarity

$$f = \sum_i \sum_j |i-j| \cdot p(i,j) \tag{3-21}$$

Energy (Angular Second Moment)

$$f = \sum_i \sum_j p(i,j)^2 \tag{3-22}$$

Entropy

$$f = -\sum_i \sum_j p(i,j) \log\left(p(i,j)\right) \tag{3-23}$$

Homogeneity (MATLAB)

$$f = \sum_i \sum_j \frac{p(i,j)}{1+|i-j|} \tag{3-24}$$

Homogeneity

$$f = \sum_i \sum_j \frac{p(i,j)}{1+(i-j)^2} \tag{3-25}$$

Maximum Probability

$$f = \max_{i,j} p(i,j) \tag{3-26}$$

Sum of Squares (Variance)

$$f = \sum_{i}\sum_{j}(j-\mu)^2 p(i,j) \tag{3-27}$$

Sum Average

$$f = \sum_{i=2}^{2N_g} i p_{x+y} \tag{3-28}$$

Sum Variance

$$f = \sum_{i=2}^{2N_g} (i - SE)^2 p_{x+y} \tag{3-29}$$

Sum Entropy

$$f = -\sum_{i=2}^{2N_g} p_{x+y} \log_2\left(p_{x+y}\right) \tag{3-30}$$

Difference Variance

$$f = \sum_{i}\sum_{j}(i-\mu)^2 p(i.j) \tag{3-31}$$

Difference Entropy

$$f = \sum_{i=0}^{N_g-1} p_{x-y}(i) \log_2\left(p_{x-y}(i)\right) \tag{3-32}$$

Information Measure of Correlation 1

$$f = \frac{HXY - HXY1}{\max(HX, HY)} \tag{3-33}$$

Information Measure of Correlation 1

$$f = \sqrt{1 - e^{-2(HXY2 - HXY)}} \tag{3-34}$$

Inverse Difference

$$f = \sum_i \sum_j \frac{1}{1 + (i-j)^2} p(i,j) \quad 3\text{-}35$$

Inverse Difference Normalised

$$f = \sum_i \sum_j \frac{p(i,j)}{1 + \frac{|i-j|}{N_g}} \quad 3\text{-}36$$

Inverse Difference Moment Normalised

$$f = \sum_i \sum_j \frac{p(i,j)}{1 + \left(\frac{|i-j|}{N_g}\right)^2} \quad 3\text{-}37$$

The image was converted into 32 distinct grey levels. Let M be the $N_g * N_r$ grey level run length matrix with N_g grey levels and N_r the maximum possible run length. Let $r_{ij} = r(i,j)$ is the number of occurrences with grey level i and j the run length. Then let N_v be the total number of voxels and

$$N_s = \sum_i \sum_j r_{ij} \quad 3\text{-}38$$

the sum of all elements. Let

$$r_i = \sum_j r_{ij} \quad 3\text{-}39$$

be the marginal sum of the runs over the run length j for grey values i, and marginal sum of the runs over the grey values i for run length j is then

$$r_j = \sum_i r_{ij} \quad 3\text{-}40$$

The mean run length is defined as

$$\mu_i = \sum_i \sum_j i p_{ij} \quad 3\text{-}41$$

and

$$\mu_j = \sum_i \sum_j j p_{ij} \quad 3\text{-}42$$

with the joint probability.

$$p_{ij} = \frac{r_{ij}}{N_s} \quad 3\text{-}43$$

The features generated using the GLRLM are define as follows (119).

Short Run Emphasis

$$f = \frac{1}{N_s} \sum_j \frac{r_j}{j^2} \quad 3\text{-}44$$

Long Run Emphasis

$$f = \frac{1}{N_s} \sum_j j^2 r_j \quad 3\text{-}45$$

Low Grey Level Run Emphasis

$$f = \frac{1}{N_s} \sum_i \frac{r_i}{i^2} \quad 3\text{-}46$$

High Grey Level Run Emphasis

$$f = \frac{1}{N_s} \sum_i i^2 r_i \quad 3\text{-}47$$

Short Run Low Grey Level Emphasis

$$f = \frac{1}{N_s} \sum_i \sum_j r_{ij} \quad 3\text{-}48$$

Short Run High Grey Level Emphasis

$$f = \frac{1}{N_s} \sum_i \sum_j \frac{i^2 r_{ij}}{j^2} \quad 3\text{-}49$$

Long Run Low Grey Level Emphasis

$$f = \frac{1}{N_s} \sum_i \sum_j \frac{j^2 r_{ij}}{j^2} \quad 3\text{-}50$$

Long Run High Grey Level Emphasis

$$f = \frac{1}{N_s}\sum_i \sum_j i^2 j^2 r_{ij} \quad \text{3-51}$$

Grey Level Non-Uniformity

$$f = \frac{1}{N_s}\sum_i r_i^2 \quad \text{3-52}$$

Grey Level Non-Uniformity Normalised

$$f = \frac{1}{N_s^2}\sum_i r_i^2 \quad \text{3-53}$$

Run Length Non-Uniformity

$$f = \frac{1}{N_s}\sum_j r_j^2 \quad \text{3-54}$$

Run Length Non-Uniformity Normalised

$$f = \frac{1}{N_s^2}\sum_j r_j^2 \quad \text{3-55}$$

Run Percentage

$$f = \frac{N_s}{N_v} \quad \text{3-56}$$

Grey Level Variance

$$f = \sum_i \sum_j (i-\mu)^2 p_{ij} \quad \text{3-57}$$

Run Length Variance

$$f = \sum_i \sum_j (j-\mu)^2 p_{ij} \quad \text{3-58}$$

Run Entropy

$$f = -\sum_i \sum_j p_{ij} \log_2(p_{ij}) \quad \text{3-59}$$

Mean

$$f = \frac{1}{N}\sum_i \sum_j p(i,j) \quad \text{3-60}$$

Median

$$f = \begin{cases} \left(\frac{n+1}{2}\right)^{th} term, \text{if odd} \\ \frac{\left(\frac{n}{2}\right)^{th} term + \left(\frac{n}{2}+1\right)^{th} term}{2}, \text{if even} \end{cases} \quad 3\text{-}61$$

Variance

$$f = E[(X - E[X])^2] \quad 3\text{-}62$$

Standard Deviation

$$f = \sqrt{\frac{1}{N-1}\sum (x_i - \bar{x})^2} \quad 3\text{-}63$$

Skewness

$$f = E\left[\left(\frac{X - E[X]}{\sigma}\right)^3\right] \quad 3\text{-}64$$

Kurtosis

$$f = E\left[\left(\frac{X - E[X]}{\sigma}\right)^4\right] \quad 3\text{-}65$$

Intensity

$$f = p(i,j) \quad 3\text{-}66$$

Gradient Magnitude

$$f = \sqrt{g_y^2 + g_x^2} \quad 3\text{-}67$$

Gradient component X direction

$$g_x = \begin{bmatrix} +1 \\ -1 \end{bmatrix} * A \quad 3\text{-}68$$

Gradient component Y direction

$$g_y = \begin{bmatrix} -1 \\ +1 \end{bmatrix} * A \quad 3\text{-}69$$

Gradient Magnitude XY scale 1

$$g_{xy}^{1} = \frac{\sqrt{g_y^2 + g_x^2}}{2^1} \quad 3\text{-}70$$

Gradient Magnitude XY scale 2

$$g_{xy}^{2} = \frac{\sqrt{g_y^2 + g_x^2}}{2^2} \quad 3\text{-}71$$

Gradient Magnitude XY scale 3

$$g_{xy}^{3} = \frac{\sqrt{g_y^2 + g_x^2}}{2^3} \quad 3\text{-}72$$

Gradient Magnitude XY scale 4

$$g_{xy}^{4} = \frac{\sqrt{g_y^2 + g_x^2}}{2^4} \quad 3\text{-}73$$

Gradient Magnitude XY scale 5

$$g_{xy}^{5} = \frac{\sqrt{g_y^2 + g_x^2}}{2^5} \quad 3\text{-}74$$

Fractional Anisotropy

$$f = \sqrt{\frac{3}{2}} \frac{\sqrt{\left(\lambda_1 - \hat{\lambda}\right)^2 + \left(\lambda_2 - \hat{\lambda}\right)^2 + \left(\lambda_3 - \hat{\lambda}\right)^2}}{\sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}} \quad 3\text{-}75$$

Where the Eigen vectors $(\lambda_1, \lambda_2, \lambda_3)$ of the gradient image's azimuth and elevation.

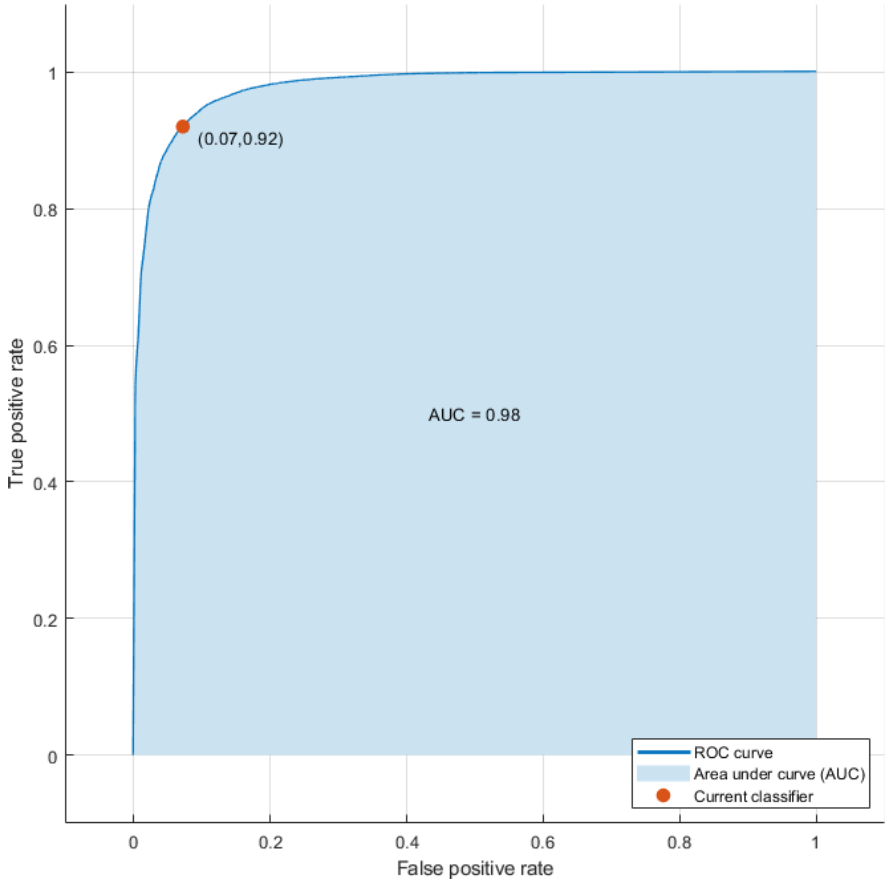
Normalised x, y, and z direction and probability overlays formed in a normalised space of the pancreas, tumour and organs segmented by the MAS tool.

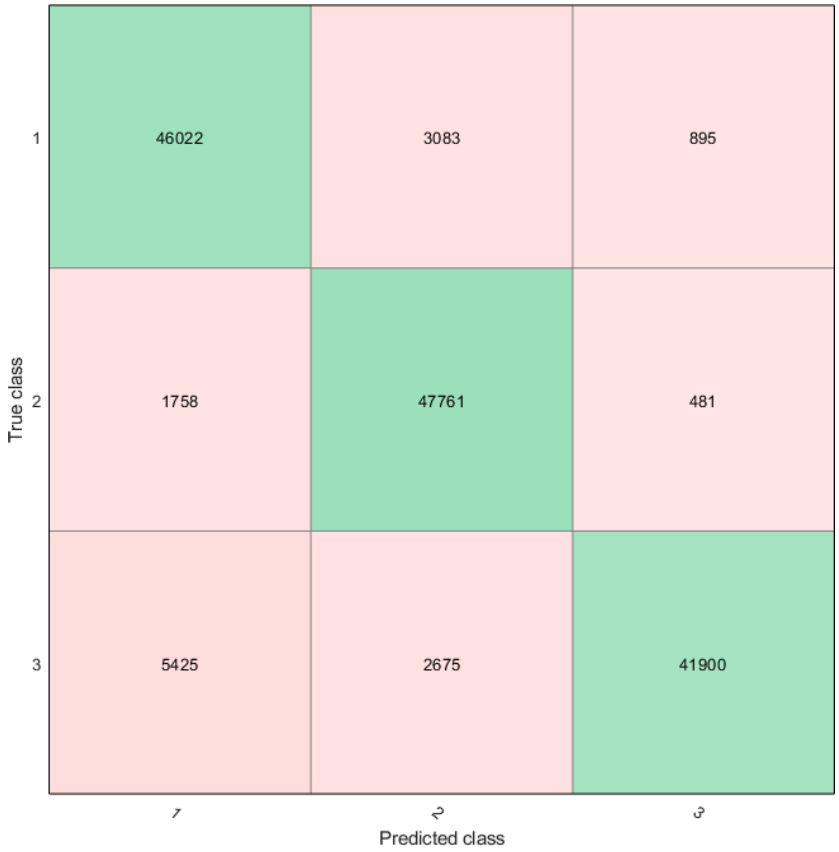
3.11 Machine Learning

Multiple machine learning methods were tested and the hyper parameters of the most suitable were also tested to find the optimum settings for this problem. This is first performed using the built in MATLAB Classification tool and is able to easily compare different models on the same data. One million voxels of the three classification types (tumour, pancreas and not tumour or pancreas) were loaded into a matrix and fed into classification tool. Multiple machine learning

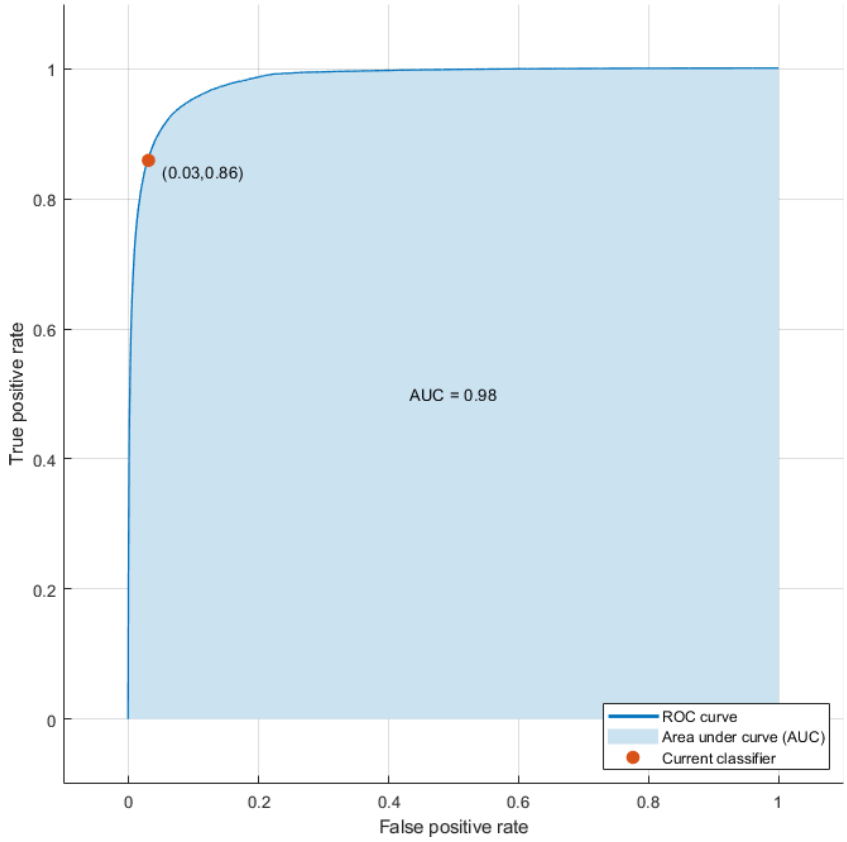
techniques were then tested on this data and compared by accuracy, ROC curves and confusion matrices.

(a) K-Nearest Neighbour





(b) Support Vector Machine



(c) Random Forest

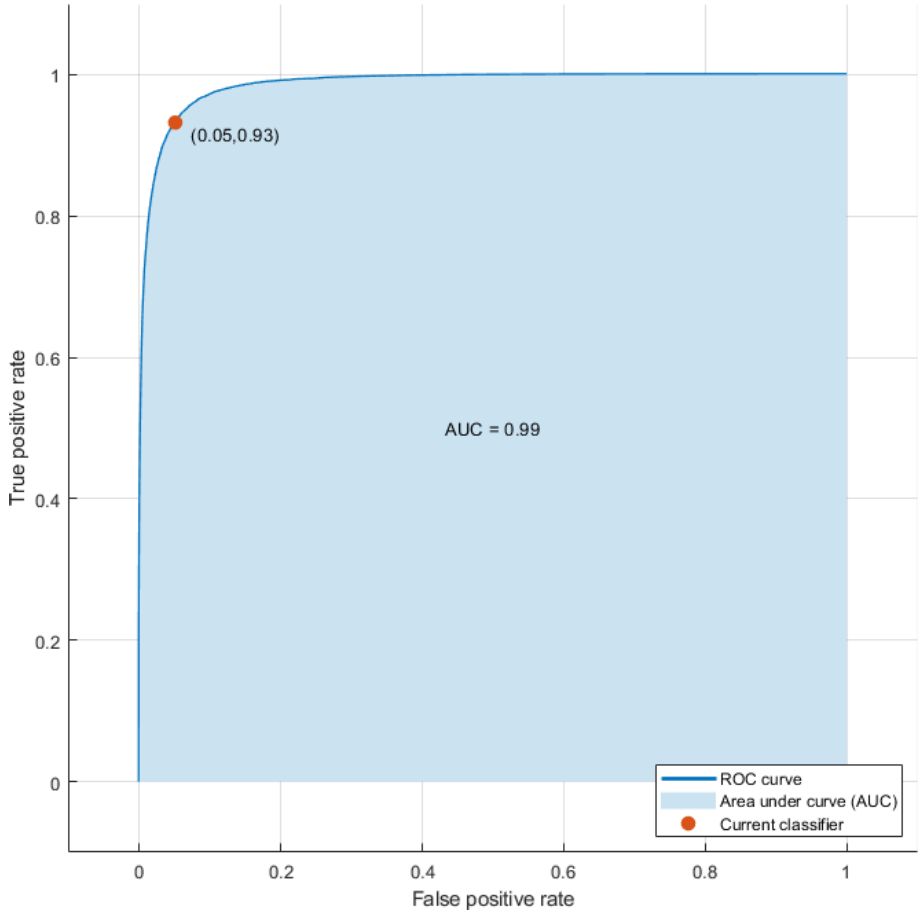


Figure 3-16: Comparison of machine learning methods. Each method was given a sample of 150,000 voxels, 50,000 voxels of tumour, pancreas and not tumour/pancreas. The models were evaluated on their accuracy to correctly classify these three sets. (a) A K-Nearest Neighbour model with an accuracy of 90.5% and AUC of 0.98. (b) A Support Vector Machine with an accuracy of 90.7% and AUC of 0.99. (c) A Random Forest with an accuracy of 93.4% and AUC of 0.99.

The models tested were decision trees (fine, medium and coarse), discriminant analysis (linear and quadratic), support vector machines (linear, quadratic, cubic, fine Gaussian, medium Gaussian and coarse Gaussian), nearest neighbour (fine, medium, course, cosine, cubic and weighted) and ensemble classifiers (boosted trees, bagged trees, subspace discriminant, subspace KNN and RUS boosted trees).

A confusion matrix displays the number of correctly predicted number of objects for each class. The diagonal of the matrix shows correctly predicted classes, while the off-diagonal shows the incorrect predictions, what class they should be, and what class they were assigned. In Figure 3-16, we can see the ROC curves and confusion matrices for 3 different types of machine learning: a) K Nearest Neighbour, b) Support Vector Machine, c) Random Forest. The ROC

curves and AUC for all 3 were almost identical, however the confusion matrices show in more detail where the 3 machine-learning methods differ. The confusion matrix A (K Nearest Neighbour) shows that the model had a tendency to assign non-pancreas / non-tumour voxels as either pancreas or tumour. The confusion matrix B (Support Vector Machine) shows the model has a tendency to classify pancreas and tumour voxels as non-pancreas / non-tumour. The confusion matrix C (Random Forest) has a much lower incidence of either of these issues.

The three best performing methods were Random Forests, K-Nearest Neighbour and Support Vector Machines. Random Forests proved to have the greatest accuracy, as shown in Figure 3-16. Random Forests are also a relatively computationally cheap machine learning method and can easily explained to non-machine learning experts, making it more approachable.

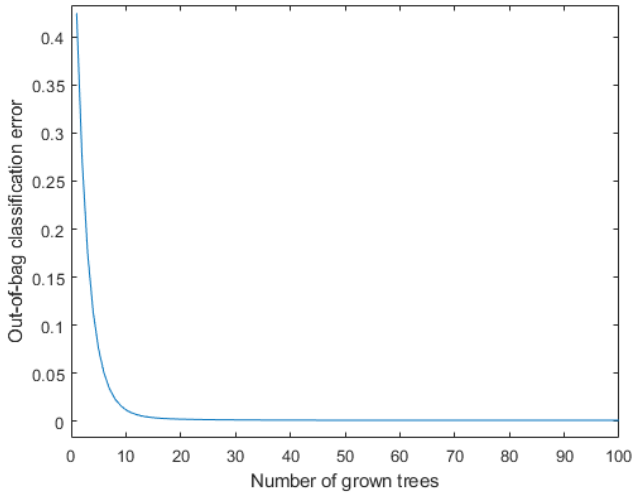


Figure 3-17: The number of grown trees built vs the out-of-bag classification error. The number of trees built was 50 as an increase in the number of trees did not decrease the out of bag error significantly and a decrease in the number of trees did not improve computational time significantly.

3.12 Random Forest Parameters

Multiple Random Forest parameters were tested to see the effect on the model's output. These included minimum leaf size, number of predictors to sample, prior knowledge and combinations of these three. The parameters tested values where:

- ‘MinLeafSize’ = 1, 5 and 10.
- ‘NumPredictorsToSample’ = 2, 5, 8, 10.
- Prior = off and on.

The best of performing of each of these is taken and the combinations where then tested:

- ‘Prior’ + ‘NumPredictorsToSample’ + ‘MinLeafSize’.
- ‘Prior’ + ‘MinLeafSize’.
- ‘Prior’ + ‘NumPredictorsToSample’.
- ‘NumPredictorsToSample’ + ‘MinLeafSize’.

The only parameter changed that improved the machine learning models performance was the number of trees grown, with 50 trees giving a low out-of-bag classification error and an increase in the number of trees not improving the classification error significantly to the computational time increase required, as shown in Figure 3-17.

In general image analysis problems, a random forest is built with hundreds of trees. The number of trees built for 3D-CAMMP was restricted to 50 as any more trees yielded an insignificant increase in model accuracy without a significant increase in computational time.

3.13 Post Processing

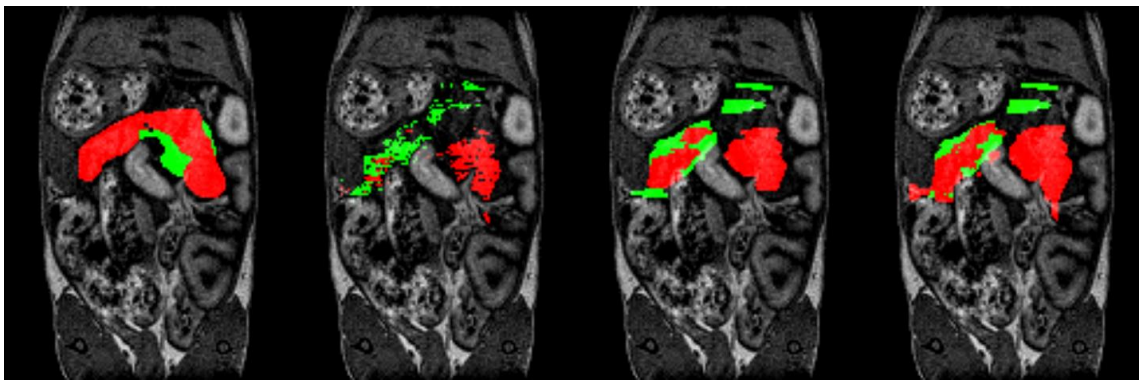


Figure 3-18: Shows model output with different forms of post processing. (a) Expert segmentation used as gold standard for model post processing comparison. (b) Direct model output has unconnected areas and does not segment large amounts of the tumour or pancreas. (c) Close geometric mean open threshold smooth cluster segmentation has closed shapes closer to the expert segmentation. (d) Threshold smooth geometric mean cluster

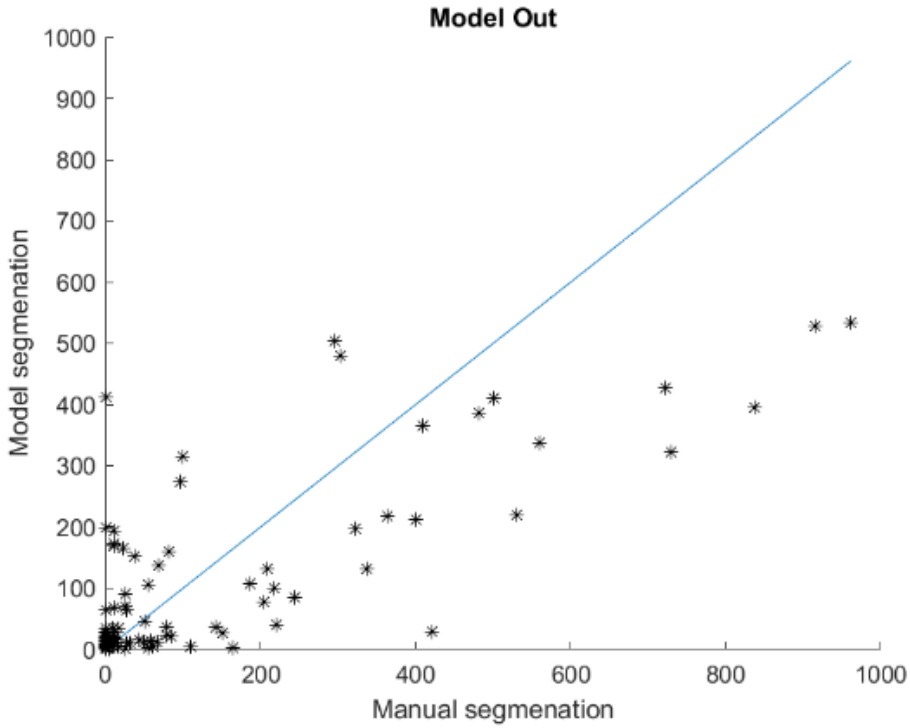
segmentation captures a larger amount of the tumour as well as the greatest stability between scans/subjects.

3.13.1 Level Sets

Level sets were not an appropriate technique for 3D-CAMMP. This is on account of level sets requiring user-input with prior knowledge, as well as the fact that pancreatic tumours tend to be structurally heterogeneous to the extreme, having very irregular patterns, i.e. some areas of smooth curve surface and some coarse sharp objects.

3.13.2 Sequential Post Processing

Multiple post processing techniques were tested, including; Thresholding Smooth Cluster, Close Geometric mean Open Threshold Smooth Connected components, Close Geometric mean Threshold Smooth Connected components, Close Open Threshold Smooth Connected components, Open Close Threshold Smooth Connected components, 3x3 moving filter, and Gaussian Smoothing. Threshold (0.3) smooth cluster has the greatest stability while also optimising the classifications, as shown in Figure 3-18 and Figure 3-19.



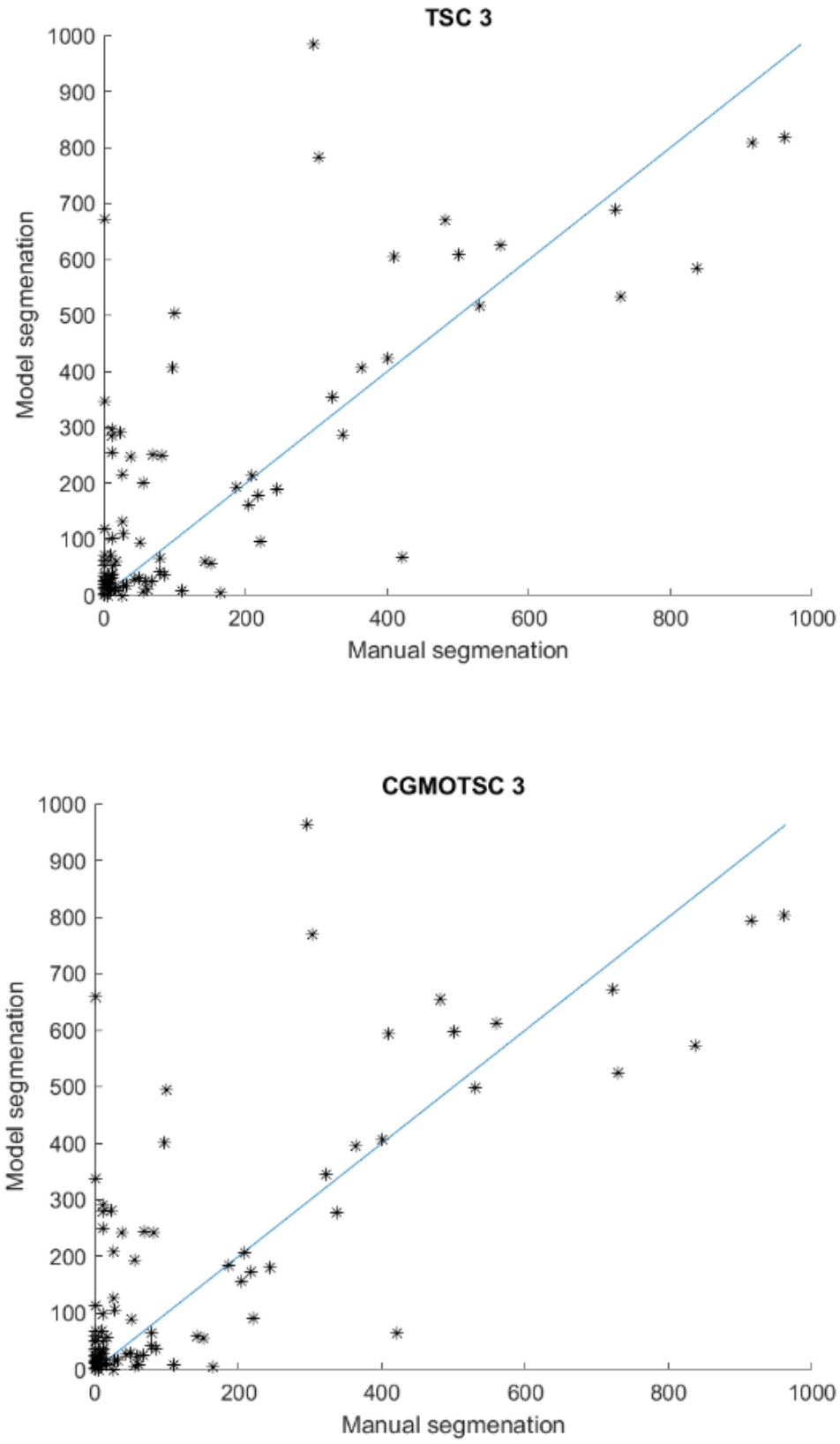


Figure 3-19: Volume of the classified tumours by expert segmentation vs model segmentation. Thresholding at 0.3 Smooth Cluster (TSC 3) has smallest variance along the x=y line and therefore has the closest approximation to expert segmentation of the model output (direct) and all other post processing techniques. Close Geometric Mean Open Threshold at 0.3 Smooth Cluster (CGMOTSC 3) was the next closet post processing technique with an almost

identical result but more than twice the computational time due to the additional steps so was not used.

