Efficient 3D Medical Image Segmentation.

MR BENJAMIN JOE FLETCHER

Master of Philosophy

ASTON UNIVERSITY

May 2014

Aston University

Efficient 3D Medical Image Segmentation

Mr Benjamin Joe Fletcher, Master of Philosophy, 2014

Synopsis

3D Medical imaging techniques have become extremely important tools in patient diagnosis. However, they produce large amounts of data that is difficult to interpret, and can currently only be analysed by highly trained people. Datasets are large – the female Visible Human dataset is around 40 Gb in size. Processing any dataset of this size will obviously be computationally demanding.

Currently segmentation of images is a predominantly manual process. Tools that are available allow segmentation to be done on a slice-by-slice basis, often using a flood-fill or region growing approach based on colour or texture space.

This report outlines research into an automated texture based segmentation technique. The research compared the effectiveness of using simple and energy efficient DCT (Discrete Cosine Transform) and Haar transforms (in both 2D and 3D forms) as a description of texture at each location within an image. This description was initially used as a vector in feature space, allowing segmentation to be carried out using a Gaussian Mixture Model and some post processing techniques. The transforms were then extended to make them independent of variations in intensity, a common issue in medical imaging. However, although now robust to intensity variations, the results were not of sufficient quality to be useful in a real application.

To improve the quality of results, a model based approach based on an AAM (Active Appearance Model) was considered. A traditional AAM uses an intensity based appearance model, which while less computationally demanding than a more complex texture based appearance model, can give poor results when subjected to intensity variations. When complex texture descriptions are used to create the appearance model results are much improved, but this is at the expense of run time, which can make the techniques less practical.

A novel combination of mDCT (modified DCT, which is intensity invariant) and an AAM was implemented and tested. When presented with 3D volumes which had been subjected to intensity variations this was seen to generate much better results than a traditional AAM, while maintaining a practical run time.

Using this approach the time taken to carry out segmentations was less than 10 minutes (when run in Matlab on a typical datacentre based Linux machine). This showed the process to be practical in terms of quality of results, run time and energy efficiency.

Keywords: Active Appearance Model, Discrete Cosine Transform, Haar Transform, Intensity Invariant

Dedication

This thesis is dedicated to Lara, who I had not met when I started working on it, and yet only through our meeting was I able to complete it.

And to Oscar, who makes sure every day is different to the last.

Also to my family who have provided endless encouragement and support.

To Max, Henry, Joe and Charlie, just because.

And lastly to Meg, who kept me company while I finished writing up.

Acknowledgements

I would like to acknowledge my supervisor I.T.Nabney for his guidance and support.

I would also like to acknowledge the Visible Human Project for providing access to their datasets.

Much of the experimental work was based on code made available by D.Kroon [20].

Table of Contents

List of Tables

List of Figures

1.0 Introduction

Modern medical diagnosis often involves the use of some type of medical imaging technique. It is invaluable to be able to look at structures inside the human body without surgery. Typically the data that is gathered is made up from a number of images of transverse slices through the patient. These slices can then be put together in order to give an accurate 3D model of the patient. However, as the technology improves and the images become higher resolution, more and more data is generated. Interpreting the images is difficult and can only be done by highly trained (and therefore expensive) medical professionals.

Figure 1: Images from the Visible Human Project.

Figure 1 shows a slice from the visible human dataset on the left, and the same slice once the individual organs and structures have been highlighted. This is a complex and technically difficult task to perform.

A crucial feature of medical imaging is the ability to distinguish different structures within the 3D data. This segmentation (delineation of anatomical structures) can be a useful technique in the early diagnosis and/or treatment of some conditions, or the repair of those structures being identified.

For example the volume of the hippocampus can act as an early indicator for altzehimers dieses [27]. If the patient's brain has been imaged, then the hippocampus can be identified and delineated, and the volume easily found. However, the hippocampus is a complex structure meaning that this

segmentation process is not trivial, and is commonly a manual and time consuming process.

As another example, consider the use of rapid prototyping techniques such as stereo lithography in surgery [28]. These techniques (similar to those used in emerging 3D printers) allow for 3D structures to be created from abstract computer models. The shape of an anatomical structure can be extracted from a medical image and then an object can be manufactured based on this shape. This technique can allow surgeons to create custom parts (e.g. hip replacements) that closely resemble the original structure being replaced.

As a final example, it is useful if a tumour can be viewed separately or in the context of the surrounding tissues. This can currently be done by using multiple imaging techniques such as PET and CT. The CT scan intensity measures the radiodensity of tissue, which means that there is usually little contrast between the tumour and the surrounding tissue. The PET scan has a higher intensity in the tumour (as tumours are more likely to readily absorb the short lived isotopes that are used in PET scans). This then allows the two datasets to be combined, giving the required views of the tumour. However, for some parts of the body, it is impossible or unsafe to use a second modality and so image processing on relatively undifferentiated scans becomes a necessity. This requires the identification of clinically relevant structures from a single 3D image. If the image is made up of slices then the contours of the structure need to be identified on each slice.

Being able to perform good quality segmentation of medical images is therefore of obvious importance. However, as already stated, medical images can be difficult and expensive to interpret (often requiring expert knowledge). In an effort to reduce the cost (both financially and in terms of time) much research has been carried out in the area of automated segmentation of medical images. These techniques are often hampered by being computationally demanding (sometimes impractically so), or by imaging artefacts such as inter- and intra-image intensity variation.

This report outlines research that has been carried out into an automated segmentation technique, based on texture analysis of the data (both in 2D and 3D). The overall aim of the research is to find a solution that, while providing high enough quality results, is as affordable as possible, and in all cases practical enough to be used in real life.

This text investigates the effectiveness of using block based transforms (such as DCT or Haar wavelet transforms) to measure the texture of medical images, and using this measure along with a GMM (and later an AAM) to perform segmentations.

The benefit (and therefore attractiveness) of using such a block based transform includes the highly parallelisable nature of the required computation. This could allow for an efficient implementation to be created based on a parallel computing environment such as C++ AMP or open CL, in conjunction with running on a GPU based platform. [29]

In this phase of the research, it was found that texture alone did not provide sufficient measure of the image to perform good quality segmentations. However, it was found that when the DCT (or Haar) transform was slightly modified and then used as a measure of texture, a good level of resilience was achieved against variations in intensity. Therefore this novel aspect of the technique was carried forward and was combined with an approach based on a deformable model (specifically the use of an AAM).

Initially the use of a standard AAM was shown to provide good quality segmentation, although as the appearance model used is based on intensity alone, it was unsurprisingly found not to be tolerant to variations in image intensity. The AAM was adapted to use an appearance model based on the modified DCT previously developed. This again provided a level of resilience against variations in image intensity.

Finally the research was concluded with an investigation into the most effective variant of the developed technique. The use of a 2D and a 3D AAM was considered, along with the effect of using appearance models based on a 2D

modified DCT against a 3D modified DCT. The effect on the quality of the segmentation as well as the effect on the computational complexity (considered to be proportional to the process run time) was considered.

A standard 2D and a standard 3D AMM (both using an intensity based appearance model) was used to provide a reference point for comparison.

2.0. Background

2.1 Image segmentation

Before going further it is useful to describe the segmentation problem, and how to measure success in any offered solution.

Image segmentation is the process by which an original image is partitioned into some homogeneous regions. More informally it is the process of splitting an image up into its component regions, each region containing pixels or voxels with something in common (such as being part of a specific object within the image).

For example, in the case of segmenting a photograph, this could mean identifying the objects within the photograph and splitting the image up accordingly. In the case of a medical image, this could mean identifying anatomical structures.

More formally, the segmentation problem is described by equation 2.1.

$$I = \left(\sum_{n=1}^{N} O_n \right) + O_b \tag{2.1}$$

Where: I represents the complete image

O_n represents the set of pixels or voxels that make up one of the N objects of interest within I.

O_b represents the set of background pixels or voxels (i.e. the set of those pixels or voxels that are not part of any of the N objects of interest found in I).

The segmentation problem can therefore be stated as the problem of identifying one or more objects, O_n, in a given image, I. In the case of this research the images being searched are 2D and 3D medical images, and the aim is to identify only a single object within each image, a specific anatomical structure

such as a bone, or an eye. As such, the segmentation problem can be restated in a simplified form, as shown in equation 2.2.

$$I = O_0 + O_b \tag{2.2}$$

Where:*I* represents the complete image

O_0 represents the set of pixels or voxels that make up the specific object being searched for.

O_b represents the set of background pixels or voxels (i.e. the pixels or voxels that are not part of O_0).

In this case, O_0 can be termed the segmentation set. O_b is the set of pixels excluded from the segmentation set.

Typically, the method used to evaluate the effectiveness of automated segmentation techniques, is to compare the techniques results against results from a different but well understood competing technique, or against manually performed segmentations. In this research comparison against manually performed segmentations was used to evaluate the success or failure of the techniques being developed and tested. The specific metrics that were used are described in section 3.3.

2.2 Overview of 3D Medical Imaging Techniques

In the following sections various medical imaging techniques are described. Often these involve imaging the patient while they are lying down. In this case they will be described with reference to a co-ordinate system with the Z-axis running through the patient's body (from head to toe), the Y-axis being vertical, and the x-axis being horizontal. This is in-line with the co-ordinate system used by many published works in the field of medical imaging, especially works on slice based processes. It is usual for slices to be created in the X-Y plane, at regular locations along the Z-axis.

2.2.1 Computed Tomography (CT) Scanning [17]

2.2.1.1 The Method

Tomography is the name given to imaging by sections. Generally it is a technique for constructing an image based on a series of views or projections.

Projection Radiography (or more commonly known as X-Rays) is one of the most commonly used medical imaging procedures. This process uses X-Rays to view objects that would be hard to image otherwise. X-Rays are commonly used to view bone structures within patients. An X-Ray source is used to project an image through the patient. The image is captured and can then be read to help with patient diagnosis. The image differentiates parts of the patient's body based on their ability to absorb the X-Rays. The strength of the X-Rays being used is varied depending on the part of the body being imaged, and the structures of interest.

Tomography is a reasonably well known technique, and was in use before computing became readily available (as early as the 1930s, and in regular use by the 1950s [30]). It helped to solve the problem of superposition in Projection Radiography. This early form of tomography was achieved by moving the X-Ray source and the film relative to the patient in order to get a sharper image along the focal plane. This allows a series of X-Rays to be taken, each focusing on a different plane within the body. Structures on the focal plane appear as sharp images on the film, whereas objects on planes a distance away from the focal plane become blurred. Once a series of images have been taken it is possible to gain an insight into the relative position of the different structures within the body, which would not have been possible if a single projection had been used.

Axial CT scanners are typically made up of a single X-Ray source and a number of X-Ray receivers. The patient lies within the scanner, the z-axis running from head to toe through the scanner (see figure 2). The source and receiver may be mounted on a ring or gantry around the patient. The gantry can be rotated.

To generate an image of a slice through the patient in the x-y plane, X-Rays are fired through the patient and the data gathered by the receivers is recorded.

Then this is repeated with the gantry having been rotated by an amount. Once the gantry has been rotated through 180 degrees, enough data has been generated to allow one slice of the patient in the x-y plane to be reconstructed.

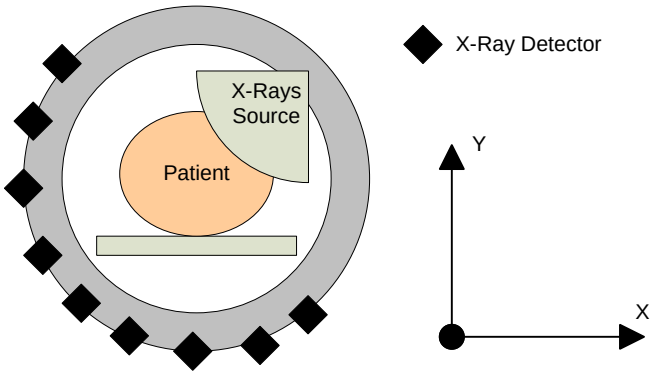


Figure 2: Diagram of a simple CT scanner.

If more slices are required then this process is repeated with the patient moved some distance along the z-axis. If a fine resolution along the z-axis is required then the table is moved a smaller distance, although this will increase the patients radiation dose, and this may be a reason for selecting a lower z-axis resolution.

Contrast agents are sometimes administered to patients. A contrast agent is able to highlight fluids, structures or vessels within the patient's body. Virtually any hollow structure within the body can be imaged using contrast agents. The agent can be positive or negative (i.e. shows up more or less opaque than the body tissues). Once it has been administered to the patient, it will show up very clearly on the image.

Iodine based contrast media are commonly used in radiology. They typically have relatively harmless interactions with the body and they are primarily used to visualise vessels

2.2.1.2 Problems with CT scanners

Patients who have a CT scan are exposed to X-Ray radiation. In some techniques the patient may be imaged a number of times. For example, the

heart can be a difficult organ to image as it is constantly beating. It takes time for the X-Ray source to pass through a 180 degree rotation around the patient, and the heart will have moved during this time. A better quality image can be achieved by imaging the heart a number of times, and cross-referencing the image data with a simultaneously performed ECG. The ECG allows data that was recorded while the heart is at a specific point in its cycle to be retained. Images taken when the heart is at other points in its cycle can be discarded. The remaining data can be combined to generate a coherent image of the heart, at rest (in diastole). The disadvantage of this technique is that a patient can receive a radiation dose equivalent to between 100 and 600 chest X-Rays during this procedure.

Also, as mentioned above, one of the limitations of CT scans is the inability to capture dynamic processes that are faster than one rotation time. The patient may move during the scan and this may be unavoidable (for example, breathing). This can lead to distortion of the images.

Another problem related to the scan speed is with the use of contrast agents. In order to get the best results out of using a contrast agent the scan should be done at an optimal moment a specific time after the agent was administered. However, this can be difficult as the scan takes a relatively long period of time, and so only a small portion of the scan receives the full benefit of the contrast agent. Another problem with contrast agents is that some patients can have a severe and life threatening reaction to the agents used.

A CT scanner can cost in the region of 1 million USD, and so a full body CT scan can be an expensive procedure. It is also likely to find other incidental problems that may then need to be investigated. This is obviously beneficial to the patient, although can present significant challenges to the medical staff to ensure the correct follow up treatment is given for all the incidental problems that have been identified. [17]

2.2.1.3 More Advanced types of CT Scanner

Various improvements have been made on the basic Axial CT scanner described above.

One such improvement is the Helical or Spiral CT Scanner. Here the table is continually moving through the scanner, and the gantry is continually rotating. This helps to reduce the scanning time, and it may be possible for the patient hold their breath for the length of the scan, to improve the quality of the scan.

Electron Beam CT is another attempt to reduce the scan time. Here the machine consists of a large vacuum tube containing an electron beam which is electro-magnetically steered towards a number of X-Ray anodes arranged around the patient. The result of this is that X-Rays can be generated from different locations around the patient very quickly (as the whole X-Ray source does not have to be physically moved around the patient). Using this technique a single slice can be obtained in the region of 50ms to 100ms.

Multi-slice CT scanners have more than one detector ring. It is possible to have 64 detector rings in one scanner. This can allow higher resolution images to be captured faster. However, the need to restrict the radiation exposure of the patient and image noise can both limit the resolution achieved.

2.2.1.4 Image Artefacts

CT images can be the subject of a number of artefacts.

Aliasing artefacts may be found on CT images [31]. Typically when sampling data, aliasing can be avoided as long as the Nyquist criteria is observed (i.e. the data being sampled has a maximum frequency no greater than half the sampling frequency). However, in the case of CT images the Nyquist criteria does not guarantee aliasing artefacts will not be observed.

This is because (unlike in discrete systems where a sample can be taken at a specific point) in the case of CT data the sample is being taken over volume (i.e. a voxel). Therefore the value sampled is some function (such as the mean) of all values present within the volume of the voxel itself.

Figure 3 shows an example of how the spacing of measurements can lead to aliasing. In the example the data being sampled is shown in blue. The samples

taken are shown in green. The data pattern being observed consists of a bar or grill pattern with a period of 2 samples. The data pattern is sampled using a sampling window (with a width equal to 1 sample), along with a simple averaging function.

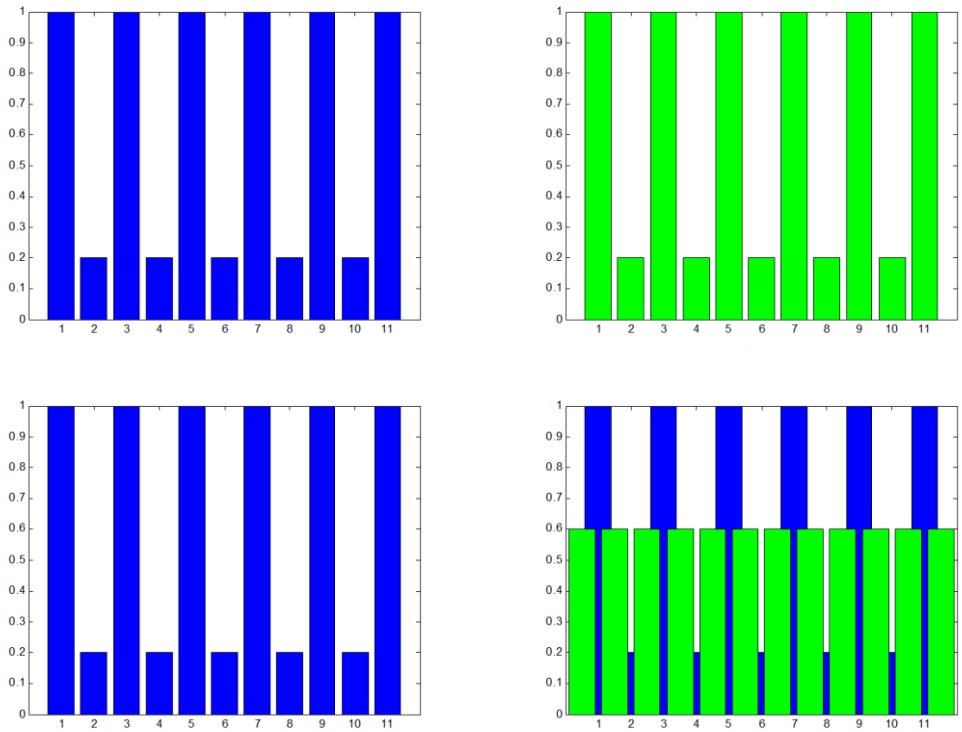


Figure 3. An example of aliasing occurring even though the Nyquist criteria has been observed. Input data (top left) sampled by aligned sampling windows (top right). Input data (bottom left) sampled by non-aligned sampling windows (bottom right). Data being sampled is shown in blue. Result of sampling is shown in green.