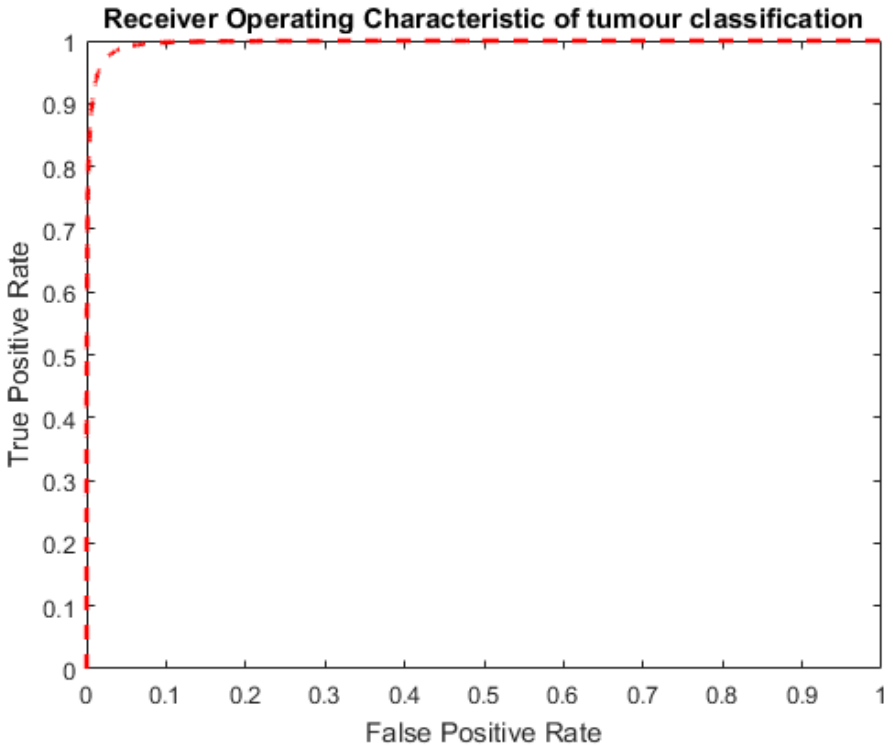


Figure 3-20: The ROC curve of tumour detection by 3D-CAMMP compared to experts. 3D-CAMMP has a sensitivity of 100% and a specificity of 85.7%. Therefore, 3D-CAMMP always detects a tumour that has been identified by expert segmentation and has a high rate of not identifying animals without tumours segmented.

3.14 3D-CAMMP

The final version of 3D-CAMMP is able to take a novel MR image of a KPC mouse abdomen and segment any pancreatic tumours, the pancreas, liver, kidneys, spleen, stomach, gall bladder, and hepatic portal vein accurately and reliably with less user input than an expert user (e.g. someone with multiple years of training/experience in identifying tumours in preclinical images, in this case MRI).

Expert image analysis was used to define ground truth. However, when multiple expert users were asked to segment the same image, there were often discrepancies between the individual segmentations. To mitigate this risk, after each scan had been segmented by the three expert users, a round-table meeting was held to discuss the individual segmentation with the objective

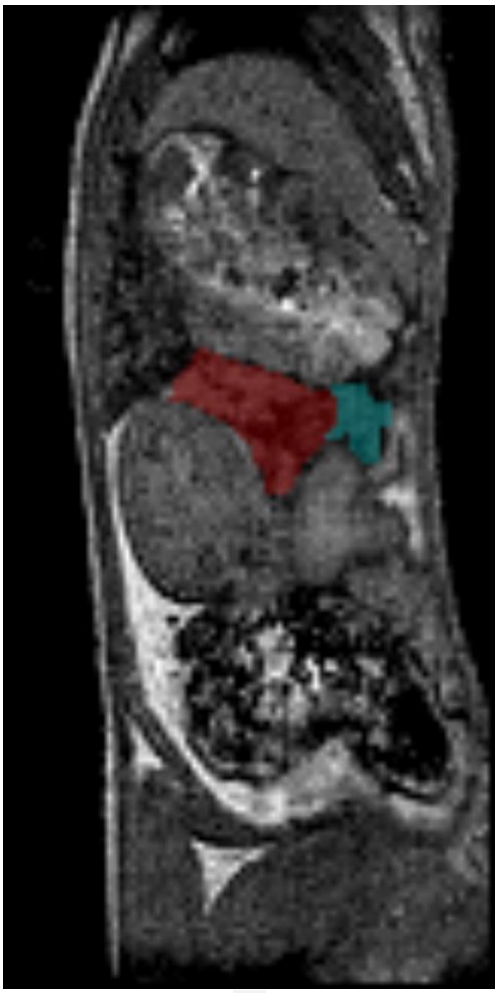
being to come to a consensus as to what the final segmentation should be. Due to the variable nature of the images and the fact that 3D data can be viewed at multiple angles, areas were sometimes missed by one expert and noticed by another. In almost all cases, the tumour area was expanded after discussion. See Figure 3-21 for details.

3D-CAMMP outperforms non-expert users consistently and obtains a DSC score with expert users of 0.76 for tumour segmentation and 0.65 for pancreas segmentation. This DSC score is as high as possible since 3D-CAMMP was in some cases able to detect tumours missed by the expert users. Receiver Operating Characteristic curve (ROC curve) analysis of 3D-CAMMP on tumour existence shows a sensitivity of 100% meaning that an animal with a tumour will always identified and a specificity of 85.7% meaning that an animal without a tumour may be identified as having a tumour, with an overall Area Under the Curve (AUC) of 0.93, as shown in Figure 3-20. This is the best-case scenario as it is advantageous for 3D-CAMMP to be over- rather than under-sensitive as user checks can more easily identify mistakes. However, when the incorrect classifications are manually checked by expert users it was found that 83% of the time 3D-CAMMP had identified small tumours that had been missed by the expert users, shown in Figure 3-22. This shows the validity of 3D-CAMMP and reiterates the difficulty of the task even for expert users.

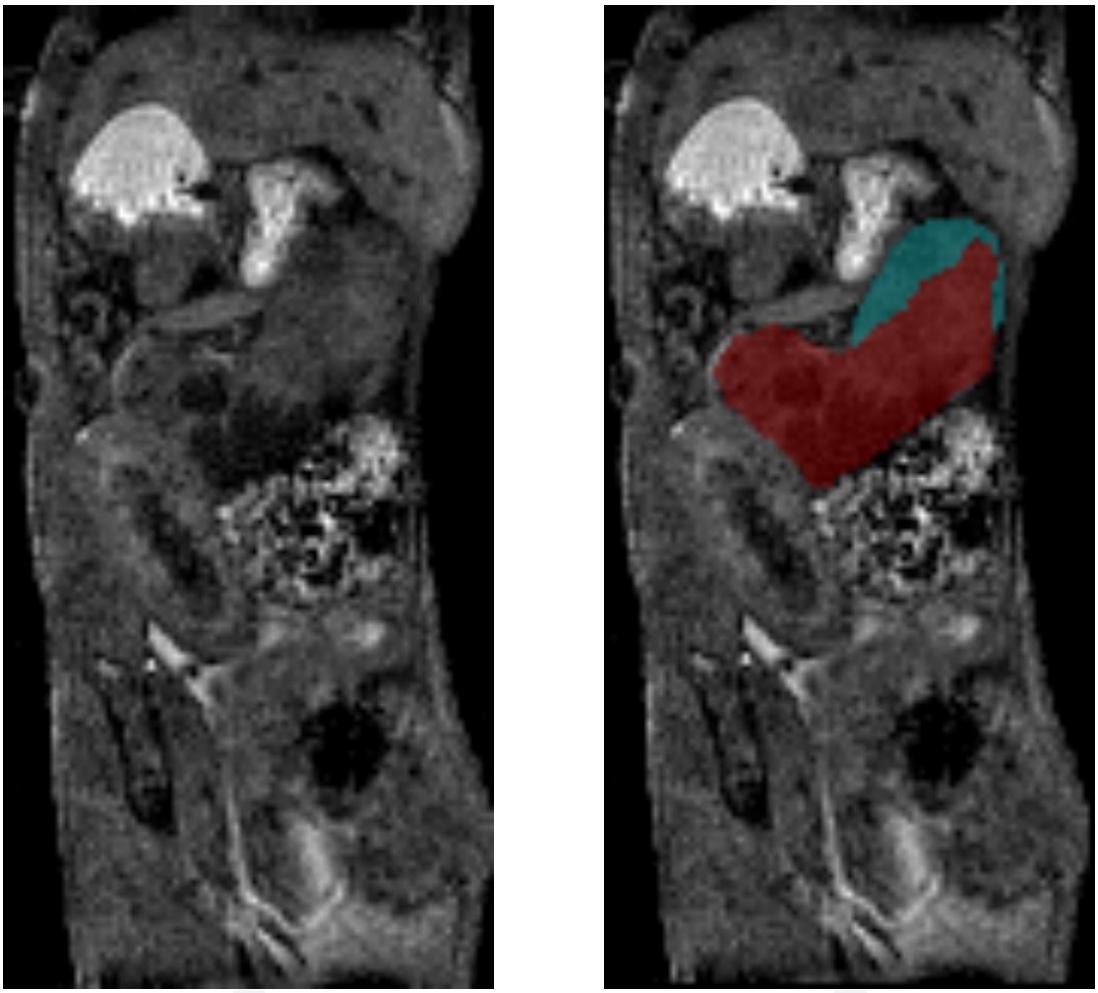
Furthermore, 3D-CAMMP is able to perform the analysis within approximately 1-2 hours. By comparison expert users could take roughly the same amount of time but in some cases took up to a total of 10 hours for some images for which 3D-CAMMP took no extra time and was comparable to the 10-hour segmentation by two expert users. The 1-2-hour process performed by 3D-CAMMP is made up of 2 parts. Stage 1, data conversion, the bias field correction, normalisation and multi atlas segmentation. This takes up to 70% of the runtime. Stage 2, creation of the bounding area, normalisation, feature generation and classification. In between these two stages, manual user input is required in order to check multi atlas segmentation. For this reason, 3D-CAMMP will run every novel scan available through stage 1 before requesting

user input. This means that 3D-CAMMP can be initiated and left while it completes stage 1 for all scans, then the user can check all the data at once and initiate stage 2. This means the user does not have to constantly check 3D-CAMMP and no time is lost when the model is left to run.

(a)



(b)



(c) (d)

Figure 3-21 shows the sagittal slice of two KPC mice with pancreatic tumours. (a) and (c) show the novel images. (b) and (d) show areas classified as tumour by all three experts in red and areas discussed at the round table meeting and then classified as tumour in teal.

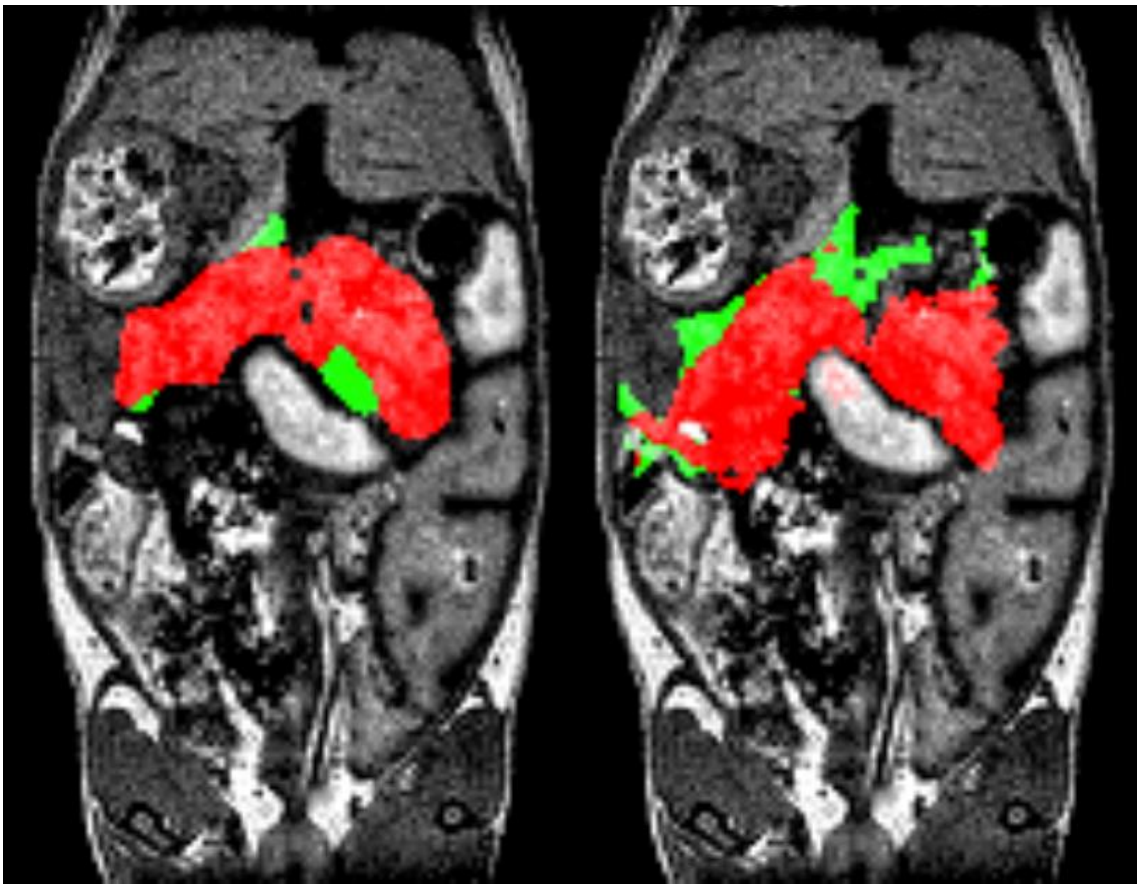


Figure 3-22: The expert segmentation (left) of tumour (red) and pancreas (green) compared to the final 3D-CAMMP output (right). The tumour identified by 3D-CAMMP has twice the volume and the pancreas 1.5 times the volume. This is an acceptable variance as it has been shown that human segmentation in general misses portions of the tumour segmentation.

A combination of 3D-CAMMP and expert user proved to be the most effective segmentation combination. 3D-CAMMP and new user greatly outperforms new users alone and is relatively similar to the classification of an expert user alone.

A version of 3D-CAMMP is also able to segment the pancreas, liver, kidneys, spleen, stomach, gall bladder, and hepatic portal vein accurately and reliably in C57BL/6 mice using the same technique.

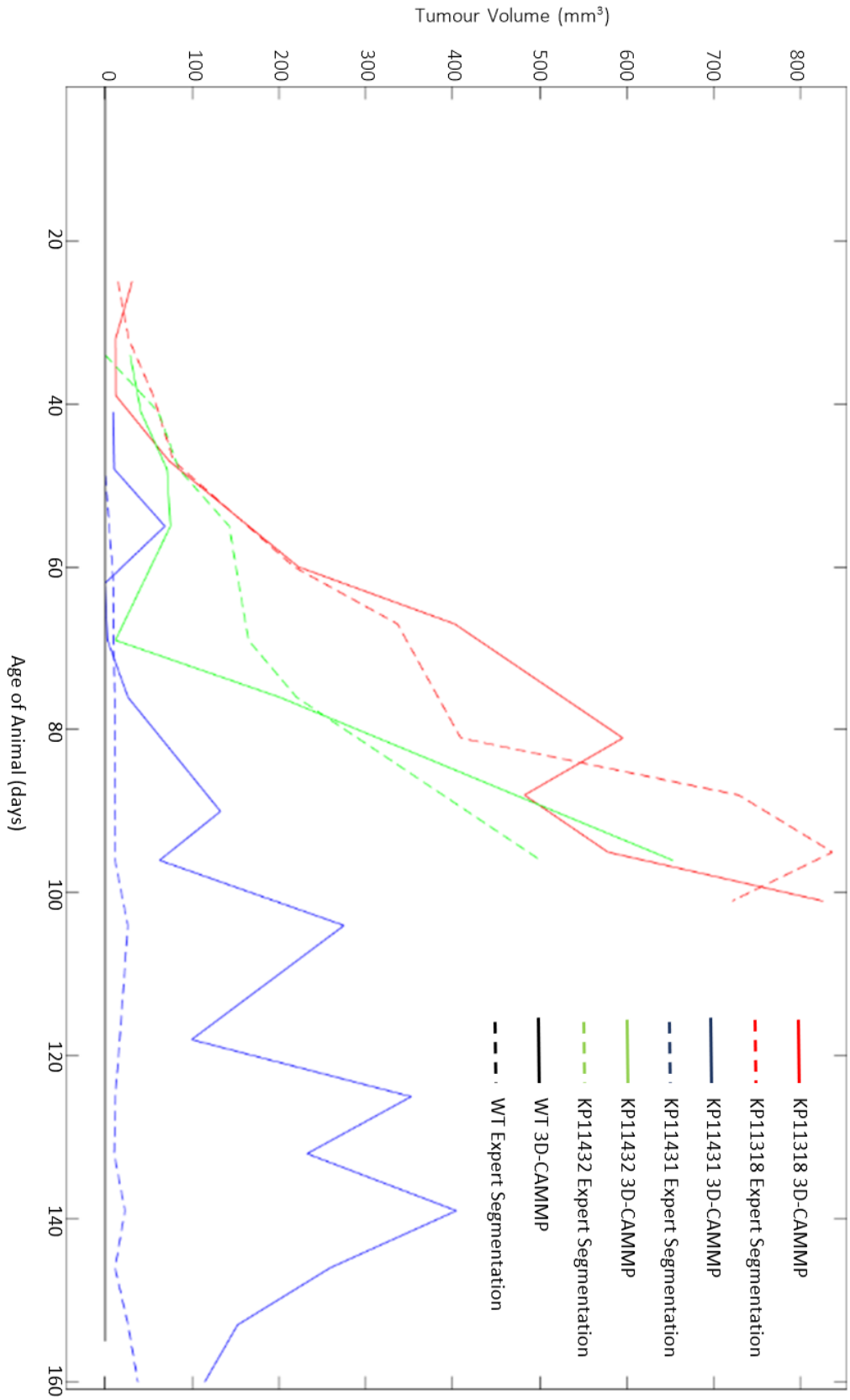


Figure 3-23: Tumour volume classified by both 3D-CAMMP and expert users. It can be seen that the classifications are very similar though 3D-CAMMP is often able to segment the tumours earlier than manual expert segmentation. Furthermore, 3D-CAMMP does not segment any tumours in the KP or WT animals throughout the study.

3D-CAMMP can also be used by expert users to give more confidence to their segmentations. It was shown that due to the difficulty of pancreatic tumour segmentation that expert users could miss tumours and 3D-CAMMP can provide another stage of analysis to help in identifying these tumours. The final DSC score of 3D-CAMMP was 0.75 when compared to expert users. This score is hindered by the fact that 3D-CAMMP is able to locate tumours that were missed by the experts but upon revisiting the images agreed that they were indeed tumours. 3D-CAMMP was also able to detect tumours at earlier stages than expert users with an average of 3 days earlier, see Figure 3-23.

4 Discussion

A major problem still facing the radiomics field as a whole is the acceptance and change in the culture of the larger medical and biological community. More published proof of radiomics power is constantly being requested, especially on the preclinical side. As more studies are proving the stability and usefulness of radiomics more researchers are brought to understand the power that it can unlock (120).

In this work a computational tool has been developed, termed 3D-CAMMP, that is able to segment multiple organs within the abdomen of a KPC mouse including the mouse pancreas and any pancreatic tumours that form. 3D-CAMMP requires very little user input which makes it accessible to researchers who do not have much experience in image analysis, but nevertheless is still very customisable if the researcher wishes it to be. The mouse pancreas and pancreatic tumours are difficult objects to segment due to their lack of clearly defined borders, spatial variance of the pancreas and the tumours being able to form anywhere in a large area. This coupled with the fact that the abdomen as a whole is extremely variable, depending on the age of the animal, if it has just eaten and even if it has just woken up, creates a difficult set of obstacles to overcome. 3D-CAMMP is built with the work done at Invicro with their Whole Body Atlas segmentation (WBA) tool that is capable of segmenting large organs within the abdomen and through the combination of the WBA, registration and machine learning algorithms this segmentation is possible (80).

Acceptance of radiomics can still be an issue when working with researchers who have never used it before as is the case with all emergent techniques. Early work in the field may also in some cases not be rigorous enough in terms of capturing the relevant information and statistical power (121) Welch *et al* published a study in 2019 showing that not all radiomic features are suitable surrogates for tumour volume measurements (122). A combined method of semi automatic segmentation and manual segmentation has been suggested to combat some of these issues (123). However, as more studies are published proving the robustness of

the techniques and the techniques become more broadly available, a quick shift will occur towards radiomics being used for long term studies as well as to enhance end of study histology (124). The emergent technology of whole animal slicing and histology will also be another major step towards the proof of radiomic features as then histological and non-invasive images can be co-registered to each other much more easily (125). The sharing of data between institutes and groups would be a huge boon to the field of Radiomics and Data Mining as a whole. The availability of data would create a much larger pool of information for researchers to use and could be extremely useful, especially when used for control studies. There is a current trend of sharing data after publication that for the imaging community in particular would be a great step forward. 3D-CAMMP has been built to be used on any workstation that might be available to the user so as not limit its use. However, a version built to use parallel programming, graphic processor programming or even cluster and supercomputer programming would be possible and would allow for the software to be used much more quickly. 3D-CAMMP and other similar tools should also be adapted to try to identify other objects (e.g. tumours, cysts) similar to the original design in either location or radiomics features, as building new tools from the ground up is often not necessary. Furthermore, tools like 3D-CAMMP often collect large amounts of data that may not be used for the original purpose but should in no way be discounted as useless. For example, 3D-CAMMP segments the liver, kidneys and spleen of an animal through its pipeline, all four of which show measurable anatomical changes throughout tumour development and so should be studied further as the data has already been collected but just remains un-analysed. This is entirely consistent with the 3R's aim of obtaining more data from each animal.

In total 30 animals where used in this research. 17 KPC, 8 KP and 5 WT where imaged a total of 300 times. None of the 30 animals used where bread specifically for this research.

This work has been presented at multiple conferences and outreach events including:

- European Molecular Imaging Meeting (EMIM), 2019. (Conference and Oral presentation).
- World Molecular Imaging Congress (WMIC), Seattle 2018. (Conference, Oral and Poster presentation, awarded best poster).
- Barts Cancer Institute PhD Day (BCI PhD Day), Queen Mary University, 2018, (Conference, Poster presentation, awarded best poster).
- London Pancreas Workshop, 2018, (Conference, Poster presentation, awarded one of top 5 abstracts).
- European Society of Magnetic Resonance in Medicine and Biology (ESMIMB), 2017, (Conference, Oral and Poster presentation)
- Barts Cancer Institute PhD Day, Queen Mary University, 2017, (Conference, Poster presentation, awarded best poster).
- Life Science Initiative, Queen Mary University, 2016, (Conference, Poster presentation, awarded best poster).
- STEMNET: Over 100 hours of public engagement and outreach, including school visits, presentations and tours.

3D-CAMMP has also received over £80,000 in grants including over £5,000 in travel grants from national and international agencies and a £75,000 NC3Rs Skills and Knowledge Transfer Grant - £75,000 to disseminate the research and have more institutes using 3D-CAMMP (The Francis Crick Institute, Cambridge University, Glasgow University and Queen Mary University of London).

5 Conclusion and Further work

As set out in section 1.23, the aim of this project was to build a computational tool that will automatically segment out the pancreas and any pancreatic tumours from an MRI scan, providing a reliable and reproducible image analysis method for volume and surface area quantification of the pancreas and pancreatic tumours, reducing effects of user discretion and thus improving the accuracy of the data.

Measured against these criteria, it has been shown that the computational model 3D-CAMMP can perform more consistently and more accurately than an expert image analyst. Furthermore, it takes less training for the user than either US or manual MRI segmentation.

The time cost of performing the MRI and running 3D-CAMMP (1-2 hours) is similar to that of performing US combined with the required volumetric analysis (2 hours), and has reduced user training needed (contrast with the operator dependency of US) and the fact that 3D-CAMMP does not need to be monitored for the majority of the time it is running (actual user time 1 hour). 3D-CAMMP offers a much more accurate, fast and cheaper way to analyse MRI images than having a dedicated expert user or training new users in image analysis. In order to reduce the amount of time for which 3D-CAMMP needs to be supervised further, an email alert could be set up in the user interface to inform users when the quality control step is required and when 3D-CAMMP has finished the classifications. 3D-CAMMP removes the need for dissection at every time point, similar to US, and in doing so reduces the issues of inter-animal variability, improving statistics for longitudinal studies, reduces group size and the cost of breeding and maintaining animals. A study was conducted between image analysis experts (n=3) and image analysis novices (n=3). The image analysis novices were given a four-hour training session on the segmentation of pancreatic tumours in KPC mice. Each group was given three images to segment. On average the image analysis experts were able to segment the pancreatic tumours within 1 hour per image with a high level of confidence. However, the image analysis novices

took on average 4 hours with a low level of confidence in the segmentation. Typical segmentations can be seen in Figure 1-15.

The reduction in group size has a substantial effect on the breeding due to fewer than 1 in 4 of the animals bred in this model having the correct genotype. Therefore, a reduction of group size from 12 to 8 in a study with 4 groups would reduce the number of animals bred by 64. Furthermore, the animals do not need to be age matched and can be inducted into a study when the individual animal has tumours of a predetermined size. This helps improve the statistical accuracy and means that a smaller colony can be used for the same studies.

3D-CAMMP has been developed on a low field MRI (Brucker ICON/Aspect 1T). This allows for the imaging protocols to be used on any MRI with a field strength of 1T or greater with minimal development. Contrast this with protocols that are developed on higher field machines that cannot always be transferred to lower field machines due to constraints of the field. Furthermore, this allows an institute to be able to purchase a low field machine (or use the low field MRI component of a hybrid system) and still be able to perform the protocols needed to use 3D-CAMMP.

Finally, any higher field MRI will be able to produce the image quality needed for 3D-CAMMP in a faster time than a low field MRI with similar tissue contrast (T1 weighted FLASH) and without any addition of contrast agent. This tool is now being implemented at centres with high field MRI.

To further improve this model, a larger set of features could be identified to increase the amount of points that the machine learning model must work from.

If the number of features is increased, then the use of feature selection may become relevant.

3D-CAMMP could be run faster using Parallel processing, GPU processing or even a cluster or supercomputer. However, if 3D-CAMMP was built in this way it would reduce its usability as it

would require anyone using it to have the correct hardware and knowledge of how to set it up. As it stands, 3D-CAMMP can be run on any computer, giving it much wider applicability.

In addition, and in parallel with the field of radiomics in humans, correlation of the outputs of 3D-CAMMP need to be validated through collection of pancreatic and pancreatic tumour histology data from scanned animals. This will provide reassurance for researchers that the model outputs correlate not only with imaging experts, but also are in agreement with results obtained from whole body tissue staining.

An NC3Rs' Skills and Knowledge Transfer Grant has been secured to implement 3D-CAMMP at Barts Cancer Institute (QMUL), The Francis Crick Institute, Cambridge University and The Beatson (Glasgow University) with collaborations with Invicro (USA). Each site will be visited over the next year and appropriate MRI protocols will be designed as well as training and support provided for the new users. Furthermore, the User Interface of 3D-CAMMP will be upgraded to make it more accessible. Each institute has also agreed that data collected using these protocols will be available online to the wider scientific community, after publication of the data to allow researchers from all over the world to use 3D-CAMMP with a large library of data and reduce the number of control experiments needed. 3D-CAMMP in this time will also be further developed and tested on its ability to detect metastatic tumours within the KPC model as well as detection of tumours in other genetically modified mouse models, for pancreatic cancer and beyond.

6 References

1. Cancer Research UK. Pancreatic cancer survival statistics: Cancer Research UK; 2015 [updated Updated:2015-05-15. Available from: http://www.cancerresearchuk.org/health-professional/cancer-statistics/statistics-by-cancer-type/pancreatic-cancer/survival.
2. Westphalen CB, Olive KP. Genetically engineered mouse models of pancreatic cancer. Cancer journal (Sudbury, Mass). 2012;18(6):502.
3. Cook MJ. The Anatomy of the Laboratory Mouse: Academic Press; 2016 [updated Updated:2008-02. Available from: http://www.informatics.jax.org/cookbook/.
4. Kersten K, de Visser KE, van Miltenburg MH, Jonkers J. Genetically engineered mouse models in oncology research and cancer medicine. EMBO Mol Med. 2017;9(2):137-53.
5. Hingorani SR, Petricoin EF, Maitra A, Rajapakse V, King C, Jacobetz MA, et al. Preinvasive and invasive ductal pancreatic cancer and its early detection in the mouse. Cancer cell. 2003;4(6):437-50.
6. Hingorani SR, Wang L, Multani AS, Combs C, Deramaudt TB, Hruban RH, et al. Trp53R172H and KrasG12D cooperate to promote chromosomal instability and widely metastatic pancreatic ductal adenocarcinoma in mice. Cancer cell. 2005;7(5):469-83.
7. Orth M, Metzger P, Gerum S, Mayerle J, Schneider G, Belka C, et al. Pancreatic ductal adenocarcinoma: biological hallmarks, current status, and future perspectives of combined modality treatment approaches. Radiation Oncology. 2019;14(1):141.
8. Sastra SA, Olive KP. Quantification of murine pancreatic tumors by high-resolution ultrasound. Methods Mol Biol. 2013;980:249-66.
9. Albanese C, Rodriguez OC, VanMeter J, Fricke ST, Rood BR, Lee Y, et al. Preclinical magnetic resonance imaging and systems biology in cancer research: current applications and challenges. Am J Pathol. 2013;182(2):312-8.
10. Richmond A, Su Y. Mouse xenograft models vs GEM models for human cancer therapeutics. Disease Models & Mechanisms. 2008;1(2-3):78-82.
11. Lee JW, Komar CA, Bengsch F, Graham K, Beatty GL. Genetically Engineered Mouse Models of Pancreatic Cancer: The KPC Model (LSL-Kras(G12D/+) ;LSL-Trp53(R172H/+) ;Pdx-1-Cre), Its Variants, and Their Application in Immuno-oncology Drug Discovery. Current protocols in pharmacology. 2016;73:14.39.1-14.39.20.
12. Carapuça EF, Gemenetzidis E, Feig C, Bapiro TE, Williams MD, Wilson AS, et al. Anti-stromal treatment together with chemotherapy targets multiple signalling pathways in pancreatic adenocarcinoma. The Journal of pathology. 2016;239(3):286-96.
13. Gopinathan A, Morton JP, Jodrell DI, Sansom OJ. GEMMs as preclinical models for testing pancreatic cancer therapies. Dis Model Mech. 2015;8(10):1185-200.
14. Matzke-Ogi A, Jannasch K, Shatirishvili M, Fuchs B, Chiblak S, Morton J, et al. Inhibition of Tumor Growth and Metastasis in Pancreatic Cancer Models by Interference With CD44v6 Signaling. Gastroenterology. 2016;150(2):513-25.e10.
15. Majumder K, Arora N, Modi S, Chugh R, Nomura A, Giri B, et al. A Novel Immunocompetent Mouse Model of Pancreatic Cancer with Robust Stroma: a Valuable Tool for Preclinical Evaluation of New Therapies. Journal of gastrointestinal surgery : official journal of the Society for Surgery of the Alimentary Tract. 2016;20(1):53-65; discussion
16. Pitarresi JR, Liu X, Sharma SM, Cuitino MC, Kladney RD, Mace TA, et al. Stromal ETS2 Regulates Chemokine Production and Immune Cell Recruitment during Acinar-to-Ductal Metaplasia. Neoplasia (New York, NY). 2016;18(9):541-52.
17. Pope JA. Medical Physics: Imaging: Pearson Education; 1999.
18. Yamada H. Visual information for categorizing facial expression of emotions. Applied Cognitive Psychology. 1993;7(3):257-70.
19. Gurcan MN, Sahiner B, Petrick N, Chan H-P, Kazerooni EA, Cascade PN, et al. Lung nodule detection on thoracic computed tomography images: Preliminary evaluation of a computer-aided diagnosis system. Medical Physics. 2002;29(11):2552-8.

20. Schmid-Saugeon P, Guillod J, Thiran J-P. Towards a Computer-Aided Diagnosis System for Pigmented Skin Lesions. 2003.
21. Verma B, Zakos J. A computer-aided diagnosis system for digital mammograms based on fuzzy-neural and feature extraction techniques. IEEE transactions on information technology in biomedicine. 2001;5(1):46-54.
22. Shen J, Baum T, Cordes C, Ott B, Skurk T, Kooijman H, et al. Automatic segmentation of abdominal organs and adipose tissue compartments in water-fat MRI: Application to weight-loss in obesity. European Journal of Radiology. 2016;85(9):1613-21.
23. Cai J, Lu L, Zhang Z, Xing F, Yang L, Yin Q, editors. Pancreas Segmentation in MRI Using Graph-Based Decision Fusion on Convolutional Neural Networks. International Conference on Medical Image Computing and Computer-Assisted Intervention; 2016: Springer.
24. Lambin P, Dubois L, Aerts H, Eriksson J, Windhorst B, De Ruysscher D, et al. Innovative molecular imaging approaches for radiation oncology: new PET biomarkers and. Strahlentherapie und Onkologie. 2011;187(9):594-.
25. Lambin P, Rios-Velazquez E, Leijenaar R, Carvalho S, Van Stiphout RG, Granton P, et al. Radiomics: extracting more information from medical images using advanced feature analysis. European journal of cancer. 2012;48(4):441-6.
26. Segal E, Sirlin CB, Ooi C, Adler AS, Gollub J, Chen X, et al. Decoding global gene expression programs in liver cancer by noninvasive imaging. Nature biotechnology. 2007;25(6):675.
27. Diehn M, Nardini C, Wang DS, McGovern S, Jayaraman M, Liang Y, et al. Identification of noninvasive imaging surrogates for brain tumor gene-expression modules. Proceedings of the National Academy of Sciences. 2008;105(13):5213-8.
28. Gollub MJ, Panicek DM, Bach AM, Penalver A, Castellino RA. Clinical importance of reinterpretation of body CT scans obtained elsewhere in patients referred for care at a tertiary cancer center. Radiology. 1999;210(1):109-12.
29. Bechtold RE, Chen MY, Ott DJ, Zagoria RJ, Scharling ES, Wolfman NT, et al. Interpretation of abdominal CT: analysis of errors and their causes. Journal of computer assisted tomography. 1997;21(5):681-5.
30. Loughrey GJ, Carrington BM, Anderson H, Dobson M, Ping FLY. The value of specialist oncological radiology review of cross-sectional imaging. Clinical radiology. 1999;54(3):149-54.
31. Siewert B, Sosna J, McNamara A, Raptopoulos V, Kruskal JB. Missed lesions at abdominal oncologic CT: lessons learned from quality assurance. Radiographics. 2008;28(3):623-38.
32. Kumar V, Gu Y, Basu S, Berglund A, Eschrich SA, Schabath MB, et al. Radiomics: the process and the challenges. Magnetic resonance imaging. 2012;30(9):1234-48.
33. Alic L, Haeck J, Klein S, Bol K, Van Tiel S, Wielepolski PA, et al. Multi-modal image registration: matching MRI with histology, 2010.
34. Lam S-C, editor Texture feature extraction using gray level gradient based co-occurence matrices. 1996 IEEE International Conference on Systems, Man and Cybernetics Information Intelligence and Systems (Cat No 96CH35929); 1996: IEEE.
35. Haralick RM, Shanmugam K. Textural features for image classification. IEEE Transactions on systems, man, and cybernetics. 1973(6):610-21.
36. Galloway MM. Texture analysis using grey level run lengths. NASA STI/Recon Technical Report N. 1974;75.
37. Castellano G, Bonilha L, Li L, Cendes F. Texture analysis of medical images. Clinical radiology. 2004;59(12):1061-9.
38. Zinovev D, Raicu D, Furst J, Armato III S. Predicting radiological panel opinions using a panel of machine learning classifiers. Algorithms. 2009;2(4):1473-502.
39. Soh L-K, Tsatsoulis C. Texture analysis of SAR sea ice imagery using gray level co-occurrence matrices. IEEE Transactions on geoscience and remote sensing. 1999;37(2):780-95.

40. Suárez J, Gancedo E, Álvarez JM, Morán A. Optimum compactness structures derived from the regular octahedron. Engineering Structures. 2008;30(11):3396-8.
41. Tang X. Texture information in run-length matrices. IEEE transactions on image processing. 1998;7(11):1602-9.
42. Aspect. Aspect SimPET [Available from: https://www.aspectimaging.com/pre-clinical-mri/simpet-simultaneous-pet-mri-complete-solution/.
43. Bruker. PET/MR 3T [Available from: https://www.bruker.com/nc/products/preclinical-imaging/nuclear-molecular-imaging/petmr-3t.html.
44. Mediso. nanoScan PET/MRI [Available from: https://www.medisousa.com/preclinical/nanoscan/pet-mri.
45. Solutions M. MRS*SPECT/CT/MR [Available from: https://www.mrsolutions.com/molecular-imaging-main/molecular-imaging/spect-ct-mr/.
46. James ML, Gambhir SS. A molecular imaging primer: modalities, imaging agents, and applications. Physiol Rev. 2012;92(2):897-965.
47. Noone TC, Semelka RC, Chaney DM, Reinhold C. Abdominal imaging studies: comparison of diagnostic accuracies resulting from ultrasound, computed tomography, and magnetic resonance imaging in the same individual. Magn Reson Imaging. 2004;22(1):19-24.
48. Schmid A, Schmitz J, Mannheim JG, Maier FC, Fuchs K, Wehrl HF, et al. Feasibility of Sequential PET/MRI Using a State-of-the-Art Small Animal PET and a 1 T Benchtop MRI. Molecular Imaging and Biology. 2013;15(2):155-65.
49. Westbrook C, Roth CK. MRI in Practice: John Wiley & Sons; 2011.
50. Westbrook C, Roth CK, Talbot J. MRI in Practice, Chapter 1-3: Wiley; 2011.
51. Curry TS, Dowdey JE, Murry RC. Christensen's Physics of Diagnostic Radiology: Lea & Febiger; 1990.
52. Slichter CP. Principles of magnetic resonance: Springer Science & Business Media; 2013.
53. J. H, A. N, H. F. RARE imaging: A fast imaging method for clinical MR. Magnetic Resonance in Medicine. 1986;3(6):823-33.
54. Haase A, Frahm J, Matthaei D, Hänicke W, Merboldt KD. FLASH imaging: Rapid NMR imaging using low flip-angle pulses. Journal of Magnetic Resonance. 2011;213(2):533-41.
55. Marzola P, Osculati F, Sbarbati A. High field MRI in preclinical research. European journal of radiology. 2003;48(2):165-70.
56. Juntu J, Sijbers J, Van Dyck D, Gielen J, editors. Bias Field Correction for MRI Images2005; Berlin, Heidelberg: Springer Berlin Heidelberg.
57. Song S, Zheng Y, He Y. A review of Methods for Bias Correction in Medical Images. Biomedical Engineering Review. 2017;1(1).
58. Tustison NJ, Avants BB, Cook PA, Zheng Y, Egan A, Yushkevich PA, et al. N4ITK: Improved N3 Bias Correction. IEEE transactions on medical imaging. 2010;29(6):1310-20.
59. Sonka M, Hlavac V, Boyle R. Image processing, analysis, and machine vision. Fourth edition, International edition. Edu 2014.
60. Joshi M, Cui J, Doolittle K, Joshi S, Van Essen D, Wang L, et al. Brain segmentation and the generation of cortical surfaces. NeuroImage. 1999;9(5):461-76.
61. Wells WM, Grimson WEL, Kikinis R, Jolesz FA. Adaptive segmentation of MRI data. Medical Imaging, IEEE Transactions on. 1996;15(4):429-42.
62. Weszka JS, Nagel RN, Rosenfeld A. A Threshold Selection Technique. IEEE Trans Comput. 1974;23(12):1322-6.
63. Otsu N. A Threshold Selection Method from Gray-Level Histograms. IEEE Transactions on Systems, Man, and Cybernetics. 1979;9(1):62-6.
64. Tsai Y-H, editor A new approach for image thresholding under uneven lighting conditions. 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007); 2007: IEEE.
65. Vala HJ, Baxi A. A review on Otsu image segmentation algorithm.
66. Wong K-P. Medical image segmentation: methods and applications in functional imaging. Handbook of biomedical image analysis: Springer; 2005. p. 111-82.

67. Mittelhäußer G, Kruggel F. Fast Segmentation of Brain Magnetic Resonance Tomograms. In: Ayache N, editor. Computer Vision, Virtual Reality and Robotics in Medicine: First International Conference, CVRMed '95, Nice, France, April 3–6, 1995 Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg; 1995. p. 237-41.
68. Kaus MR, Warfield SK, Nabavi A, Black PM, Jolesz FA, Kikinis R. Automated segmentation of mr images of brain tumors 1. Radiology. 2001;218(2):586-91.
69. Bieniecki W, editor Oversegmentation avoidance in watershed-based algorithms for color images. Modern Problems of Radio Engineering, Telecommunications and Computer Science, 2004 Proceedings of the International Conference; 2004: IEEE.
70. Letteboer M, Niessen W, Willems P, Dam EB, Viergever M, editors. Interactive multi-scale watershed segmentation of tumors in MR brain images. Proc of the IMIVA workshop of MICCAI; 2001.
71. Dam E, Loog M, Letteboer M, editors. Integrating automatic and interactive brain tumor segmentation. Pattern Recognition, 2004 ICPR 2004 Proceedings of the 17th International Conference on; 2004: IEEE.
72. Cates JE, Whitaker RT, Jones GM. Case study: an evaluation of user-assisted hierarchical watershed segmentation. Medical Image Analysis. 2005;9(6):566-78.
73. Ratan R, Sharma S, Sharma SK. Multiparameter Segmentation and Quantization of Brain Tumor from MRI Images2009.
74. Salman S, Bahrani AA. Segmentation of tumor tissue in gray medical images using watershed transformation method. Computer Science and Applied Mathematics. 2010(2):123--7.
75. Maintz JA, Viergever MA. A survey of medical image registration. Medical image analysis. 1998;2(1):1-36.
76. Cuadra MB, Pollo C, Bardera A, Cuisenaire O, Villemure J-G, Thiran J-P. Atlas-based segmentation of pathological MR brain images using a model of lesion growth. IEEE transactions on medical imaging. 2004;23(10):1301-14.
77. Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the EM algorithm. Journal of the royal statistical society Series B (methodological). 1977:1-38.
78. Moon N, Bullitt E, Van Leemput K, Gerig G, editors. Automatic brain and tumor segmentation. International Conference on Medical Image Computing and Computer-Assisted Intervention; 2002: Springer.
79. Prastawa M, Bullitt E, Moon N, Van Leemput K, Gerig G. Automatic brain tumor segmentation by subject specific modification of atlas priors 1. Academic radiology. 2003;10(12):1341-8.
80. Hesterman J, Ghayoor A, Novicki A, Wang X, Cadoret Y, Becerra L, et al. Multi-atlas Approaches for Image Segmentation across Modality, Species and Application Area. future. 2019;6:7..
81. Invicro. Multi-Atlas Segmentation Tool [Available from: https://invicro.com/capabilities/software/multi-atlas-segmentation-tool/.
82. Ekin A, editor Pathology-Robust MR Intensity Normalization With Global And Local Constraints. 2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro; 2011 March 30 2011-April 2 2011.
83. Brankov JG. Evaluation of the channelized Hotelling observer with an internal-noise model in a train-test paradigm for cardiac SPECT defect detection. Phys Med Biol. 2013;58(20):7159-82.
84. Shidahara M, Inoue K, Maruyama M, Watabe H, Taki Y, Goto R, et al. Predicting human performance by channelized Hotelling observer in discriminating between Alzheimer's dementia and controls using statistically processed brain perfusion SPECT. Annals of nuclear medicine. 2006;20(9):605.
85. Gonzalez RC, Woods RE. Digital image processing. 2002.

86. Yuh EL, Cooper SR, Mukherjee P, Yue JK, Lingsma HF, Gordon WA, et al. Diffusion tensor imaging for outcome prediction in mild traumatic brain injury: a TRACK-TBI study. Journal of neurotrauma. 2014;31(17):1457-77.
87. Bishop CM. Pattern recognition. Machine Learning. 2006;128:1-58.
88. Alpaydin E. Introduction to Machine Learning: The MIT Press; 2010. 584 p.
89. Kotsiantis S. Supervised Machine Learning: A Review of Classification Techniques. Informatica (Ljubljana). 2007;31.
90. John Lu Z. The elements of statistical learning: data mining, inference, and prediction. Journal of the Royal Statistical Society: Series A (Statistics in Society). 2010;173(3):693-4.
91. Le QV, editor Building high-level features using large scale unsupervised learning. 2013 IEEE international conference on acoustics, speech and signal processing; 2013: IEEE.
92. Chapelle O, Schlkopf B, Zien A. Semi-Supervised Learning: The MIT Press; 2010. 528 p.
93. Christakou C, Lefakis L, Vrettos S, Stafylopatis A, editors. A Movie Recommender System Based on Semi-supervised Clustering. International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06); 2005 28-30 Nov. 2005.
94. Li CH, Yuen PC. Semi-supervised Learning in Medical Image Database. In: Cheung D, Williams GJ, Li Q, editors. Advances in Knowledge Discovery and Data Mining: 5th Pacific-Asia Conference, PAKDD 2001 Hong Kong, China, April 16–18, 2001 Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg; 2001. p. 154-60.
95. Xu M, Franti P, editors. A heuristic K-means clustering algorithm by kernel PCA. Image Processing, 2004 ICIP'04 2004 International Conference on; 2004: IEEE.
96. Bezdek JC. Pattern recognition with fuzzy objective function algorithms: Springer Science & Business Media; 2013.
97. Vapnik V. The nature of statistical learning theory: Springer science & business media; 2013.
98. Zhou J, Chan KL, Chong VF, Krishnan SM. Extraction of brain tumor from MR images using one-class support vector machine. Conference proceedings : Annual International Conference of the IEEE Engineering in Medicine and Biology Society IEEE Engineering in Medicine and Biology Society Annual Conference. 2005;6:6411-4.
99. Cai H, Verma R, Ou Y, Lee S-k, Melhem ER, Davatzikos C, editors. Probabilistic segmentation of brain tumors based on multi-modality magnetic resonance images. Biomedical Imaging: From Nano to Macro, 2007 ISBI 2007 4th IEEE International Symposium on; 2007: IEEE.
100. Ruan S, Lebonvallet S, Merabet A, Constans Jm, editors. Tumor Segmentation From A Multispectral Mri Images By Using Support Vector Machine Classification. 2007 4th IEEE International Symposium on Biomedical Imaging: From Nano to Macro; 2007 12-15 April 2007.
101. Verma R, Zacharaki EI, Ou Y, Cai H, Chawla S, Lee SK, et al. Multiparametric tissue characterization of brain neoplasms and their recurrence using pattern classification of MR images. Acad Radiol. 2008;15(8):966-77.
102. Zhang N, Ruan S, Lebonvallet S, Liao Q, Zhu Y, editors. Multi-kernel SVM based classification for brain tumor segmentation of MRI multi-sequence. Image Processing (ICIP), 2009 16th IEEE International Conference on; 2009: IEEE.
103. Bauer S, Nolte LP, Reyes M. Fully automatic segmentation of brain tumor images using support vector machine classification in combination with hierarchical conditional random field regularization. Medical image computing and computer-assisted intervention : MICCAI International Conference on Medical Image Computing and Computer-Assisted Intervention. 2011;14(Pt 3):354-61.
104. Tibshirani R, Hastie T, Friedman J. Regularized Paths for Generalized Linear Models Via Coordinate Descent. Journal of Statistical Software. 2010;33.
105. Dudani SA. The Distance-Weighted k-Nearest-Neighbor Rule. IEEE Transactions on Systems, Man, and Cybernetics. 1976;SMC-6(4):325-7.
106. Liaw A, Wiener M. Classification and Regression by RandomForest. Forest. 2001;23.

107. Strobl C, Boulesteix A-L, Zeileis A, Hothorn T. Bias in random forest variable importance measures: Illustrations, sources and a solution. BMC Bioinformatics. 2007;8(1):25.
108. Day C-P, Merlino G, Van Dyke T. Preclinical mouse cancer models: a maze of opportunities and challenges. Cell. 2015;163(1):39-53.
109. NC3Rs. The 3Rs [Available from: https://www.nc3rs.org.uk/the-3rs.
110. Olive KP, Jacobetz MA, Davidson CJ, Gopinathan A, McIntyre D, Honess D, et al. Inhibition of Hedgehog signaling enhances delivery of chemotherapy in a mouse model of pancreatic cancer. Science (New York, NY). 2009;324(5933):1457-61.
111. Husain K, Centeno BA, Chen D-T, Hingorani SR, Sebti SM, Malafa MP. Vitamin E δ-Tocotrienol Prolongs Survival in the LSL-Kras G12D/+LSL-Trp53R172H/+ Pdx-1-Cre (KPC) Transgenic Mouse Model of Pancreatic Cancer. Cancer Prevention Research. 2013;6(10):1074-83.
112. Hermann PC, Sancho P, Cañamero M, Martinelli P, Madriles F, Michl P, et al. Nicotine Promotes Initiation and Progression of KRAS-Induced Pancreatic Cancer via Gata6-Dependent Dedifferentiation of Acinar Cells in Mice. Gastroenterology. 2014;147(5):1119-33.e4.
113. Weissmueller S, Manchado E, Saborowski M, Morris John P, Wagenblast E, Davis Carrie A, et al. Mutant p53 Drives Pancreatic Cancer Metastasis through Cell-Autonomous PDGF Receptor β Signaling. Cell. 2014;157(2):382-94.
114. Kreyszig E. Advanced engineering mathematics: John Wiley & Sons; 2010.
115. Aljabar P, Heckemann RA, Hammers A, Hajnal JV, Rueckert D. Multi-atlas based segmentation of brain images: atlas selection and its effect on accuracy. Neuroimage. 2009;46(3):726-38.
116. Iglesias JE, Sabuncu MR. Multi-atlas segmentation of biomedical images: A survey. Medical Image Analysis. 2015;24(1):205-19.
117. Lancelot S, Roche R, Slimen A, Bouillot C, Levigoureux E, Langlois JB, et al. A multi-atlas based method for automated anatomical rat brain MRI segmentation and extraction of PET activity. PLoS One. 2014;9(10):e109113.
118. Kulkarni S, Khurd P, Hsiao I, Zhou L, Gindi G. A channelized Hotelling observer study of lesion detection in SPECT MAP reconstruction using anatomical priors. Phys Med Biol. 2007;52(12):3601-17.
119. Zwanenburg A, Leger S, Vallières M, Löck S. Image biomarker standardisation initiative. arXiv preprint arXiv:161207003. 2016.
120. Parmar C, Velazquez ER, Leijenaar R, Jermoumi M, Carvalho S, Mak RH, et al. Robust radiomics feature quantification using semiautomatic volumetric segmentation. PloS one. 2014;9(7):e102107.
121. Morin O, Vallières M, Jochems A, Woodruff HC, Valdes G, Braunstein SE, et al. A deep look into the future of quantitative imaging in oncology: a statement of working principles and proposal for change. International Journal of Radiation Oncology* Biology* Physics. 2018;102(4):1074-82.
122. Welch ML, McIntosh C, Haibe-Kains B, Milosevic MF, Wee L, Dekker A, et al. Vulnerabilities of radiomic signature development: The need for safeguards. Radiotherapy and Oncology. 2019;130:2-9.
123. Tixier F, Um H, Young RJ, Veeraraghavan H. Reliability of tumor segmentation in glioblastoma: impact on the robustness of MRI-radiomic features. Medical physics. 2019.
124. Baeßler B, Weiss K, dos Santos DP. Robustness and reproducibility of radiomics in magnetic resonance imaging: a phantom study. Investigative radiology. 2019;54(4):221-8.
125. Brevard M, Farhoud M, Dimant H, Sahagian G, Rosenthal E, Nguyen Q-D. Abstract LB-366: Cryo-fluorescence tomography as a new tool in 3D visualization of tumor heterogeneity, metastatic proliferation and immuno-oncology. AACR; 2018.