Even though the Nyquist criteria has been satisfied, it can be seen that it is possible (when the sampling windows are not aligned with the original data) for the grill pattern to be lost.

The issue of aliasing, as described above, is related to that of the partial volume effect, which is a common effect in medical imaging, and is shown up as a blurring of sharp edges. It is a limitation due to the resolution of the scan. Each voxel represents a physical volume within the patient. The voxel has been evaluated based on some property (in this case X-Ray absorption). One voxel is represented as a single scalar value representing this property, but in reality the

physical space may contain two or more types of tissue. If this is the case then the voxel will be represented as an averaged absorption over all these tissues. This effect can be reduced by increasing the 2D resolution, or reducing the inter-slice distance. [4]

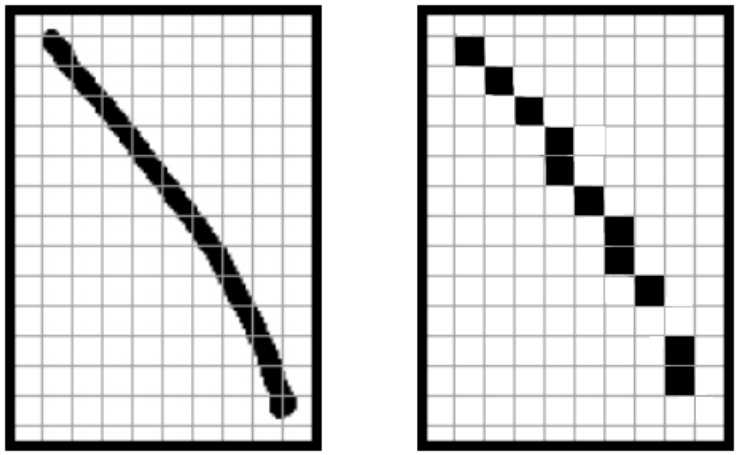


Figure 4: Illustration of the Partial Volume Effect. On the left, the real object (with a grid representing the sampling resolution). On the right, the resulting sampled image.

The images in figure 4 attempt to explain the Partial Volume Effect graphically. The left image shows the object we are trying to image (in this case a solid black line), and a grid representing the resolution we are using to sample the image.

A square on the grid represents one pixel, and each pixel can only be given a single colour as we generate the sampled the image (in this case we will choose either black or white as the colour for the pixel). However, the squares on the grid fall into one of three categories, not just two. They can be empty, full, or neither empty nor full.

A full square is one shown as completely black. There are no pixels that fall into this category. If there were then these would map to black pixels on the right hand image.

An empty square on the grid is completely white. There are many empty squares and they map to white pixels in the sampled image.

The remaining squares contain some black and some white, where the solid black line partially passes through them. As we sample the image we need to decide what colour the resulting pixel for each of these squares should be. To generate the sampled image, the pixels are coloured black if the original square was mostly black, and white otherwise. It is this sampling technique that has caused the reduction in quality, and the break in the line within the sampled image.

If, as in this case, a break in a structure is seen as a result of the partial volume effect, then this can impact the effectiveness of post processing techniques that may be used on the image. For example any region growing techniques are very likely to be adversely affected by the Partial Volume Effect.

The example given above is a simplistic 2-dimensional situation, given for illustrative purposes. It is obvious that using a smaller sampling frequency would help reduce the partial volume effect, and this is also true for 3D CT images; voxels that have a larger volume are more likely to contain two or more tissue types.

Reducing the slice thickness used in a CT scan can therefore reduce the partial volume effect. However, it is well documented that increasing the slice thickness helps to reduce image noise [32]. This is because the increased volume of the voxels has a low pass filtering effect on the image. A trade-off exists between image quality and slice thickness, but it is a complex one; if the slice is too thick then the partial volume effect will be worse, but reducing the thickness of the slice can introduce noise due to the smaller volume of the voxel.

The issue is not peculiar to CT images. For example, the partial volume effect has been seen to introduce significant errors into the measurement of the volume of structures in the brain, measured from MRI images [33]. Various approaches have been considered in counteracting the partial volume effect. Some approaches try and correct for the issue [34] while others [33] attempt to provide an estimation of the uncertainty introduced by the effect.

Motion artefacts can also be seen when the patient moved during the scan. The image may become blurred or streaky. [6]

2.2.2 Positron emission tomography

2.2.2.1 The Method

A PET scan is a medical imaging technique that provides a 2D or 3D map of functional activity within the body.

A patient who is undergoing a PET scan is administered a short-lived radioactive isotope tracer. This tracer is a metabolically active molecule that has had the isotope incorporated into it. The result from the scan will be an image showing where the metabolically active molecules are concentrated. Therefore the image will also show the levels of corresponding metabolic function. For example, PET scans are used to detect brain activity in different regions of the brain.

The patient must undergo a waiting period to allow the isotope time to reach the part of the body that is of interest. When the waiting period is over the patient is placed inside the scanner. The scanner has a similar physical structure to the CT scanner, with the patient lying on a table which passes through the centre of a ring of sensors (see figure 5). The sensors on the PET scanner do not rotate, but are fixed in place.

As the isotope starts to decay, it will give out positrons. These positrons may travel up to a few millimetres before colliding with an electron. This collision will, in turn, result in two photons travelling in opposite directions away from the site of the collision. The sensors around the patient are trying to detect this pair of photons travelling in opposite directions.

Therefore two sensors situated directly opposite each other are required to detect the isotope decaying. When one of the sensors detects a photon, it is only considered to be evidence of an isotope decaying if the other sensor also detected a photon at the same time. If this is not the case then the single photon that was detected is ignored.

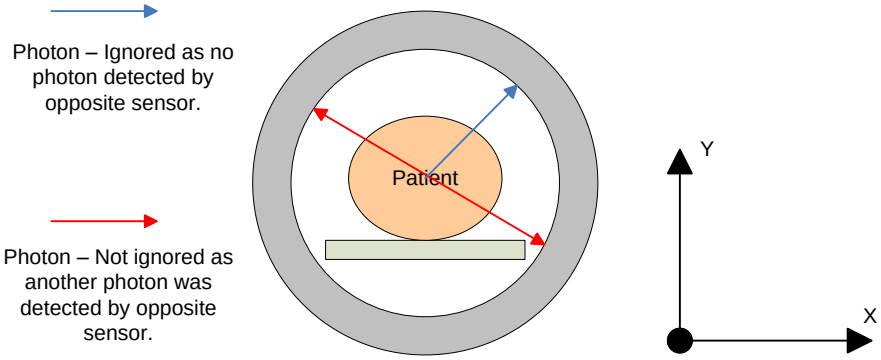


Figure 5: Photon pairs as observed by a PET scanner.

A process similar to that used in CT is then used to reconstruct the image. However, this process is not as reliable for PET data as it is in the case of CT data, due to the specific physics related limitations in the scanning process, specifically positron range and photon non-collinearity. [35].

There are other limitations (such as those related to the physical size of the detector, but here we will focus on the physics related limitations, rather than the technological limitations, which may be improved over time, as new technologies become available).

In a CT scan, X-rays are used to differentiate the ability of tissue (at different locations within the patient) to absorb X-rays. However, in a PET scan the location of the isotope being observed is measured indirectly. The observation being made is actually of the location of the annihilation of a positron. As previously stated the positron range (i.e. the distance a positron may travel before reaching the thermal energies required in order to be annihilated) can be in the order of a few millimetres (although this figure varies based on the isotopes being used), and therefore the location recorded can be a few millimetres from the true location of interest. This leads to a limitation of the spatial resolution that can be readily achieved using PET scanners. It is, however, noted that once the data has been captured, statistical techniques [36] (beyond the scope of this text) can be applied to take account of positron range, and therefore improve the resolution of the resulting images.

Photon non-collinearity is caused by the momentum of the emitted positron. This small, but finite, momentum causes some small variation in the trajectory of the photons emitted at the point of annihilation. This means the photons are not traveling in exactly opposite directions to each other, and this adds further uncertainty as to the true location of the decaying isotope. This uncertainty worsens with diameter of the detector ring. For a detector ring of diameter in the range of 80cm to 90cm, the uncertainty added could be in the region of 2mm. Photon non-collinearity leads to a further reduction in the spatial resolution that can readily be obtained from the captured data.

Another issue revolves around the relative size of a PET data set. Typically a PET data set may have millions of samples, whereas a CT data set could have billions. The reduced size of the dataset means that noise has a larger effect on the data.

If the scanner has a single ring of sensors, the image will be a single slice (in the x-y plane) of the patient. Some scanners have a number of rings forming a tube. This can be used to capture a 3D image, although the image is harder to reconstruct and this process calls for more computational power.

Some scanners are able to perform both CT and PET scans. This is useful because it means both scans can be done at the same time, and the data can be considered together (the anatomical images from the CT and the metabolic functional images from the PET).

The total dose of radiation a patient will receive from a PET scan is quite low at around 350 times that of a single chest X-Ray.

A PET scanner can cost several million USD.

2.2.3 MRI

2.2.3.1 The Method

An MRI scanner has a similar physical form to CT and PET scanners (see figure 6). The patient lies on a table in the centre of the scanner. The scanner will generate a set of image slices through the patient in the x-y plane.

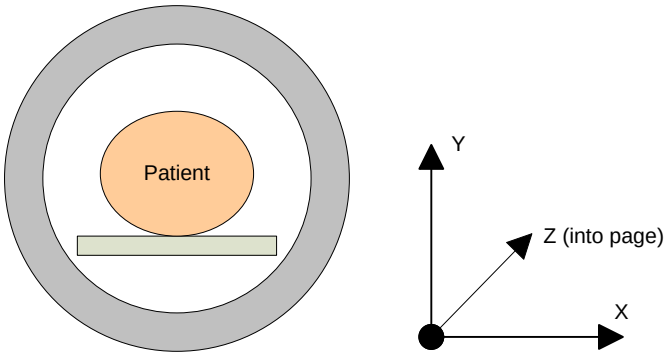


Figure 6: Physical layout of an MRI scanner.

While the detailed description of how an MRI scanner works is beyond the scope of this text, a brief outline is provided below.

When the scanner is activated a strong magnetic field is generated along the z-axis (running along a line from the patients head towards their toes). Radio frequency (RF) electro-magnetic pulses are then generated in a direction orthogonal to the magnetic field.

Tissue within the field will become slightly magnetic. By generating an EM pulse it is possible to make the tissue's magnetic field change, and this change is large enough to generate a current in a suitably orientated coil (the receiver coil) located outside of the patient. This is how the tissue is observed by the scanner.

Benefits of MRI scanners over other scanning methodologies include:

1) not using ionizing radiation.
2) contrast agents (when used) only have a low incidence of side effects.
3) Unlike other scanning techniques slices can be imaged along any plane.

2.2.3.2 Problems with MRI scanners

MRI scanners provide one of the best ways of looking inside a patient without requiring surgery. However, they do present some problems.

The scanning process itself can be an uncomfortable experience. The patient is required to lie within the scanner for a long time (typically 20 to 90 minutes). Slight movements of the patient's body can result in distorted images, and may mean that the scan has to be repeated. Some patients may have problems with being inside the scanner (claustrophobia or they may just not fit). The scanners also make a large amount of noise when they are in operation.

Patients that have pacemakers cannot be scanned as the magnetic field will interfere with the pacemaker's operation.

The scanners themselves are extremely expensive (in the order of several million USD), and this has the knock-on effect of making the scans expensive to perform.

More importantly (in the context of this text) the images that are generated by an MRI scanner are also not without issue. While they are also affected by general imaging artefacts (such as the partial volume effect), but one of the biggest issues is that of intensity inhomogeneity. The variability of brightness for similar tissue at different locations within an image can make an MRI scan difficult to interpret, as well as being extremely problematic to segmentation techniques based on absolute pixel intensity.

This intensity variation can be quite significant. For example, when considering brain tissue, white matter and grey matter have distinct signal intensities, but due to the magnitude of the inhomogeneity the absolute pixel intensities generated by these two types of tissues can overlap [37].

The variation is not consistent between MRI scanning equipment, or even between images generated on the same equipment. Inter-image as well as intra-image inhomogeneity exists. It can be affected by the age of the patient, as well as the region of the body being imaged (information taken from private

communication with Dr. Woods from Birmingham University). It can also be affected by the operating conditions and status of the scanning equipment [25].

The intensity inhomogeneity is not trivial to remove from previously created images, or is it easy to accurately model (although work has been carried out in this area [24]). However, a simple model that is sometimes used during the testing of MRI segmentation methods is that of simply adjusting the contrast of the image over the volume of the image. A more detailed discussion of this (and related topics) is held in section 5.

2.2.4 Cryosection

This is a frozen-section laboratory procedure. The procedure is to freeze the tissue sample rapidly to about -20 degrees Celsius. Then a microtome is used to slice the tissue into very thin slices, which are then placed on glass slide and stained. It is reasonably quick to prepare such a slide (in the order of 10 minutes) but the quality of samples is lower than for traditional histology.

It is obviously not appropriate for diagnostic purposes as the tissues are physically cut into slices as part of the process. However, a brief introduction is included here as the majority of the work presented in this text is based on the high resolution cryosection dataset provided by the visible human project.

Section 2.4 gives a detailed review of how this dataset was generated.

2.3 Image Processing and Segmentation Techniques

A review is carried out in this section covering some of the approaches, techniques and datasets that was used throughout the research.

2.3.1 Thresholding approaches [6]

In a thresholding approach the image is segmented based on intensity. Pixels are grouped into one of two classes; those which have intensity higher than the threshold value, and those which do not. By finding the correct threshold this can be an effective method of segmenting an image. It is also possible to use a multi-thresholding technique, to allow the image to be split into more than two regions.

This process is fast enough to be run in real-time and so the threshold values can be adjusted interactively by the operator. However, a basic thresholding technique is only useful for processing data from a single source, and is susceptible to noise. It is commonly used in conjunction with other techniques. For example it has been used in conjunction with region growing to extract bronchus regions from 3D chest X-rays [11].

2.3.2 Region-growing approaches [6]

This approach allows a region of an image to be defined based on a start point (from within the region) and a set of rules for defining the borders of the region. The rules may be based on intensity (similar to a thresholding approach), or may look for edges within the image. It is a commonly used (if basic) technique in the field of image segmentation, for example consider the work done by Adams et al [21] and Hojjatoleslami et al [22].

Region growing can be affected by noise and imaging effects such as the Partial Volume Effect (see section 2.2.1.4) which can cause regions of the image that should remain separate to become connected.

Another disadvantage is that the start point has to be defined, and this is not an automatic process.

However, an advantage is the simplicity of the technique and the ease with which it can be implemented. Figure 7 shows a small section of pseudo code expressing a generalised region growing algorithm, used as the basis of the region growing processes used later in the text (see section 4.1).

```
While(region is growing)

For (each location in an image or volume)
        If (location is already part of the region) then
                For (each neighbouring location)
                        If(neighbouring location should be part of the region) then
                                Make the neighbouring location part of the region.
                        End If
                End For
        End If
End For

End While
```

Figure 7: A typical region growing algorithm (in pseudo code).

Typically the variation between different region growing approaches is based on the criteria used to decide on inclusion in the region, and also the criteria for defining a location's neighbouring locations.

2.3.3 Atlas-guided approaches [6]

Atlas guided approaches are a generalised set of techniques that attempt to map a predefined template or atlas of a body part to a new image. This approach obviously relies on an atlas being available, or it being possible to create such an atlas. A process called atlas warping is then used to transform the atlas via a series of linear or non-linear transformations until a close fit to the new image has been found. A useful outcome of using an atlas based approach is that once the warping process is complete, any labels, landmarks, or segmentations that are associated with the atlas can then be transferred to the new image. This style of approach has been used in a number of applications related to medical image processing [3].

Atlas-guided approaches are most commonly (although not exclusively) used on MR brain imaging, and pre-existing atlases for work in this area are available [38] [37].

This technique is best suited to segmentation of structures that do not show a large amount of variation across the population. An Active Appearance Model

(AAM) [18] is used in chapter 7 to build such an atlas. This model is then warped to find the best fit against new 2-dimensional and 3-dimensional images, and if a good fit is found, can be used to perform segmentation.

2.3.4 Discrete Cosine Transform (DCT)

The Discrete Cosine Transform was first proposed (along with an efficient algorithm for calculation, based on the fast Fourier transform) in 1973 by N Ahmed et al [39].