Appendix 1: 3D-CAMMP

6.1 ThreeDAMMP.m

```
%% Data Pipeline
% 11/12/2018
function ThreeDCAMMP()
% addpath('Code')
currentFolder = pwd;
masDir = CharParse(currentFolder,'\D',0);


%% Find the maximum magnitude of vectors from centre of mass of organs to general centre
f = 'data/refLib/*1.rmha'; % Get all rmha files
F = dir(f);
a = size(F,1);
mag = zeros(1,a);
for i = 1:a
   fROI = sprintf('data/refLib/%s',F(i).name);
   [ROI,~] = mhdImport(fROI);
   %% Centre of each ROI
   [xN,yN,zN] = deal(zeros(7,1));
   for j = 1:7 %for loop for each ROI
      Roi = ROI;
      Roi(Roi~=j)=0;
      Roi = Roi./j;
      cen = regionprops(Roi,'centroid');
      xN(j) = cen.Centroid(2);
      yN(j) = cen.Centroid(1);
      zN(j) = cen.Centroid(3);
      xyzN(j,[2,1,3]) = cen.Centroid; %Reorder as output of cen is y,x,z
   end
   xyzNm =  round(mean(xyzN));
   xyzNm = reshape(xyzNm, [1 3]);
   mag(i) = mean(sum(([xyzN(:,1),xyzN(:,2),xyzN(:,3)] -
[xyzNm(1),xyzNm(2),xyzNm(3)]).^2,2).^(1/2));
end
magColl = round(mean(mag));

%% Pancreas and Tumour Cloud
id = {};
[PancCloudWarp, TumCloudWarp] = deal(zeros(120,80,160));
[roiPanc, roiTum] = deal([]);
for m = 1:size(F,1)
   fROI = sprintf('data/refLib/%s',F(m).name);
   [ROI,~] = mhdImport(fROI);
   %% Centre of each ROI
   xyzN = zeros(7,3);
   for n = 1:7 %for loop for each ROI Kidneys Spleen Stomach, Liver Gallbladder and Hepatic
Portal Vein
      Roi = ROI;
      Roi(Roi~=n)=0;
      Roi = Roi./n;
      cen = regionprops(Roi,'centroid');
      xyzN(n,[2,1,3]) = cen.Centroid; %Reorder as output of cen is y,x,z
```

```
    end
    xyzNm = round(mean(xyzN));
    xyzNm = reshape(xyzNm, [1 3]);
    shiftT = eye(4);
    PancCenterVec = size(Roi)/2;
    shiftImg{m} = -( xyzNm - PancCenterVec);
    shiftT(4,1:3) = shiftImg{m}([2,1,3]);
    tformShift = affine3d();
    tformShift.T = shiftT;
    mag = mean(sum(([xyzN(:,1),xyzN(:,2),xyzN(:,3)] -
[xyzNm(1),xyzNm(2),xyzNm(3)]).^2,2).^(1/2));
    scaleFac = magColl/mag; %Calcuate the scaling factor
    Rin = imref3d(size(Roi));
    Rin.XWorldLimits = Rin.XWorldLimits-2*mean(Rin.XWorldLimits); %Find the limits for the
translation
    Rin.YWorldLimits = Rin.YWorldLimits-2*mean(Rin.YWorldLimits);
    Rin.ZWorldLimits = Rin.ZWorldLimits-2*mean(Rin.ZWorldLimits);

    %% Pancreas
    Roi = ROI;
    Roi(Roi~=8)=0;
    Roi = Roi./8;
    img = imwarp(Roi,Rin,tformShift,'linear','OutputView', Rin,'FillValues', min(Roi(:)));
    img = imresize3(img,scaleFac);
    img = FitImageToMatrix(img,size(Roi));
    PancCloudWarp = PancCloudWarp+img;
    roiPanc = cat(4,roiPanc,img);

    %% Tumour
    Roi = ROI;
    Roi(Roi~=9)=0;
    Roi = Roi./9;
    imgTum = imwarp(Roi,Rin,tformShift,'linear','OutputView', Rin,'FillValues', min(Roi(:)));
    imgTum = imresize3(imgTum,scaleFac);
    imgTum = FitImageToMatrix(imgTum,size(Roi));
    TumCloudWarp = TumCloudWarp+imgTum;
    roiTum = cat(4,roiTum,imgTum);

    %% Index
    idTemp = CharParse(F(m).name,'.',0);
    id = [id;idTemp];

end

DiMask(:,:,1) = [0 1 0; 1 1 1; 0 1 0];
DiMask(:,:,2) = 1;
DiMask(:,:,3) = DiMask(:,:,1);

% Pancreas
PancDi = imdilate(PancCloudWarp,DiMask);
PancSmooth = imgaussfilt3(PancDi, 1);
PancSmooth(PancSmooth<0) = 0;
```

```
PancSmooth = PancSmooth./max(max(max(PancSmooth)));
IDs.ID = id;
PancCloudKPC = struct('PancreasCloud',PancSmooth,'AnimalID', IDs,'NumberOfScans',size(id,1));

% Tumour
TumSmooth = imgaussfilt3(TumCloudWarp, 1);
TumSmooth(TumSmooth<0) = 0;
TumSmooth = TumSmooth./max(max(max(TumSmooth)));
IDs.ID = id;
TumCloud = struct('TumourCloud',TumSmooth,'AnimalID', IDs,'NumberOfScans',size(id,1));

%% Bias filter correction, Normalisation & Data convertion - Data in Input/BiasCon
%% runBMC
scriptPath = '"runBMC-Matlab.vqs"';
openVQ = ['"C:\Program Files\inviCRO\VivoQuant\Vivoquant.exe" --script ' scriptPath ' & exit
&'];
system (openVQ);
uiwait(msgbox('Complete runBMC code in VivoQuant and then click OK.','Success','modal'));

%% Multi Atlas Segmentation tool - Data in Rois
%% runWBA
% Run the VivoQuant Whole Body Atlas tool in order to classify the Kidneys,
% Spleen, Stomach, Liver, Gallbladder and Hepatic Portal Vein

% Build a reference library from the data contained in folder library
d = 'Library/*KPC*1.mhd';
D = dir(d);
inDir = 'WBA';
inputD = 'Library';
if ~exist(inDir,'dir'),mkdir(inDir);end

fid = fopen( 'WBA/refList.txt', 'wt' );
for i = 1:length(D)
  dataName = CharParse(D(i).name,'.',0);
  fprintf(fid,'%s/%s,%s/%s.rmha\n', D(i).folder, D(i).name, D(i).folder, dataName);

  id = sprintf('%s/%s',inputD,D(i).name);
  headerid = fopen(id,'r+');
  [A,~] = fscanf(headerid,'%s');
  if contains(A,'StudyDescription') == 0
     fprintf(headerid, 'StudyDescription = MRI');
  end
  fclose(headerid);
end
fclose(fid);

% Run the WBA
indir = 'data';
%
probOut = sprintf('./data/probabilities/');
if ~exist(probOut,'dir'),mkdir(probOut);end

scriptPath = '"MAS_tool.vqs"';
```

```
openVQ = ['"C:\Program Files\inviCRO\VivoQuant\Vivoquant.exe" --script ' scriptPath ' & exit
&'];
system (openVQ);

uiwait(msgbox('Once the MAS tool in VivoQuant has finished for all scans click
OK.','Success','modal'));

%% Create the probability map of all organs segmented by the WBA
d = dir(fullfile(indir,'*1.mhd'));
for i = 1:length(d)
   dDir = fullfile(indir,d(i).name);
   basename = strrep(d(i).name,'.mhd','');
   [zScan, rScan, klScan, krScan, liScan, spScan, stScan, gScan, hScan] = deal(zeros(120,80,160));

         % Find the ROI output
   [~,sInfo] = mhdImport(dDir);
   odir =
sprintf('%s/AppData/Roaming/inviCRO/VivoQuant/cache/WBAtlas%s',masDir,sInfo.sopinstanceu
id);

   % Create probability map
   pname = sprintf('%s/average_BestN-kidneyLeft-linear-deform.mhd',odir);
   [pScan,~] = mhdImport(pname);
   zScan = zScan+pScan;
   pScan(pScan >= 0.5) = 1;
   pScan(pScan < 0.5) = 0;
   CC = bwconncomp(pScan);
   numPixels = cellfun(@numel,CC.PixelIdxList);
   [~,idxG] = max(numPixels);
   klScan(CC.PixelIdxList{idxG}) = 1;
   klScan = round(smooth3(klScan,'box',[3,3,3]));
   rScan(klScan == 1) = 1;

   pname = sprintf('%s/average_BestN-kidneyRight-linear-deform.mhd',odir);
   [pScan,~] = mhdImport(pname);
   zScan = zScan+pScan;
   pScan(pScan >= 0.5) = 1;
   pScan(pScan < 0.5) = 0;
   CC = bwconncomp(pScan);
   numPixels = cellfun(@numel,CC.PixelIdxList);
   [~,idxG] = max(numPixels);
   krScan(CC.PixelIdxList{idxG}) = 1;
   krScan = round(smooth3(krScan,'box',[3,3,3]));
   rScan(krScan == 1) = 2;

   pname = sprintf('%s/average_BestN-Spleen-linear-deform.mhd',odir);
   [pScan,~] = mhdImport(pname);
   zScan = zScan+pScan;
   pScan(pScan >= 0.5) = 1;
   pScan(pScan < 0.5) = 0;
   CC = bwconncomp(pScan);
   numPixels = cellfun(@numel,CC.PixelIdxList);
```

```
[~,idxG] = max(numPixels);
spScan(CC.PixelIdxList{idxG}) = 1;
spScan = round(smooth3(spScan,'box',[3,3,3]));
rScan(spScan == 1) = 3;

pname = sprintf('%s/average_BestN-Stomach-linear-deform.mhd',odir);
[pScan,~] = mhdImport(pname);
zScan = zScan+pScan;
pScan(pScan >= 0.5) = 1;
pScan(pScan < 0.5) = 0;
CC = bwconncomp(pScan);
numPixels = cellfun(@numel,CC.PixelIdxList);
[~,idxG] = max(numPixels);
stScan(CC.PixelIdxList{idxG}) = 1;
stScan = round(smooth3(stScan,'box',[3,3,3]));
rScan(stScan == 1) = 4;

pname = sprintf('%s/average_BestN-Liver-linear-deform.mhd',odir);
[pScan,~] = mhdImport(pname);
zScan = zScan+pScan;
pScan(pScan >= 0.5) = 1;
pScan(pScan < 0.5) = 0;
CC = bwconncomp(pScan);
numPixels = cellfun(@numel,CC.PixelIdxList);
[~,idxG] = max(numPixels);
liScan(CC.PixelIdxList{idxG}) = 1;
liScan = round(smooth3(liScan,'box',[3,3,3]));
rScan(liScan == 1) = 5;

pname = sprintf('%s/average_BestN-Gallbladder-linear-deform.mhd',odir);
[pScan,~] = mhdImport(pname);
pScan(pScan >= 0.15) = 1;
pScan(pScan < 0.15) = 0;
CC = bwconncomp(pScan);
numPixels = cellfun(@numel,CC.PixelIdxList);
[~,idxG] = max(numPixels);
gScan(CC.PixelIdxList{idxG}) = 1;
gScan = round(smooth3(gScan,'box',[3,3,3]));
rScan(gScan == 1) = 6;

pname = sprintf('%s/average_BestN-HepaticPortalVein-linear-deform.mhd',odir);
[pScan,pInfo] = mhdImport(pname);
zScan = zScan+pScan;
pScan(pScan >= 0.2) = 1;
pScan(pScan < 0.2) = 0;
CC = bwconncomp(pScan);
numPixels = cellfun(@numel,CC.PixelIdxList);
[~,idxG] = max(numPixels);
hScan(CC.PixelIdxList{idxG}) = 1;
hScan = round(smooth3(hScan,'box',[3,3,3]));
rScan(hScan == 1) = 7;

zScan(zScan>1) = 1;
```

```
    % Save the probability to new location
    probName = sprintf('%s%s_OtherProb.mhd',probOut,basename);
    mhdExport(zScan,probName,pInfo);

    % Save the ROIs to new location
    mROI = CharParse(dDir,'.m',0);
    dataName = sprintf('%sMASTool.rmha',mROI);
    [~,rInfo] = mhdImport(dataName);
%     newName = sprintf('%s%s.rmha',outDir,basename);
    mhdExport(rScan,dataName,rInfo);
end

%% Manual QC
% uiwait(msgbox('Check the output from the MAS tool ROIs in the data folder. When you are
happy with an ROI save it into the same folder, once all ROIs have been check click
OK.','Success','modal'));

%% ROI and Other Prob Combine - Data in Rois
%% runCombineOther
% Take the probability cloud of the organs from the WBA and combine this
% with the user verified ROIs and after some smoothing taking the largest
% number from either to update the probability cloud

runDir = sprintf('data/*MASTool.rmha'); %Identify input folder
C = dir(runDir);

id = struct2cell(C);
for i = 1:size(C,1)
%     zScan = zeros(120,80,160); %Create a new space
    rName = sprintf('data/%s',C(i).name);
    [rScan,~] = mhdImport(rName); %Load the ROI data
    rScan(rScan > 7) = 0; %Remove any extra ROIs
    rScan(rScan > 1) = 1; %Convert to a binary map
        DiMask(:,:,1) = [0 1 0; 1 1 1; 0 1 0]; %Create a dilation mask
    DiMask(:,:,2) = 1;
    DiMask(:,:,3) = DiMask(:,:,1);
    dScan = imdilate(rScan,DiMask); %Dilate the ROI data
    dScan = dScan - rScan; %Smooth the edges of the dilated data
    dScan = dScan.*0.5;
        rScan = rScan+dScan;
    idC = CharParse(id{1,i},'MASTool.',0);
    oName = sprintf('%s%s_OtherProb.mhd',probOut,idC);
    [oScan,sInfo] = mhdImport(oName); %Load the other probability
    Scan = max(rScan,double(oScan)); %Take the max value from either the ROIs or the Other
probability
    mhdExport(Scan,oName,sInfo); %Save the data out
end

%% Create bounding area and normalise the scan - Data in normalisedData
%% runArea
% Load in the tumour and pancreas clouds that are built from the library
```

```
% data. These probability clouds are then then used to identify where the
% Machine Learning should look to identify the pancreas and tumour. Any
% voxels already classified as other organs by the WBA and user are removed
% from this area. Finally perform mean standard deviation normailsation


normOut = 'data/normalised';
if ~exist(normOut,'dir'),mkdir(normOut);end

%% Load in the novel scan
d = 'data/*MASTool.rmha'; % Get all rmha files
D = dir(d);

for m = 1:size(D,1)
   fROI = sprintf('data/%s',D(m).name); %Load in the ROI
   [ROI,rInfo] = mhdImport(fROI);
   id = CharParse(D(m).name,'MASTool.',0);
   idType = CharParse(id,'-',2);
   idType = idType{3};

   fScan = sprintf('data/%s.mhd',id); %Load in the scan
   [scan,sInfo] = mhdImport(fScan);
   scanRe = reshape(scan,[size(scan,1)*size(scan,2)*size(scan,3),1]);
   T = otsuthresh(scanRe); %Find the Otsu's threshold of the scan
   scanD = scan;
   scanD(scanD < T) = 0;
   scanD(scanD >= T) = 1; %Create binary map
   di(:,:,1) = [0 0 0; 0 1 0; 0 0 0];
   di(:,:,2) = [0 1 0; 1 1 1; 0 1 0];
   di(:,:,3) = di(:,:,1);
   for n = 1:3 %Dilate the binary map 3 times
      scanD = imdilate(scanD,di);
   end

   %% Centre of each ROI + the average distance of that ROI to the Pancreas
   xyzN = zeros(7,3);
   for n = 1:7 %for loop for each ROI
      Roi = ROI;
      Roi(Roi~=n)=0;
      Roi = Roi./n;
      cen = regionprops(Roi,'centroid');
      xyzN(n,[2,1,3]) = cen.Centroid; %Reorder as output of cen is y,x,z
   end
   xyzNm = round(mean(xyzN));
   xyzNm = reshape(xyzNm, [1 3]);
   shiftT = eye(4);
   PancCenterVec = size(Roi)/2;
   shiftImg{m} = ( xyzNm - PancCenterVec);
   shiftT(4,1:3) = shiftImg{m}([2,1,3]);
   tformShift = affine3d();
   tformShift.T = shiftT;
   mag = mean(sum(([xyzN(:,1),xyzN(:,2),xyzN(:,3)] -
[xyzNm(1),xyzNm(2),xyzNm(3)]).^2,2).^(1/2));
```

```
    scaleFac = magColl/mag;

    Rin = imref3d(size(Roi));
    Rin.XWorldLimits = Rin.XWorldLimits-2*mean(Rin.XWorldLimits);
    Rin.YWorldLimits = Rin.YWorldLimits-2*mean(Rin.YWorldLimits);
    Rin.ZWorldLimits = Rin.ZWorldLimits-2*mean(Rin.ZWorldLimits);

    %% Transform probability cloud
    PancCloudNovel = PancCloudKPC.PancreasCloud;
    PancCloudNovel = imresize3(PancCloudNovel,1/scaleFac);
    PancCloudNovel = FitImageToMatrix(PancCloudNovel,size(Roi));
    PancCloudNovel = imwarp(PancCloudNovel,Rin,tformShift,'linear','OutputView',
Rin,'FillValues', min(Roi(:)));
    PancCloudNovel(PancCloudNovel<0)=0;
    pCloud = sprintf('%s/%s_PancCloud.mhd',probOut,id);
    mhdExport(PancCloudNovel,pCloud,sInfo);

    TumCloudNovel = TumCloud.TumourCloud;
    TumCloudNovel = imresize3(TumCloudNovel,1/scaleFac);
    TumCloudNovel = FitImageToMatrix(TumCloudNovel,size(Roi));
    TumCloudNovel = imwarp(TumCloudNovel,Rin,tformShift,'linear','OutputView',
Rin,'FillValues', min(Roi(:)));
    TumCloudNovel(TumCloudNovel<0)=0;
    tCloud = sprintf('%s/%s_TumCloud.mhd',probOut,id);
    mhdExport(TumCloudNovel,tCloud,sInfo);
    areaRoi = PancCloudNovel+TumCloudNovel;

    areaRoi(areaRoi<0.02) = 0;
    areaRoi(areaRoi>0)=1;
    areaRoi = areaRoi.*scanD;
    aRoi = sprintf('%s/%s_AreaRoi.rmha',probOut,id);
    mhdExport(areaRoi,aRoi,rInfo);

    %% Normalisation
    idx = scan>=T;
    scanRe(scanRe<T)=[];
    scanStd = (scanRe-mean(scanRe))/std(scanRe);
    [mk,mkI] = maxk(scanStd,round(size(scanStd,1)*0.001));
    scanStd(mkI) = mk(end);
    scanStdBuild = zeros(120,80,160);
    if min(scanStd) < 0
       scanStdBuild = scanStdBuild + min(scanStd);
    end
    scanStdBuild(idx) = scanStd;
    scanStdBuild = scanStdBuild - min(scanStd);

    %% Save out
    StdName = sprintf('%s/%s.mhd',normOut,id);
    mhdExport(scanStdBuild,StdName,sInfo);

end
```

```
%% Feature Calculation - Data in Features
% runFeatures
% Calculate the features of the area defined in runArea

normDir = ('./data/normalised/');
probDir = ('./data/probabilities/');

d = 'data/normalised/*1.mhd'; % Get all ROIs
D = dir(d);
for l = 1:length(D)
   dName{l} = CharParse(D(l).name,'.',0);
end

x = size(D,1);

Outdir = sprintf('./data/features/');
if ~exist(Outdir,'dir')
   mkdir(Outdir);
end

DiMask(:,:,1) = [0 1 0; 1 1 1; 0 1 0];
DiMask(:,:,2) = 1;
DiMask(:,:,3) = DiMask(:,:,1);
% load('magMean.mat');

OffSet = [1 2];
glcm_window = 11;

for i = 1:length(D)
   %% Read volumes
   fscan = sprintf('%s%s',normDir,D(i).name);
   [Data,~]=mhdImport(fscan);
   DataP = padarray(Data,[glcm_window,glcm_window,glcm_window]);
   DataP = double(DataP);

   pscan = sprintf('%s%s_PancCloud.mhd',probDir,dName{i});
   [Panc,~]=mhdImport(pscan);
   Panc = padarray(Panc,[glcm_window,glcm_window,glcm_window]);
   idType = CharParse(dName{i},'-',2);
   idType = idType{3};
   if strcmp('KPC',idType)
      tscan = sprintf('%s%s_TumCloud.mhd',probDir,dName{i});
      [Tum,~]=mhdImport(tscan);
      Tum = padarray(Tum,[glcm_window,glcm_window,glcm_window]);
   else
      Tum = zeros(size(Panc));
   end

   % mask
   fROI = sprintf('data/%sMASTool.rmha',dName{i});
   [ROI,~]=mhdImport(fROI);

            [Gmag, Gazi, Gele] = imgradient3(DataP);
```

```
    [Gx, Gy, ~] = imgradientxyz(DataP);
    Gxy1 = (Gx.^2+Gy.^2).^.5/(2^0);
    Gxy2 = (Gx.^2+Gy.^2).^.5/(2^1);
    Gxy3 = (Gx.^2+Gy.^2).^.5/(2^2);
    Gxy4 = (Gx.^2+Gy.^2).^.5/(2^3);
    Gxy5 = (Gx.^2+Gy.^2).^.5/(2^4);
    FA = FACalcArea(Gmag,Gazi,Gele,glcm_window);

    %% COM
    xyzN = zeros(7,3);
    for n = 1:7 %for loop for each ROI
        Roi = ROI;
        Roi(Roi~=n)=0;
        Roi = Roi./n;
        cen = regionprops(Roi,'centroid');
        xyzN(n,[2,1,3]) = cen.Centroid; %Reorder as output of cen is y,x,z
    end

    xyzNm = round(mean(xyzN));
    xyzNm = reshape(xyzNm, [1 3]);
    mag = mean(sum(([xyzN(:,1),xyzN(:,2),xyzN(:,3)] -
[xyzNm(1),xyzNm(2),xyzNm(3)]).^2,2).^(1/2));
    scaleFac = magColl/mag;

   %% Bounding Box
    % mask
    fROI = sprintf('%s%s_AreaRoi.rmha',probDir,dName{i});
    [ROIb,~]=mhdImport(fROI);
    ROInames = {'BoundingBox'};
%     Rs = size(rInfo.extras.val{end},1);
    RoiP = padarray(ROIb,[glcm_window,glcm_window,glcm_window]);


    %% Find slices with label
    idxP = find(RoiP);
    [~,~,i3] = ind2sub(size(RoiP), idxP);
    indZ = unique(i3);
    RoiZ = RoiP;
    for kk = 1:glcm_window
        RoiZ = imdilate(RoiZ,DiMask);
    end
    DataP = padarray(DataP,[0,0,glcm_window]);
    DataP = double(DataP);

    %% extract features
    feature_matrix=[];
    Glrlm = [];
    for k = 1:length(indZ)
        [indx,indy] = find((RoiP(:,:,indZ(k)))>0);
        %GLCM
        [mean_im,std_im,entropy_im,~]= calculate_main_features((DataP(:,:,indZ(k))));
```

```
glcm_struct=calculate_glcm_Run_IdxTest((DataP(:,:,indZ(k))),glcm_window,OffSet,indx,indy);

[featur_vector,~]=features_vector_Run(ROInames(1),mean_im,std_im,entropy_im,glcm_struct,
indx,indy);
      feature_matrix= cat(1, feature_matrix, featur_vector);
      %GLRLM
      [RLFeatures0,RLFeatures45,RLFeatures90,RLFeatures135] = deal([]);
      [rlm0,rlm45,rlm90,rlm135,max0,max45,max90,max135] =
calculate_glrlm_Run_IdxTest(DataP(:,:,indZ(k)),glcm_window,indx,indy);
      for jj = 1:size(rlm0,1)
         [RLFeatures0(jj,:)] = calculate_glrlm_features(rlm0{jj,1},max0(:,jj),glcm_window);
      end
      for jj = 1:size(rlm45,1)
         [RLFeatures45(jj,:)] = calculate_glrlm_features(rlm45{jj,1},max45(:,jj),glcm_window);
      end
      for jj = 1:size(rlm90,1)
         [RLFeatures90(jj,:)] = calculate_glrlm_features(rlm90{jj,1},max90(:,jj),glcm_window);
      end
      for jj = 1:size(rlm135,1)
         [RLFeatures135(jj,:)] = calculate_glrlm_features(rlm135{jj,1},max135(:,jj),glcm_window);
      end
      Glrlm = cat(1,Glrlm,[RLFeatures0, RLFeatures45, RLFeatures90, RLFeatures135]);

   end
   feature_matrix= cat(1, feature_matrix);
   Features = cat(2,feature_matrix(:,1:23), Glrlm);

   %% Spatial features
   fea = MoCalc(DataP,idxP,glcm_window);
   Features = cat(2,Features, fea);

    %% Calculate the Gradient Magnitude and the FA values
    Grad = Gmag(idxP);
    FA3 = FA(idxP);
    Gxx = Gx(idxP);
    Gyy = Gy(idxP);
    Gxy1C = Gxy1(idxP);
    Gxy2C = Gxy2(idxP);
    Gxy3C = Gxy3(idxP);
    Gxy4C = Gxy4(idxP);
    Gxy5C = Gxy5(idxP);
    Features = cat(2,Features,Grad,Gxx,Gyy,Gxy1C,Gxy2C,Gxy3C,Gxy4C,Gxy5C,FA3);


   %% COM
   [xx,yy,zz] = Gen3DCoords(RoiP);
   difCOM = zeros(size(xx,1),3);
   for k = 1:size(xx,1)
      difCOM(k,1) = (xyzNm(1)-xx(k,1))/scaleFac;
      difCOM(k,2) = (xyzNm(2)-yy(k,1))/scaleFac;
      difCOM(k,3) = (xyzNm(3)-zz(k,1))/scaleFac;
   end
```

```
    Features = cat(2,Features, difCOM);


   %% Other organ probability
   PancC = Panc(idxP);
   TumC = Tum(idxP);
   OPname = sprintf('%s%s_OtherProb.mhd',probDir,dName{i});
   [oProb, ~] = mhdImport(OPname);
   oProb = padarray(oProb,[glcm_window,glcm_window,glcm_window]);
   opGath = oProb(idxP);
   Features = cat(2,Features, PancC, TumC, opGath);


    %% Save features
    saveName = dName{i};
    fname = sprintf('%s%s_BoundingArea_Features.mat',Outdir,saveName);
    parsave(fname,Features);

end

%% Run Model - Data in Classification
%% runModel
% Run the Bagged Random Forest Classification model and perform postprocessing

% clearvars
inputD = 'data/features';
d = sprintf('%s/*Area*.mat',inputD);
D = dir(d);
load('TreeBagger.mat');

outdir = 'Classification';
if ~exist(outdir),mkdir(outdir);end

%%
windSize = 11;
for i = 1:size(D,1)
    bname = sprintf('%s/%s',inputD,D(i).name);
    load(bname);
    dName = CharParse(D(i).name,'_',0);

    [~, yscore] = predict(Mdl,x);

    brName = sprintf('data/probabilities/%s_AreaRoi.rmha',dName);
    [Bb, ~] = mhdImport(brName);
    Bb = padarray(Bb,[windSize,windSize,windSize]);
    Ind = find(Bb == 1);

    rName = sprintf('data/%sMASTool.rmha',dName);
    [ROI, rInfo] = mhdImport(rName);

    %% Other ROIs
    Roi = ROI;
```

```
    Roi(Roi > 7) = 0;
    ROIidx = find(Roi > 0);

    %% Save out ROIs
    [yFitVisCloudPanc,yFitVisCloudTum] = deal(zeros(142,102,182));
    for j = 1:size(Ind,1)
        yFitVisCloudPanc(Ind(j)) = yscore(j,1);
        yFitVisCloudTum(Ind(j)) = yscore(j,2);
    end

    %% Tumour - Threshold, Smooth, Connected Componets (0.3)
    yTum = yFitVisCloudTum(1+windSize:end-windSize,1+windSize:end-windSize,1+windSize:end-
windSize);
    imT = zeros(size(yTum));
    imT(yTum>=0.3) = 1;
    imT = round(smooth3(imT,'ball',3));

    TumourROI = imT;
    CC2 = bwconncomp(TumourROI,18);
    numPixels2 = cellfun(@numel,CC2.PixelIdxList);
    [~,idx2] = find(numPixels2 < 64);
    for j = 1:size(idx2,2)
        TumourROI(CC2.PixelIdxList{idx2(j)}) = 0;
    end
    TumourROI(ROIidx) = 0;
    tumourIdx = find(TumourROI > 0);

    vTumour = length(tumourIdx);
    voxVol = rInfo.p_size(1)*rInfo.p_size(2)*rInfo.p_size(3);
    tumVol = vTumour*voxVol;

    %% Pancreas - Threshold, Smooth, Connected Componets (0.3)
    yPanc = yFitVisCloudPanc(1+windSize:end-windSize,1+windSize:end-
windSize,1+windSize:end-windSize);
    imP = zeros(size(yPanc));
    imP(yPanc>=0.3) = 1;
    imP = round(smooth3(imP,'ball',3));

    PancreasROI = imP;
    CC2 = bwconncomp(PancreasROI,18);
    numPixels2 = cellfun(@numel,CC2.PixelIdxList);
    [~,idx2] = find(numPixels2 < 64);
    for j = 1:size(idx2,2)
        PancreasROI(CC2.PixelIdxList{idx2(j)}) = 0;
    end
    PancreasROI(ROIidx) = 0;
    PancreasROI(tumourIdx) = 0;
    pancreasIdx = find(PancreasROI > 0);
    vPancreas = length(pancreasIdx);
    pancVol = vPancreas*voxVol;

    %% Combine segmentations
    dNameTum = CharParse(dName,'-',2);
```

```
    if tumVol > 0
        dNameTum{4} = '1';
            else
                    dNameTum{4} = '0';
    end
    dNameTumComb = sprintf('%s-%s-%s-%s-
%s',dNameTum{1},dNameTum{2},dNameTum{3},dNameTum{4},dNameTum{5});
    ROI(pancreasIdx) = 8;
    ROI(tumourIdx) = 9;
    rInfo.extras.val{end}(8) = {'Pancreas'};
    rInfo.extras.val{end}(9) = {'Tumour'};
    sName = sprintf('%s/%s.rmha',outdir,dNameTumComb);
    mhdExport(ROI,sName,rInfo);

    tName = sprintf('%s/%s.csv',outdir,dNameTumComb);
    fid = fopen(tName,'w');
    fprintf(fid,'Animal ID,Tumour Size (mm3),Pancreas Size (mm3)\n');
    fprintf(fid,'%s,%.2f,%.2f\n',dNameTumComb,tumVol,pancVol);
    fclose(fid);

    dataPull = sprintf('data/normalised/%s.mhd',dName);
    [Scan, sInfo] = mhdImport(dataPull);
    sName = sprintf('%s/%s.mhd',outdir,dNameTumComb);
    mhdExport(Scan,sName,sInfo);


end

%% Remove unnecessary files
% rmdir normalisedData s
% rmdir Rois s
% rmdir WBA/WBAOutput s

uiwait(msgbox('Classifictaion Complete!','Success','modal'));
end

function [out] = cad_glcm_features(glcm)

% VECTORIZED CODE: FASTER

size_glcm_1 = size(glcm,1);
size_glcm_2 = size(glcm,2);
size_glcm_3 = size(glcm,3);

% checked
out.autoc = zeros(1,size_glcm_3); % Autocorrelation: [2]
out.contr = zeros(1,size_glcm_3); % Contrast: matlab/[1,2]
out.corrm = zeros(1,size_glcm_3); % Correlation: matlab
out.corrp = zeros(1,size_glcm_3); % Correlation: [1,2]
out.cprom = zeros(1,size_glcm_3); % Cluster Prominence: [2]
out.cshad = zeros(1,size_glcm_3); % Cluster Shade: [2]
out.dissi = zeros(1,size_glcm_3); % Dissimilarity: [2]
```

```
out.energ = zeros(1,size_glcm_3); % Energy: matlab / [1,2]
out.entro = zeros(1,size_glcm_3); % Entropy: [2]
out.homom = zeros(1,size_glcm_3); % Homogeneity: matlab
out.homop = zeros(1,size_glcm_3); % Homogeneity: [2]
out.maxpr = zeros(1,size_glcm_3); % Maximum probability: [2]
out.sosvh = zeros(1,size_glcm_3); % Sum of sqaures: Variance [1]
out.savgh = zeros(1,size_glcm_3); % Sum average [1]
out.svarh = zeros(1,size_glcm_3); % Sum variance [1]
out.senth = zeros(1,size_glcm_3); % Sum entropy [1]
out.dvarh = zeros(1,size_glcm_3); % Difference variance [4]
out.denth = zeros(1,size_glcm_3); % Difference entropy [1]
out.inf1h = zeros(1,size_glcm_3); % Information measure of correlation1 [1]
out.inf2h = zeros(1,size_glcm_3); % Informaiton measure of correlation2 [1]
out.indnc = zeros(1,size_glcm_3); % Inverse difference normalized (INN) [3]
out.idmnc = zeros(1,size_glcm_3); % Inverse difference moment normalized [3]

% Indices
[i,j] = meshgrid(1:size_glcm_1,1:size_glcm_2);
idx1 = (i+j)-1;
idx2 = abs(i-j)+1;
ii = (1:(2*size_glcm_1-1))';
jj = (0:size_glcm_1-1)';

for k = 1:size_glcm_3 % number glcms
% Normalize GLCM
glcm_sum = sum(sum(glcm(:,:,k)));
Pij = glcm(:,:,k)./glcm_sum; % Normalize each glcm
glcm_mean = mean(Pij(:)); % compute mean after norm
%
p_x = squeeze(sum(Pij,2));
p_y = squeeze(sum(Pij,1))';
%
u_x = sum(sum(i.*Pij));
u_y = sum(sum(j.*Pij));
%
p_xplusy = zeros((2*size_glcm_1 - 1),1); %[1]
p_xminusy = zeros((size_glcm_1),1); %[1]
for aux = 1:max(idx1(:))
p_xplusy(aux) = sum(Pij(idx1==aux));
end
for aux = 1:max(idx2(:))
p_xminusy(aux) = sum(Pij(idx2==aux));
end

% Contrast
out.contr(k) = sum(sum((abs(i-j).^2).*Pij));
% Dissimilarity
out.dissi(k) = sum(sum(abs(i-j).*Pij));
% Energy
out.energ(k) = sum(sum(Pij.^2));
% Entropy
out.entro(k) = -sum(sum(Pij.*log(Pij+eps)));
% Homogeneity Matlab
```

```
out.homom(k) = sum(sum(Pij./(1+abs(i-j))));
% Homogeneity Paper
out.homop(k) = sum(sum(Pij./(1+abs(i-j).^2)));
% Sum of squares: Variance
out.sosvh(k) = sum(sum(Pij.*((j-glcm_mean).^2)));
% Inverse difference normalized
out.indnc(k) = sum(sum(Pij./(1+(abs(i-j)./size_glcm_1))));
% Inverse difference moment normalized
out.idmnc(k) = sum(sum(Pij./(1+((i-j)./size_glcm_1).^2)));
% Maximum probability
out.maxpr(k) = max(Pij(:));
% Sum average
out.savgh(k) = sum((ii+1).*p_xplusy);
% Sum entropy
out.senth(k) = -sum(p_xplusy.*log(p_xplusy+eps));
% Sum variance
out.svarh(k) = sum((((ii+1) - out.senth(k)).^2).*p_xplusy);
% Difference entropy
out.denth(k) = -sum(p_xminusy.*log(p_xminusy+eps));
% Difference variance
out.dvarh(k) = sum((jj.^2).*p_xminusy);
% Computes correlation
hxy1 = -sum(sum(Pij.*log(p_x*p_y' + eps)));
hxy2 = -sum(sum((p_x*p_y').*log(p_x*p_y' + eps)));
hx = -sum(p_x.*log(p_x+eps));
hy = -sum(p_y.*log(p_y+eps));
hxy = out.entro(k);
% Information measure of correlation 1
out.inf1h(k) = (hxy-hxy1)/(max([hx,hy]));
% Information measure of correlation 2
out.inf2h(k) = (1-exp(-2*(hxy2-hxy)))^0.5;
% Cluster Prominence
out.cprom(k) = sum(sum(Pij.*((i+j-u_x-u_y).^4)));
% Cluster Shade
out.cshad(k) = sum(sum(Pij.*((i+j-u_x-u_y).^3)));
%
s_x = sum(sum(Pij.*((i-u_x).^2)))^0.5;
s_y = sum(sum(Pij.*((j-u_y).^2)))^0.5;
corp = sum(sum(Pij.*(i.*j)));
corm = sum(sum(Pij.*(i-u_x).*(j-u_y)));
% Autocorrelation
out.autoc(k) = corp;
% Correlation paper
out.corrp(k) = (corp-u_x*u_y)/(s_x*s_y);
% Correlation Matlab
out.corrm(k) = corm/(s_x*s_y);
end
end

function glcm_struct=calculate_glcm_Run_IdxTest(im,ws,OffSet,indx,indy)

glcm_struct=cell(length(indx),1);
```

```
for i=1:length(indx)

    temp_im=im(indx(i)-ws:indx(i)+ws,indy(i)-ws:indy(i)+ws);  % crop a small window around
pixel
    glcm=graycomatrix(temp_im,'offset',OffSet,'NumLevels',32,'GrayLimits',[0 1.0]);
    %glcm=round(mean(glcm,3));
    glcm_struct{i}=cad_glcm_features(glcm);

end

end

function [ RLFeatures ] = calculate_glrlm_features( grl,maxcount,ws )
mx = size(grl,1);
mn = 1;
% gl = (mx-mn)+1;
gl = mx;

m=grl(mn:mx,:);
m1=m';
maxrun=max(max(maxcount));
S=0;
p = 120+(2*ws);
q = 80+(2*ws);
n=p*q;
G(gl)=0;
R(q)=0;
for u=1:gl
   for v=1:size(m,2)
      G(u)=G(u)+m(u,v);
      S=S+m(u,v);
   end
end
for u1=1:size(m,2)
   for v1=1:gl
      R(u1)=R(u1)+m1(u1,v1);
   end
end

SRE=0; LRE=0; GLN=0; RLN=0; RP=0; LGRE=0; HGRE=0;
for h1= 1:maxrun
   SRE=SRE+(R(h1)/(h1*h1)); %Short Run Length
   LRE=LRE+(R(h1)*(h1*h1)); %Long Run Length
   RLN=RLN+(R(h1)*R(h1)); %Run Length Non-Uniformity
   RP=RP+R(h1); %Run Percentage
end
SRE1=SRE/S;%%
LRE1=LRE/S;%%
RLN1=RLN/S;%%
RP1=RP/n;%%
for h2=1:gl
   GLN=(GLN+G(h2)^2); %Grey-Level Non-Uniformity
```

```
   LGRE=LGRE+(G(h2)/(h2*h2)); %Low Grey-Level Emphasis
   HGRE=HGRE+(h2*h2)*G(h2); %High Grey-Level Emphasis
end
GLN1=GLN/S;
LGRE1=LGRE/S;
HGRE1=HGRE/S;
RLFeatures = [SRE1,LRE1,RLN1,RP1,GLN1,LGRE1,HGRE1];
end

function [
glrlm0,glrlm45,glrlm90,glrlm135,maxcount0,maxcount45,maxcount90,maxcount135,idx ] =
calculate_glrlm_Run_IdxTest(im,ws,indx,indy)


[glrlm0,glrlm45,glrlm90,glrlm135] = deal(cell(length(indx),1));
for ii=1:length(indx)
   temp_im=im(indx(ii)-ws:indx(ii)+ws,indy(ii)-ws:indy(ii)+ws);

   %% Set image to 32 levels (16 bit)
   Imin = min(min(temp_im));
   newim = temp_im-Imin;
   Nmax = max(max(newim));
   Q = round(Nmax/32);
   [m,n] = size(newim);
   Quant = 0;
   for i = 1:m
      for j = 1:n
         I = temp_im(i,j);
         for B = 1:32
            if (I > Quant) && ( I <= Quant+Q)
               temp_im(i,j) = B/32;
               Quant = Quant+Q;
            end
         end
      end
   end
   newmax = max(max(temp_im));
   newim1 = temp_im/newmax;
   temp_im = round(newim1*32)+1;
   dir=0;
   dist1=1;
   if (dir == 1)
      temp_im=temp_im';
   end
   temp_im(isnan(temp_im))=0;

   %% Calculate grey level run length matrix 90 degrees  mx = max(max(temp_im));

   mx = max(max(temp_im));
%     mn = min(min(temp_im));
   [p,q] = size(temp_im);
   count=1;
```

```
c=1;
col=1;
grl = zeros(mx,p);
if ~exist('maxcount90')
   maxcount90 = zeros(p*q,length(indx));
end
mc=0;
for j=1:p
   for k=1:q-dist1
      mc=mc+1;
      g=temp_im(j,k);
      f=temp_im(j,k+dist1);
      if (g==f)&&(g~=0)
         count=count+1;
         c=count;
         col=count;
         maxcount90(mc,ii)=count;
      else grl(g,c)=grl(g,c)+1;
         col=1;
         count=1;
         c=1;
      end
   end
   grl(f,col)=grl(f,col)+1;
   count=1;
   c=1;
end
grl(1,:) = [];
glrlm90{ii} = grl;


%% Calculate grey level run length matrix 135 degrees
count=1;
c=1;
col=1;
grl = zeros(mx,p);
if ~exist('maxcount135')
   maxcount135 = zeros(p*q,length(indx));
end
mc=0;
for k=1:q-dist1
   for j=1:p-dist1
      mc=mc+1;
      g=temp_im(j,k);
      f=temp_im(j+dist1,k+dist1);
      if (g==f)&&(g~=0)
         count=count+1;
         c=count;
         col=count;
         maxcount135(mc,ii)=count;
      else grl(g,c)=grl(g,c)+1;
         col=1;
         count=1;
```

```
            c=1;
        end
    end
    grl(f,col)=grl(f,col)+1;
    count=1;
    c=1;
end
grl(1,:) = [];
glrlm135{ii} = grl;

%% Calculate grey level run length matrix 0 degrees
count=1;
c=1;
col=1;
grl = zeros(mx,p);
if ~exist('maxcount0')
    maxcount0 = zeros(p*q,length(indx));
end
mc=0;
for k=1:q
    for j=1+dist1:p
        mc=mc+1;
        g=temp_im(j,k);
        f=temp_im(j-dist1,k);
        if (g==f)&&(g~=0)
            count=count+1;
            c=count;
            col=count;
            maxcount0(mc,ii)=count;
        else grl(g,c)=grl(g,c)+1;
            col=1;
            count=1;
            c=1;
        end
    end
    grl(f,col)=grl(f,col)+1;
    count=1;
    c=1;
end
grl(1,:) = [];
glrlm0{ii} = grl;


%% Calculate grey level run length matrix 45 degrees
count=1;
c=1;
col=1;
grl = zeros(mx,p);
if ~exist('maxcount45')
    maxcount45 = zeros(p*q,length(indx));
end
mc=0;
```

```
    for k=1+dist1:q
        for j=1+dist1:p
            mc=mc+1;
            g=temp_im(j,k);
            f=temp_im(j-dist1,k-dist1);
            if (g==f)&&(g~=0)
                count=count+1;
                c=count;
                col=count;
                maxcount45(mc,ii)=count;
            else grl(g,c) = grl(g,c)+1;
                col=1;
                count=1;
                c=1;
            end
        end
        grl(f,col)=grl(f,col)+1;
        count=1;
        c=1;
    end
    grl(1,:) = [];
    glrlm45{ii} = grl;


end
end

function [mean_im,std_im,entropy_im,gf]= calculate_main_features(im)

% mean of the images
w=5;
h = 1/w*ones(w,1);
H = h*h';
mean_im = filter2(H,im);

% standard deviation
std_im= stdfilt(im,ones(w));

% entropyfilt

e=11;
entropy_im= entropyfilt(im,ones(e));

% gabor filter
gf=gabor_feature(im);

end

function f_out = CharManip(f_in,s_cut,op,s_add)
%
% Function to manipulate strings
%
% f_out = CharManip(f_in,s_cut,op,[s_add])
```

```
%
% f_out --> output string
% f_in --> input string
% s_cut --> string to remove, replace, or shift
% op --> operation
%   0 --> remove s_cut
%   1 --> replace s_cut with s_add
%   2 --> insert s_add before s_cut
% s_add --> string to add (optional)

if nargin == 3
   s_add = '';
end

occur = 0;
for i = 1:(length(f_in)-length(s_cut)+1)
 cnt = 0;
 flag = 0;
 while cnt < length(s_cut) && flag == 0
          cnt = cnt + 1;
          if f_in(i+cnt-1) ~= s_cut(cnt)
           flag = 1;
          end
 end
 if (cnt == length(s_cut) && flag == 0)
          occur = occur + 1;
          idx(occur) = i;
 end
end

if occur == 0
 f_out = f_in;
 %return;
 %disp('No occurrences found.');
 %msg = sprintf('String %s not found!',s_cut);
 %error(msg);
elseif occur == 1
 switch op
  case 0
          f_out = sprintf('%s%s',f_in(1:idx(1)-1),f_in(idx(1)+length(s_cut):length(f_in)));
  case 1
          f_out = sprintf('%s%s%s',f_in(1:idx(1)-1),s_add,f_in(idx(1)+length(s_cut):length(f_in)));
  case 2
          f_out = sprintf('%s%s%s',f_in(1:idx(1)-1),s_add,f_in(idx(1):length(f_in)));
 end
elseif occur > 1
 switch op
  case 0
          f_out = sprintf('%s',f_in(1:idx(1)-1));
          for i = 1:occur-1
           f_out = sprintf('%s%s',f_out,f_in(idx(i)+length(s_cut):idx(i+1)-1));
          end
```

```
            f_out = sprintf('%s%s',f_out,f_in(idx(occur)+length(s_cut):length(f_in)));
  case 1
            f_out = sprintf('%s%s',f_in(1:(idx(1)-1)),s_add);
            for i = 1:occur-1
              f_out = sprintf('%s%s%s',f_out,f_in(idx(i)+length(s_cut):idx(i+1)-1),s_add);
            end
            f_out = sprintf('%s%s',f_out,f_in(idx(occur)+length(s_cut):length(f_in)));
  case 2
            f_out = sprintf('%s%s',f_in(1:idx(1)),s_add);
            for i = 1:occur-1
              f_out = sprintf('%s%s%s',f_out,f_in(idx(i):idx(i+1)-1),s_add);
            end
            f_out = sprintf('%s%s',f_out,f_in(idx(occur):length(f_in)));
 end
end
end

function f_out = CharParse(f_in,s_delimit,op)
%
% Function to manipulate strings
%
% f_out = CharManip(f_in,s_delimit,op)
%
% f_out --> output string or cell
% f_in --> input string
% s_delimit --> character delimiter
% op --> operation
%   0 --> return all before delimiter
%   1 --> return all after delimiter
%   2 --> parse chunks around delimiter (f_out = cell)

sL = length(s_delimit);

occur = 0;
for i = 1:(length(f_in)-sL+1)
   cnt = 0;
   flag = 0;
   while cnt < length(s_delimit) && flag == 0
      cnt = cnt + 1;
      if f_in(i+cnt-1) ~= s_delimit(cnt)
         flag = 1;
      end
   end
   if (cnt == length(s_delimit) && flag == 0)
      occur = occur + 1;
      idx(occur) = i;
   end
end

if occur == 0
   %disp(sprintf('String %s not found!',s_delimit));
            if op == 0 || op == 1
              f_out = f_in;
```

```
            else
              f_out{1} = f_in;
            end
elseif occur == 1
   switch op
      case 0
         f_out = f_in(1:idx(1)-1);
      case 1
         f_out = f_in(idx(1)+sL:length(f_in));
      case 2
         f_out{1} = f_in(1:idx(1)-1);
         f_out{2} = f_in(idx(1)+sL:length(f_in));
   end
elseif occur > 1
   switch op
      case 0
         f_out = f_in(1:idx(1)-1);
      case 1
         f_out = f_in(idx(1)+sL:length(f_in));
      case 2
         f_out{1} = f_in(1:idx(1)-1);
         for i = 1:occur-1
            f_out{i+1} = f_in(idx(i)+sL:idx(i+1)-1);
         end
         f_out{occur+1} = f_in(idx(occur)+sL:length(f_in));
   end
end
end

function [FA] = FACalcArea(Mag,Azi,Ele,windowSize)
   %% One slice of an image at a time

   sP = 1+windowSize;
   ePX = size(Mag,1)-windowSize;
   ePY = size(Mag,2)-windowSize;
   ePZ = size(Mag,3)-windowSize;
   FA = zeros(size(Mag));
   for i = sP:ePX
      for j = sP:ePY
         for k = sP:ePZ
            if Mag(i,j,k) == 0
               continue
            end
            MagM = mean2(Mag(i-windowSize:i+windowSize,j-windowSize:j+windowSize,k-
windowSize:k+windowSize));
            AziM = mean2(Azi(i-windowSize:i+windowSize,j-windowSize:j+windowSize,k-
windowSize:k+windowSize));
            EleM = mean2(Ele(i-windowSize:i+windowSize,j-windowSize:j+windowSize,k-
windowSize:k+windowSize));

            A(1,1) = MagM*sind(EleM)*cosd(AziM);
            A(2,1) = MagM*sind(EleM)*sind(AziM);
```

```
            A(3,1) = MagM*cosd(EleM);
            T = A*A';
            R = T/trace(T);
            FA(i,j,k) = sqrt((1/2)*(3-(1/trace(R.^2))));
          end
        end
    end

end

function
[features_vector,label_mat]=features_vector_Run(label,mean_im,std_im,entropy_im,glcm_stru
ct,indx,indy)

% [indx,indy]=find((mask)>0);

features_vector=zeros(length(indx),24);
label_mat=cell(length(indx),1);

for i =1: length(indx)

 features_vector(i,1)=mean_im(indx(i),indy(i));
     features_vector(i,2)=std_im(indx(i),indy(i));
     features_vector(i,3)=entropy_im(indx(i),indy(i));
     features_vector(i,4)=mean_im(indx(i),indy(i));
      features_vector(i,5)=glcm_struct{i}.corrm;
      features_vector(i,6)=glcm_struct{i}.contr;
      features_vector(i,7)=glcm_struct{i}.corrp;
      features_vector(i,8)=glcm_struct{i}.cprom;
      features_vector(i,9)=glcm_struct{i}.cshad;
      features_vector(i,10)=glcm_struct{i}.dissi;
      features_vector(i,11)=glcm_struct{i}.energ;
      features_vector(i,12)=glcm_struct{i}.entro;
      features_vector(i,13)=glcm_struct{i}.homom;
      features_vector(i,14)=glcm_struct{i}.maxpr;
      features_vector(i,15)=glcm_struct{i}.sosvh;
      features_vector(i,16)=glcm_struct{i}.savgh;
      features_vector(i,17)=glcm_struct{i}.senth;
      features_vector(i,18)=glcm_struct{i}.dvarh;
      features_vector(i,19)=glcm_struct{i}.denth;
      features_vector(i,20)=glcm_struct{i}.inf1h;
      features_vector(i,21)=glcm_struct{i}.inf2h;
      features_vector(i,22)=glcm_struct{i}.indnc;
      features_vector(i,23)=glcm_struct{i}.idmnc;

      label_mat{i,1}=label;

end



end
```

```
% function com = Find_COM(img,b_flag)
% %
% % Function to find the center-of-mass of a 2D or 3D image
% %
% % com = Find_COM(img,b_flag)
% %
% % com --> center-of-mass
% % img --> input image
% % b_flag --> binary flag (default = 1).  If 0, then each voxel value will be
% % used.  If 1, then all > 0 voxel values = 1
%
% if nargin == 1, b_flag = 1;end
%
% if b_flag == 1
%   img(img>0) = 1;
%   img(img~=1) = 0;
% end
%
% if length(size(img))==2
%
%   com(1) = sum(sum(img,2)'.*(1:size(img,1)))/sum(img(:));
%   com(2) = sum(sum(img,1).*(1:size(img,2)))/sum(img(:));
%
% elseif length(size(img))==3
%
%   tmp = squeeze(sum(img,1));
%   com_1(2) = sum(sum(tmp,2)'.*(1:size(tmp,1)))/sum(tmp(:));
%   com_1(1) = 0;
%   com_1(3) = sum(sum(tmp,1).*(1:size(tmp,2)))/sum(tmp(:));
%
%   tmp = squeeze(sum(img,2));
%   com_2(2) = 0;
%   com_2(1) = sum(sum(tmp,2)'.*(1:size(tmp,1)))/sum(tmp(:));
%   com_2(3) = sum(sum(tmp,1).*(1:size(tmp,2)))/sum(tmp(:));
%
%   tmp = squeeze(sum(img,3));
%   com_3(2) = sum(sum(tmp,1).*(1:size(tmp,2)))/sum(tmp(:));
%   com_3(1) = sum(sum(tmp,2)'.*(1:size(tmp,1)))/sum(tmp(:));
%   com_3(3) = 0;
%
% %  if (com_1(1)~=com_3(1) | com_1(3)~=com_2(3) | com_2(2)~=com_3(2))
% %      disp('Warning; Individual COM Mismatch');
% %  end
%
%   com = [com_3(1) com_3(2) com_2(3)];
%
% end
% end

function [img_out,rge_in,rge_out] = FitImageToMatrix(img_in,coords,varargin)
%
% Function to fit a 3D image into the size defined in the 3x1 coords
```

```
%
% img_out = FitImageToMatrix(img_in,coords,<backgroundPixelValue=min>)
%
% img_out --> 3D image out size of coords
% img_in --> 3D image in
% coords --> 3 x 1 size vector

d_in = size(img_in);
d_out = coords;
d_diff = d_out-d_in;

backgroundPixelValue = double(min(img_in(:)));
if length(varargin) > 0
   backgroundPixelValue = varargin{1};
end
img_out = backgroundPixelValue*ones(coords(1),coords(2),coords(3));
if length(find(d_diff<0))==0
   d_diff = round(d_diff/2);
   for i = 1:3
     rge_out{i} = (d_diff(i)+1):(d_diff(i)+d_in(i));
     rge_in{i} = 1:size(img_in,i);
   end
   img_out(rge_out{1},rge_out{2},rge_out{3}) = img_in;
else
   for i = 1:3
      if d_diff(i) < 0
         rge_out{i} = 1:d_out(i);
         idx = round(-1*d_diff(i)/2);
         rge_in{i} = (idx+1):(idx+d_out(i));
      else
         d_diff(i) = round(d_diff(i)/2);
         rge_out{i} = (d_diff(i)+1):(d_diff(i)+d_in(i));
         rge_in{i} = 1:d_in(i);
      end
   end
   img_out(rge_out{1},rge_out{2},rge_out{3}) = img_in(rge_in{1},rge_in{2},rge_in{3});
end
end

function gf=gabor_feature(img)

imageSize = size(img);
numRows = imageSize(1);
numCols = imageSize(2);

wavelengthMin = 4/sqrt(2);
wavelengthMax = hypot(numRows,numCols);
n = floor(log2(wavelengthMax/wavelengthMin));
wavelength = 2.^(0:(n-2)) * wavelengthMin;

deltaTheta = 45;
orientation = 0:deltaTheta:(180-deltaTheta);
```

```
g = gabor(wavelength,orientation);

%%
gabormag = imgaborfilt(img,g);


%%
for i = 1:length(g)
   sigma = 0.5*g(i).Wavelength;
   K = 3;
   gabormag(:,:,i) = imgaussfilt(gabormag(:,:,i),K*sigma);
end


%%
X = 1:numCols;
Y = 1:numRows;
[X,Y] = meshgrid(X,Y);
featureSet = cat(3,gabormag,X);
featureSet = cat(3,featureSet,Y);

%%

% numPoints = numRows*numCols;
X = reshape(featureSet,numRows*numCols,[]);

%%
X = bsxfun(@minus, X, mean(X));
X = bsxfun(@rdivide,X,std(X));


coeff = pca(X);


% gf1=X*coeff(:,1);
gf = reshape(X*coeff(:,1),numRows,numCols);

end

function [x,y,z] = Gen3DCoords(img,value)
%
% Function to find the x,y,z coordinates of non-zero voxels in a 3D image
%
% [x,y,z] = Gen3DCoords(img,[value])
%
% img --> input image
% num --> find only coordinates for voxels of a particular value
% x,y,z --> the coordinates

if ~exist('value','var') || isempty(value)
  [x,yz] = find(img~=0);
else
```

```
  [x,yz] = find(img == value);
end
y = mod(yz,size(img,2));
y(y==0) = size(img,2);
z = ceil(yz/size(img,2));

end

function parsave(fname, x)
save(fname, 'x')
end

function struct = getOrSetField(struct, field, default)
% takes in a struct and a desired field and populates with default if it
% does not exist, it is case insensitive, and will force the case of field
% even if it exists. This function is great for passing options structures
% to functions
%
% Elliot January 2016

fields = fieldnames(struct);
[a, b] = ismember(lower(field), lower(fields));
if ~a
   struct.(field) = default;
else
   %ensure case
   tmp =     struct.(fields{b(1)});
   struct = rmfield(struct, (fields{b(1)}));
   struct.(field) = tmp;
end
end

function mhdExport(dta,filename,varargin)
%
% Function to write .mhd and .raw data
%
% mhdEport(dta,filename,varargin)
%
% dta --> data to be written
% filename --> path and filename where the .mhd and .raw files should be
% written
% varargin must be a structure, examples bellow:
%   '-p', pixel_size --> pixel size
%   '-c', class-type --> class (i.e., 'float')
%   '-s', scale factor --> ScalingFactor
%   '-si',scale intercept --> ScalingIntercept
%   '-u', scaling unit --> ScalingUnit
%   '-r', roi.type,roi.vals --> ROI
%   '-m', modality --> modality
%   '-n', patientsname --> PatientsName
%   '-seriesdec','-d', seriesdescription --> SeriesDescription
%   '-studydesc', studydescription --> StudyDescription
%   '-v', version --> Version
```

```
%   '-o', offset --> use a zero offset (needed for AC)
%   '-w', slices thrown away (specific to the Atlas MR tool)
%   '-e', extras (see mhdImport to find all extras)
%   '-g', magnification (used by NM/CT tool)
%   '-h', header (assumes unsigned char)
%   '-i', image type (i.e., volume, planar)
%   '-id', patient id
%   '-ipacs', write the header for iPACS upload
%   '-t', transpose the image data before writing
%   '-studyuid', study instance uid
%   '-seriesuid', series instance uid
%   '-sopuid', sop instance uid
%   '-weight', animal weight
%   '-dose', injected dose
%   '-dosetime', time of injected dose
%   '-names', ROI names
%   '-colors', ROI colors
%   '-cr', cropping range
%   '-hdr', write the header only (must be MHD)
%   '-sz', manually write the image size
%   '-zlib', compress raw using zlib library (writes mhd/zraw pair)
%   '-off', writes custom offset
%   '-frameduration', writes duration of each frame (vector)
%   '-framestart', writes start time of each frame (vector)
%   '-framemidpoint', writes center point of each frame (vector)
%   '-frameunit', writes time unit (string) (example:s)
%   '-framedesc', writes optional description about frame time info (example:time)

%   CiQuant
%   '-fd', Registration fidical centers [x,y,z]
%   '-fdr',Registration fiducials diameter (mm)
%   '-fdc',Registration fiducials colors in RGB format
%   '-swd', splitting window diemensions
%   '-sw', splitting window centers
%   '-swc', splitting window colors in RGB

%   '-cal', calibration fiducials centers
%   '-calr', calibration fiducials radius
%   '-calc', calibration fiducials colors
%   '-calseq', calibration sequence, sequence of the actual consntration of standards
%   '-calpro', calibration procedure
%   '-caleq', calibration equation used [a,b,z] y=ax+b, z is plate location in z
%   '-calunit', calibration sequence units

%   '-rgl', registration parameters local  [cos sin xt yt]
%   '-rgg', registration parameters global [a b c d xt yt]
%   '-zwl', position of WL in Z
%   '-zarg' position of Autorad in Z

% ipacs use
%   '-studyuid'
```

```
% CFT
%   '-fluro', fluorophore name
%   '-flurowv', [excitation emission]

% Joseph Brook - 11/11/2019



if nargin < 3
   error('Must include data, a filename and an info structure');
end

if ~isa(varargin{1},'struct')
   error('Must include data, a filename and an info structure');
else
   info = varargin{1};
   [~, name, ext] = fileparts(filename);


   p_size = getFieldOrDefault(info,'p_size', 0.02);

   info = getOrSetField(info,'patientsname', name);
   info = getOrSetField(info,'patientsid', name);
   info = getOrSetField(info,'seriesdescription', name);

   type = getFieldOrDefault(info,'type', class(dta));
   scaling = getFieldOrDefault(info,{'scaling' 'slope'}, 1);
   intercept = getFieldOrDefault(info,'intercept',0);
   scaling_unit =  getFieldOrDefault(info,{'scaling_unit' 'ScalingUnit' 'Unit' 'unit'},'');
   header_only =  getFieldOrDefault(info,{'header_only' 'hdr'},0);
   compress =  getFieldOrDefault(info,{'compress' 'zlib'},0);
   sz =size(dta);


   varargin = {'-p',p_size,'-n',info.patientsname,'-d',info.seriesdescription,'-s',scaling,'-si',
intercept,'-u', scaling_unit,'-id',info.patientsid};
   if strcmp(ext,'.rmha')
      dta = uint8(dta);
      type = 'uint8';
      varargin = {varargin{:}, '-names',   info.extras.val{end}};
   end
   if header_only
      varargin = {varargin{:}, '-hdr'};
   else
      varargin = {varargin{:},'-sz',sz,'-c', type};
   end

   if compress, varargin = {varargin{:}, '-zlib'}; end


   n_in = size(varargin,2);
   n_cnt = 1;
   p_flag = 0; % for pixel size
```

```
c_flag = 0; % for type to write
v_flag = 0;
r_flag = 0;
m_flag = 0;
d_flag = 0;
n_flag = 0;
e_flag = 0;
s_flag = 0;
o_flag = 0;
off_flag = 0;
w_flag = 0;
h_flag = 0;
g_flag = 0;
t_flag = 0;
id_flag = 0;
cr_flag = 0;
ipacs_flag = 0;
study_flag = 0;
series_flag = 0;
sop_flag = 0;
weight_flag = 0;
studydesc_flag = 0;
dose_flag = 0;
dosetime_flag = 0;
names_flag = 0;
colors_flag = 0;
sfactor_flag = 0;
sifactor_flag = 0;
hdronly_flag = 0;
zlib_flag = 0;
jp2_flag = 0;
jpg_flag=0;
png_flag = 0;
tif_flag = 0;
sz_flag = 0;
framedesc_flag = 0; % description of frame time info (optional)
frameunit_flag = 0; % time unit of frames (required for any frame info)
frameduration_flag = 0;
framemidpoint_flag = 0;
framestart_flag = 0;
s_factor = 1;
s_unit = 'UNKNOWN';
fd_flag = 0;
fdr_flag= 0;
fdc_flag=0;
sw_flag = 0;
swd_flag = 0;
swc_flag = 0;
cal_flag =0;
calr_flag=0;
calc_flag =0;
calseq_flag=0;
```

```
calpro_flag =0;
caleq_flag=0;
calunit_flag=0;
rgg_flag=0;
rgl_flag=0;
zwl_flag=0;
zarg_flag=0;
fluor_flag=0;
fluorwv_flag=0;

while n_cnt <= n_in
   switch varargin{n_cnt}
      case '-m'
         m_flag = 1;
         m_val = varargin{n_cnt+1};
         n_cnt = n_cnt + 2;
      case '-p'
         p_flag = 1;
         p_val = varargin{n_cnt+1};
         n_cnt = n_cnt + 2;
      case '-d'
         d_flag = 1;
         seriesdesc = varargin{n_cnt+1};
         n_cnt = n_cnt + 2;
      case '-seriesdesc'
         d_flag = 1;
         seriesdesc = varargin{n_cnt+1};
         n_cnt = n_cnt + 2;
      case '-studydesc'
         studydesc_flag = 1;
         studydesc = varargin{n_cnt+1};
         n_cnt = n_cnt + 2;
      case '-e'
         e_flag = 1;
         extras = varargin{n_cnt+1};
         n_cnt = n_cnt + 2;
      case '-n'
         n_flag = 1;
         n_val = varargin{n_cnt+1};
         n_cnt = n_cnt + 2;
      case '-id'
         id_flag = 1;
         patientid = varargin{n_cnt+1};
         n_cnt = n_cnt + 2;
      case '-ipacs'
         ipacs_flag = 1;
         ipacshdr = varargin{n_cnt + 1};
         n_cnt = n_cnt + 2;
      case '-c'
         c_write = varargin{n_cnt+1};
         switch c_write
            case 'ushort'
               c_cast = 'uint16';
```

```
            c_print = 'MET_USHORT';
        case 'char'
            c_cast = 'int8';
            c_print = 'MET_CHAR';
        case 'uchar'
            c_cast = 'uint8';
            c_print = 'MET_UCHAR';
        case 'uint8'
            c_cast = 'uint8';
            c_print = 'MET_UCHAR';
        case 'float'
            c_cast = 'single';
            c_print = 'MET_FLOAT';
        case 'int'
            c_cast = 'int32';
            c_print = 'MET_INT';
        case 'double'
            c_cast = 'double';
            c_print = 'MET_DOUBLE';
        case 'uint'
            c_cast = 'uint32';
            c_print = 'MET_UINT';
        case 'short'
            c_cast = 'int16';
            c_print = 'MET_SHORT';
        otherwise
            error('File type not supported');
    end
    n_cnt = n_cnt + 2;
    c_flag = 1;
case '-s'
    sfactor_flag = 1;
    s_factor = varargin{n_cnt+1};
    n_cnt = n_cnt + 2;
case '-si'
    sifactor_flag = 1;
    si_factor = varargin{n_cnt+1};
    n_cnt = n_cnt + 2;
case '-u'
    s_flag = 1;
    s_unit = varargin{n_cnt+1};
    n_cnt = n_cnt + 2;
case '-v'
    version = varargin{n_cnt+1};
    n_cnt = n_cnt + 2;
    v_flag = 1;
case '-h'
    h_flag = 1;
    header = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
case '-names'
    names_flag = 1;
```

```
    roiNames = varargin{n_cnt+1};
    n_cnt = n_cnt + 2;
case '-colors'
    colors_flag = 1;
    colorNames = varargin{n_cnt+1};
    n_cnt = n_cnt + 2;
case '-colormap'
    colors_flag = 2;
    cmap = varargin{n_cnt+1};
    n_cnt = n_cnt + 2;
case '-cr'
    cr_flag = 1;
    crRange = varargin{n_cnt+1};
    n_cnt = n_cnt + 2;
case '-t'
    t_flag = 1;
    n_cnt = n_cnt + 1;
case '-r'
    roi = varargin{n_cnt+1};
    n_cnt = n_cnt + 2;
    r_flag = 1;
case '-o'
    o_flag = 1;
    n_cnt = n_cnt + 1;
case '-off'
    off_flag = 1;
    offsetin = varargin{n_cnt+1};
    n_cnt = n_cnt + 2;
case '-w'
    w_flag = 1;
    slices = varargin{n_cnt+1};
    n_cnt = n_cnt + 2;
case '-g'
    g_flag = 1;
    mag = varargin{n_cnt+1};
    n_cnt = n_cnt + 2;