The DCT transforms allow any series of data points to be faithfully expressed as a sum of cosine functions. Each cosine function is oscillating at a different frequency, and the functions are often depicted as a set of basis images. Figure 8 shows an example of the basis images that would be required when using a DCT over a data series with 8 samples.

Figure 8: basis functions used by a DCT, being calculated over a data series of 8 samples.

As can be seen in Figure 8, each basis image is unique and contains a cosine function of a unique frequency. By combining these 8 images, with suitable weighting, any series of 8 data samples could be reconstructed.

More formally, Equation 2.3 shows how a DCT is calculated from a series of input data.

$$X_k = \sum_{n=0}^{N-1} x_n cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right] \tag{2.3}$$

Where: *N* represents the number of values in the input vector

$x = \{x_0, x_1, x_2, \ldots x_{N-1}\}$ and is the input data being transformed.

$X = \{X_0, X_1, X_2, \ldots X_{N-1}\}$ and is the output data after the transform has been carried out.

The DCT has been commonly used in lossy compression techniques (typically used to compress images, video, or audio) where the smaller, higher frequency components can be removed with little consequence to the perceived quality of the reconstructed data.

Although the DCT is defined for a single dimensional data series, it can be more generally applied to a block of N dimensional data. For each dimension, a number of single dimensional DCTs are carried out in sequence. The transformed data replaces the input data in each case, and then the process is continued for all dimensions in turn. In the case of a 2-dimensional transform a DCT is carried out on each of the rows of the input data, and then on each of the columns.

In the general case, for N dimensions, the number of transforms carried out for each dimension is the product of the size of the data block in each of the other dimensions. This is more formally stated in equation 2.4, and an example is shown in the following text, considering the calculation of a 2-dimensional DCT. This process is also outlined in equations 2.5 to 2.11.

$$T = \sum_{d=1}^{D} \left[\left[\prod_{n=1}^{D} s_n \right] / s_d \right] \tag{2.4}$$

Where: *T* is the total number of DCTs required.

D is the dimensionality of the input data.

S_d is the width of the input data with respect to dimension d.

$$d = \begin{bmatrix} d_{0,0} & d_{1,0} \\ d_{0,1} & d_{1,1} \end{bmatrix} \tag{2.5}$$

Where: *d* represents a 2-dimensional data series which we want to transform using the DCT.

$$[dx_{0,0}\ dx_{1,0}] = DCT([d_{0,0}\ d_{1,0}])$$

(2.6)

$$\left[dx_{0,1}\ dx_{1,1}\right] = DCT\left(\left[d_{0,1}\ d_{1,1}\right]\right)$$

(2.7)

$$dx = \begin{bmatrix} dx_{0,0} & dx_{1,0} \\ dx_{0,1} & dx_{1,1} \end{bmatrix}$$

(2.8)

Where:*dx* represents the result of performing two DCTs in the direction of the x-axis of the data series.

$$\left[dxy_{0,0}\ dxy_{0,1}\right] = DCT\left(\left[dx_{0,0}\ dx_{0,1}\right]\right)$$

(2.9)

$$\left[dxy_{1,0}\ dxy_{1,1}\right] = DCT\left(\left[dx_{1,0}\ dx_{1,1}\right]\right)$$

(2.10)

$$dxy = \begin{bmatrix} dxy_{0,0} & dxy_{1,0} \\ dxy_{0,1} & dxy_{1,1} \end{bmatrix}$$

(2.11)

Where:*dy* represents the result of performing a further two DCTs in the direction of the y-axis of the data series.

The process starts by performing two DCTs on the input data *d*. The DCTs are performed in the direction of the x-axis, as shown in equations 2.6 and 2.7. The output from the DCTs are then used to create an interim result (*dx*). The process is then repeated, only this time using *dx* as the input to the DCTs, and performing the DCTs in the direction of the y-axis, as shown in equations 2.9 and 2.10. Finally the result is found in *dxy,* as shown in equation 2.11.

Again, it is common to depict the different cosine functions considered by the 2-dimensional DCT as a set of basis images. A set of basis images (for use with a 2-dimensional DCT operating over data blocks of 8x8 pixels) are shown in figure 9. It can be seen that the top left basis image is the DC component (i.e. there are no horizontal or vertical frequencies present in this component at all), and the bottom right basis image contains high horizontal and vertical frequencies.

All the other components contain various combinations of horizontal and vertical frequencies based on their position within the figure.

Using suitable weights to combine these basis images it is possible to create any set of 8x8 pixels.

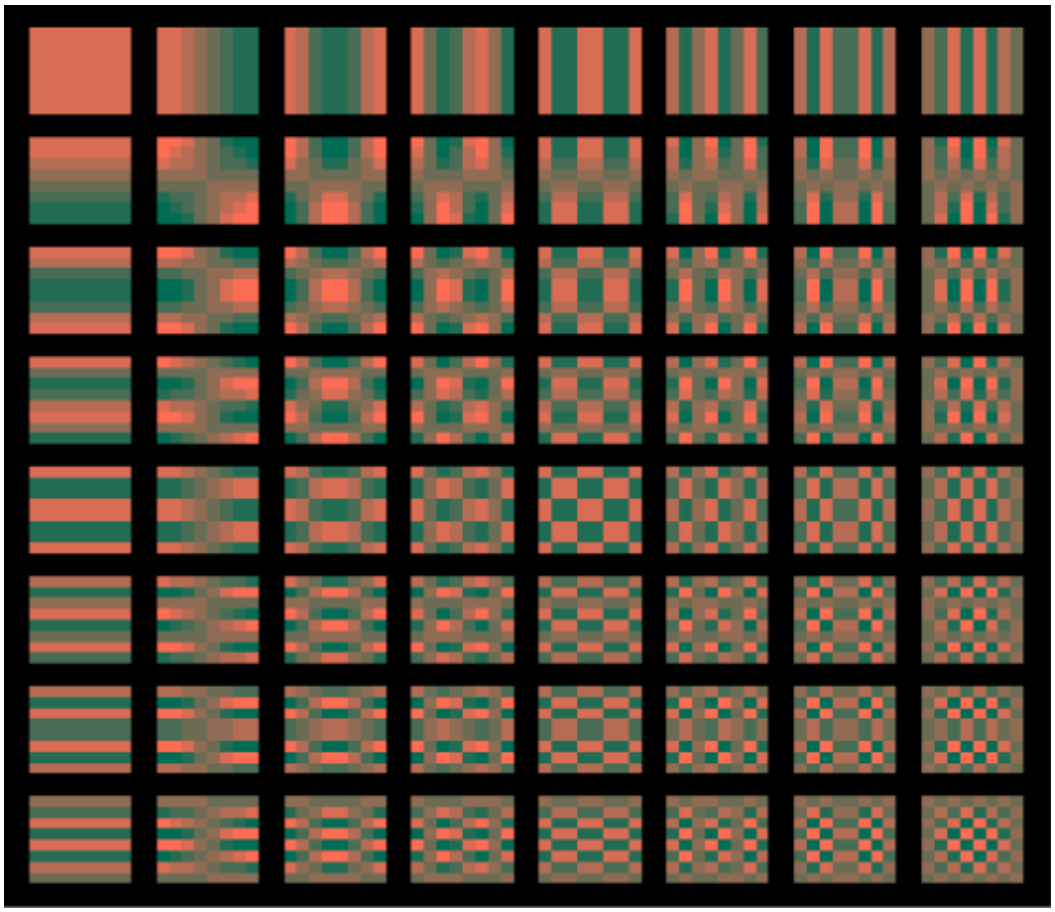


Figure 9: basis functions used by a DCT, being calculated over a 2-dimensional data series of 8x8 samples.

As mentioned above, DCTs are commonly used in lossy compression techniques applied to digital media. As such, much research has been carried out into efficient implementations of the DCT. For example Ananthashayana et. al. developed a novel, recursive, multiplierless algorithm for calculating 2-dimensional DCTs [40]. This algorithm can be implemented using only addition and shifting. Other work includes the development of high performance algorithms for calculating 2D DCTs, specifically targeted at GPU architectures [29].

2.3.5 Gaussian Mixture Models

A normal or Gaussian distribution is a continuous probability distribution. It is commonly used to model the behaviour of random variables whose distributions are not known, and (in its simplest form) is described by equation 2.12.

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.12}$$

Where: *f* is some random variable over *x*.

μ is the mean of the distribution

σ is the standard deviation of the distribution.

The Gaussian distribution takes the form of a bell curve (as can be seen from either of the two curves shown n Figure 10). If μ = 0 and σ = 1 then the distribution can be called the standard normal distribution.

Many situations can be modelled using Gaussian distributions, and once suitable values for μ and σ have been found, the distribution can be used to better understand the origin of previously unobserved data points on the same axis.

For example, consider a target shooting game at a fair ground. Players aim at, and try to hit a target, in order to win some prize. To make the example more straightforward we will consider the player can only adjust their aim in the horizontal direction, and that the aim in the vertical direction is fixed.

Consider that two players each play the game a number of times. Each time they fire at the target they hit somewhere along the horizontal axis, but at the correct height (as this is fixed in our example). They each record their results, and we end up with two sets of data, each belonging to a different player.

After playing the game a number of times they calculate the mean, variance and standard deviation of the data, and then plot a Gaussian distribution on a graph, with probability of result shown on the y-axis, and distance from centre of target on the x-axis. It is supposed that *x*=0 marks the central location on the target.

Figure 10 shows what this (fictitious) graph might look like. Some anecdotal information can be extracted from the plot;

- player one (represented by the green curve) demonstrates a more consistent set of results, but their distribution is not central. Perhaps this suggests that they were possibly the more skilful player, but that they may have done better by adjusting their sight.
- player two (represented by the red curve) seems to be the less skilful player, but has a much more centrally distributed set of results.

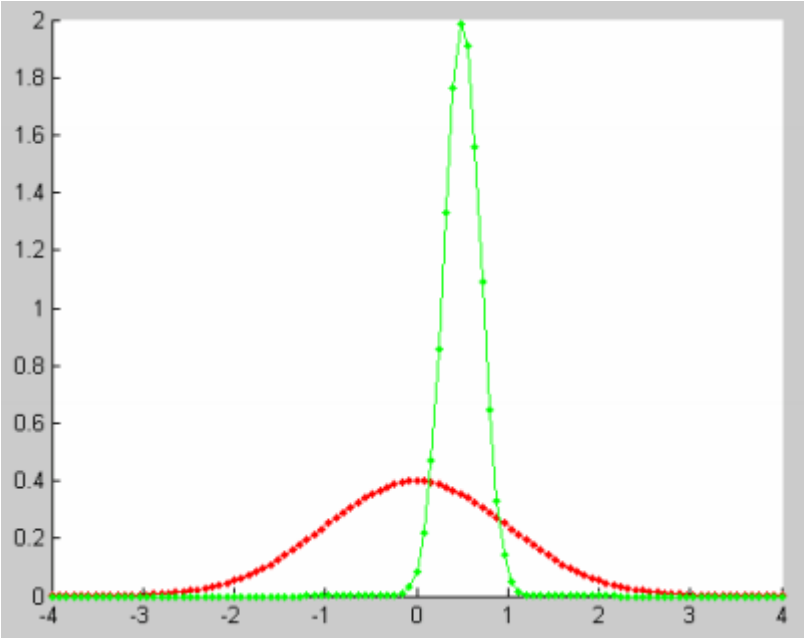


Figure 10: Example showing the shape of two Gaussian distributions, which have different values for mean and standard deviation.

Now, if one of the two players (selected at random) plays the game again, and records their new result, we can use the plot in Figure X to work out the probability that either player was the shooter.

For example, if the record result was -1, it is unlikely player one was the shooter. However, if the result was 0.5, it is likely (but not certain) that player one was the shooter.

Using a Gaussian distribution in this way might allow some simple situations to be modelled, but generally situations are not this straight forward, and might

need a more complex distribution to provide a good fit between the model and the observed data.

For example, what would happen if (instead of trying to work out which player was the shooter in a given instance) we now want to find the probability that a given location on the target would be hit, if player one and player two are the only players? A Gaussian Mixture Model can be used to investigate this more complex situation.

In a Gaussian Mixture Model, two or more distributions are combined in a linear and weighted manor). This is shown in equation 2.13.

$$p(x) = \sum_{j=1}^{K} w_j . f(x, \mu, \sigma) \tag{2.13}$$

Where *p(x)* is the probability of *x.*

K is the number of distributions being combined in the model.

W_j is the weighting applied to distribution *j*.

So in the example, a weighting might be applied to each player which is proportional to how often they play the game. If player two (represented by the red distribution in figure 10) played the game twice as often as player one, then the combined probability distribution might look something that shown in figure 11.

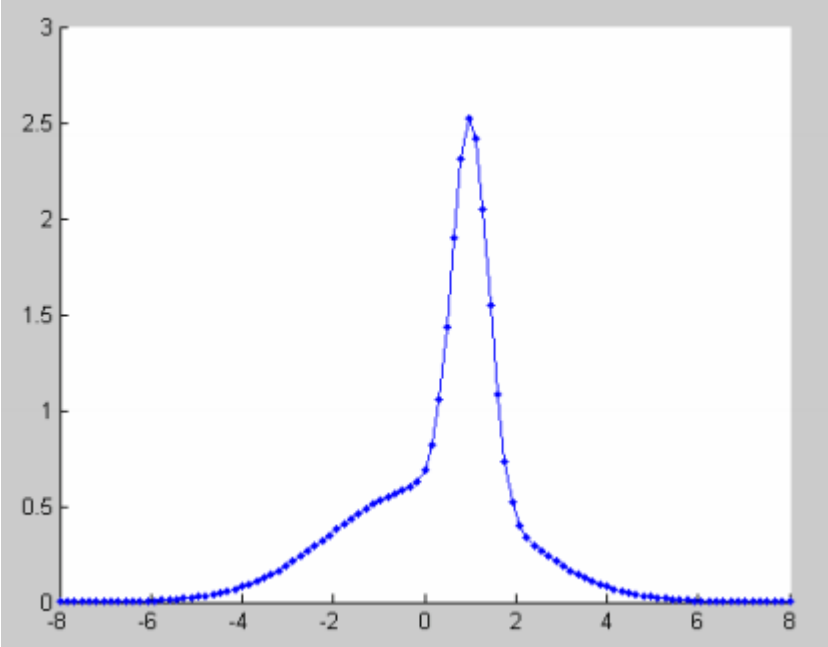


Figure 11: A probability distribution based on combining the two Gaussian distributions shown in figure 10.

So far, in this description, only the use of 1-dimensional Gaussian distributions has been considered, but this technique can be generalised for N dimensions. A common approach in image processing is to map some features of an input image into an N-dimensional feature space, and then to use a suitably trained GMM to find the probability that the image satisfies some criteria [41].

Typically an Expectation Maximisation algorithm [42] is used to adjust the parameters of the GMM to get the best fit to the observed data. This is a two-step, iterative algorithm:

1) E-Step: for each point, estimate the probability that each Gaussian in the model generated it.
2) M-Step: modify the parameters of the model to maximize the likelihood of the data.

2.3.6. Wavelets and the Haar transform

Wavelet transforms, like the DCT, Fourier transform (FT), and other variations of the FT (such as the Fractional Fourier Transform [43], and Short Time Fourier transform [44]) are time-domain to frequency-domain transforms. However,

whereas the FT, Fractional FT, STDT and DCT are fixed resolution transforms, the wavelet transform is a multi-scale and multi-resolution transform.

This is significant in time-domain to frequency domain transforms, due to the limitations such a process faces as implied by Heisenberg's uncertainty principle [45] – mainly that the momentum and the position of a moving particle cannot be known simultaneously. In the case of a time-domain to frequency domain transform, this tells us that we cannot know what spectral component exists at any given instant in time. Stated another way, a higher frequency component can be located better in time, whereas a lower frequency component can be located better in frequency.

Considering fixed resolution transforms (such as FT, DCT and STFT) each are found to have limitations due to the uncertainty principle. The FT operates over the entire length of input signal. This results in issues trying to identify higher frequency signal components that are local to a specific region of the input signal. However, the FT does perform well at identifying lower frequency components that are present over the whole input data.

In an attempt to address this short coming of the FT, the STFT was developed. The STFT uses a windowing function, shifted through time. This allows the higher frequency components found in the input signal, to be described in terms of not just the frequency itself, but also with some temporal information. However, lower frequency components cannot be identified.

Wavelet transforms are an attempt to improve this situation, by providing frequency information at various locations within the input data, and at various resolutions. This is achieved by adapting a mother function (by shifting and dilation) to enable the transform to be applied at different scales and locations within the input data.

For example consider the simplest wavelet transform of all, the Haar transform [46]. Its mother function is based on a single period of a square wave, as described in equation 2.14.

$$\varphi(t) = \begin{cases} 1 & 0 \leq t < 1/2 \\ -1 & \frac{1}{2} \leq t < 1 \\ 0 & otherwise \end{cases} \tag{2.14}$$

Where $\varphi(t)$ is the wavelet's mother function.

Now consider the Haar transform being performed on some input data made up of 8 samples. The basis functions (derived from the mother function) can be expressed graphically as a set of waveforms (see figure 12), or as a set of vectors (as shown in figure 13).