case '-i'
    itype = varargin{n_cnt+1};
    n_cnt = n_cnt + 2;
case '-weight'
    weight_flag = 1;
    weight = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
case '-dose'
    dose_flag = 1;
    dose = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
case '-dosetime'
    dosetime_flag = 1;
    dosetime = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
case '-hdr'
    hdronly_flag = 1;
```

```
    %           fprintf('Writing MHD header file only.\n');
    n_cnt = n_cnt + 1;
  case '-sz'
    sz_flag = 1;
    sz = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
  case '-studyuid'
    study_flag = 1;
    studyuid = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
  case '-seriesuid'
    series_flag = 1;
    seriesuid = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
  case '-sopuid'
    sop_flag = 1;
    sopuid = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
  case '-zlib'
    zlib_flag = 1;
    n_cnt = n_cnt + 1;
  case '-jp2'
    jp2_flag = 1;
    n_cnt = n_cnt + 1;
  case '-jpeg'
    jpg_flag = 1;
    n_cnt = n_cnt + 1;
  case '-jpg'
    jpg_flag = 1;
    n_cnt = n_cnt + 1;
  case '-png'
    png_flag = 1;
    n_cnt = n_cnt + 1;
  case '-tif'
    tif_flag = 1;
    n_cnt = n_cnt + 1;
  case '-tiff'
    tif_flag = 1;
    n_cnt = n_cnt + 1;
  case '-framestart'
    framestart_flag = 1;
    framestart = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
  case '-frameduration'
    frameduration_flag = 1;
    frameduration = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
  case '-framemidpoint'
    framemidpoint_flag = 1;
    framemidpoint = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
  case '-frameunit'
```

```
    frameunit_flag = 1;
    frameunit = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
case '-framedesc'
    framedesc_flag = 1;
    framedesc = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
case '-fd'
    fd_flag = 1;
    fdLocation = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
case '-fdr'
    fdr_flag = 1;
    fd_radius = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
case '-fdc'
    fdc_flag = 1;
    fd_color = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
case '-swd'
    swd_flag = 1;
    splittingWindowDim = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
case '-sw'
    sw_flag = 1;
    splittingWindowCen = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
case '-swc'
    swc_flag = 1;
    splittingWindowColor = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
case '-cal'
    cal_flag = 1;
    cal_centers = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
case '-calr'
    calr_flag = 1;
    cal_radius = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
case '-calc'
    calc_flag = 1;
    cal_color = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
case '-calseq'
    calseq_flag = 1;
    calibSequence = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;
case '-calpro'
    calpro_flag = 1;
    calibProcedure = varargin{n_cnt + 1};
    n_cnt = n_cnt + 2;

case '-caleq'
```

```
            caleq_flag = 1;
            cal_eq = varargin{n_cnt + 1};
            n_cnt = n_cnt + 2;

        case '-calunit'
            calunit_flag = 1;
            calunit = varargin{n_cnt + 1};
            n_cnt = n_cnt + 2;

        case '-rgl'
            rgl_flag = 1;
            registration_local = varargin{n_cnt + 1};
            n_cnt = n_cnt + 2;
        case '-rgg'
            rgg_flag = 1;
            registration_global = varargin{n_cnt + 1};
            n_cnt = n_cnt + 2;
        case '-zwl'
            zwl_flag = 1;
            z_location_wl = varargin{n_cnt + 1};
            n_cnt = n_cnt + 2;
        case '-zarg'
            zarg_flag = 1;
            z_location_arg = varargin{n_cnt + 1};
            n_cnt = n_cnt + 2;
        case '-fluor'
            fluor_flag=1;
            fluro_name = varargin{n_cnt + 1};
            n_cnt = n_cnt + 2;
        case '-fluorwv'
            fluorwv_flag=1;
            fluor_wavelength = varargin{n_cnt + 1};
            n_cnt = n_cnt + 2;
        otherwise
            error('Improper variable input argument');
    end
end

if c_flag == 0
    if ~jp2_flag
        c_cast = 'single';c_print = 'MET_FLOAT';c_write = 'float';
    else
        if strcmp(class(dta),'uint8')
            c_cast = 'uint8';c_print = 'MET_UCHAR';c_write = 'uint8';
        elseif strcmp(class(dta),'uint16')
            c_cast = 'uint16';c_print = 'MET_USHORT';c_write = 'uint16';
        else
            fprintf(['\nWARNING: %s not supported for JPEG2000. Converting ' ...
                'to uint8\n\n'],class(dta));
            dta = cast(dta,'uint8');
            c_cast = 'uint8';c_print = 'MET_UCHAR';c_write = 'uint8';
        end
```

```
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% CHECK THE NAME
tmp = CharParse(filename,'.',2);
doRMHA = 0;
if strcmp(tmp{length(tmp)},'mhd')
    doMHA = 0;
elseif strcmp(tmp{length(tmp)},'mha')
    doMHA = 1;
elseif strcmp(tmp{length(tmp)},'rmha')
    doMHA = 1;
    doRMHA = 1;
    % FORCE MODALITY TO RO
    m_flag = 1;m_val = 'RO';
    if ~strcmp(c_print,'MET_UCHAR')
        c_print = 'MET_UCHAR';
        c_write = 'uchar';
        %disp('Warning: Must use unsigned char with RMHA');
    end
else
    disp('Error: Filename must end in either mhd,mha,or rhma');
end

% MAKE SURE HEADER ONLY ONLY ON FOR MHD
if (doMHA || doRMHA) && hdronly_flag
    error('Header only option only available for MHD');
end

% GET THE SIZE,
if ~sz_flag, sz = size(dta);end
ndims = length(sz);

if jp2_flag || jpg_flag || png_flag || tif_flag, sz(ndims) = 1;end

if jp2_flag || jpg_flag || png_flag || tif_flag

    sz=size(dta);
    ndims=length(sz);
    if ndims==3 && size(dta,3) ~=3 %% for fluorescence stack
        sz(ndims) = [];
        ndims=2;
    elseif ndims==4  % for white light image stack
        sz(ndims-1)=sz(ndims);
        sz(ndims)=[];
        ndims=3;
        dta = permute(dta,[2 1 3 4]);
    elseif (size(dta,3) == 3 && size(dta,4) == 1) %% for single white light image
        ndims = 4;
        sz = [size(dta,1) size(dta,2) size(dta,3) size(dta,4)];
        sz(ndims-1) = sz(ndims);
```

```
        sz(ndims) = [];
        ndims = 3;
        dta = permute(dta,[2 1 3 4]);
      end

    end
    % CHECK FOR TRANSPOSITION
    if t_flag
      tmp = sz;sz(2) = tmp(1);sz(1) = tmp(2);
      if ndims == 2, dta = permute(dta,[2 1 3]);
      elseif ndims == 3, dta = permute(dta,[2 1 3 4]);
        %elseif ndims == 4, dta = permute(dta,[2 1 3 4]);
      end
    end

    % CHECK THE PIXEL SIZE
    if p_flag == 0
      if ndims == 2
        p_val = [1.0 1.0];
      elseif ndims == 3
        p_val = [1.0 1.0 1.0];
      elseif ndims == 4
        p_val = [1.0 1.0 1.0 1.0];
      end
    else
      if ndims == 2 && length(p_val)==1
        p_val = [p_val p_val];
      elseif ndims == 3 && length(p_val)==1
        p_val = [p_val(1) p_val(1) p_val(1)];
      elseif ndims == 4 && length(p_val)==1
        p_val = [p_val(1) p_val(1) p_val(1) p_val(1)];
      elseif length(p_val)~=ndims
        if hdronly_flag
          fprintf('Warning: Voxel mismatch, but only writing header.\n');
        else
          error('Either the length of the voxel dimension vector is incorrect or the number of
dimensions is not supported.');
        end
      end
    end


    [fid,message] = fopen(filename,'w');

    if fid == -1
      error('Could not open MHD file %s for writing: %s.',filename,message);
    end
    if doRMHA==1
      fprintf(fid,'ObjectType = ROI\n');
    else
      fprintf(fid,'ObjectType = Image\n');
    end
```

```
if exist('itype')
   fprintf(fid,'ImageType = %s\n',itype);
end
fprintf(fid,'NDims = %d\n',ndims);
fprintf(fid,'BinaryData = True\n');
fprintf(fid,'BinaryDataByteOrderMSB = False\n');
if doRMHA == 0
   if zlib_flag
      fprintf(fid,'CompressedData = True\n');
   elseif jp2_flag
      fprintf(fid,'CompressedData = JP2\n');
   elseif jpg_flag
      fprintf(fid,'CompressedData = JPEG\n');
   elseif png_flag
      fprintf(fid,'CompressedData = PNG\n');
   elseif tif_flag
      fprintf(fid,'CompressedData = TIFF\n');
   else
      fprintf(fid,'CompressedData = False\n');
   end
else
   fprintf(fid,'CompressedData = RLE\n');
end
if e_flag
   idx = find(strcmp(extras.key,'CompressedData'));
   if ~isempty(idx)
      extras.val(idx) = [];extras.key(idx) = [];
   end
end
if ndims == 2
   fprintf(fid,'TransformMatrix = -1 0 0 -1\n');
elseif ndims == 3
   fprintf(fid,'TransformMatrix = -1 0 0 0 -1 0 0 0 -1\n');
elseif ndims == 4
   fprintf(fid,'TransformMatrix = -1 0 0 0 0 -1 0 0 0 0 -1 0 0 0 0 -1\n');
end
% DICOM --> RAI
% IVS --> LPS
fprintf(fid,'AnatomicalOrientation = LPS\n');

p_val = abs(p_val);

if ndims == 2
   if off_flag == 1
      fprintf(fid, 'Offset = %g %g\n', offsetin(1), offsetin(2));
      disp('Printing custom offset.');
   elseif o_flag == 1
      fprintf(fid,'Offset = 0 0\n');
      disp('Printing a zero offset.');
   else
      fprintf(fid,'Offset = %g %g\n',sz(1)*p_val(1),sz(2)*p_val(2));
   end
   fprintf(fid,'CenterOfRotation = 0 0\n');
```

```
        fprintf(fid,'ElementSpacing = %g %g\n',p_val(1),p_val(2));
        fprintf(fid,'DimSize = %d %d\n',sz(1),sz(2));
    elseif ndims == 3
        if off_flag == 1
            fprintf(fid, 'Offset = %g %g %g\n', offsetin(1), offsetin(2),...
                offsetin(3));
            disp('Printing custom offset.');
        elseif o_flag == 1
            fprintf(fid,'Offset = 0 0 0\n');
            disp('Printing a zero offset.');
        else

            fprintf(fid,'Offset = %g %g %g\n',sz(1)*p_val(1),sz(2)*p_val(2),sz(3)*p_val(3));
        end
        fprintf(fid,'CenterOfRotation = 0 0 0\n');
        fprintf(fid,'ElementSpacing = %g %g %g\n',p_val(1),p_val(2),p_val(3));
        fprintf(fid,'DimSize = %d %d %d\n',sz(1),sz(2),sz(3));
    elseif ndims == 4
        if off_flag == 1
            fprintf(fid, ['Offset = %g %g %g %g\n'], offsetin(1), offsetin(2), ...
                offsetin(3), offsetin(4));
            disp('Printing custom offset.');
        elseif o_flag == 1
            fprintf(fid,'Offset = 0 0 0 0\n');
            disp('Printing a zero offset.');

            %    elseif jpg_flag
            %        fprintf(fid,'Offset = %g %g %g\n',sz(1)*p_val(1),sz(2)*p_val(2),sz(3)*p_val(3));
        else
            fprintf(fid,'Offset = %g %g %g
%g\n',sz(1)*p_val(1),sz(2)*p_val(2),sz(3)*p_val(3),sz(4)*p_val(4));
        end
        fprintf(fid,'CenterOfRotation = 0 0 0 0\n');
        fprintf(fid,'ElementSpacing = %g %g %g %g\n',p_val(1),p_val(2),p_val(3),p_val(4));
        fprintf(fid,'DimSize = %d %d %d %d\n',sz(1),sz(2),sz(3),sz(4));
    end
    fprintf(fid,'Rows = %d\n',sz(1));
    fprintf(fid,'Columns = %d\n',sz(2));

    % fiducial location FiducialSetSequence (0070,031C)

    if fd_flag == 1

        fprintf(fid,'RegistrationFiducialCenters = ');
        fprintf(fid,'[');
        for fnum=1:size(fdLocation,2)-1
            fprintf(fid,'%.2f %.2f %.2f\\ ',...
                fdLocation{fnum}(1),fdLocation{fnum}(2),fdLocation{fnum}(3));

        end
        fprintf(fid,'%.2f %.2f %.2f]\n',...
```

```
fdLocation{size(fdLocation,2)}(1),fdLocation{size(fdLocation,2)}(2),fdLocation{size(fdLocation,2)}
(3));

   end

   if fdr_flag == 1

      fprintf(fid,'RegistrationFiducialRadius = ');
      fprintf(fid,'[%.2f]\n',fd_radius);

   end

   if fdc_flag == 1

      fprintf(fid,'RegistrationFiducialColors = ');
      fprintf(fid,'[');
      for fnum=1:size(fd_color,2)-1
         fprintf(fid,'%.2f %.2f %.2f\\ ',...
            fd_color{fnum}(1),fd_color{fnum}(2),fd_color{fnum}(3));

      end
      fprintf(fid,'%.2f %.2f %.2f]\n',...
         fd_color{size(fd_color,2)}(1),fd_color{size(fd_color,2)}(2),fd_color{size(fd_color,2)}(3));


   end

   if swd_flag == 1

      fprintf(fid,'SplitingWindowDim = %0.2f
%0.2f\n',splittingWindowDim(1),splittingWindowDim(2));
   end

   if sw_flag == 1

      fprintf(fid,'SplittingWindowCenters = ');
      fprintf(fid,'[');
      for fnum=1:size(splittingWindowCen,2)-1
         fprintf(fid,'%.2f %.2f %.2f\\ ',...

splittingWindowCen{fnum}(1),splittingWindowCen{fnum}(2),splittingWindowCen{fnum}(3));
      end

      fprintf(fid,'%.2f %.2f %.2f]\n',...

splittingWindowCen{size(splittingWindowCen,2)}(1),splittingWindowCen{size(splittingWindowC
en,2)}(2),splittingWindowCen{size(splittingWindowCen,2)}(3));

   end

   if swc_flag == 1
```

```
        fprintf(fid,'SplittingWindowColors = ');
        fprintf(fid,'[');
        for fnum=1:size(splittingWindowColor,2)-1
            fprintf(fid,'%.2f %.2f %.2f\\ ',...

splittingWindowColor{fnum}(1),splittingWindowColor{fnum}(2),splittingWindowColor{fnum}(3))
;

        end
        fprintf(fid,'%.2f %.2f %.2f]\n',...

splittingWindowColor{size(splittingWindowColor,2)}(1),splittingWindowColor{size(splittingWind
owColor,2)}(2),splittingWindowColor{size(splittingWindowColor,2)}(3));

    end
    % calibration related parameters

    if cal_flag == 1

        fprintf(fid,'CalibrationStandardsCenters = ');
        fprintf(fid,'[');
        for fnum=1:size(cal_centers,2)-1
            fprintf(fid,'%.2f %.2f %.2f\\ ',...
                cal_centers{fnum}(1),cal_centers{fnum}(2),cal_centers{fnum}(3));
        end
        fprintf(fid,'%.2f %.2f %.2f]\n',...

cal_centers{size(cal_centers,2)}(1),cal_centers{size(cal_centers,2)}(2),cal_centers{size(cal_cente
rs,2)}(3));

    end

    if calr_flag == 1

        fprintf(fid,'CalibrationStandardsRadius = [%.2f]\n',cal_radius);

    end

    if calc_flag == 1

        fprintf(fid,'CalibrationStandardsColors = ');
        fprintf(fid,'[');
        for fnum=1:size(cal_color,2)-1
            fprintf(fid,'%.2f %.2f %.2f\\ ',...
                cal_color{fnum}(1),cal_color{fnum}(2),cal_color{fnum}(3));

        end
        fprintf(fid,'%.2f %.2f %.2f]\n',...

cal_color{size(cal_color,2)}(1),cal_color{size(cal_color,2)}(2),cal_color{size(cal_color,2)}(3));

    end
```

```
if calpro_flag == 1

   fprintf(fid,'CalibrationProcedure = %s\n',calibProcedure);

end

if calseq_flag == 1

   fprintf(fid,'CalibrationSequence =');

   for cs=1:length(calibSequence)
      fprintf(fid,' %.2f',calibSequence{cs});
   end

   fprintf(fid,'\n');

end

if calunit_flag == 1

   fprintf(fid,'CalibrationSequenceUnit = %s\n',calunit);

end


if caleq_flag == 1


   fprintf(fid,'CalibrationStandardsEquation = ');
   fprintf(fid,'[');
   for fnum=1:size(cal_eq,2)-1
      fprintf(fid,'%.2f %.2f %.2f\\ ',...
         cal_eq{fnum}(1),cal_eq{fnum}(2),cal_eq{fnum}(3));

   end

   fprintf(fid,'%.2f %.2f %.2f]\n',...
      cal_eq{size(cal_eq,2)}(1),cal_eq{size(cal_eq,2)}(2),cal_eq{size(cal_eq,2)}(3));

end


if rgl_flag == 1

   fprintf(fid,'RegistrationSequence = ');
   fprintf(fid,'[');
   for fnum=1:size(registration_local,2)-1
      fprintf(fid,'%.2f %.2f %.2f %0.2f\\ ',...
         registration_local{fnum}(1),registration_local{fnum}(2),...
         registration_local{fnum}(3),registration_local{fnum}(4));
```

```
        end
        fprintf(fid,'%.2f %.2f %.2f %0.2f]\n',...
registration_local{size(registration_local,2)}(1),registration_local{size(registration_local,2)}(2),...
registration_local{size(registration_local,2)}(3),registration_local{size(registration_local,2)}(4));


    end

    if rgg_flag == 1

        fprintf(fid,'RegistrationMatrixGlobal = ');
        fprintf(fid,'[');
        for fnum=1:size(registration_global,2)
            fprintf(fid,'%.2f %.2f %.2f %0.2f %.2f %0.2f',...
                registration_global{fnum}(1),registration_global{fnum}(2),...
                registration_global{fnum}(3),registration_global{fnum}(4),...
                registration_global{fnum}(5),registration_global{fnum}(6));

        end
        fprintf(fid,']\n');

    end


    if zwl_flag == 1

        fprintf(fid,'BlockSliceLocationWL = ');
        fprintf(fid,'[');
        for fnum=1:length(z_location_wl)-1
            fprintf(fid,'%.2f\\',...
                z_location_wl(fnum));

        end
        fprintf(fid,'%.2f]\n',...
            z_location_wl(length(z_location_wl)));


    end

    if zarg_flag == 1

        fprintf(fid,'BlockSliceLocationARG = ');
        fprintf(fid,'[');
        for fnum=1:length(z_location_arg)-1
            fprintf(fid,'%.2f\\',...
                z_location_arg(fnum));

        end
        fprintf(fid,'%.2f]\n',...
            z_location_arg(length(z_location_arg)));
```

```
end

if fluor_flag == 1

   fprintf(fid,'Fluorophore = %s\n',fluro_name);
end

if fluorwv_flag == 1

   fprintf(fid,'FluorophoreWavelength = %d %d\n',fluor_wavelength(1),fluor_wavelength(2));
end

if ipacs_flag
   key = ipacshdr.key;val = ipacshdr.val;
   for i = 1:length(key)
      fprintf(fid,'%s = %s\n',key{i},val{i});
   end

else
   if v_flag == 1
      fprintf(fid,'Version = %0.3f\n',version);
   end
   if r_flag == 1
      fprintf(fid,'ROI = %s ',roi.type);
      switch roi.type
         case 'box'
            fprintf(fid,'%d %d %d %d %d %d\n',roi.vals(1),roi.vals(2),roi.vals(3),...
               roi.vals(4),roi.vals(5),roi.vals(6));
         case 'sqbox'
            fprintf(fid,'%d %d %d %d %d %d\n',roi.vals(1),roi.vals(2),roi.vals(3),...
               roi.vals(4),roi.vals(5),roi.vals(6));
         otherwise
            disp('That ROI type is not supported');
            fprintf(fid,'UNK\n');
      end
   end
   if g_flag == 1
      if ndims == 2
         if length(mag)==1
            mag = [mag mag];
         end
         fprintf(fid,'Magnification = %g %g\n',mag(1),mag(2));
      elseif ndims == 3
         if length(mag)==1
            mag = [mag mag mag];
         end
         fprintf(fid,'Magnification = %g %g %g\n',mag(1),mag(2),mag(3));
      elseif ndims == 4
         if length(mag)==1
            mag = [mag mag mag mag];
         end
```

```
            fprintf(fid,'Magnification = %g %g %g %g\n',mag(1),mag(2),mag(3),mag(4));
        end
    end

    % Cropping range
    if cr_flag == 1
        if ndims == 2
            fprintf(fid,'Cropping = [%d %d %d %d]\n',...
                crRange{1}(1),crRange{1}(end),crRange{2}(1),crRange{2}(end));
        elseif ndims == 3
            fprintf(fid,'Cropping = [%d %d %d %d %d %d]\n',...

crRange{1}(1),crRange{1}(end),crRange{2}(1),crRange{2}(end),crRange{3}(1),crRange{3}(end));
        end
    end

    if w_flag == 1
        fprintf(fid,'Slice = [');
        if ~isempty(slices)
            for i = 1:(length(slices)-1)
                fprintf(fid,'%d ',slices(i));
            end
            for i = length(slices)
                fprintf(fid,'%d',slices(i));
            end
        end
        fprintf(fid,'];\n');
    end

    if m_flag == 1
        fprintf(fid,'Modality = %s\n',m_val);
        if e_flag
            idx = find(strcmp(extras.key,'Modality'));
            if ~isempty(idx), extras.val(idx) = [];extras.key(idx) = [];end
        end
    end

    if n_flag == 1
        fprintf(fid,'PatientsName = %s\n',n_val);
        if e_flag
            idx = find(strcmp(extras.key,'PatientsName'));
            if ~isempty(idx), extras.val(idx) = [];extras.key(idx) = [];end
        end
    end

    if id_flag == 1
        fprintf(fid,'PatientID = %s\n',patientid);
        if e_flag
            idx = find(strcmp(extras.key,'PatientID'));
            if ~isempty(idx), extras.val(idx) = [];extras.key(idx) = [];end
        end
    end
```

```
if d_flag == 1
   fprintf(fid,'SeriesDescription = %s\n',seriesdesc);
   if e_flag
      idx = find(strcmp(extras.key,'SeriesDescription'));
      if ~isempty(idx), extras.val(idx) = [];extras.key(idx) = [];end
   end
end

if studydesc_flag == 1
   fprintf(fid,'StudyDescription = %s\n',studydesc);
   if e_flag
      idx = find(strcmp(extras.key,'StudyDescription'));
      if ~isempty(idx), extras.val(idx) = [];extras.key(idx) = [];end
   end
end

if study_flag == 1

   if length(studyuid)==1
      cdate = datestr(now,'yyyymmdd');
      ctime = datestr(now,'HHMMSS.FFF');
      studyuidnew = sprintf('1.1.05001.07005.%s.%s',cdate,ctime);
      imagetype='Volume';
      opt.Method = 'MD5';opt.Format = 'double';
      uidInput = [cdate ctime imagetype];
      uidStr = strrep(num2str(DataHash(uidInput,opt)),' ','');
      uidStr = [uidStr(1:6) uidStr((end-3):end)];
      studyuidnew = [studyuidnew '.' uidStr];

      fprintf(fid,'StudyInstanceUID = %s\n',studyuidnew);
   else
      fprintf(fid,'StudyInstanceUID = %s\n',studyuid);
   end
end

if series_flag == 1
   fprintf(fid,'SeriesInstanceUID = %s\n',seriesuid);
   if e_flag
      idx = find(strcmp(extras.key,'SeriesInstanceUID'));
      if ~isempty(idx), extras.val(idx) = [];extras.key(idx) = [];end
   end
end

if sop_flag == 1
   fprintf(fid,'SOPInstanceUID = %s\n',sopuid);
   if e_flag
      idx = find(strcmp(extras.key,'SOPInstanceUID'));
      if ~isempty(idx), extras.val(idx) = [];extras.key(idx) = [];end
   end
end

sfOff = 0;
```

```
if sfactor_flag == 1
   fprintf(fid,'ScalingFactor = %g\n',s_factor);
   sfOff = 1; % To overwrite the '-extras' value
end

if sifactor_flag == 1
   fprintf(fid,'Intercept = %g\n',si_factor);
end

suOff = 0;
if s_flag == 1
   fprintf(fid,'ScalingUnit = %s\n',s_unit);
   suOff = 1; % To overwrite the '-extras' value
end

if weight_flag == 1
   fprintf(fid,'PatientsWeight = %g\n',weight);
end

if dose_flag == 1
   fprintf(fid,'TotalDose = %g\n',dose);
end

if dosetime_flag == 1
   fprintf(fid,'TotalDoseTime = %s\n',dosetime);
end

if h_flag == 1
   fprintf(fid,'HeaderSize = %d\n',length(header));
end


%% Enter in frame stuff here andrew
if framedesc_flag == 1
   fprintf(fid,'FrameDesc = %s\n',framedesc);
   if e_flag
      idx = find(strcmp(extras.key,'FrameDesc'));
      if ~isempty(idx), extras.val(idx) = []; extras.key(idx) = [];end
   end
end

if frameunit_flag == 1
   fprintf(fid,'FrameUnit = %s\n',frameunit);
   if e_flag
      idx = find(strcmp(extras.key,'FrameUnit'));
      if ~isempty(idx), extras.val(idx) = []; extras.key(idx) = [];end
   end
end

if framemidpoint_flag ==1
   if length(framemidpoint) ~= size(dta,4)
      error('Number of frames in -framemidpoint does not match size of data');
```

```
    end
    if length(unique(framemidpoint)) ~= length(framemidpoint)
        disp('WARNING: There are duplicate entries in -framemidpoint');
    end
    if find(diff(framemidpoint) <= 0)
        disp('WARNING: -framemidpoint is not increasing in time');
    end
    if size(framemidpoint,1) ~= 1
        framemidpoint = framemidpoint';
    end
    if size(framemidpoint,1) ~= 1
        error('WARNING: -framemidpoint must be a 1-D vector');
    end
    fprintf(fid,'FrameMidpoint = %s\n',regexprep(num2str(framemidpoint),'\s+',' '));
    if e_flag
        idx = find(strcmp(extras.key,'FrameMidpoint'));
        if ~isempty(idx), extras.val(idx) = []; extras.key(idx) = [];end
    end
end

if framestart_flag ==1
    if length(framestart) ~= size(dta,4)
        error('Number of frames in -framestart does not match size of data');
    end
    if length(unique(framestart)) ~= length(framestart)
        disp('WARNING: There are duplicate entries in -framestart');
    end
    if find(diff(framestart) <= 0)
        disp('WARNING: -framestart is not increasing in time');
    end
    if size(framestart,1) ~= 1
        framestart = framestart';
    end
    if size(framestart,1) ~= 1
        error('WARNING: -framestart must be a 1-D vector');
    end

    fprintf(fid,'FrameStart = %s\n',regexprep(num2str(framestart),'\s+',' '));
    if e_flag
        idx = find(strcmp(extras.key,'FrameStart'));
        if ~isempty(idx), extras.val(idx) = []; extras.key(idx) = [];end
    end
end

if frameduration_flag ==1
    if length(frameduration) ~= size(dta,4)
        error('Number of frames in -frameduration does not match size of data');
    end
    if size(frameduration,1) ~= 1
        frameduration = frameduration';
    end
    if size(frameduration,1) ~= 1
        error('WARNING: -frameduration must be a 1-D vector');
```

```
      end

      if e_flag
         idx = find(strcmp(extras.key,'FrameDuration'));
         if ~isempty(idx), extras.val(idx) = []; extras.key(idx) = [];end
      end
   end

   if e_flag == 1
      if ~isempty(fieldnames(extras))
         for i = 1:length(extras.key)
            if strcmp(extras.key{i},'ScalingUnit') && suOff == 1
               %              disp('Using modified scaling unit.');
            elseif strcmp(extras.key{i},'ScalingFactor') && sfOff == 1
               %              disp('Using modified scaling factor.');
            elseif strcmp(extras.key{i},'SOPInstanceUID') && doRMHA == 1
               %              fprintf(fid,'ReferenceUID = %s\n',extras.val{i});
            elseif strcmp(extras.key{i},'roiNames')
               if ~names_flag
                  names_flag = 1;
                  roiNames = cell(length(extras.val{i}),1);
                  for j = 1:length(extras.val{i}),roiNames{j} = extras.val{i}{j};end
               end
            else
               fprintf(fid,'%s = %s\n',extras.key{i},regexprep(num2str(extras.val{i}),'\s+',' '));
            end
         end
      end
   end
end

fprintf(fid,'ElementType = %s\n',c_print);

%fname = sprintf('%s.raw',filename);
fname = filename;
if contains(fname,'/')
   fname = CharParse(fname,'/',2);
   fname = fname{length(fname)};
elseif contains(fname,'\')
   fname = CharParse(fname,'\',2);
   fname = fname{length(fname)};
end

% CLEAN UP THE ROI FOR RMHA OUTPUT
if doRMHA == 1
   if ~isempty(find(mod(dta,1)>0, 1))
      disp('Warning: Only positive integer values supported with RMHA');
      dta = round(dta);
   end
   if ~isempty(find(dta<0, 1))
      disp('Warning: Only positive integer values supported with RMHA');
      dta(dta<0) = 0;
```

```
end

%colorIdx = unique(dta);
%colorIdx(colorIdx==0) = [];
%if islogical(colorIdx), colorIdx = cast(colorIdx,'uint8');end
colorIdx = 1:double(max(dta(:)));

if length(colorIdx) > 255
   disp('Warning:Only 255 ROIs supported.');
   colorIdx = colorIdx(1:255);
end

if colors_flag == 0
   if length(colorIdx) < 16
      colorNames = {'red','green','blue','cyan','magenta','yellow',...
         'gray','white','darkRed','darkGreen','darkBlue',...
         'darkCyan','darkMagenta','darkgoldenrod','darkgray','lightgray'};
   else
      colorFull = {'khaki','violet','plum','aquamarine','lightsteelblue',...
         'lightskyblue','silver','skyblue','palegreen','orchid',...
         'burlywood','hotpink','lightsalmon','tan','lightgreen',...
         'aqua','cyan','fuchsia','magenta','yellow',...
         'darkgray','darkgrey','darksalmon','sandybrown','lightcoral',...
         'turquoise','salmon','cornflowerblue','mediumturquoise','mediumorchid',...
         'darkkhaki','mediumpurple','palevioletred','mediumaquamarine','greenyellow',...
         'darkseagreen','rosybrown','gold','mediumslateblue','coral',...
         'deepskyblue','dodgerblue','tomato','deeppink','orange',...
         'darkturquoise','goldenrod','cadetblue','yellowgreen','lightslategray',...
         'lightslategrey','blueviolet','darkorchid','mediumspringgreen','peru',...
         'slateblue','darkorange','royalblue','indianred','gray',...
         'grey','slategray','slategrey','chartreuse','springgreen',...
         'lightseagreen','steelblue','lawngreen','darkviolet','mediumvioletred',...
         'mediumseagreen','chocolate','darkgoldenrod','orangered','dimgray',...
         'dimgrey','limegreen','crimson','sienna','olivedrab',...
         'darkcyan','darkmagenta','darkslateblue','seagreen','olive',...
         'purple','teal','blue','lime','red',...
         'brown','firebrick','darkolivegreen','saddlebrown','forestgreen',...
         'darkslategray','darkslategrey','indigo','mediumblue','midnightblue',...
         'darkblue','darkred','green','maroon','navy',...
         'white','snow','ghostwhite','azure','ivory',...
         'mintcream','floralwhite','aliceblue','lavenderblush','seashell',...
         'honeydew','whitesmoke','lightcyan','lightyellow','oldlace',...
         'cornsilk','linen','beige','lavender','lemonchiffon',...
         'lightgoldenrodyellow','mistyrose','papayawhip','antiquewhite','blanchedalmond',...
         'bisque','moccasin','gainsboro','peachpuff','paleturquoise',...
         'navajowhite','pink','wheat','palegoldenrod','lightgray',...
         'lightgrey','lightpink','powderblue','thistle','lightblue',...
         'darkgreen'};
      if length(colorIdx) > 145
         colorFull = [colorFull colorFull];
         colorFull = colorFull(1:255);
      end
      colorSub = round(linspace(1,length(colorFull),length(colorIdx)));
```

```
            colorNames = colorFull(colorSub);
        end
    elseif colors_flag == 1
        if length(colorNames)~=length(colorIdx)
            error('Number of colors does not match number of ROI regions');
        end
    else
        cmat = vqpalette(cmap);
        cx = 1:size(cmat,1);
        cxi = linspace(1,size(cmat,1),255);
        for cc = 1:3
            cmati(:,cc) = interp1(cx,cmat(:,cc),cxi);
            cmati(:,cc) = round(cmati(:,cc)./max(cmati(:,cc))*255);
        end
        for cc = 1:255, colorNames{cc} = '#';
            for ee = 1:3
                colorNames{cc} = [colorNames{cc} dec2hex(cmati(cc,ee),2)];
            end
        end
        colorSub = round(linspace(1,length(colorNames),length(colorIdx)));
        colorNames = colorNames(colorSub);
    end


    % MORE AVAILABLE
    % http://www.w3.org/TR/SVG/types.html#ColorKeywords

%       roiCnt = 0;
    if ~isempty(colorIdx)
        for i = 1:max(colorIdx)

            %% SET ROI NAME
            if names_flag && (numel(roiNames)>=i)
                roiName = roiNames{i};
            else
                roiName = sprintf('ROI-%d',i);
            end

            %% CHECK FOR NON-ZERO
            if find(colorIdx == i)
                fprintf(fid,'ROI[%d] = %s:%s:0:0:128:255\n',i,roiName,colorNames{i});
            else
                fprintf('WARNING: No voxels in %s\n',roiName);
                fprintf(fid,'ROI[%d] = %s:transparent:1:0:128:255\n',i,roiName);
            end
        end
    end
end

% define data to write here (so that we know compressed data size)
towrite = [];
if doRMHA==1
```

```
        % USE RLE ENCODING
        dta = dta(:)';
        tmp = [find(dta(1:end-1) ~= dta(2:end)) length(dta)];
        nums = diff([0 tmp]);
        vals = dta(tmp);
        towrite = zeros(size(dta));
        cnt = 1;
        charmax = 255;
        for i = 1:length(tmp)
          if nums(i) > charmax
            ntmp = [repmat([vals(i) charmax],1,floor(nums(i)/charmax)) vals(i)
mod(nums(i),charmax)];
            towrite(cnt:(cnt+length(ntmp)-1)) = ntmp;
            cnt = cnt + length(ntmp);
          else
            towrite(cnt:cnt+1) = [vals(i) nums(i)];
            cnt = cnt + 2;
          end
        end
        towrite = towrite(1:cnt-1);
      else
        if hdronly_flag == 0
          if zlib_flag
            dta = cast(dta, c_cast);
            towrite = zlib_deflate(dta);
            c_write = 'uint8';
          elseif jp2_flag

            if length(size(dta))~=3
              fprintf('\nJPEG data must have 3 dimensions for writing. Exiting.\n\n');
              return;
            end
            if size(dta,3)~=3 && size(dta,3)~=4
              fprintf(['\nJPEG data 3rd dimension must be length 3 (RGB) ' ...
                'or 4 (RGBA). Exiting.\n\n']);
              return;
            else
              fprintf(fid,'ElementNumberOfChannels = %d\n',size(dta,3));
            end

            towrite = dta;

          elseif jpg_flag || png_flag || tif_flag

            if length(size(dta))<3
              fprintf('\nJPEG, PNG, and TIFF data must have 3 dimensions for writing.
Exiting.\n\n');
              return;

            else
              fprintf(fid,'ElementNumberOfChannels = %d\n',size(dta,3));
            end
```

```
            towrite = dta;

        else

            towrite = dta;
        end
    end
end

% finish writing header
if doMHA == 0

    if zlib_flag
        filename((end-2):(end+1)) = 'zraw';
        fname((end-2):(end+1)) = 'zraw';
        fprintf(fid, 'CompressedDataSize = %d\n', length(towrite));
    elseif jp2_flag
        filename((end-2):end) = 'jpg';
        fname((end-2):end) = 'jpg';
    elseif jpg_flag
        fprintf(fid,'RGBRepresentation = %d\n', 4);
        fprintf(fid,'ShowRGB = %d\n', 1);

        filename((end-3):end+4) = '%04d.jpg'; % image names are zero padded
        fname((end-3):end+4) = '%04d.jpg';
        if length(sz)==3
            fname = sprintf('%s %d %d %d',fname,0,sz(end)-1,1);
        else
            fname = sprintf('%s %d %d %d',fname,0,0,1);
        end
    elseif png_flag
        fprintf(fid,'RGBRepresentation = %d\n', 4);
        fprintf(fid,'ShowRGB = %d\n', 1);

        filename((end-3):end+4) = '%04d.png';
        fname((end-3):end+4) = '%04d.png';
        if length(sz)==3
            fname = sprintf('%s %d %d %d',fname,0,sz(end)-1,1);
        else
            fname = sprintf('%s %d %d %d',fname,0,0,1);
        end
    elseif tif_flag
        fprintf(fid,'RGBRepresentation = %d\n', 4);
        fprintf(fid,'ShowRGB = %d\n', 1);

        filename((end-3):end+4) = '%04d.tif';
        fname((end-3):end+4) = '%04d.tif';
        if length(sz)==3
            fname = sprintf('%s %d %d %d',fname,0,sz(end)-1,1);
        else
            fname = sprintf('%s %d %d %d',fname,0,0,1);
        end
```

```
    else

      filename((end-2):end) = 'raw';
      fname((end-2):end) = 'raw';

    end

    fprintf(fid,'ElementDataFile = %s\n',fname);
    fclose(fid);

    % open file for writing raw
    if hdronly_flag == 0 && ~jpg_flag && ~jp2_flag && ~png_flag && ~tif_flag
%       fname = sprintf('%s.raw',filename);
      fid = fopen(filename,'w','l');
    end

  else
    if zlib_flag
      fprintf(fid, 'CompressedDataSize = %d\n', length(towrite));
    end
    fprintf(fid,'ElementDataFile = LOCAL\n');
  end

  if h_flag == 1 && hdronly_flag == 0
    fwrite(fid,header);
  end

  % write data if needed
  if hdronly_flag == 0

    if jp2_flag

      imwrite(towrite,filename,'jp2','mode','lossless');

    elseif jpg_flag

      filename=strrep(filename,'\','\\');
      if length(sz)==3
        for jj=1:sz(end)

          jpgim=towrite(:,:,:,jj);
          imwrite(jpgim,sprintf(filename,jj-1),'jpg','BitDepth',12);
        end
      elseif length(sz)==2
        imwrite(towrite,sprintf(filename,0),'jpg');
      end
    elseif png_flag

      filename=strrep(filename,'\','\\');
      if length(sz)==3
        for jj=1:sz(end)

          pngim=towrite(:,:,:,jj);
```

```
            imwrite(pngim,sprintf(filename,jj-1),'png');
          end
        elseif length(sz)==2
          imwrite(towrite,sprintf(filename,0),'png');
        end
      elseif tif_flag

        filename=strrep(filename,'\','\\');
        if length(sz)==3
          for jj=1:sz(end)

            tifim=towrite(:,:,:,jj);
            imwrite(tifim,sprintf(filename,jj-1),'tif');
          end
        elseif length(sz)==2
          imwrite(towrite,sprintf(filename,0),'tif');
        end

      else

        fwrite(fid,towrite,c_write);
        fclose(fid);
      end
    end

end



end

function val = getFieldOrDefault(info,field, default)
if ~ iscell(field); field = (68); end
useDefault = 1;
for f = field
  if isfield(info,field{1})
    if ~isempty(info.(field{1}))
      useDefault = 0;
      val = info.(field{1});
    end
  end
end
if useDefault
  val = default;
end
return
end

function [img,i_info] = mhdImport(fname)
%
% Function to read in an image of the form fname.mhd & fname.raw
%
```

```
% [img,i_info] = mhdImport(fname, <dflag>)
%
% img --> resulting image
% i_info --> other information (pixel size, data type, etc)
% fname --> filename.mhd
%
% Joseph Brook - 11/11/2019

dflag = 1;

% IDENTIFY PATH
dpath = '';
if ~isempty(findstr(fname,'/'))
  fname_tmp = CharManip(fname,'\',1,'/');
  str_sets = CharParse(fname_tmp,'/',2);
  dpath = CharManip(fname_tmp,str_sets{length(str_sets)},0);
elseif ~isempty(findstr(fname,'\'))
  str_sets = CharParse(fname,'\',2);
  dpath = CharManip(fname,str_sets{length(str_sets)},0);
end

% CHECK FOR MHD OR MHA
l = length(fname);
if lower(fname((l-2):l))=='mhd'
  doMHA = 0;
elseif lower(fname((l-2):l))=='mha'
  doMHA = 1;
else
  error('Filename should end in mhd or mha');
end

zip_str = [];
byte_rd = 'l';
roiCnt = 0;
headersize = 0;
i_info.scaling = 1.0;
i_info.intercept = 0.0;
n_dims = 0;
[fid,message] = fopen(fname,'r');
if fid == -1
    err_msg = sprintf('Unable to open file %s: %s',fname,message);
    error(err_msg);
end
i_info.extras = struct;
eCnt = 0;
endCheck = 0;
roiCnt = 0;
numchan = 1;
while endCheck == 0
  in = fgetl(fid);
  if strfind(in,'=')>0
    in = CharParse(in,' = ',2);
    if length(in)>1
```

```
tagname = cast(in{1},'char');
tagname(ismember(tagname,', ''')) = [];
if contains(tagname,'ROI[')
 tagroi = tagname;
 tagname = '3DROI';
end
switch tagname
 case 'ObjectType'
  o_type = char(in{2});
 case 'ImageType'
  i_info.imagetype = char(in{2});
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
 case 'NDims'
  n_dims = str2num(char(in{2}));
  i_info.n_dims = n_dims;
 case 'BinaryData'
  bin_str = char(in{2});
 case 'BinaryDataByteOrderMSB'
  byte_str = lower(char(in{2}));
  byte_str(byte_str==' ') = '';
  i_info.bit_type = byte_str;
  if strcmp(byte_str,'true')==1
   byte_rd = 'b';
  else
   byte_rd = 'l';
  end
 case 'CompressedData'
  zip_str = char(in{2});
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,zip_str,0);
 case 'TransformMatrix'
  t_matr = str2num(in{2});
  i_info.tMatrix = t_matr;
 case 'Offset'
  i_info.offset = str2num(in{2});
 case 'CenterOfRotation'
  i_info.cor = str2num(in{2});
 case 'AnatomicalOrientation'
  i_info.orientation = in{2};
 case 'HeaderSize'
  headersize = str2num(in{2});
 case 'ElementSpacing'
  spacing = str2num(in{2});
  i_info.p_size = str2num(in{2});
 case 'ElementNumberOfChannels'
  numchan = str2num(in{2});
  i_info.numchannels = numchan;
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,numchan,0);
 case 'Cropping'
  i_info.cr = eval(in{2});
 case 'DimSize'
  i_info.size = str2num(in{2});
 case 'ElementType'
```

```
  el_str = in{2};
  switch el_str
    case 'MET_USHORT'
      el_read = 'ushort';
      ml_read = 'uint16';
    case 'MET_CHAR'
      el_read = 'char';
      ml_read = 'int8';
    case 'MET_UCHAR'
      el_read = 'uchar';
      ml_read = 'uint8';
    case 'MET_FLOAT'
      el_read = 'float';
      ml_read = 'single';
    case 'MET_INT'
      el_read = 'int';
      ml_read = 'int32';
    case 'MET_DOUBLE'
      el_read = 'double';
      ml_read = 'double';
    case 'MET_UINT'
      el_read = 'uint';
      ml_read = 'uint32';
    case 'MET_SHORT'
      el_read = 'short';
      ml_read = 'int16';
  end
  i_info.type = el_read;
case 'ElementDataFile'
  dname = in{2};
case 'DoseUnits'
  i_info.doseUnit = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'TotalDose'
  i_info.DICOMid = str2num(in{2});
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},1);
case 'DoseValue'
  i_info.DICOMid = str2num(in{2});
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},1);
case 'Dose'
  i_info.DPid = str2num(in{2});
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'PatientsWeight'
  i_info.DICOMweight = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'PatientsSex'
  i_info.patientssex = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'PatientsBirthDate'
  i_info.patientsbirthdate = str2num(in{2});
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},1);
case 'Magnification'
  i_info.magnification = str2num(in{2});
```

```
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,str2num(in{2}),1);
case 'SoftwareVersion'
  i_info.softwareversion = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'Weight'
  i_info.DPweight = str2num(in{2});
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},1);
case 'Intercept'
  i_info.intercept = str2num(in{2});
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},1);
case 'ScalingFactor'
  i_info.scaling = str2num(in{2});
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},1);
case 'ScalingUnit'
  i_info.scaling_unit = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'Modality'
  i_info.modality = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'PatientsName'
  i_info.patientsname = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'PatientID'
  i_info.patientsid = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'SeriesDescription'
  i_info.seriesdescription = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'SeriesDate'
  i_info.seriesdate = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'SeriesTime'
  i_info.seriestime = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'StudyDate'
  i_info.studydate = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'StudyTime'
  i_info.studytime = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'StudyInstanceUID'
  i_info.studyinstanceuid = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'SeriesInstanceUID'
  i_info.seriesinstanceuid = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'SOPInstanceUID'
  i_info.sopinstanceuid = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'SOPClassUID'
  i_info.sopclassuid = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
```

```
case 'ContentDate'
  i_info.contentdate = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'ContentTime'
  i_info.contenttime = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'InstanceNumber'
  i_info.instancenumber = str2num(in{2});
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},1);
case 'ImageSelection'
  i_info.i_flag = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'AcquisitionDate'
  i_info.acqdate = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'AcquisitionTime'
  i_info.acqtime = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'AcquisitionDateTime'
  i_info.acqdatetime = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'ManufacturersModelName'
  i_info.modelname = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'StudyDescription'
  i_info.studydescription = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'SeriesNumber'
  i_info.seriesnumber = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'StudyID'
  i_info.studyid = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'AccessionNumber'
  i_info.accessionnumber = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
% Frame info
case 'FrameDesc'
  i_info.framedesc = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'FrameUnit'
  i_info.frameunit = in{2};
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},0);
case 'FrameMidpoint'
  i_info.framemidpoint = str2num(in{2});
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},1);
case 'FrameStart'
  i_info.framestart = str2num(in{2});
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},1);
case 'FrameDuration'
  i_info.frameduration = str2num(in{2});
  [i_info.extras,eCnt] = AddExtra(i_info.extras,tagname,eCnt,in{2},1);
case 'ROI'
```

```
        roiCnt = roiCnt + 1;
        tmp = CharParse(in{2},' ',2);
        roi.type = tmp{1};
        i_info.roi.type{roiCnt} = roi.type;
        switch i_info.roi.type{roiCnt}
         case 'box'
          for i = 1:6, i_info.roi.vals{roiCnt}(i) = str2num(tmp{i+1});end
         case 'sqbox'
          for i = 1:6, i_info.roi.vals{roiCnt}(i) = str2num(tmp{i+1});end
         otherwise
          disp('That ROI type is not supported');
          i_info.roi.vals{roiCnt} = 0;
        end
       case '3DROI'
        roinum = str2num(CharParse(CharParse(tagroi,']',0),'[',1));
        roiinfo = CharParse(in{2},':',2);
        nSet = {'name','color','hidden','immutable','alphaSlice', ...
              'alphaSurface'};
        for rr = 1:length(roiinfo)
         if rr < 3
          cmd = sprintf('i_info.roi3d(%d).%s = ''%s'';',roinum,nSet{rr},roiinfo{rr});
         else
          cmd = sprintf('i_info.roi3d(%d).%s = %d;',roinum,nSet{rr},str2num(roiinfo{rr}));
         end
         eval(cmd);
        end
       case 'Slice'
        i_info.slices = eval(in{2});
       otherwise
%         cmd = sprintf('i_info.%s = ''%s'';',strrep(in{1},' ',''),in{2});
%         eval(cmd);
     end
    end
  end
  if doMHA == 0, endCheck = feof(fid);
  else
     endCheck = strcmp(tagname,'ElementDataFile');
  end
end

% ADD ROINAMES TO EXTRAS
if isfield(i_info,'roi3d')
  roiNames = cell(length(i_info.roi3d),1);
  for i = 1:length(i_info.roi3d)
    roiNames{i} = i_info.roi3d(i).name;
  end
  i_info.extras.key{end+1} = 'roiNames';
  i_info.extras.val{end+1} = roiNames;
end

% CHECK THE INJECTED DOSE AND WEIGHT
if isfield(i_info,'DPweight'), i_info.weight = i_info.DPweight;
```

```
elseif isfield(i_info,'DICOMweight'), i_info.weight = i_info.DICOMweight;end

if isfield(i_info,'DPid'), i_info.id = i_info.DPid;
elseif isfield(i_info,'DICOMid'), i_info.id = i_info.DICOMid;end

% READ IN THE IMAGE DATA
if dflag >= 1
 if ~isempty(dpath), dname = [dpath dname]; end

 if doMHA==0
  fclose(fid);
  [fid,message] = fopen(dname,'r',byte_rd);
  if fid == -1
%      err_msg = sprintf('Unable to open file %s: %s',dname,message);
    error('Unable to open file %s: %s',dname,message);
  end
 end
 i_info.hdr = fread(fid,headersize,'uchar');
 switch (zip_str)
  case 'RLE'
   img = zeros(prod(i_info.size),1);
   rle = fread(fid,inf,el_read);
   cnt = 1;
   for i = 1:2:length(rle)
    img(cnt:(cnt+rle(i+1)-1)) = rle(i);
    cnt = cnt + rle(i+1);
   end
  case 'True'
   imgz = fread(fid,inf,'uchar=>uint8');
   img = zlib_inflate(imgz, ml_read);
  case 'JP2'
   img = imread(dname);
   i_info.size(3) = size(img,3);
  otherwise
   img = fread(fid,inf,['*' el_read]);
 end
 %img = cast(fread(fid,inf,el_read),'single');
 fclose(fid);

 % RESHAPE THE IMAGE DATA
 dims = i_info.size;
 if numchan == 1
  if n_dims == 3
   img = reshape(img,[dims(1) dims(2) dims(3)]);
  elseif n_dims == 2
   img = reshape(img,[dims(1) dims(2)]);
  elseif n_dims == 4
   img = reshape(img,[dims(1) dims(2) dims(3) dims(4)]);
  end
 else
  tmp = img;
  img = zeros([dims numchan]);
  switch n_dims
```

```
   case 2, for i = 1:numchan,img(:,:,i) = reshape(tmp(i:numchan:end),dims);end
   case 3, for i = 1:numchan,img(:,:,:,i) = reshape(tmp(i:numchan:end),dims);end
   case 4, for i = 1:numchan,img(:,:,:,:,i) = reshape(tmp(i:numchan:end),dims);end
  end
end

% SCALE THE IMAGE DATA
if i_info.scaling ~= 1 % if scaling is present
   if ismember(class(img),{'uint16', 'uint8', 'ushort'}) %and is in integer for,
      img = single(img); % cast to a single
      i_info.type = 'float';
   end
end
img = img.*i_info.scaling+i_info.intercept;
i_info.scaling = 1;
i_info.intercept = 0;

% FLIP THE IMAGE DATA
if (exist('t_matr') && exist('spacing'))
          if n_dims == 3
           if i_info.p_size(3) < 0
                    i_info.p_size(3) = i_info.p_size(3)*-1;
                    t_matr(9) = t_matr(9)*-1;
           end
           if isfield(i_info,'orientation')
                    if strcmp(i_info.orientation(1),'R')
                      t_matr(1) = t_matr(1)*-1;
                      i_info.orientation(1) = 'L';
                    end
                    if strcmp(i_info.orientation(2),'A')
                      t_matr(5) = t_matr(5)*-1;
                      i_info.orientation(2) = 'P';
                    end
                    if strcmp(i_info.orientation(3),'I')
                      t_matr(9) = t_matr(9)*-1;
                      i_info.orientation(3) = 'S';
                    end
           end
          end

          if n_dims == 2
           %if t_matr(1)==1, img=fliplr(img);end
           %if t_matr(4)==1, img=flipud(img);end
           %img = img';
          elseif n_dims == 3
           if t_matr(1)==1, img=flip3d(img,1);end
           if t_matr(5)==1, img=flip3d(img,2);end
           if t_matr(9)==1, img=flip3d(img,3);end
           i_info.tMatrix = [-1 0 0 0 -1 0 0 0 -1];
   if ~isequal(abs(t_matr([1,5,9])), [1,1,1])
      disp(['Warning: Given Transform Matrix not supported at this ' ...
          'time']);
```

```
    end
          elseif n_dims == 4
    if t_matr(1)==1, img=img(end:-1:1,:,:,:);end
    if t_matr(6)==1, img=img(:,end:-1:1,:,:);end
    if t_matr(11)==1, img=img(:,:,end:-1:1,:);end
    i_info.tMatrix = [-1 0 0 0 0 -1 0 0 0 -1 0 t_matr(13:end)];
    if ~isequal(abs(t_matr([1,6,11])), [1,1,1])
       disp(['Warning: Given Transform Matrix not supported at this ' ...
            'time.']);
    end
end
 end

 % CHECK FOR TRANSPOSITION
 if dflag == 2
    if isfield(iinfo,'p_size')
       tmp = iinfo.p_size;
       iinfo.p_size(1) = tmp(2);
       iinfo.p_size(2) = tmp(1);
    end
    if isfield(iinfo,'size')
       tmp = iinfo.size;
       iinfo.size(1) = tmp(2);
       iinfo.size(2) = tmp(1);
    end
    if n_dims == 2, img = img';
    elseif n_dims == 3, img = permute(img,[2 1 3]);
    elseif n_dims == 4, img = permute(img,[2 1 3 4]);
    end
 end
else
 img = 0;
 fclose(fid);
end

function [extras,eCnt] = AddExtra(extras,key,eCnt,val,evalStr)
eCnt = eCnt + 1;
%extras(eCnt).key = key;
%extras(eCnt).val = val;
extras.key{eCnt} = key;
if evalStr
 extras.val{eCnt} = str2num(val);
else
 extras.val{eCnt} = val;
end
end
end

function [ features ] = MoCalc( img, idx, win )
%Calculate the central tendancies

[x,y,z] = size(img);
[a,b,c] = ind2sub([x,y,z],idx);
```

```
features = [];
for i = 1:size(idx)
   A = img(a(i)-win:a(i)+win,b(i)-win:b(i)+win,c(i)-win:c(i)+win);
   B = reshape(A,[size(A,1)*size(A,2)*size(A,3),1]);
   meanB = mean(B);
   modeB = mode(B);
   medB = median(B);
   varB = var(B);
   stdB = std(B);
   kurtB = kurtosis(B);
   skewB = skewness(B);
   inte = img(idx(i));
   features = [features; meanB,modeB,medB,varB,stdB,kurtB,skewB,inte];
end
end

function out = zlib_deflate(in)
% ZLIB_DEFLATE compresses input using the zlib package in java
%
% INPUTS
% in    input values (of uncompressed type)
%
% OUTPUTS
% out   compressed bytes (uint8)

import com.mathworks.mlwidgets.io.InterruptibleStreamCopier

% convert input to byte stream
in_bytes = typecast(in(:), 'uint8');
in_stream = java.io.ByteArrayInputStream(in_bytes);

% create a deflater stream
deflater_stream = java.util.zip.DeflaterInputStream(in_stream);

% copy deflated bytes into output byte array
stream_copier = InterruptibleStreamCopier.getInterruptibleStreamCopier();
out_stream = java.io.ByteArrayOutputStream();
stream_copier.copyStream(deflater_stream, out_stream);
out = typecast(out_stream.toByteArray(), 'uint8');
end

function out = zlib_inflate(in, type)
% ZLIB_INFLATE decompresses input using the zlib package in java
%
% INPUTS
% in    compressed input bytes (uint8)
% type  output type (e.g., uint8, single, double - see typecast)
%
% OUTPUTS
% out   decompressed values of output type
```

```
import com.mathworks.mlwidgets.io.InterruptibleStreamCopier

% create an inflater input stream from the compressed bytes
in_stream = java.io.ByteArrayInputStream(in);
inflater_stream = java.util.zip.InflaterInputStream(in_stream);

% copy stream
stream_copier = InterruptibleStreamCopier.getInterruptibleStreamCopier();
out_stream = java.io.ByteArrayOutputStream();
stream_copier.copyStream(inflater_stream, out_stream);
out = typecast(out_stream.toByteArray(), type);
end
```