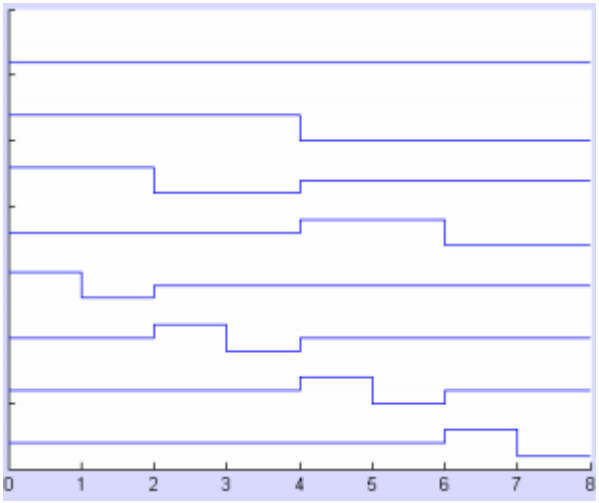


Figure 12: a graphical depiction of the Haar wavelets basis functions –based on the mother function, shifted and dilated for use at various scales and locations across the input data.

```
< 1,  1,  1,  1,  1,  1,  1,  1 >
< 1,  1,  1,  1, -1, -1, -1, -1 >
< 1,  1, -1, -1,  0,  0,  0,  0 >
< 0,  0,  0,  0,  1,  1, -1, -1 >
< 1, -1,  0,  0,  0,  0,  0,  0 >
< 0,  0,  1, -1,  0,  0,  0,  0 >
< 0,  0,  0,  0,  1, -1,  0,  0 >
< 0,  0,  0,  0,  0,  0,  1, -1 >
```

Figure 13: The Haar wavelet basis functions, expressed as a series of vectors.

Note that while the vectors listed in figure 13 are all orthogonal, they are not orthonormal. Creating a set of orthonormal vectors is easily accomplished by dividing the vectors above by their magnitude.

The complete set of orthogonal basis vectors also implies that the Haar wavelet transform only makes sense when operating on data sets of dimensions of a power 2.

To carry out the Haar transform, the dot product of the input sample data and each (orthonormal) basis vector is taken. Each result is a different coefficient in the output of the Haar transform, and each coefficient represents the magnitude of a specific frequency component as at specific scale and location within the input data.

The Haar wavelet transform is natively a one dimensional transform. To perform a Haar wavelet transform on a 2D input matrix, a number of one dimensional Haar transforms are performed; firstly across all the rows of the input data, and subsequently across all the columns (the same technique as was described for performing DCTs on multi-dimensional data, in section 2.3.4).

2.3.7. JPEG compression

The JPEG standard [13] defines a set of techniques, methods and formats used for compressing images. Baseline JPEG is one of the more commonly used methods and will form the basis of the discussion here. JPEG compression is a lossy compression technique, meaning that information is lost when the image is compressed. However, by exploiting the properties of human vision JPEG compression aims to retain the most relevant information, so that when the compressed image is reconstructed the subjective reduction in quality when compared to the original is minimal.

It is also important to note that the techniques employed by JPEG compression are essentially colour blind, and if a colour image is to be compressed then it must first be split into three images (one for each colour channel) and each image is then compressed separately. The techniques can actually be applied to any greyscale image, and so any colour space could be used to represent a full colour image. However, JPEG favours the YCbCr colour space, which allows a higher image quality for a given level of compression, as the three channels being used (Y=luma or brightness, Cb = blue difference, Cr = red difference)

maps more closely to perception of colour in the human visual system, which is more sensitive to luminance rather than colour (due to the relative densities of cone and rod photoreceptors in the retina).

The Cb and Cr channels are therefore typically down sampled, and this reduction in the resolution of the Cb and Cr channels has little effect on the perceived quality of the reconstructed image.

Another feature of the human eye is that it is more sensitive to small variations in intensity over a large area, than to the exact strength of high frequency variations in intensity [26]. This means that the perceived image quality of a compressed image will be higher, if it is the higher frequency components of that image that have been disrupted and the lower frequency components that have been better preserved.

To take advantage of this feature of human vision, the next step taken in JPEG compression is to transform the image into the frequency domain, using DCT transforms. Each of the three images (one image for each channel) is split into blocks of 8x8 pixels. The image dimensions may not be an exact multiple of the block size, so blocks at the edges of the image may need to be padded with dummy data. Edge blocks might be padded with a single colour (say black), but this can lead to some edge effects or artefacts. Using a repeat of the pixels along the edge of the image helps to reduce (but not eliminate) the artefacts.

Then the 2-dimensional DCT is carried out on each block. Before doing this however, each blocks data is shifted to sit in a range that is centred on zero, rather than one that is between zero and some maximum value. This is done to make the computation more efficient.

The result from the DCT is a matrix of coefficients, each coefficient representing the weight that should be applied to the corresponding basis image (as shown in figure 9) if the original image was to be reconstructed. The next step is to quantize each of the coefficients. Quantizing a value involves dividing the value by a quantisation factor using integer division. It should be noted that this part of the process is lossy, and that while dividing by a higher quantisation factor

means that fewer bits are required to store the value, it also means that the value is less accurately stored. The opposite is also true, that if a lower quantisation factor is used, more bits are required to store the quantised value, but the original value can be reconstructed with a higher degree of accuracy. As different coefficients from the DCT represent different basis images (and considering human vision is more sensitive to some of the basis images than others), the quantisation factor that is used for each coefficient is tuned to the basis image in question. Therefore a quantisation matrix is used to represent the quantisation factors that are used.

While the JPEG standard does not specify a specific matrix that should be used, a common one is shown in equation 2.15. As a general observation, it can be seen that lower quantisation factors are used at the top left of the matrix (the factors applied to the lower frequency coefficients) than those towards the bottom right of the matrix (the factors applied to the higher frequency coefficients).

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \quad (2.15)$$

Where *Q* is a quantisation matrix commonly used in JPEG compression.

This weighting makes sense given the previous statement that human vision is more sensitive to lower frequency components than to the higher frequency components within an image.

Typically, as the result of the quantization step, many of the higher frequency coefficients are reduced to zero or to very small values. This fact is once again taken advantage of in the final step of the compression, that of loss-less entropy encoding of the quantised block.
To carry this step out, the coefficients of the quantised block are converted from a matrix into a linear sequence of integers, and the way in which this is done is

key to maximising the amount of compression achieved by this step. By attempting to group long runs of zero's together in the sequence, a higher level of compression can be achieved, and this is done by starting with the DC component (in the top left hand corner of the matrix) and proceeding to construct the linear sequence of coefficients in a zigzag pattern (as shown in figure 14) ending with the higher frequency coefficients in the lower right hand corner. When considering typical photographic images, this zigzag pattern tends to result in the sequence ending with a large number of zero's, separated by only a few small values. This type of sequence is particularly efficient to encode using an entropy encoder.

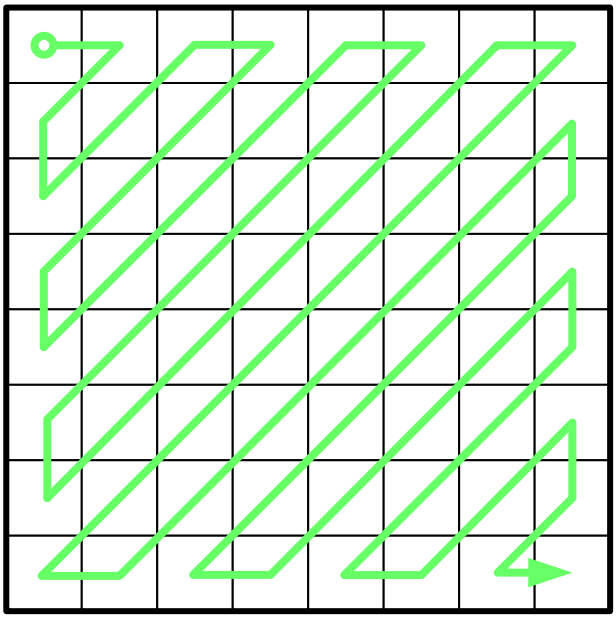


Figure 14: The zigzag ordering of the coefficients that is used to construct the optimal sequence for entropy encoding.

As previously stated, the compression ratio used for JTAG is not fixed; instead it can be specified on an image by image basis. The mechanism used to increase or decrease the compression ratio is to scale the quantisation matrix up or down. If the matrix is scaled up, then the block's coefficients will be recorded with less accuracy, and more of the higher frequency components will be reduced to small values or to zero. As the compression ratio is increased still further, artefacts in the images will start to become visible. As might be expected, the first artefacts to be observed will be those associated with the higher frequency components of the image. Figure 15 shows how increasing the compression ratio effects image quality. It can be seen that flesh tones (which

are generally made up of lower frequency components) are handled well, even at higher levels of compression. However, higher frequency parts of the image start to reveal the blocky nature of JTAG. It is interesting to note that the DCT basis images which have been assigned higher quantisation factors start to become more visible in the blocks. This makes sense, as the higher quantisation means that when the blocks of the image are reconstructed, the weightings used for those basis images are less accurately recreated.

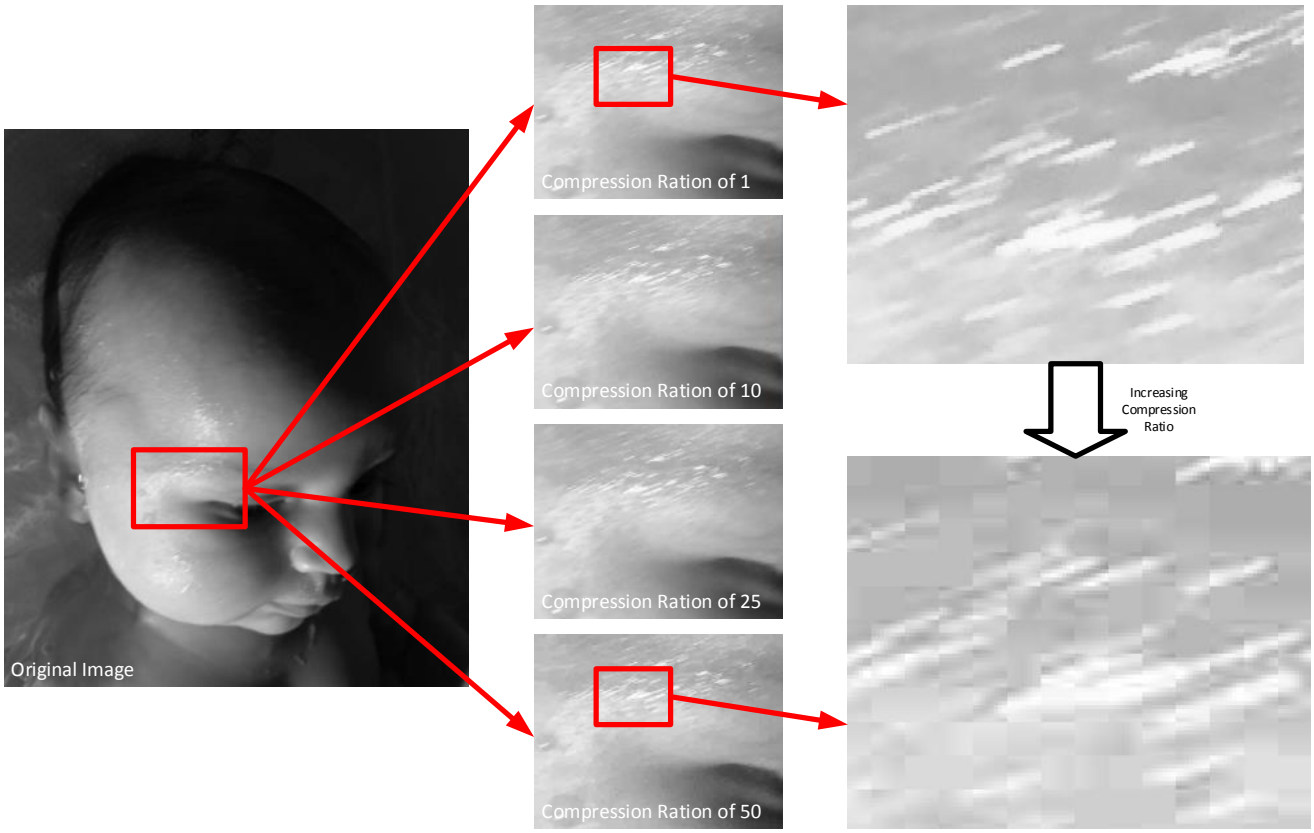


Figure 15: the effect of increasing the compression ration used when storing an image in JPEG format.

2.3.8 Active Appearance Model [18, 19]

Active Appearance Models (AAMs) were proposed by Cootes et al in 1998 [18], and combine both a shape model and a texture model to allow the contents of images to be better understood. By creating a model that understands both texture and shape, and understanding how that model relates to some

previously unseen image, it is possible to extrapolate and interpret the contents of that previously unseen image.

To create an AAM for a specific type of object requires a set of training images showing various examples of that object. Each image should be augmented with a number of landmarks, each of which corresponds to the position of key features of the object shown in the image.

For example, if the model being built is to represent a face, then the landmarks could be based around the position of the features of the face such as eyes, nose, or mouth.

Each image in the training set will be augmented with the same number of landmarks, relating to the same set of object features. The complete set of landmarks for a specific image defines the shape of the object shown in that image.

The complete set of augmented training images allows a statistical model of shape variation to be constructed. The landmarks for each of the images are considered in turn. Firstly they are aligned into a common co-ordinate frame using procrustes analysis. The locations of each set of landmarks can then be represented as a vector, x. Principle Component Analysis (PCA) can then be carried out on the set of vectors, leading to equation 2.16 which shows how the approximation of any shape can be achieved based on knowing the mean shape and a set of shape parameters.

$$x = \bar{x} + P_s b_s \tag{2.16}$$

Where: x is the vector of landmarks for a new shape that is being approximated.

$\bar{x}$ is vector of landmarks for the mean shape.

P_s is the set of orthogonal directions of variation (generated by the PCA).

b_s is a set of shape parameters.

A statistical model of the texture variation also needs to be constructed. To do this each training image is shape normalised, that is it is warped so that its

shape matches that of the mean shape. This is done using triangulation (such as Delaunay triangulation) and interpolation (as it is unlikely that each pixel in the shape normalised image corresponds to exactly one pixel in the pre-warped image).

Figure 16 shows two triangles. Each triangle has landmarks at its vertices, but with the landmarks in different relative locations for each. A point within the original triangle can be mapped to an equivalent location within the new triangle by considering the location of the point in terms of the triangle itself, and the distance of the point from the vertices. In practice the location of the point in the new triangle is unlikely to correspond to exactly one pixel, so it is more usual to do the reverse operation, which is for each pixel that falls within the new triangle using triangulation to find the equivalent location within the original triangle. Once this has been done, interpolation between the pixels in that local area can be used to find the value that should be assigned to the pixel in the new triangle.

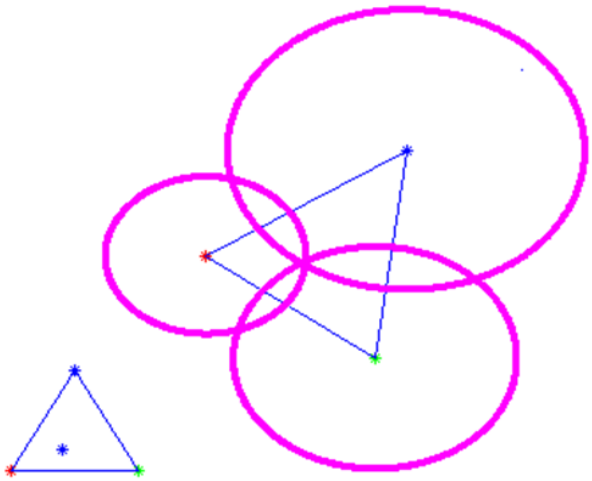


Figure 16: Example of warping using triangulation. A point in the original triangle (bottom left) is mapped to the equivalent point in the new triangle (top right).

More formally, if a triangle is described as two vectors (A_0 and B_0, representing two of the triangles sides) then the location of a point inside that triangle, P_0, can be described by two scaling factors, a and b, as shown in equation 2.17.

$$P_0 = a\,.\overrightarrow{A_0} + b.\overrightarrow{B_0}$$

(2.17)

Where: P_0 is a point inside the triangle

a is a scaling factor.

A_0 is a vector representing one of the triangles sides.

b is a scaling factor.

B_0 is a vector representing one of the triangles other sides.

The scaling factors (a and b) can then be used to map location P_0 to the equivalent location (P_1) within a different triangle (described by vectors A_1 and B_1), as shown in equation 2.18.

$$P_1 = a\,.\overrightarrow{A_1} + b.\overrightarrow{B_1}$$

(2.18)

Where: P_1 is a point inside the triangle

a is a scaling factor.

A_1 is a vector representing one of the triangles sides.

b is a scaling factor.

B_1 is a vector representing one of the triangles other sides.

Once each training image has been shape normalised it can then be sampled (over the area of the mean shape) to obtain the texture data for that image. Typically the measure of texture used is simply the grey level information from the image. The grey level information can be normalised at this point to minimise the effect of variable lighting effects across the set of training images. The resulting normalised texture data for each training image can be represented as a vector, g.

Performing PCA over the complete set of these vectors leads to equation 2.19, which allows the appearance of any example to be approximated based on knowing the mean appearance and applying a set of texture parameters.

$$g = \bar{g} + P_g b_g$$

(2.19)

Where: g is the vector of landmarks for a new shape that is being approximated.

$\bar{g}$ is vector of landmarks for the mean shape.

P_g is the set of orthogonal directions of variation (generated by the PCA).

b_g is a set of texture parameters.