6.2 runBMC-MATLAB.vqs

```
//Brucker MRI Conversion
//Script to take Bruker data and organise it into raw format, ignoring localisers and applying bias filter correction
//Looks for folders that begin with 201 as extracted from the Paravisions (Bruker, MA, USA) zip files
//Impanated using VivoQuant (inviCRO, MA, Boston) ver3.5
//Written by Joseph Brook
//version 4.0 (11/12/2018) - J. Brook

//Define Libraries and prebuilt functions
#include "VQSTools.vqs"
#include "ipacs.vqs"
var bruk = VQ.brukerImporter();
var dm = VQ.dataManager();

//Designate where to save the files
var dataLoc = 'Input';
var oLoc = 'data/';
rmkdir(oLoc);
var patients = VQ.lsDir(dataLoc,'201*'); //Input data

//Bring the Reorientation/Registration menu
var mw = VQ.mainWin();
var dm = VQ.dataManager();
var ctl = VQ.controler();
mw.setViewMode('Slice View','Reorientation/Registration');
var reg = VQ.currentOp();

//For each folder
for (var i = 0 ;i < patients.length; i++){
   var cdir = dataLoc + "/" +  patients[i];
   var strs = bruk.scanStudies(cdir);
   var strs = bruk.scanStudies(cdir);
   var st1 = bruk.loadDesc(cdir,'1');
   VQ.debug(strs);

   //For each scan in the folder
   for (var j = 0; j < strs.length; j++){
      var tok = strs[j].split('-');
      var idx = tok[0];
```

```
var val = tok[1];
var str = bruk.loadDesc(cdir,idx);

        //Skip the localizer
if (wildcardMatch(str,'*Localizer*'))
   continue;

//Load the data
        var path = VQ.fileExists(cdir + "/" + idx + '/pdata/' + val + '/2dseq');
            if (path === false){
            continue;}

        //Get info for the naming
bruk.load(cdir + "/" + idx + '/pdata/' + val + '/2dseq');
bruk.transferData(0,dm);

var tr = dm.getDcmString(0,'RepetitionTime');
var te = dm.getDcmString(0,'EchoTime');
        var psplit = patients[i].split('_');
        var sdesc = dm.getDesc(0,'seriesdescription');
var tak = sdesc.split('_');
var type = tak[0];
        var tw = type.split(' - ');
        if (tw[1] === 'T2') {
            continue;
        }
        var age = VQ.getInt('Age of ' + psplit[3] + psplit[4] + psplit[5] + ' on date ' +
psplit[0],'Input age of the animal',value = 0,min = 0,max = 2147483647,step = 1);

var fname = psplit[3] + psplit[4] + psplit[5] + "-" + age + "-KPC-0-" +  tw[1];

        //Prepare to flip the data
mw.setViewMode('Slice View','Reorientation/Registration');
var reg = VQ.currentOp();
        VQ.getWidget('dataSelector').setSelectionString('0');
        VQ.suspend('Tick the orientations needed to be flipped. The stomach should be
on the left and the head at the top of the coronal orientation and the spine on the left of the
sagital orientation');

//Apply transformation
VQ.currentOp().applyTransformation();

        //Apply Bias Filter
        mw.setViewMode('Slice View','Filtering');
        VQ.getWidget('dataSelector').setSelectionString('0');
        VQ.getWidget('cbSelect').setCurrentIndex(8);
        VQ.getWidget('DownsampleFactor').setValue('2');
        VQ.getWidget('cbAppendData').setChecked(false);
        VQ.getWidget('cbForce2D').setChecked(false);
        VQ.getWidget('ConvergeThreshold').setValue('0.5000');
        VQ.getWidget('buttonBox').accepted();
```

```
      //Save the data as raw
                  dm.setDesc(VQ.index(0),'StudyDescription','Library')
      var stddesc = dm.getDesc(0,'studydescription');
                  VQ.storeAsRaw(oLoc + fname + ".zraw",0);

                  //Unload the data
      dm.unloadData(0,-1);
   }
         VQ.quit()
}
```