A complete approximation of any example (including the shape and texture) can therefore be summarised by simply specifying the vectors b_s and b_g. As it is likely that the texture and shape are linked, the shape and texture data is combined so that PCA can once again be applied. Equation 2.20 shows the concatenated vector that is generated for each training example.

$$b = \begin{pmatrix} W_s b_s \\ b_g \end{pmatrix}$$

(2.20)

Where: W_s is a diagonal matrix of weights (used to take into account the difference in units between b_s and b_g).

b is the combined vector of shape and texture parameters.

Equation 2.21 shows the new appearance model (combining texture and shape into a single model), which has been obtained by carrying out PCA on the set of vectors b for each training image.

$$b = Qc$$

(2.21)

Where: Q are the eigenvectors of the complete set of vectors b.

c is the set of appearance parameters.

The linear nature of the model allows the shape and texture to be expressed independently from each other as shown in equation 2.22, 2.23 and 2.24.

$$x = \bar{x} + P_s W_s Q_s c \tag{2.22}$$

$$g = \bar{g} + P_g Q_g c \tag{2.23}$$

$$Q = \begin{pmatrix} Q_s \\ Q_g \end{pmatrix} \tag{2.24}$$

For any given c an example image can be created. This is done by first calculating the texture data, g, and then calculating the shape data, x. The texture data can then be warped into shape described by x.

Once the appearance model has been created it can be used to search a previously unseen image for the object of interest, provided a reasonable starting point is available.

The search is treated as an optimisation problem, where we are trying to minimise the difference between an image synthesised from the model (I_m), and the previously unseen image (I_i). δI (the difference vector between I_m and I_i) and can therefore be defined in equation 2.25.

$$\delta I = I_i - I_m \tag{2.25}$$

Where: δI is the difference vector between I_i and I_m

I_i is the previously unseen image.

I_m is the synthesised image.

Adjustments are iteratively made to the appearance parameters, c, in order to reduce the magnitude of δI. However, this is not necessarily a straight forward process as the number of parameters that make up c can be large.

Gaining an understanding of the relationship between c and δI is obviously an important step in performing the optimisation. It turns out that assuming a linear relationship, as detailed in equation 2.26, is good enough to achieve useful results.

$$\delta c = A\,\delta I \tag{2.26}$$

Where: δc is a change in the appearance parameters, c.

A describes the linear relationship between δc and δI.

Multiple multivariate linear regression on a series of values for δc and the corresponding value of δI are used to find A. This is done by considering images for which the appearance parameters, c, are known, and then adjusting the appearance parameters by a small amount, and recording the affect this has.

Images from the training set could be used for this, or new images synthesised from the model itself. There is some advantage in using images synthesised from the model, as the appearance parameters are precisely know, and no noise is added to the images during the process.

In either case a known set of appearance parameters is used, c_0. The parameters are adjusted by some small amount, δc, to obtain new parameters as shown in equation 2.27.

$$c = \delta c + c_0 \tag{2.27}$$

Where: c represents the new appearance parameters.

δc represents the small adjustment being made to the parameters.

c_0 represents the initial appearance parameters.

For the new parameters c, the shape, x, and texture, g, are calculated as shown in equations 2.22 and 2.23, and can be used to create a new image. The shape, x_0, and texture, g_0, of the original image are also available, and so the error between the original and new image can now be calculated. If the shape of both images is normalised the sample error can be defined in terms of the texture alone, δg, as shown in equation 2.28.

$$\delta g = g_0 - g \tag{2.28}$$

Where: g_0 is the shape normalised texture of the original image.

g is the shape normalised texture of the new image.

δg is the sample error.

To carry out the training this process is repeated a number of times, using random, small adjustments δc to c, recording the resulting values of δc and δg, so that multi-variate regression can be used to obtain the value for A in equation 2.29.

$$\delta c = A\,\delta g \tag{2.29}$$

Where: δc is a change in the appearance parameters, c.

A describes the localised linear relationship between δc and δg.

It would be ideal to find a value for A that holds over a large range of errors (δg), but experiments carried out by Cootes et al. have shown that in practice this linear relationship only holds over a limited range of errors (δg), but that this range is enough to allow AAMs to provide useful results.

Cootes et al. have also shown that it is more efficient to employ a multi-resolution approach when carrying out the search. To do this an initial search is carried out on a low resolution image. The results of this search are carried forward, and used as a starting point on a higher resolution image, and the search is repeated. This is process is continued until the search has been carried out on the image at the target resolution.

The AAM described by Cootes et al. uses grey values or intensity to form the texture model. Work has been done by Ya Su et al. to create and test an AAM which uses a more complex measure of texture, based on Gabor wavelets combined with Local Binary Patterns [19]. This can give better resilience to variations in environment (such as lighting levels in photographs), but comes at the cost of run time, which (when working with 2D images) was seen to increase

by over 10x as compared against using intensity to construct the appearance model, as shown in table 1.

Texture representation	Fitting Time of AAM (seconds)
Intensity	77.8
GLBP	865.1

Table 1: Fitting Time of AAM in seconds when using appearance model based on Intensity and GLBP [19].

2.4 Datasets

2.4.1 The Visible Human Project (VHP)

To test any techniques that are developed as part of this research, some 3D medical image data is required. The VHP dataset was used for this research. The VHP was run by the United States National Library of Medicine. Its goal was to provide a complete and anatomically detailed 3D representation of a normal male and female human. The project aimed to include digital images derived from computerized tomography, magnetic resonance imaging and photographic images from cryosectioning of the cadavers. [49]

The project makes available CT, MR and cryosection images of male and female cadavers, with the aim of allowing the data set to be used for a wide range of research purposes (including modelling of the human body [16]).

2.4.2 Selection of the Cadavers.

Human bodies that are made available for purposes of medical research or teaching in the United States, are generally obtained after being gifted to the State Anatomical Board (SAB) by the states citizens. The search for suitable cadavers for use in the visible human project was carried out by a consortium of SABs representing Texas, Colorado and Maryland. The consortium put in place a screening process to access the suitability of the possible candidates. The process considered:

1) Available medical records, specifically looking at evidence of infectious or meta-static disease.
2) Surgical history.
3) Medical conditions that could cause or lead to altered anatomy.
4) Physical condition of the cadaver (looking for scars or distortions in anatomy).
5) Candidate's height and weight.

Obese or emaciated cadavers where ruled out, and cadavers with a height of more than 6 feet were also rejected due to physical limitations of the imaging equipment to be used.

2.4.3 How the Dataset was Created. [12]

Although multiple cadavers were selected, the case of Joseph Paul Jernigan is well documented and it is interesting to review his case, along with the associated timeline and techniques used in the imaging of the cadaver.

Jernigan died aged 38 years old, at 12:31am on August the 5th 1993, at the Texas Department of Corrections in Huntsville. The cause of death was due to a court ordered lethal injection. In other cases of court ordered lethal injection, cadavers had been seen to suffer from significant deterioration within the first 24 hours from the time of death. However, the risk of this was managed by an infusion of 19 litres of 1% formalin and the administration of an anticoagulant. This process caused a minimal amount of disruption (around the site of the injections) to the anatomy of the candidate, however, the injection sites were sutured and the body cleaned before being transferred to the Colorado SAB morgue, arriving 8 hours after death.

Jernigan did have some altered anatomy from previous medical treatments, including:

1) Left Orchiectomy at age 15.
2) Appendectomy at age 21.
3) Number 14 tooth removed at age 38.

It was considered important to capture scanned images from the unfrozen cadaver, as pilot studies had shown CT and MRI images to be degraded by the freezing process.

Initially anteroposterior film radiographs of the entire cadaver were carried out at 12.5 hours after death.

MRI scanning was carried out at 18.5 hours after death at Colorado University Hospital. A General Electric 1.5 Tesla Sigma MRI system was used, resulting in images taken at a 4mm interval, with a resolution of 256 x 256 voxels per slice. Each voxel was stored with 16-bit accuracy, and the images were initially stored in General Electric Genesis format.

CT scanning was carried out after the cadaver has been immobilized (by use of a specially constructed plywood mould, and a foaming agent). This was done so that the CT images could be precisely correlated with the cryosection images that would be captured later in the process. It took around 15 minutes for the foaming agent to expand and solidify to fully immobilize the cadaver.

The CT scanning process was commenced at 22.5 hours after death. Three different slice intervals were used in the CT scanning; 1mm for the head and neck, 3mm for imaging of the thorax, abdomen, and pelvis, and 5mm for the lower extremities.

At 26 hours after death the cadaver was returned to the SAB morgue and placed into a specially constructed dry-ice freezing chamber. Other than being moved so that some further CT scans could be carried out (performed on 29th August and 14th October) the cadaver remained in the freezing chamber until 15th October, at which point it was transferred to a walk in freezer, and stored in conditions of minus 7 degrees Celsius or below.

The final selection of the Cadaver (and along with it the decision to go ahead with the cryosection imaging process) was made on 2nd September. The cryosection imaging process required that the cadaver by split into four blocks (due to physical limitations of the equipment being used). This process was started on 11th November with further CT scans to work out the best locations for the cadaver to be split into the four blocks. The cadaver was subsequently transferred back to the freezing chamber (on 20th January 1994) in preparation for the sectioning, which it had been decided could be more accurately carried out with the cadaver at a lower temperature. The sectioning was carried out on 24th January, and the cadaver was split into 4 sections;

1) Head, neck and thorax.
2) Abdomen and pelvis.
3) Thighs and knees.
4) Legs, ankles and feet.

The cryosections were prepared using a cryomacrotome that had been specially modified to allow continued accurate operation in extremely cold conditions. It

was developed within the former department of anatomy of the University of Colorado's Medical School. The cutting disk was 14 inches in diameter, and span at 300 rpm, and was held in a fixed position – the specimen was attached to a moveable table, and a cut was made when the specimen was moved past the cutting disk.

The cryomacrotome used had previously been used to with slice intervals as low as 0.1mm. The slice interval used for the VHP was 1mm, but this still required that the machine was maintained on a scheduled basis, and regularly recalibrated.

The process was repetitive, and the number of cycles each day was around 50. The time of each cycle varied from 3 minutes to 15 minutes – the variation due to the variation in manual work that had to be carried out during each cycle. This could include the use of a scalpel to trim or cut away any structures that did not cut cleanly (such as tendons) or to clean away any debris that may be present after a cut. Compressed air was also used to clean the surface, and blue latex was used to fill any cavities (required to add stability during the cutting process). Once the surface was ready to be imaged, a black mask (with a grey scale strip) was used to frame the specimen. A post-it note was then applied to the mask and used to display the original slice number. An example image, where the mask, greyscale strip, and post-it note can be seen is shown in figure 17.

Figure 17: Example slice from the Visible Human Project (cryosection dataset).

The digital images were captured by a Leaf CCD camera and a colour filter wheel, allowing the red, green and blue images to be captured separately. Each of the three images was stored in 2048 x 2048 x 14-bit TIFF format. The red, green and blue images were combined to create a 42-bit full colour image, this being subsequently reduced to 24-bits, by logarithmically compressing the 14-bits used for each channel down to 8-bits. This was done independently for each channel. Each pixel (in the final full colour images) therefore represents a voxel of volume 0.32 x 0.32 x 1.0 mm.

Film based images were also captured for archival and redundancy purposes. The films were all processed at the same time, once all images had been captured.

2.4.4 Technical Specification of the Dataset.

The Visible human project created a number of different datasets based on the cadavers being imaged, including CT, MR and Cryosection data. Each modality generated datasets of differing resolutions and qualities. Specific data about each data set is presented in the Table 2.

Dataset	Male			Female		
Property	Resolution (per slice)	Colour Depth	Slice Interval	Resolution (per slice)	Colour Depth	Slice Interval
CT	512x512	12-bit	1, 3 and 5mm.	512x512	12-bit	1, 3 and 5mm.
MR	256x256	12-bit	4mm	256x256	12-bit	4mm
Cryosection	2048x1216	24-bit	1mm	2048x1216	24-bit	0.33mm
High Resolution Cryosection	4096x2700	24-bit	1mm	4096x2700	24-bit	0.33mm

Table 2: Details of the Visible Human Project data.

The work carried out here is based on the cryosection data, of both the male and female datasets. This may seem like a strange choice, given the obvious unsuitability of cryosection for diagnosis. The main driver for this is the availability of the datasets required for training, validation and testing purposes.

For more general purpose image processing techniques, datasets are widely available. Either standard datasets can be used, or internet search engine results can be used to provide data [15].

However, the datasets required here are quite specific, and it is not only the image data that is required, but also an accompanying segmentation. While pre-segmented datasets of MRI data do exist, it can be difficult to get access to them, and MRI images can be difficult to interpret without specialist knowledge.

In comparison, the cryosection images are easier to segment, allowing an individual inexpert in the field to generate datasets.

3.0 Initial research into the use of DCT and Haar transforms

The aim of this research has been to develop, validate and review a more automated technique to perform segmentation of medical images, than is currently available. The current state of the art in medical image segmentation is often a manual process relying on the skill and training of medical professionals. [26].

There are a number of different types of medical imaging techniques. Although the technique used for generating the images is different, the images themselves tend to be intensity based, rather than full colour. Also, they tend to be generated by combining a number of slices into a 3D model, meaning that there can be a different (often poorer) resolution along the z-axis than along the x-axis and y-axis. The imaging techniques also tend to have a variation in intensity over any given scan.

The research is not aimed at any specific medical imaging technique, but is rather the development of a general texture based segmentation technique for intensity based 3D images, which is robust to unpredictable variations in intensity such as those which commonly afflict medical imaging techniques (such as MRI).

3.1 Over view of the technique

At this point it is useful to describe the outline of the technique being developed to carry out the segmentation. Aspects of this technique will be described in more detail later on in the text, along with a narrative describing the development of various enhancements.

There were three phases to the technique:

1) training the GMM (using data from a training data set).
2) tuning (searching for a threshold value, using data from a validation data set)

3) operation (segmenting previously unseen images from a test data set)

For each of these three phases an input data set was created. Each of these datasets contained different and independent images, which took the form of a series of ordered slices. The slices can be considered individually (as 2-dimensional data) or combined into a 3-dimensional data set. Each slice was manually segmented and this segmentation data was used depending on the specific phase; as training data in the training phase, or for scoring the results in the training and operational phases.

The approach taken was to create a descriptor in some feature space for each voxel within the images, and then to classify each voxel as being a member of the segmentation set or not based on analysis of its associated descriptor.

This method is similar to the Bag-of-features approach in some ways. Patches of the image are mapped into feature space, and then classified according to the location in feature space that the patches occupy. Generally speaking, when considering a Bag-of-features style approach, the spatial location of the image patches is not considered, and not mapped into feature space. Some work has been carried out to investigate the effect of including some measure of coarse geometry [5] or spatially weighting the features [8] on the techniques effectiveness. These styles of enhancement were not considered here, instead the spatial relationship between pixels and voxels was taken into account by a spatial domain post processing procedure, that were carried out after an initial segmentation had been performed.

Another significant difference from the bag-of-features approach, is that no key point generator [14] was used. Instead, the technique splits the entire image into a number of blocks (such that every part of the image is mapped to one and only one block), and analyses each and every block. Once all the blocks have been mapped into feature space, and classified as being a member of the segmentation set or not, then it is also possible to consider the spatial relationship between the blocks to further refine the results.

The transform that maps the patches into feature space is akin to the SIFT [9] transform that is commonly used for 2D image analysis [1]. The SIFT transform is invariant to rotational and scale. The process being developed here also needs to be invariant to such variations in scale and rotation. The process should also be invariant to the variations in intensity that medical images are often subject to.

However, the transform of a block into feature space does not necessarily have to be invariant to all of these, for the overall process to be. For example, given a small enough block size, and a large enough number of blocks being available for the training of the GMM, it was thought likely that variations in rotation and scale could be accounted for in the training and response of the GMM. Given that the number of blocks in each image is reasonably large, this assumption was carried forward for the first part of the research, and no specific testing against variation in rotation and scale was carried out.

As mentioned above, the overall process should also be invariant to the type of variations in brightness and intensity that commonly afflicts medical images. It was thought that while invariance to rotation and scale could be dealt with via suitably training the GMM, invariance to intensity variations should be taken into account earlier in the process, when the block was being mapped into feature space for example. This was decided on the basis that, given a small enough block size, the intra-block intensity variation seen is likely to also be small. It therefore also seemed likely (at a superficial level) that the transform mapping the block into feature space could be made invariant to these small intra-block variations in intensity.

Once more issue that could affect a block based approach is offset. Consider Figure X. It can be seen that the same area of pixels can be used to generate blocks with very different appearances, depending on the offset of the blocks. However, as with variations of rotation and scale, it is assumed that given a small enough block size, and a large enough sample size, the GMM can be trained to recognise the various possible block appearances.

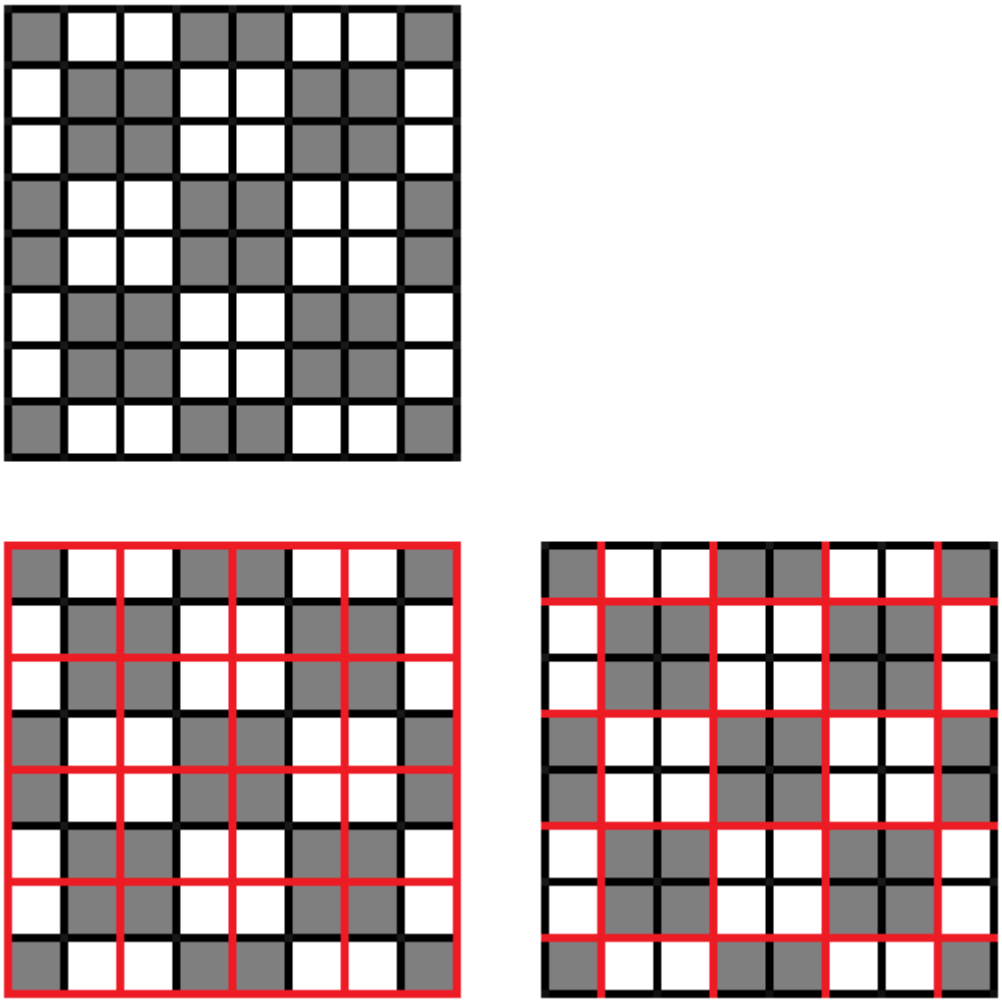


Figure 18: A close up of an area of 8x8 pixels taken from the centre of an image (top). The same area split into blocks (bottom left and bottom right).
Pixel boundaries shown in black. Block boundaries shown in red.

The descriptors were calculated based on the intensity of the voxel in question, but crucially based also on the intensity of voxels contained within a local rectangular (or cuboid) block, with the voxel in question at the blocks origin. The block dimensions were fixed for a given experiment, although the effect of using different dimensions was also investigated.

For the training phase, only descriptors that were associated with voxels from within the segmentation set were used. These descriptors were grouped together and used to train the GMM.

The GMM was implemented using netlab and its library of GMM related functions [10]. Each descriptor was flattened into a 1 x d vector (where d is the number of coefficients in the descriptor). The descriptors were then combined into a d x n matrix (where n is the number of descriptors). This matrix was then used as the input data required to train the GMM, which was set to use a spherical co-variance with 10 centres. The decision to use 10 centres was initially taken after a brief investigation and some initial experimental work. It was never revisited, and it is noted that it would be interesting to investigate the effect on results using other values may have.

Once trained, the GMM could be used to generate the probability that a voxel was also a member of the segmentation set, based on that specific voxels descriptor. As inclusion in the segmentation set is a binary operation, a probability cannot (by itself) specify inclusion within the segmentation set. Some threshold was also needed to classify a voxel as included or not, based on the output of the GMM.

If the threshold was set too low then too many voxels were included within the set, and if the threshold was set too high, too few voxels were included in the set. Choosing the correct value for the threshold was critical in achieving good results, and minimising errors. The threshold was found during the tuning phase of the technique. During this phase a simple search was carried out, using different threshold values to segment previously unseen images, and comparing the results against manual segmentation results. The search looked for the threshold that generated the least errors on the validation set.

Once the GMM had been trained, and a threshold value had been found, the operational phase could be started. In this phase, new (previously unseen) images could be segmented using the technique. The results were then evaluated and reviewed.

There was a reasonably limited amount of data available for validation and test, so resampling was used to increase the amount of data available. This was done by discarding the first N rows, M columns, and (where appropriate) S slices of the image. Note that (in order to avoid aliasing effects) the values that

can be assigned to N, M and S are restricted, such that the resultant new image will meaningfully add to the validation and test data sets. This restriction is shown in equation 2.30.

$$M < x, \qquad N < y, \qquad S < z \tag{2.30}$$

Where:*M* is the number of columns being discarded from image.

N is the number of rows being discarded from image.

S is the number of slices being discarded from the image.

x is the width of the block size being used.

y is the height of the block size being used.

z is depth of the block size being used.

By taking this approach there is also the added benefit of testing the assumption, made earlier in the text, that the training of the GMM will provide robustness against blocks containing similar textures having very different appearances (due to the exact offset of the block within the image).

3.2 Input Data and Manual Segmentations

The visible human data set has been used to develop and test the technique. The high resolution data set has been used [49], which is provided in full colour. The visible human dataset is made up of a large amount of data. The technique was developed and tested on a sub-set of the data. Three datasets were used. These were from a bone in the upper arm, a bone in the left upper leg and a bone in the upper right leg. Bones were chosen as an in-expert manual segmentation of these is relatively straight forward, and an automatic segmentation is not trivial.

In total 173 sections of slices, each 201x201 pixels in size were manually segmented, summarised in tables 3 to 8.

Once the manual segmentation has been completed this does not have to be done again, and the segmentation is used to train, tune and test the GMM.

Some slices from each data set are shown in tables 4, 6 and 8. It can be seen that the size, shape and even the relative location of the segmentation on different slices through the sets is quite varied.

Some medical imaging techniques generate colour (or multi-dimensional images) where as some generate grey-scale (or single dimension) images. Throughout this research the dataset has been used in both contexts. Where a grey-scale image is being considered, the full colour images have been reduced to a monochrome image, by taking a mean value across the datasets 3 dimensions. As already stated, the human eye is more sensitive to certain aspects of texture, and certain ranges of the colour spectrum. Therefore, and in hindsight, it may have been more appropriate to create a monochrome image by performing a weighted sum of the RGB components, in line with the sensitivity of the human eye over the spectrum, rather than just calculating the mean of the RGB components. Much research has been carried out into weightings that can be used to generate more faithful monochrome images from full colour images [7] however this was not considered.

Name	Upper Arm Bone
X size	201
Y size	201
Z size (number of slices)	71

Table 3: Details about the upper arm bone dataset.

Table 4: Example slices and segmentations from the upper arm bone dataset.

Name	Upper Left Leg Bone
X size	201
Y size	201
Z size (number of slices)	51

Table 5: Details about the upper left leg bone dataset.

Table 6: Example slices and segmentations from the upper left leg bone dataset.

Name	Upper Right Leg Bone
X size	201
Y size	201
Z size (number of slices)	51

Table 7: Details about the upper right leg bone dataset.

Table 8: Example slices and segmentations from the upper right leg bone dataset.

3.3 Evaluation of Results

To test any segmentation technique, a scoring mechanism has to be defined. When quantifying the success of a given technique it is usual to compare the results against a manual segmentation, and this is what was done. The automated segmentation technique outputs a 3D matrix of 1s and 0s. A 1 in a given location signifies that the given voxel is a member of the segmentation set. A 0 signifies it is not.

The colour images from the visible human data set where used to carry out a manual segmentation, which was then used to generate a similar matrix to compare the results to. The slice images tended not to have a high green component within them, so pure green (24-bit RGB value 0x00FF00) was used to highlight the segmentation set.

Code was written that could load in the manually segmented image and find the locations marked in pure green. These locations were then combined to generate a 3D target mask, to which the results of the automated technique could be compared.

In doing the comparison a number of metrics were collected, as using just a single percentage value to show the success of the process could be misleading. Consider the case where 98% of the voxels are outside the segmentation set. In this situation if the technique failed completely and resulted in an empty segmentation set, this could be portrayed as a result of 98% successfully classified voxels. This is clearly misleading. For this reason the following metrics were calculated, to give a better overall picture of the success of the segmentation:

1) Number of correct ones. This is the number of voxels that are in the segmentation set (according to the manual segmentation) that were correctly identified. (i.e. True Positive).
2) Number of incorrect ones. This is the number of voxels that are in the segmentation set (according to the manual segmentation) that were not correctly identified. (i.e. False Positive).

3) Number of correct zeros. This is the number of voxels that are not in the segmentation set (according to the manual segmentation) that were correctly identified. (i.e. True Negative).
4) Number of incorrect zeros. This is the number of voxels that are not in the segmentation set (according to the manual segmentation) that were incorrectly identified. (i.e. False Negative).
5) Number of correct voxels. This is the total number of voxels that were correctly classified when compared with the manual segmentation.
6) Number of incorrect voxels. This is the total number of voxels that were incorrectly classified when compared with the manual segmentation.

With hindsight it would also have been useful to capture and record some measure of the errors inherent to these results, so that the significance of variations made to the technique could be properly evaluated. This was not done as part of the original experimental work.

To provide some idea of the size of the errors inherent in the results presented, an experiment was rerun. The experiment was based on the “upper arm bone” dataset, and used a block size of 4x4 pixels, and was based on a 2D DCT transform. In order to increase the amount of test data available the input data was often resampled (keeping the resolution fixed, but varying the starting offset, as described in section 4.6). In this experiment, a block size of 4x4 pixels allowed the input data to be resampled 16 times, generating 16 different results.

The results are presented in table 9, and the mean, variance and standard deviation are presented in Table 10.

X offset	Y offset	% Correct Ones	% Correct Zeros	% Correct Voxels
0	0	80.5	99.7	95.4
0	1	80.5	99.6	95.3
0	2	86.9	99.5	96.7
0	3	87.2	99.5	96.7
1	0	87.2	99.6	96.8
1	1	86.5	99.5	96.6
1	2	87.6	99.5	96.9
1	3	87.6	99.5	96.8
2	0	87.8	99.5	96.9
2	1	87.4	99.5	96.7
2	2	86.3	99.5	96.5
2	3	85.2	99.3	96.1
3	0	85.8	99.5	96.4
3	1	85.2	99.5	96.2
3	2	74.9	99.7	94.1
3	3	75.2	99.5	94.0

Table 9. results from a groups of experiments based on a block size of 4x4 pixels, and a 2D DCT transform.

	% Correct 1s	% Correct 0s	% Correct Voxels
Mean	84.48	99.53	96.13
Variance	17.50	0.01	0.83
Standard Deviation	4.18	0.09	0.91

Table 10. The Mean, variance and Standard deviation for the data presented in table 9.

It is interesting to note the percentage of correct zeros has a very much lower variance than the percentage of correct ones. This is because the dataset being

used contains a much higher number of voxels outside of the segmentation set rather than inside, and the process (due to use of region growing) tends to correctly classify the vast majority of voxels outside of the segmentation set. It may, therefore, be more interesting to consider the standard deviation of the percentage of correct ones when reviewing the results presented later on in the text.

The above were recorded as absolute values, which were then used to generate the percentage of ones, zeros and voxels that were correctly classified. These percentages were then used to compare the relative success of each of the techniques.

There is also a subjective side to the quality of the results being achieved. For this reason a plot was created for each slice, showing the comparison between the manual segmentation and the automated segmentation. Such a plot is shown in figure 19

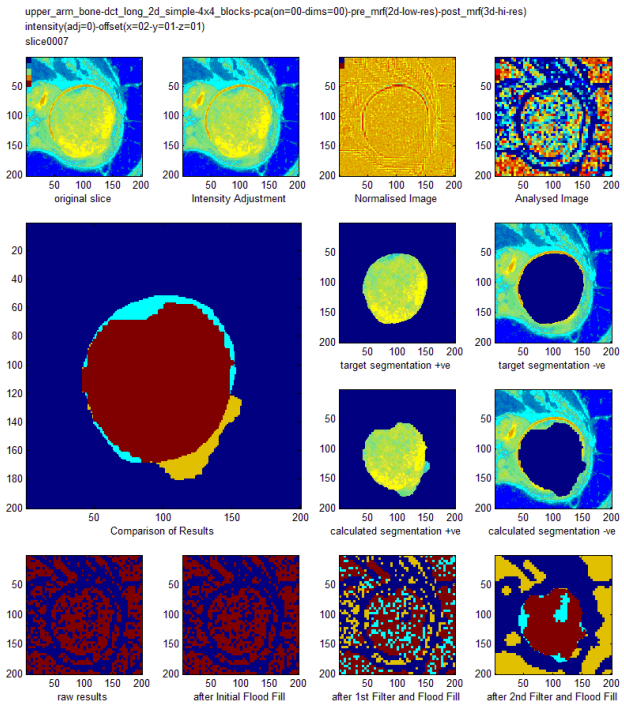


Figure 19: A sample of a plot showing the results of a segmentation.

The plot is made up of a number of sections, each of which is described below (from left to right, and then from top to bottom).

1) (Top Left) This panel shows the original image, as loaded from a file.
2) This panel shows the image after its intensity has been adjusted. This is done to examine the techniques robustness to variations in intensity.
3) Normalised image. This panel is included for information only, to give a representation of the image once the DC component of the image has been removed.
4) (Top Right) This panel shows a plot of the probability that each location is a member of the segmentation set. Four blocks of colour are shown in the top left corner of this panel. The top block shows the colour of the lowest probability in the plot, and the bottom block shows the colour of the highest probability in the plot.
5) (Large panel on the left). This panel shows a comparison between the manual segmentation and the automatically generated segmentation. The key to this panel is:
    a. Dark Blue voxels are correctly identified zeros.
    b. Dark Red voxels are correctly identified ones.
    c. Light Blue voxels are ones (according to the manual segmentation) but have been identified as zeros according to the automatic segmentation.
    d. Yellow voxels are zeros (according to the manual segmentation) but have been identified as ones according to the automatic segmentation.
6) This panel is labelled “target segmentation +ve” and shows the portion of the intensity adjusted image that is a member of the segmentation set (according to the manual segmentation).
7) This panel is labelled “target segmentation –ve” and shows the portion of the intensity adjusted image that is not a member of the segmentation set (according to the manual segmentation).
8) This panel is labelled “calculated segmentation +ve” and shows the portion of the intensity adjusted image that is a member of the segmentation set (according to the automatic segmentation).

9) This panel is labelled “calculated segmentation –ve” and shows the portion of the intensity adjusted image that is not a member of the segmentation set (according to the automatic segmentation).
10) (Bottom Left) This panel shows the raw results from the classification, before any post processing takes place. The raw results are calculated by placing thresholds on the probability shown in the panel labelled “Analysed Image”. The key to this panel is:
    a. Dark red voxels have been classified as being within the segmentation set.
    b. Dark blue voxels have been classified as being outside the segmentation set.
11) This panel shows the results after the initial region growing process has been applied. Note that an initial region growing process was not always used, so in some examples this panel will show no change from the raw results. The key to this panel is:
    a. Dark Blue voxels are zeros that remain unchanged by this stage.
    b. Dark Red voxels are ones that remain unchanged by this stage.
    c. Light Blue voxels are zeros that have been changed to ones by this stage.
    d. Yellow voxels are ones that have been changed to zeros by this stage.
12) This shows the results after the first filter and another region growing process (when used) have been applied. The key to this panel is the same as the panel showing the initial region growing process.
13) This shows the results after the second filter and final region growing process (when used) have been applied. The key to this panel is the same as the panel showing the initial region growing process.

3.4 Descriptors

When segmentation is performed manually or semi-manually the user has to review each slice of the medical image and select the voxels that form part of the organ or region being outlined. Although tools exist to help with this function

(region growing etc.) the user still has to differentiate between the different tissues on the slice. This is done by eye, and so texture obviously plays an important part in this selection.

Image and video, storage and compression techniques (such as JPEG [13]) are able to pick out the features of an image that are most important to the subjective quality of an image. (This is essentially how compression of an image file is achieved, by discarding any information that does not add to the subjective quality of the image). As manual segmentation is carried out by eye, it seems a reasonable supposition that if a texture descriptor was created from those transforms commonly used in image storage and compression (such as DCT and wavelet transforms), there should be enough information to perform the segmentation.

Effort could be put into developing a segmentation technique based on texture, which considered all available information about the textures in question. This technique could build texture descriptors describing the information held in the image and could then be used as an input to the segmentation process. However, considering sources such as the JPEG compression standard [2] and other research into the sensitivity of human vision [reference sensitivity of cones to colour, and to texture], it is apparent that the human eye is more sensitive to some variations in texture and colour than others.

JPEG compression is a lossy compression technique. That is, information is thrown away when JPEG compression is used. However, the concept behind JPEG compression is to make sure the information being lost is the least important information when it comes to reconstructing the image. It is still true that if a high level of compression is being used, then the reconstructed image will be of a lower quality than if a small level of compression is being used, but the most important information is always retained. As stated in Chapter 2, the human eye is more sensitive to small variations of intensity over a large area, than to the exact strength of high frequency variations. Therefore when compressing an image using JPEG, more importance is given to retaining the lower frequency components of the image, rather than the higher frequency components.