6.3 MAS_Tool.vqs

```
#include "VQSTools.vqs"
#include "ipacs.vqs"

//Used to run MAS tool on local data given path to local reference library
//Reference library directory must contain MASsettings.txt and ref_list.txt files

var ref_lib_path =  getLocalDir() + 'data/refLib/'; //reference library directory
var data_dir = getLocalDir() + 'data/'; //Input data directory


//Initial setup
var dm = VQ.dataManager();
var mw = VQ.mainWin();

dm.unloadData(0,-1);
var maxVoxRatio = VQ.getConfig("Data/maxVoxRatioVol");
VQ.setConfig("Data/maxVoxRatioVol", 20.0, true);


//Find and list data to run MAS tool on (wildcard filter acceptable)
var studies = VQ.lsDir(data_dir,'*mhd');
var ref_lib_count = VQ.lsDir(ref_lib_path,'*rmha');

//Define which ROI names in the reference library rmha files to run for the input data
var roi_list =
['KidneyLeft','KidneyRight','Spleen','Stomach','Liver','Gallbladder','HepaticPortalVein'];

//Run for each subject
for (var i = 0;i<studies.length;i++){

         //remove file extension from input just for formatting output file name
         var pname = studies[i].split('.').slice(0, -1).join('.')
         var output_rmha = data_dir+pname+'MASTool.rmha';

         //check if output exists
         if (VQ.fileExists(output_rmha))
         {
                  //VQ.showMessage('Output for subject - ' + pname + ' - is already
found...skipping');
                  //continue;
         }
```

```
        //load preprocessed input image
        dm.openDat(0,[data_dir + studies[i]]);


    //Initialize and run MAS tool
        var MAS = VQ.wbAtlas(ref_lib_path);
        MAS.initialize();

        //Check off ROIs to run
        var mas_rois = MAS.getROIsList();
        for(var ridx=0; ridx<roi_list.length; ridx++){
            mas_rois.setChecked(roi_list[ridx].toLowerCase(),true);
        }

        //Load in MAS tool settings and run
        var settings_file = ref_lib_path + 'MASsettings.txt';
        MAS.loadSettings(settings_file);
        MAS.getWidget('dataSelector').setSelectedIndex(0);
        MAS.run()
        if(MAS.hasError()){
            VQ.suspend(MAS.takeError());
        }



        //Save resulting ROIs to local rmha file in data directory
        mw.setViewMode('Slice View','3D ROI Operator');
        var roi = VQ.currentOp();
        roi.saveROI(output_rmha);



        //Clean-up
        roi.clearAllROIs();
        MAS.close();

    dm.unloadData(0,-1);
}

VQ.setConfig("Data/maxVoxRatioVol", maxVoxRatio, true);

VQ.showMessage('Done for all subjects!');




//Returns the directory of the current VivoScript.
function getLocalDir(){
    // Is it run from the App
    if (VQ.isInteractive()){
        var localDir = VQ.getConfig('cwdVQScript');
    }
```

```
    else{
        var localDir = VQ.getConfig('VQScript/QuickAccess/'+arguments[0])
    }
    // Switch to all forward slashes
    while (localDir.indexOf('\\') > -1)
        localDir = localDir.replace('\\','/');
    // Remove last piece (which is the file name)
    var tok = localDir.split('/');
    tok = tok.splice(0,tok.length-1);
    var dirout = tok.join('/');
    dirout = dirout + '/';

    return dirout;
}
```