To demonstrate this concept, and to investigate the reduction of quality caused by removing various frequency components from an image, some experimentation was carried out. An image was extracted from the high resolution visible human data set. The image is shown in figure 20. The image is a full colour image, and so was split into three images (the red, green and blue components of the original). Each of these images was then split into blocks of pixels (4x4 pixels per block). A 2-dimensional DCT was then performed on each block, I_{block}, resulting in a set of coefficients, D_{block}, as shown in equations 3.1 and 3.2.

$$I_{block} = \begin{bmatrix} i_{0,0} & i_{0,1} & i_{0,2} & i_{0,3} \\ i_{1,0} & i_{1,1} & i_{1,2} & i_{1,3} \\ i_{2,0} & i_{2,1} & i_{2,2} & i_{2,3} \\ i_{3,0} & i_{3,1} & i_{3,2} & i_{3,3} \end{bmatrix} \tag{3.1}$$

Where:I_{block} represents a 4-by-4 block of pixels from the input image.

$I_{x,y}$ represents the intensity of a specific pixel within that block.

$$D_{block} = \begin{bmatrix} d_{0,0} & d_{0,1} & d_{0,2} & d_{0,3} \\ d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,0} & d_{2,1} & d_{2,2} & d_{2,3} \\ d_{3,0} & d_{3,1} & d_{3,2} & d_{3,3} \end{bmatrix} \tag{3.2}$$

Where:D_{block} represents a 4-by-4 matrix of coefficients that have been generated by performing a 2-dimensional DCT on I_{block}.

$d_{x,y}$ represents a specific coefficient within D_{block}

D_{block} was then subjected to three separate transformations, where some of the coefficients were set to zero. The specific transforms that were used are shown in equation 3.3, 3.4 and 3.5.

$$D_1 = \begin{bmatrix} d_{0,0} & d_{0,1} & 0 & 0 \\ d_{1,0} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

(3.3)

Where:D_1 represents a 4-by-4 matrix of coefficients after high frequency components have been set to zero.

$$D_2 = \begin{bmatrix} 0 & d_{0,1} & d_{0,2} & d_{0,3} \\ d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,0} & d_{2,1} & d_{2,2} & d_{2,3} \\ d_{3,0} & d_{3,1} & d_{3,2} & d_{3,3} \end{bmatrix}$$

(3.4)

Where:D_2 represents a 4-by-4 matrix of coefficients after the DC coefficient has been set to zero.

$$D_3 = \begin{bmatrix} 0 & d_{0,1} & 0 & 0 \\ d_{1,0} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

(3.5)

Where:D_3 represents a 4-by-4 matrix of coefficients after the DC coefficient has been set to zero.

Once the transforms had been applied, a 2-dimensional inverse DCT was performed, and the red, green and blue components of the image were reconstructed and then combined into a single image. The resulting three reconstructed images were then reviewed along with the original to investigate the effect of removing:

1) the DC coefficient,
2) the high frequency components,
3) both the DC coefficient and the high frequency components.

The images are shown in figure 20.

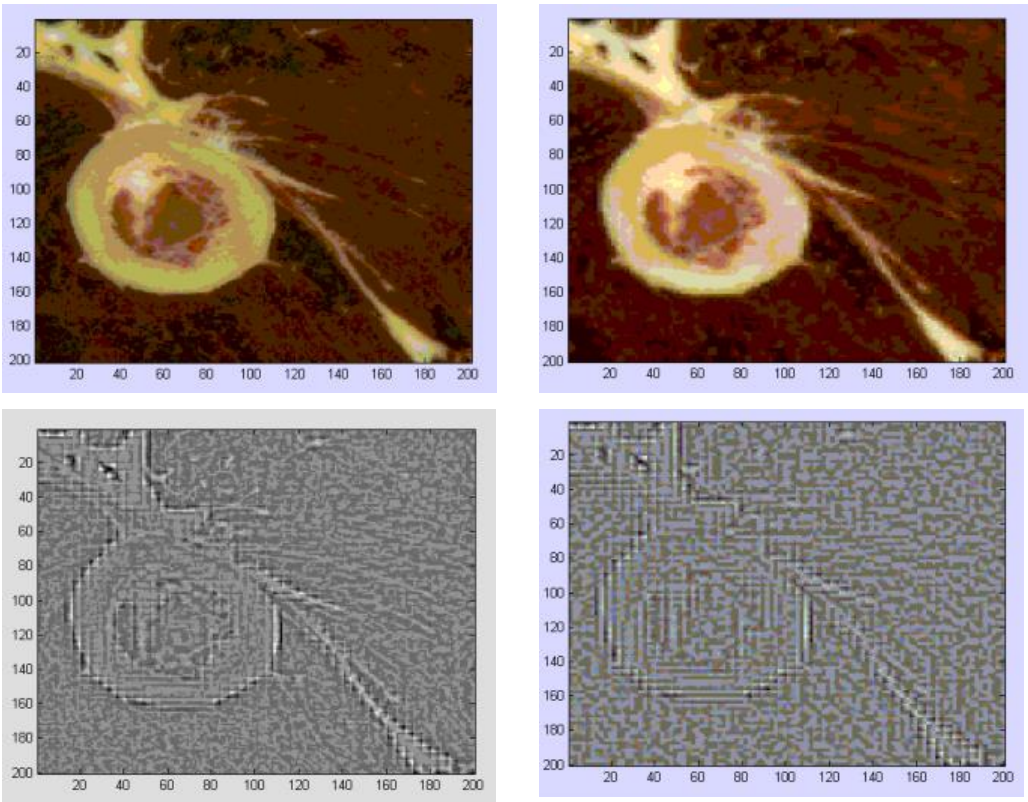


Figure 20. Original image (top left). Image with high frequency components removed, using transform D_1 (top right). Image with DC coefficient removed, using transform D_2 (bottom left). Image with both DC coefficients and high frequency components removed, using transform D_3 (bottom right).

The results show that there is a much lower impact to the reconstructed image when the higher frequency components are removed. Removing the DC coefficient results in a significant loss of quality (from the point of view of an observer), but it has to be also noted that the various textures within the image are (to some extent) still distinguishable from each other. Removing both the high frequency and DC coefficients results in the lowest quality reconstructed image, and in this case many of the various textures are no longer distinguishable. This is perhaps not surprising, considering how much information has been lost in the reconstruction of this image.

These (largely anecdotal) results suggest that it might be possible to develop an efficient technique based on a descriptor built from the more visible or invariant aspects (in terms of the distortions commonly introduced by the various medical imaging techniques being considered) of the texture in question. Any reduction in complexity or dimensionality of the descriptors being used (assuming that

enough information is still retained within the descriptors to allow effective segmentation) will result in an improved efficiency and a reduced computational complexity (and therefore cost) in using the technique.

The results also suggested that using a spatial to frequency domain transform (such as the DCT or wavelet transform) in the calculation of the descriptors could be feasible and effective. The initial research focused on creating and evaluating such descriptors based on DCT and wavelet transforms.

3.5 Development of descriptors.

Initially an investigation was carried out into the suitability of DCT and Wavelet transforms (which are commonly used as the basis of image storage and compression) for identifying different tissue types – an essential part of any automated segmentation technique.

A two stage process was created for mapping blocks into descriptors in feature space:

1) Analytical phase, transforming the block from the spatial domain into the frequency domain.
2) Transform phase, mapping some or all of the frequency domain coefficients into feature space.

As mentioned earlier, it was suspected that a good balance between quality of results and efficiency (in terms of run time) might be achieved by only using some of the frequency domain coefficients to create a descriptor in feature space.

Throughout this research two methods were used in the analytical phase; DCT and Haar transforms. These were applied as 2 or 3-dimensional transforms, as required.

In parallel, two transforms were also considered; Long and Short, which are described in more detail throughout the rest of this section.

The Long transform creates a single descriptor per block, using all of the frequency domain coefficients to do so, although it should be noted that the DC component was optionally removed in some cases to test improved invariance against intensity variation. The long transform is shown in equations 3.6 to 3.9.

$$I_{block} = \begin{bmatrix} i_{0,0}\ i_{0,1}\ i_{0,2}\ i_{0,3} \\ i_{1,0}\ i_{1,1}\ i_{1,2}\ i_{1,3} \\ i_{2,0}\ i_{2,1}\ i_{2,2}\ i_{2,3} \\ i_{3,0}\ i_{3,1}\ i_{3,2}\ i_{3,3} \end{bmatrix} \tag{3.6}$$

$$DCT(I_{block}) = \begin{bmatrix} d_{0,0}di_{0,1}d_{0,2}d_{0,3} \\ d_{1,0}d_{1,1}d_{1,2}\ d_{1,3} \\ d_{2,0}\ d_{2,1}\ d_{2,2}\ d_{2,3} \\ d_{3,0}\ d_{3,1}\ d_{3,2}\ d_{3,3} \end{bmatrix} \tag{3.7}$$

$$D_{block} = \left[d_{0,0}d_{0,1}d_{0,2}d_{0,3}d_{1,0}d_{1,1}d_{1,2}d_{1,3}d_{2,0}d_{2,1}d_{2,2}d_{2,3}d_{3,0}d_{3,1}d_{3,2}d_{3,3}\right] \tag{3.8}$$

$$D_{mblock} = \left[d_{0,1}d_{0,2}d_{0,3}d_{1,0}d_{1,1}d_{1,2}d_{1,3}d_{2,0}d_{2,1}d_{2,2}d_{2,3}d_{3,0}d_{3,1}d_{3,2}d_{3,3}\right] \tag{3.9}$$

Where:I_{block} represents the input block from the image.

$DCT(I_{block})$ represents the DCT transform is of $I_{block.}$

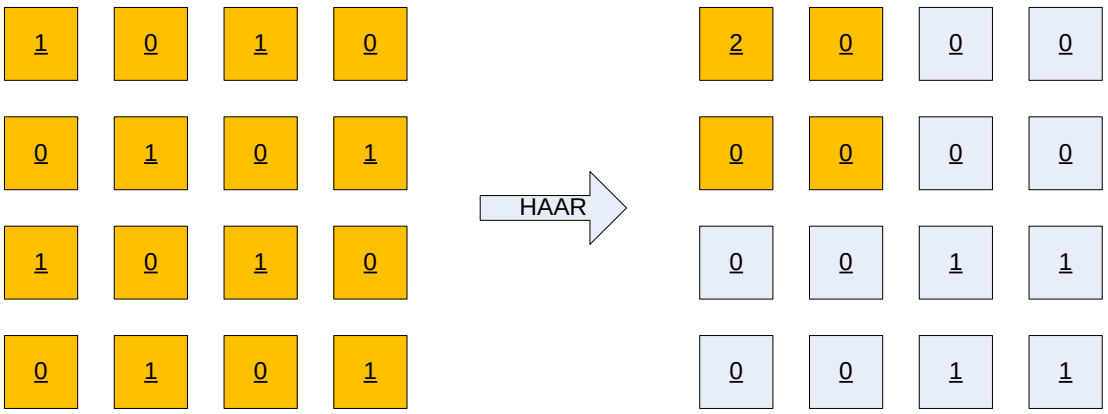
D_{block} represents the Long transform descriptor of I_{block}.

D_{mblock} represents the Long transform descriptor of I_{block} with the DC component removed.

The short transform splits a block into a number of sub-blocks, and an individual descriptor is then generated for each sub-block, using a subset of the frequency domain coefficients for each. This has the effect of increasing the resolution of segmentation (as classification can now be carried out at the sub-block level) and also reducing the dimensionality of the descriptors in feature space.

To decide which frequency domain coefficients to use in the descriptors of each sub-block, some spatial information is required. As such, the short transform is only suitable for use when the Haar transform was used in the analytical phase. This is because, the Haar transform is a wavelet transform, and so It provides not only information on what frequencies are present in a given sample, but also, where in the sample the frequencies can be found. Each coefficient within the transformed data represents a frequency component over a specific region of the data sample.

For example, the top left coefficient in the results from a Haar transform represents the DC component which is calculated from all the data in the sample. The bottom right coefficient represents the highest frequency component of the bottom right group of 2x2 pixels in the data sample. Figure 20 shows some example 4-by-4 Haar transforms, and uses colour coding to show which coefficients in the input data sample are required to calculate the value of each of the coefficients in the output data.



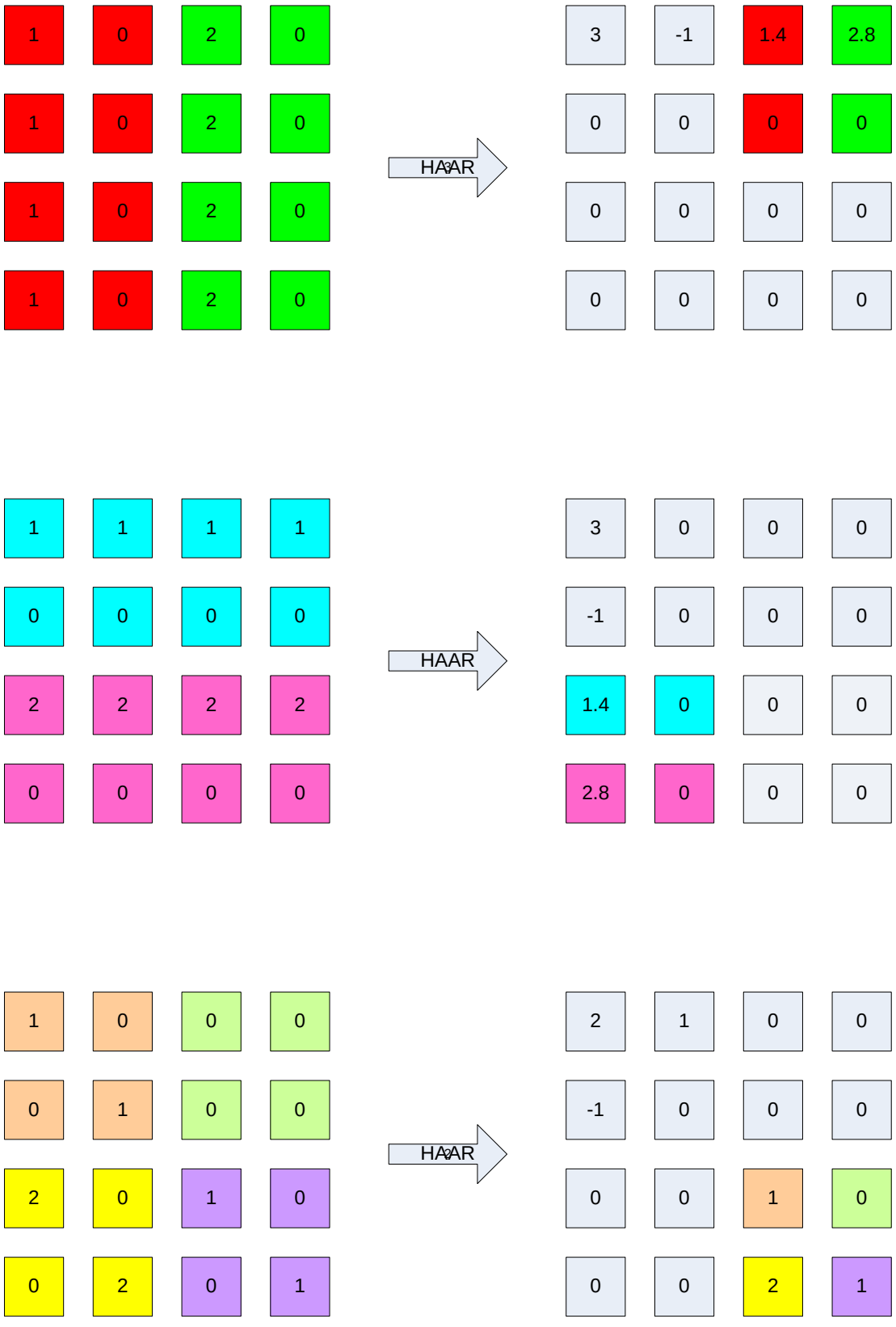


Figure 21: Mapping between output coefficients of a Haar transform, and the input coefficients that affect them.

From figure 21 it can be seen that some specific frequency domain coefficients (generated by the Haar transform) are completely unaffected by the some of the input data samples. The short transform splits the image block into a number of 2x2 sub-blocks, and while the Haar transform is still carried out at the block level (and only carried out once), an individual descriptor is created for each sub-block, and contains only those frequency domain coefficients who are affected by the values in the input sub-block.

For example, consider figure 22. Here a 4x4 block is being mapped into feature space. The figure shows the input image data on the left hand side, and the results of carrying out a Haar transform on this data, on the right hand side. As described above, the 4x4 block is split into 4 sub-blocks, of which we will consider just one, the bottom right sub-block, shown shaded in the diagram below.

The frequency domain coefficients can be grouped into 9 groups, and the descriptor for any sub-block should be made up of only a single value from each group. Note that the grouping aligns to the concepts demonstrated by Figure 16.

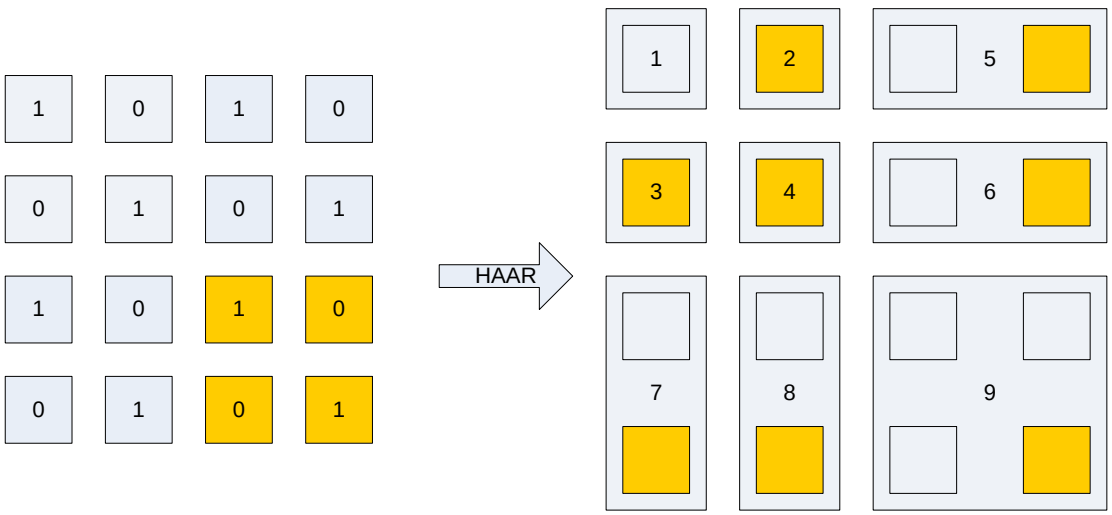


Figure 22: Mapping of Haar transform coefficients to a Haar short descriptor for a specific sub-block.