6.4 MASToolSetting.txt

```
OutputFileExtension = mhd
UseCompression = 1
SaveRegistrationMovieImages = 0
SaveMetric = 1
BoundingBoxName = PerOrgan
BoundingBoxPadding = 0
AverageName = BestN
AverageBestN = 6
BestNMetric = MI
AverageThreshold = 0.44
ThresholdName = Fixed
HistogramMatchingFlag = 1
HistogramMatchingBins = 1024
HistogramMatchingPoints = 10
Winsorize = 0
WinsorizeLowerQuantile = 0.005
WinsorizeUpperQuantile = 0.995
LinearTransformTypes = Affine
LinearInitName = Moments
LinearMetricType = MattesMutualInformation
MetricHistogramBins = 64
LinearOptimizerType = ConjugateGradientDescent
LinearSamplingPercentage = 0.02
LinearNumIter = 200
LinearMinStepLength = 0.0001
LinearLearningRate = 8
LinearRelaxation = 0.8
LinearMultiResFlag = 1
LinearShrinkFactors = 2 1
LinearSmoothingSigmas = 1 0
DeformablePreRegName = None
DeformableName = DiffeomorphicDemons
DeformableMetricName = MattesMutualInformation
DeformableSigma = 0.5
DeformableTolerance = 0.01
DeformableNumIter = 150
DeformableMaxStep = 0.5
DeformableMultiResFlag = 1
```

```
DeformableShrinkFactors = 4 2 1
DeformableSmoothingSigmas = 2 1 0
Verbose = 2
```

6.5 RebuildModle.m

```
%% Rebuild Model
clearvars
addpath('Code')
tic
inDir = ('Library');

d = 'Library/*.mhd';
D = dir(d);
k = size(D,1);
for l = 1:k
   dNameTemp = CharParse(D(l).name,'.',0);
   dName{l} = CharParse(dNameTemp,'_',0);
end
duName = unique(dName);
%%%%%%
outDir = 'Library';


%% Find the maximum magnitude of vectors from centre of mass of organs to general centre
f = 'Library/*1.rmha'; % Get all rmha files
F = dir(f);
a = size(F,1);
mag = zeros(1,a);
for i = 1:a
   fROI = sprintf('%s/%s',outDir,F(i).name);
   [ROI,rInfo] = mhdImport(fROI);
   %% Centre of each ROI
   [xN,yN,zN] = deal(zeros(7,1));
   for j = 1:7 %for loop for each ROI
      Roi = ROI;
      Roi(Roi~=j)=0;
      Roi = Roi./j;
      cen = regionprops(Roi,'centroid');
      xN(j) = cen.Centroid(2);
      yN(j) = cen.Centroid(1);
      zN(j) = cen.Centroid(3);
      xyzN(j,[2,1,3]) = cen.Centroid; %Reorder as output of cen is y,x,z
   end
   xyzNm =  round(mean(xyzN));
   xyzNm = reshape(xyzNm, [1 3]);
   mag(i) = mean(sum(([xyzN(:,1),xyzN(:,2),xyzN(:,3)] -
[xyzNm(1),xyzNm(2),xyzNm(3)]).^2,2).^(1/2));
end
magColl = round(mean(mag));
save('Code/magMean.mat','magColl');

%% Pancreas and Tumour Cloud
id = {};
```

```
[PancCloudWarp, TumCloudWarp, PancCloudWarpKPWT] = deal(zeros(120,80,160));
[roiPanc, roiTum] = deal([]);
for m = 1:size(F,1)
   fROI = sprintf('%s/%s',outDir,F(m).name);
   [ROI,~] = mhdImport(fROI);
   %% Centre of each ROI
   xyzN = zeros(7,3);
   for n = 1:7 %for loop for each ROI Kidneys Spleen Stomach, Liver Gallbladder and Hepatic
Portal Vein
      Roi = ROI;
      Roi(Roi~=n)=0;
      Roi = Roi./n;
      cen = regionprops(Roi,'centroid');
      xyzN(n,[2,1,3]) = cen.Centroid; %Reorder as output of cen is y,x,z
   end
   xyzNm = round(mean(xyzN));
   xyzNm = reshape(xyzNm, [1 3]);
   shiftT = eye(4);
   PancCenterVec = size(Roi)/2;
   shiftImg{m} = -( xyzNm - PancCenterVec);
   shiftT(4,1:3) = shiftImg{m}([2,1,3]);
   tformShift = affine3d();
   tformShift.T = shiftT;
   mag = mean(sum(([xyzN(:,1),xyzN(:,2),xyzN(:,3)] -
[xyzNm(1),xyzNm(2),xyzNm(3)]).^2,2).^(1/2));
   scaleFac = magColl/mag; %Calcuate the scaling factor
   Rin = imref3d(size(Roi));
   Rin.XWorldLimits = Rin.XWorldLimits-2*mean(Rin.XWorldLimits); %Find the limits for the
translation
   Rin.YWorldLimits = Rin.YWorldLimits-2*mean(Rin.YWorldLimits);
   Rin.ZWorldLimits = Rin.ZWorldLimits-2*mean(Rin.ZWorldLimits);

   %% Pancreas
   Roi = ROI;
   Roi(Roi~=8)=0;
   Roi = Roi./8;
   img = imwarp(Roi,Rin,tformShift,'linear','OutputView', Rin,'FillValues', min(Roi(:)));
   img = imresize3(img,scaleFac);
   img = FitImageToMatrix(img,size(Roi));
   PancCloudWarp = PancCloudWarp+img;
   roiPanc = cat(4,roiPanc,img);

   %% Tumour
   Roi = ROI;
   Roi(Roi~=9)=0;
   Roi = Roi./9;
   imgTum = imwarp(Roi,Rin,tformShift,'linear','OutputView', Rin,'FillValues', min(Roi(:)));
   imgTum = imresize3(imgTum,scaleFac);
   imgTum = FitImageToMatrix(imgTum,size(Roi));
   TumCloudWarp = TumCloudWarp+imgTum;
   roiTum = cat(4,roiTum,imgTum);

   %% Index
```

```
    idTemp = CharParse(F(m).name,'.',0);
    id = [id;idTemp];

end

DiMask(:,:,1) = [0 1 0; 1 1 1; 0 1 0];
DiMask(:,:,2) = 1;
DiMask(:,:,3) = DiMask(:,:,1);

% Pancreas
PancDi = imdilate(PancCloudWarp,DiMask);
PancSmooth = imgaussfilt3(PancDi, 1);
PancSmooth(PancSmooth<0) = 0;
PancSmooth = PancSmooth./max(max(max(PancSmooth)));
IDs.ID = id;
PancCloudKPC = struct('PancreasCloud',PancSmooth,'AnimalID', IDs,'NumberOfScans',size(id,1));
save('Code/PancCloudKPC.mat','PancCloudKPC');

% Tumour
TumSmooth = imgaussfilt3(TumCloudWarp, 1);
TumSmooth(TumSmooth<0) = 0;
TumSmooth = TumSmooth./max(max(max(TumSmooth)));
IDs.ID = id;
TumCloud = struct('TumourCloud',TumSmooth,'AnimalID', IDs,'NumberOfScans',size(id,1));
save('Code/TumCloud.mat','TumCloud');

OffSet = [1 2];
glcm_window = 11;
for i = 1:size(duName,2)
    otherProbName = sprintf('%s/%s_OtherProb.mhd',inDir,duName{i});

    tempName = sprintf('%s/%s.mhd',inDir,duName{i});
    runWBAModel;

    fROI = sprintf('%s/%s',outDir,F(i).name);
    [ROI,rInfo] = mhdImport(fROI);
    id = CharParse(F(i).name,'.',0);
    idType = CharParse(id,'-',2);
    idType = idType{3};
    if strcmp('KPC',idType)
        PancCloud = PancCloudKPC;
    else
        PancCloud = PancCloudKP;
        TumCloud.TumourCloud = zeros(size(ROI));
    end
    fScan = sprintf('%s/%s.mhd',outDir,id);
    [scan,sInfo] = mhdImport(fScan);
    scanRe = reshape(scan,[size(scan,1)*size(scan,2)*size(scan,3),1]);
    T = otsuthresh(scanRe);
    scanD = scan;
    scanD(scanD < T) = 0;
    scanD(scanD >= T) = 1;
```

```
    di(:,:,1) = [0 0 0; 0 1 0; 0 0 0];
    di(:,:,2) = [0 1 0; 1 1 1; 0 1 0];
    di(:,:,3) = [0 0 0; 0 1 0; 0 0 0];
    for n = 1:3
        scanD = imdilate(scanD,di);
    end

    %% Centre of each ROI + the average distance of that ROI to the Pancreas
    xyzN = zeros(7,3);
    for n = 1:7 %for loop for each ROI
        Roi = ROI;
        Roi(Roi~=n)=0;
        Roi = Roi./n;
        cen = regionprops(Roi,'centroid');
        xyzN(n,[2,1,3]) = cen.Centroid; %Reorder as output of cen is y,x,z
    end
    xyzNm = round(mean(xyzN));
    xyzNm = reshape(xyzNm, [1 3]);
    shiftT = eye(4);
    PancCenterVec = size(Roi)/2;
    shiftImg{m} = ( xyzNm - PancCenterVec);
    shiftT(4,1:3) = shiftImg{m}([2,1,3]);
    tformShift = affine3d();
    tformShift.T = shiftT;
    mag = mean(sum(([xyzN(:,1),xyzN(:,2),xyzN(:,3)] -
[xyzNm(1),xyzNm(2),xyzNm(3)]).^2,2).^(1/2));
    scaleFac = magColl/mag;

    Rin = imref3d(size(Roi));
    Rin.XWorldLimits = Rin.XWorldLimits-2*mean(Rin.XWorldLimits);
    Rin.YWorldLimits = Rin.YWorldLimits-2*mean(Rin.YWorldLimits);
    Rin.ZWorldLimits = Rin.ZWorldLimits-2*mean(Rin.ZWorldLimits);

    %% Transform probability cloud
    PancCloudNovel = PancCloud.PancreasCloud;
    PancCloudNovel = imresize3(PancCloudNovel,1/scaleFac);
    PancCloudNovel = FitImageToMatrix(PancCloudNovel,size(Roi));
    PancCloudNovel = imwarp(PancCloudNovel,Rin,tformShift,'linear','OutputView',
Rin,'FillValues', min(Roi(:)));
    PancCloudNovel(PancCloudNovel<0)=0;
    pCloud = sprintf('%s/%s_PancCloud.mhd',outDir,id);
    mhdExport(PancCloudNovel,pCloud,sInfo);

    if strcmp('KPC',idType)
        TumCloudNovel = TumCloud.TumourCloud;
        TumCloudNovel = imresize3(TumCloudNovel,1/scaleFac);
        TumCloudNovel = FitImageToMatrix(TumCloudNovel,size(Roi));
        TumCloudNovel = imwarp(TumCloudNovel,Rin,tformShift,'linear','OutputView',
Rin,'FillValues', min(Roi(:)));
        TumCloudNovel(TumCloudNovel<0)=0;
        tCloud = sprintf('%s/%s_TumCloud.mhd',outDir,id);
        mhdExport(TumCloudNovel,tCloud,sInfo);
        areaRoi = PancCloudNovel+TumCloudNovel;
```

```
else
   areaRoi = PancCloudNovel;
end

areaRoi(areaRoi<0.02) = 0;
areaRoi(areaRoi>0)=1;
areaRoi = areaRoi.*scanD;
aRoi = sprintf('%s/%s_AreaRoi.rmha',outDir,id);
mhdExport(areaRoi,aRoi,rInfo);



%% Other area
Roi = ROI;
Roi(Roi==9) = 8;
Roi(Roi~=8) = 1;
Roi(Roi==8) = 0;
otherRoi = areaRoi.*Roi;

oRoi = sprintf('%s/%s_OtherRoi.rmha',outDir,id);
mhdExport(otherRoi,oRoi,rInfo);

%%
%% Read volumes
fscan = sprintf('%s/%s',inDir,D(i).name);
[Data,~]=mhdImport(fscan);
DataP = padarray(Data,[glcm_window,glcm_window,glcm_window]);
DataP = double(DataP);

pscan = sprintf('%s/%s_PancCloud.mhd',inDir,dName{i});
[Panc,~]=mhdImport(pscan);
Panc = padarray(Panc,[glcm_window,glcm_window,glcm_window]);
idType = CharParse(dName{i},'-',2);
idType = idType{3};
if strcmp('KPC',idType)
   tscan = sprintf('%s/%s_TumCloud.mhd',inDir,dName{i});
   [Tum,~]=mhdImport(tscan);
   Tum = padarray(Tum,[glcm_window,glcm_window,glcm_window]);
else
   Tum = zeros(size(Panc));
end

% mask
fROI = sprintf('%s/%s.rmha',inDir,dName{i});
[ROI,rInfo]=mhdImport(fROI);
ROIName = rInfo.extras.val{end}; %ROI Names
Rs = size(rInfo.extras.val{end},1);
ROInames = {'KidneyLeft','KidneyRight','Spleen','Stomach','Liver',...
   'Gallbladder','HepaticPortalVein','Pancreas','Tumour'};

[Gmag, Gazi, Gele] = imgradient3(DataP);
[Gx, Gy, Gz] = imgradientxyz(DataP);
```

```
    Gxy1 = (Gx.^2+Gy.^2).^.5/(2^0);
    Gxy2 = (Gx.^2+Gy.^2).^.5/(2^1);
    Gxy3 = (Gx.^2+Gy.^2).^.5/(2^2);
    Gxy4 = (Gx.^2+Gy.^2).^.5/(2^3);
    Gxy5 = (Gx.^2+Gy.^2).^.5/(2^4);
    FA = FACalcArea(Gmag,Gazi,Gele,glcm_window);

    %% COM
    xyzN = zeros(7,3);
    for n = 1:7 %for loop for each ROI
        Roi = ROI;
        Roi(Roi~=n)=0;
        Roi = Roi./n;
        cen = regionprops(Roi,'centroid');
        xyzN(n,[2,1,3]) = cen.Centroid; %Reorder as output of cen is y,x,z
    end

    xyzNm = round(mean(xyzN));
    xyzNm = reshape(xyzNm, [1 3]);
    mag = mean(sum(([xyzN(:,1),xyzN(:,2),xyzN(:,3)] -
[xyzNm(1),xyzNm(2),xyzNm(3)]).^2,2).^(1/2));
    scaleFac = magColl/mag;
    %% ROI
    if Rs(1) == 9
        jl = [8,9];
    else
        jl = 8;
    end
    for j = jl%1:Rs %for loop for each ROI
        ROIChoice = j;
        Roi = ROI;
        Roi(Roi~=ROIChoice)=0;
        Roi = Roi./ROIChoice;
        RoiP = padarray(Roi,[glcm_window,glcm_window,glcm_window]);

        %% Find slices with label
        idxP = find(RoiP);
        [~,~,i3] = ind2sub(size(RoiP), idxP);
        indZ = unique(i3);
        RoiZ = RoiP;
        for kk = 1:glcm_window
            RoiZ  = imdilate(RoiZ,DiMask);
        end

        %% extract GLCM & GLRLM features
        feature_matrix=[];
        Glrlm = [];
        for k = 1:length(indZ)
            [indx,indy] = find((RoiP(:,:,indZ(k)))>0);
            %GLCM
            [mean_im,std_im,entropy_im,gf]= calculate_main_features((DataP(:,:,indZ(k))));

glcm_struct=calculate_glcm_Run_IdxTest((DataP(:,:,indZ(k))),glcm_window,OffSet,indx,indy);
```

```
[featur_vector,label_mat]=features_vector_Run_IdxTest(ROIName(j,:),mean_im,std_im,entropy
_im,glcm_struct,indx,indy);
        feature_matrix= cat(1, feature_matrix, featur_vector);
        %GLRLM
        [RLFeatures0,RLFeatures45,RLFeatures90,RLFeatures135] = deal([]);
        [rlm0,rlm45,rlm90,rlm135,max0,max45,max90,max135] =
calculate_glrlm_Run_IdxTest(DataP(:,:,indZ(k)),glcm_window,indx,indy);
        for jj = 1:size(rlm0,1)
          [RLFeatures0(jj,:)] = calculate_glrlm_features(rlm0{jj,1},max0(:,jj),glcm_window);
        end
        for jj = 1:size(rlm45,1)
          [RLFeatures45(jj,:)] = calculate_glrlm_features(rlm45{jj,1},max45(:,jj),glcm_window);
        end
        for jj = 1:size(rlm90,1)
          [RLFeatures90(jj,:)] = calculate_glrlm_features(rlm90{jj,1},max90(:,jj),glcm_window);
        end
        for jj = 1:size(rlm135,1)
          [RLFeatures135(jj,:)] =
calculate_glrlm_features(rlm135{jj,1},max135(:,jj),glcm_window);
        end
        Glrlm = cat(1,Glrlm,[RLFeatures0, RLFeatures45, RLFeatures90, RLFeatures135]);

    end
    feature_matrix= cat(1, feature_matrix);
    Features = cat(2,feature_matrix(:,1:23), Glrlm);

    %% Spatial features
    fea = MoCalc(DataP,idxP,glcm_window);
    Features = cat(2,Features, fea);

    %% Calculate the Gradient Magnitude and the FA values
    Grad = Gmag(idxP);
    FA3 = FA(idxP);
    Gxx = Gx(idxP);
    Gyy = Gy(idxP);
    Gxy1C = Gxy1(idxP);
    Gxy2C = Gxy2(idxP);
    Gxy3C = Gxy3(idxP);
    Gxy4C = Gxy4(idxP);
    Gxy5C = Gxy5(idxP);
    Features = cat(2,Features,Grad,Gxx,Gyy,Gxy1C,Gxy2C,Gxy3C,Gxy4C,Gxy5C,FA3);

    %% COM
    [xx,yy,zz] = Gen3DCoords(RoiP);
    difCOM = zeros(size(xx,1),3);
    for k = 1:size(xx,1)
      difCOM(k,1) = (xyzNm(1)-xx(k,1))/scaleFac;
      difCOM(k,2) = (xyzNm(2)-yy(k,1))/scaleFac;
      difCOM(k,3) = (xyzNm(3)-zz(k,1))/scaleFac;
    end
    Features = cat(2,Features, difCOM);
```

```
    %% Other organ probability
    PancC = Panc(idxP);
    TumC = Tum(idxP);
    OPname = sprintf('%s/%s_OtherProb.mhd',inDir,dName{i});
    [oProb, ~] = mhdImport(OPname);
    oProb = padarray(oProb,[glcm_window,glcm_window,glcm_window]);
    opGath = oProb(idxP);
    Features = cat(2,Features, PancC, TumC, opGath);

    %% Save features
    saveName = dName{i};
    fname = sprintf('%s/%s_%s_Features.mat',outDir,saveName,char(ROIName(j,:)));
    parsave(fname,Features);
  end


  %% Bounding Box
  % mask
  fROI = sprintf('%s/%s_AreaRoi.rmha',inDir,dName{i});
  [ROIb,rInfo]=mhdImport(fROI);
  ROInames = {'BoundingBox'};
  Rs = size(rInfo.extras.val{end},1);
  RoiP = padarray(ROIb,[glcm_window,glcm_window,glcm_window]);


  %% Find slices with label
  idxP = find(RoiP);
  [~,~,i3] = ind2sub(size(RoiP), idxP);
  indZ = unique(i3);
  RoiZ = RoiP;
  for kk = 1:glcm_window
    RoiZ = imdilate(RoiZ,DiMask);
  end
  DataP = padarray(DataP,[0,0,glcm_window]);
  DataP = double(DataP);

  %% extract features
  feature_matrix=[];
  Glrlm = [];
  for k = 1:length(indZ)
    [indx,indy] = find((RoiP(:,:,indZ(k)))>0);
    %GLCM
    [mean_im,std_im,entropy_im,gf]= calculate_main_features((DataP(:,:,indZ(k))));

glcm_struct=calculate_glcm_Run_IdxTest((DataP(:,:,indZ(k))),glcm_window,OffSet,indx,indy);

[featur_vector,label_mat]=features_vector_Run_IdxTest(ROInames(1),mean_im,std_im,entropy
_im,glcm_struct,indx,indy);
    feature_matrix= cat(1, feature_matrix, featur_vector);
    %GLRLM
    [RLFeatures0,RLFeatures45,RLFeatures90,RLFeatures135] = deal([]);
```

```
    [rlm0,rlm45,rlm90,rlm135,max0,max45,max90,max135] =
calculate_glrlm_Run_IdxTest(DataP(:,:,indZ(k)),glcm_window,indx,indy);
    for jj = 1:size(rlm0,1)
      [RLFeatures0(jj,:)] = calculate_glrlm_features(rlm0{jj,1},max0(:,jj),glcm_window);
    end
    for jj = 1:size(rlm45,1)
      [RLFeatures45(jj,:)] = calculate_glrlm_features(rlm45{jj,1},max45(:,jj),glcm_window);
    end
    for jj = 1:size(rlm90,1)
      [RLFeatures90(jj,:)] = calculate_glrlm_features(rlm90{jj,1},max90(:,jj),glcm_window);
    end
    for jj = 1:size(rlm135,1)
      [RLFeatures135(jj,:)] = calculate_glrlm_features(rlm135{jj,1},max135(:,jj),glcm_window);
    end
    Glrlm = cat(1,Glrlm,[RLFeatures0, RLFeatures45, RLFeatures90, RLFeatures135]);

  end
  feature_matrix= cat(1, feature_matrix);
  Features = cat(2,feature_matrix(:,1:23), Glrlm);

  %% Spatial features
  fea = MoCalc(DataP,idxP,glcm_window);
  Features = cat(2,Features, fea);

  %% Calculate the Gradient Magnitude and the FA values
  Grad = Gmag(idxP);
  FA3 = FA(idxP);
  Gxx = Gx(idxP);
  Gyy = Gy(idxP);
  Gxy1C = Gxy1(idxP);
  Gxy2C = Gxy2(idxP);
  Gxy3C = Gxy3(idxP);
  Gxy4C = Gxy4(idxP);
  Gxy5C = Gxy5(idxP);
  Features = cat(2,Features,Grad,Gxx,Gyy,Gxy1C,Gxy2C,Gxy3C,Gxy4C,Gxy5C,FA3);


  %% COM
  [xx,yy,zz] = Gen3DCoords(RoiP);
  difCOM = zeros(size(xx,1),3);
  for k = 1:size(xx,1)
    difCOM(k,1) = (xyzNm(1)-xx(k,1))/scaleFac;
    difCOM(k,2) = (xyzNm(2)-yy(k,1))/scaleFac;
    difCOM(k,3) = (xyzNm(3)-zz(k,1))/scaleFac;
  end
  Features = cat(2,Features, difCOM);


  %% Other organ probability
  PancC = Panc(idxP);
  TumC = Tum(idxP);
  OPname = sprintf('%s/%s_OtherProb.mhd',inDir,dName{i});
```

```
    [oProb, ~] = mhdImport(OPname);
    oProb = padarray(oProb,[glcm_window,glcm_window,glcm_window]);
    opGath = oProb(idxP);
    Features = cat(2,Features, PancC, TumC, opGath);


    %% Save features
    saveName = dName{i};
    fname = sprintf('%s/%s_BoundingArea_Features.mat',outDir,saveName);
    parsave(fname,Features);
    toc

    %% Other featrues
    Oname = sprintf('%s/%s_OtherRoi.rmha',inDir,dName{i});
    [other, oInfo] = mhdImport(Oname);
    other = padarray(other,[glcm_window,glcm_window,glcm_window]);
    idxO = find(other == 1);
    q = ismember(idxP,idxO);
    idxOt = find(q == 1);
    OtherFeatures = Features(idxOt,:);
    fname = sprintf('%s/%s_Other_Features.mat',outDir,saveName);
    parsave(fname,OtherFeatures);
end

%% Build Model
Ogat = [];
for l = 1:size(duName,2)
    o = sprintf('%s/*%s*Other*.mat',inDir,duName{l});
    O = dir(o);
    for i = 1:size(O,1)
        lOname = sprintf('%s/%s',inDir,O(i).name);
        OFe = load(lOname);
        Ogat = [Ogat;OFe.x];
    end
end

Pgat = [];
for l = 1:size(duName,2)
    polyN = sprintf('%s/*%s*Pancreas*.mat',inDir,duName{l});
    P = dir(polyN);
    for i = 1:size(P,1)
        lPname = sprintf('%s/%s',inDir,P(i).name);
        PFe = load(lPname);
        Pgat = [Pgat;PFe.x];
    end
end

Tgat = [];
for l = 1:size(duName,2)
    t = sprintf('%s/*%s*Tumour*.mat',inDir,duName{l});
    T = dir(t);
    for i = 1:size(T,1)
        lTname = sprintf('%s/%s',inDir,T(i).name);
```

```
      TFe = load(lTname);
      Tgat = [Tgat;TFe.x];
   end
end

%% Choose random pixels
rng(1);
M = min(size(Tgat,1),size(Pgat,1));
PgatR = datasample(Pgat,M,'Replace',false);
OgatR = datasample(Ogat,M,'Replace',false);
TgatR = datasample(Tgat,M,'Replace',false);

fullData = cat(1,PgatR,TgatR,OgatR);
cList = [repmat({'Pancreas'},M,1);repmat({'Tumour'},M,1);repmat({'Other'},M,1);];

names = {'Autocorrelation', 'Contrast', 'CorrelationMatlab', 'Correlation',...
   'ClusterProminence', 'ClusterShade', 'Dissimilarity', 'Energy',...
   'Entropy', 'HomogeneityMatlab', 'Homogeneity', 'MaximumProbability',...
   'SumofSquares', 'SumAverage', 'SumVariance', 'SumEntropy',...
   'DifferenceVariance', 'DifferenceEntropy',...
   'InformationMeasureofCorrelation1', 'InformationMeasureofCorrelation2',...
   'InverseDifference', 'InverseDifferenceNormalised',...
   'InverseDifferenceMomentNormalised', 'SRE0', 'LRE0', 'RLN0', 'RP0',...
   'GLN0', 'LGRE0', 'HGRE0', 'SRE45', 'LRE45', 'RLN45', 'RP45', 'GLN45',...
   'LGRE45', 'HGRE45', 'SRE90', 'LRE90', 'RLN90', 'RP90', 'GLN90',...
   'LGRE90', 'HGRE90', 'SRE135', 'LRE135', 'RLN135', 'RP135','GLN135',...
   'LGRE135', 'HGRE135', 'Mean', 'Mode', 'Median', 'Variance',...
   'StandardDeviation', 'Kurtosis', 'Skewness', 'Intensity', 'Gradient',...
   'GradientX', 'GradientY', 'GradientXY0', 'GradientXY1', 'GradientXY2',...
   'GradientXY3', 'GradientXY4', 'FractionalAnisotropy', 'xDist',...
   'yDist', 'zDist', 'PancCloud', 'TumCloud','OtherOrganProb'};

Mdl = TreeBagger(50,fullData,cList,'OOBPrediction','On','OOBPredictorImportance',...

'On','Method','classification','PredictorNames',names,'ClassNames',{'Pancreas','Tumour','Other'}
);
save('Code/TreeBagger.mat','Mdl')
toc
```