Therefore the descriptor for the sub-block under consideration is made up from the 8 shaded frequency domain coefficients and optionally the DC component

(which is shown as group 1). Note that the DC component was also discarded at some points during the research to test improved invariance against intensity variation (as was also detailed for the long transform).

More formally the short transform process (for a block size of 4x4 pixels) is shown in Equation 3.10 and 3.11.

$$I_{block} = \begin{bmatrix} i_{0,0}\ i_{0,1}\ i_{0,2}\ i_{0,3} \\ i_{1,0}\ i_{1,1}\ i_{1,2}\ i_{1,3} \\ i_{2,0}\ i_{2,1}\ i_{2,2}\ i_{2,3} \\ i_{3,0}\ i_{3,1}\ i_{3,2}\ i_{3,3} \end{bmatrix} \tag{3.10}$$

$$S_{subblock0,0} = \begin{bmatrix} i_{0,0} & i_{0,1} \\ i_{1,0} & i_{1,1} \end{bmatrix} \tag{3.11}$$

$$S_{subblock1,0} = \begin{bmatrix} i_{0,2} & i_{0,3} \\ i_{1,2} & i_{1,3} \end{bmatrix} \tag{3.12}$$

$$S_{subblock0,1} = \begin{bmatrix} i_{2,0} & i_{2,1} \\ i_{3,0} & i_{3,1} \end{bmatrix} \tag{3.13}$$

$$S_{subblock1,1} = \begin{bmatrix} i_{2,2} & i_{2,3} \\ i_{3,2} & i_{3,3} \end{bmatrix} \tag{3.14}$$

$$Haar(I_{block}) = \begin{bmatrix} h_{0,0}\ h_{0,1}\ h_{0,2}\ h_{0,3} \\ h_{1,0}\ h_{1,1}\ h_{1,2}\ h_{1,3} \\ h_{2,0}\ h_{2,1}\ h_{2,2}\ h_{2,3} \\ h_{3,0}\ h_{3,1}\ h_{3,2}\ h_{3,3} \end{bmatrix} \tag{3.15}$$

$$D_{subblock0,0} = \begin{bmatrix} h_{0,0}\ h_{0,1}\ h_{1,0}\ h_{1,1}\ h_{0,2}\ h_{1,2}\ h_{2,0}\ h_{2,1}\ h_{2,2} \end{bmatrix} \tag{3.16}$$

$$D_{subblock1,0} = \left[h_{0,0}\ h_{0,1}\ h_{1,0}\ h_{1,1}\ h_{0,3}\ h_{1,3}\ h_{2,0}\ h_{2,1}\ h_{2,3}\right] \tag{3.17}$$

$$D_{subblock0,1} = \left[h_{0,0}\ h_{0,1}\ h_{1,0}\ h_{1,1}\ h_{0,2}\ h_{1,2}\ h_{3,0}\ h_{3,1}\ h_{3,2}\right] \tag{3.18}$$

$$D_{subblock1,1} = \left[h_{0,0}\ h_{0,1}\ h_{1,0}\ h_{1,1}\ h_{0,3}\ h_{1,3}\ h_{3,0}\ h_{3,1}\ h_{3,3}\right] \tag{3.19}$$

Where:I_{block} represents the input block from the image.

S represents the 4 sub-blocks.

Haar(I_{block}) represents the Haar transform is of *I*

D represents the descriptors of each of the sub-blocks.

3.6 Implementation of technique

As previously explained, at a high level, the process can be split into 3 stages:

1) Training a GMM. In this step a GMM is trained with the features from blocks within the segmentation set.
2) Tuning the model. In this step validation data is used to adjust the thresholds being used to classify blocks with the GMM.
3) Operation (testing the model). Here test data is passed into the model to see how the model performs. The amount of test data available is increased by resampling the test data using a different offset as a starting position.

Steps 2, 3 and 4(described above) are based around the same process (to a greater or lesser extent). The process is outlined below and described in the flow chart shown in figure 23.

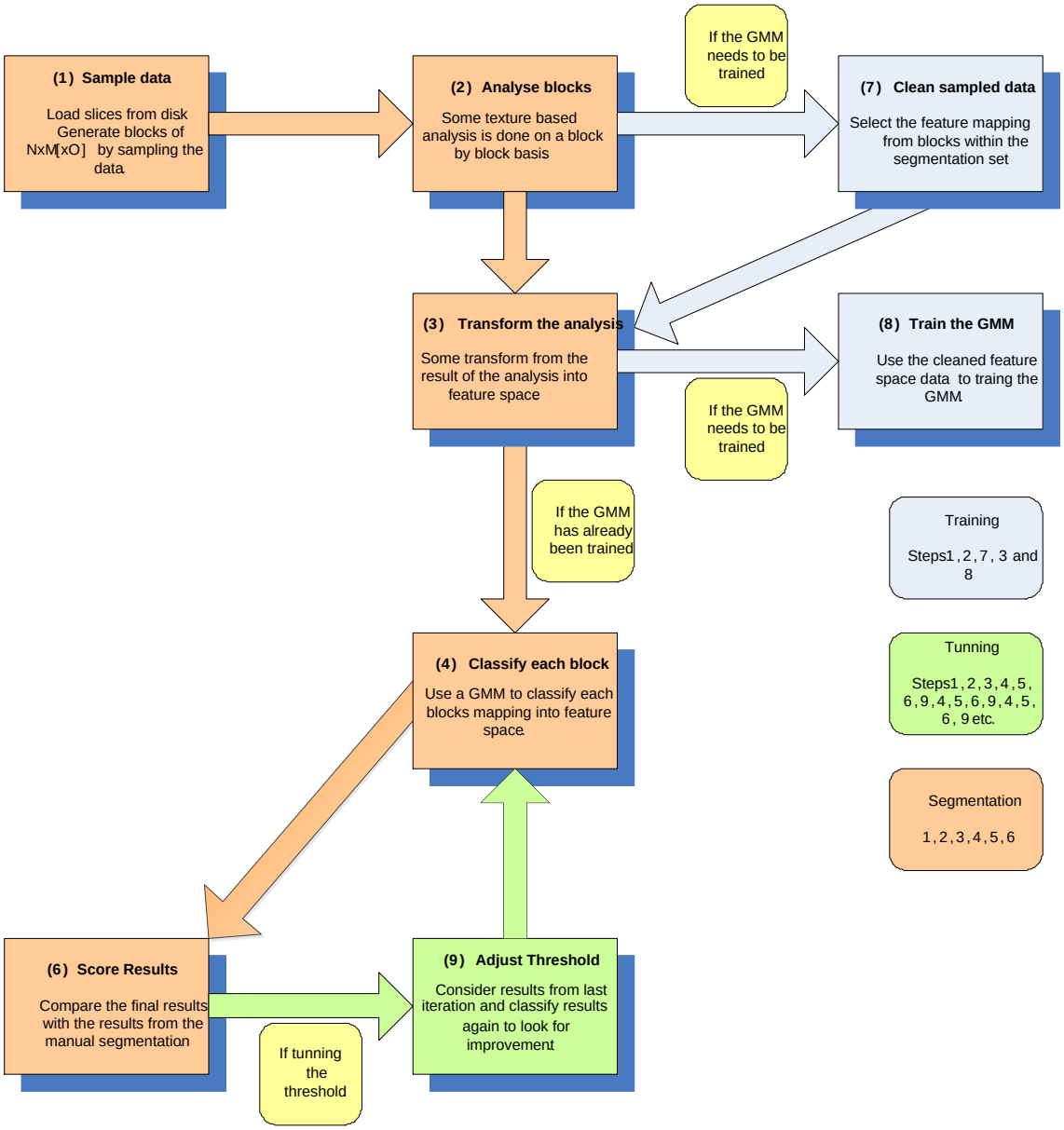


Figure 23: Flow chart showing the process used to investigate the DCT and Wavelet transforms as a measure of texture in medical images.

The training phase starts by loading in and sampling data from the dataset being used. This is step (1) in figure 23. Each slice from a dataset is stored as a separate image. The manual segmentation is also stored in the same way. Both of these are loaded in and combined into a 3D matrix of intensity values, and a 3D matrix of 1s and 0s which represents the segmentation set. Once this has been done these matrixes are then sampled. This process splits the matrixes into a number of smaller blocks, for later processing. The blocks are either 2D or

3D depending on the technique being tested, and also hold the location of the block within the dataset as this is useful later on.

To increase the amount of data generally available for the amount of manual segmentation that had to be carried out, the sampling process can be started from different offsets.

Once the data has been loaded and sampled it will then be analysed (step (2) in figure 23). The analysis that has been carried out as part of this research was using either a DCT or a Haar transform. This was carried out on a block by block basis and the result of this step is a matrix of the same size and shape as the input block.

Blocks that are not part of the target segmentation set are removed in step (7), so that the GMM can be trained in step (8).

The results of the analysis in step (2) are then transformed into feature space (step (3) of figure 13). This step is included in the process to allow some flexibility in how the results of the analysis are used to generate a feature. In its simplest form this step can consist of taking the matrix that was the result of the analysis step, and using this as co-ordinates into the feature space.

Once the GMM has been trained each feature from the segmentation set is then passed to the GMM and the result recorded, so that an initial threshold for classification can be obtained.

The training process is then complete. The next step is to tune the threshold so that the best possible results can be obtained.

This is done by performing a two phase search for the best threshold value. An initial value for the threshold is found by considering the probability generated for each of the items in the training set by the GMM. These are ranked (in order of probability) and then the probability of the feature towards the lower end of the probability scale is taken as a starting point, nominally 20% from the lower end of the training set.

The tuning process can then begin. A validation data set (including images and the target masks used for scoring) is loaded in. It is sampled, analysed and transformed, in the same way as before. This part of the process is only run once.

Next the features that have been generated are classified using the GMM and the threshold level that is currently being tested. Some post-processing can be carried out on the resulting calculated mask (as appropriate to the specific investigation being carried out), and the final results are then scored against the target mask.

An adjustment is then made to the threshold value being used (based on the scoring) and the process is run through again from the classification onwards. The search is split into two phases, the first of which is a rapid, course search.

At the start of phase one, the threshold is set to a value that will exclude 20% of the members of the segmentation set. It is therefore initially set too high, relatively large adjustments are made to reduce it until it has been observed to have been reduced too much. This decision is based on the relative percentages of incorrectly assigned ones and zeros.

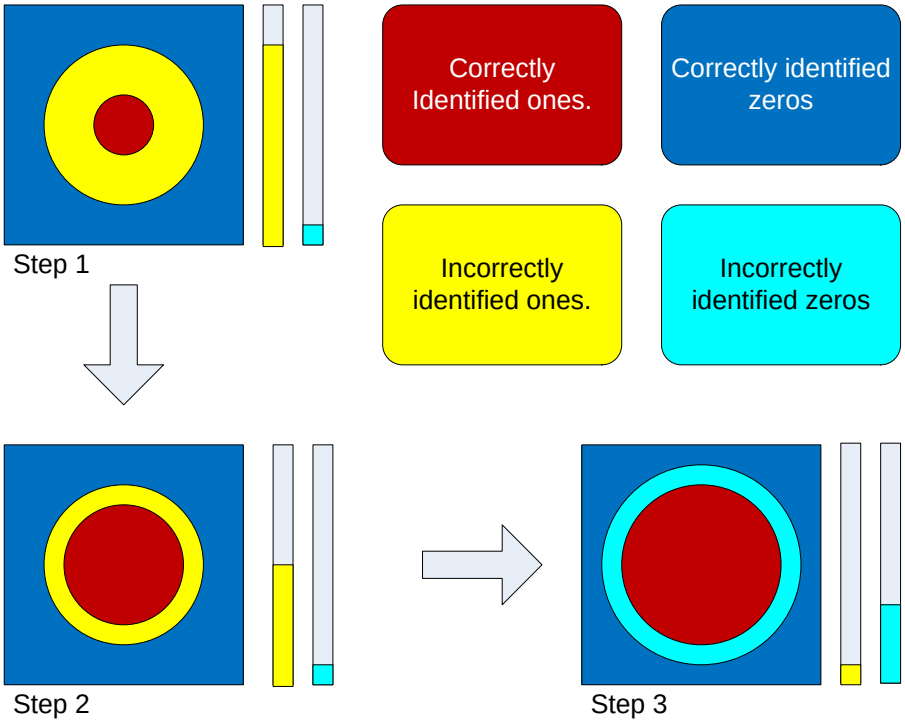


Figure 24: phase one of search.

Figure 24 shows the technique used in phase one of the search. If the percentage of incorrect zeros is smaller than the percentage of incorrect ones (as it is in Step 1 and Step 2 in the example, shown by the yellow and light blue bars), this suggests that the calculated segmentation is too small. If this is the case then not enough features are being classified as members of the segmentation set, so the threshold should be lowered. It is lowered by 55% in each step.

Once the percentage of incorrect zeros is larger than the percentage of incorrect ones, that suggests that the region that has been marked is now too big, and the threshold needs to be increased again, back towards its previous value. To do this the search enters the second phase (shown in Figure 26).

Phase one of the search is also described in the pseudo code, shown in Figure 25.

```
mix = GMM (previously configured and trained)
descriptors = descriptors of all blocks. (previously generated hash)
current_threshold = starting_threshold (previously calculated)


while(in phase one)
        clear segmentation set

        foreach this_block in all_blocks
                this_descriptor = descriptors[this_block]

// find the probability this block is a member of the segmentation set.
                prob = gmmprob(mix, d)

                if(prob > current_threshold) then
                        add block to segmentation set
                else
                        do *not* add block to segmentation set
                end if

        end for

// analyse results to calculate percent_correct, percent_incorrect_zeros and
percent_incorrect_ones.
        analyse_results()

// previous_percent_correct is used by phase two.
        previous_percent_correct = percent_correct

        if(percent_incorrect_zeros > percent_incorrect_ones) then
// phase one is now complete.
                exit while loop.
        else
// previous_threshold is used by phase two.
                previous_threshold = current_threshold
                current_threshold = current_threshold * 0.55
        end if

end while
```

Figure 25: Blocks are sampled, and descriptors calculated in advance. The starting threshold is set such that 80% of the segmentation set are above the threshold (based on the manual segmentation).

When phase one is completed, the current threshold is fed into the second phase, and used as the starting point for a more localised and more detailed search.

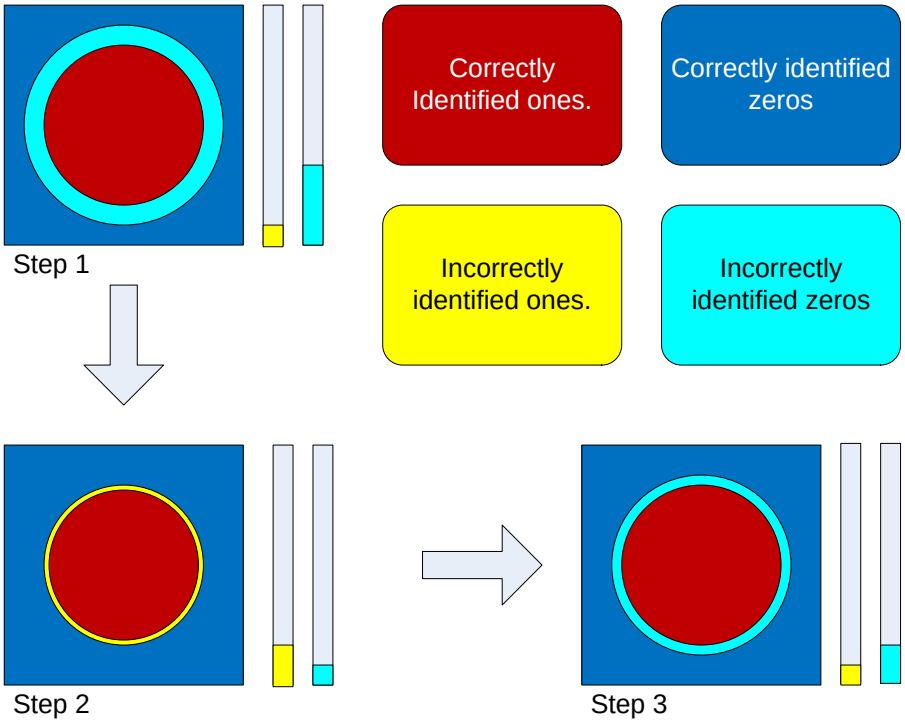


Figure 26: phase 2 of search.

In the second phase progressively smaller and smaller adjustments are made to the threshold value, and the results should steadily improve (provided that the adjustments to the threshold are made in the correct direction). Each adjustment that is made is half the size of the previous adjustment. This process continues until the results stop improving, or some timeout for this loop is reached.

The first adjustment to the threshold raises it half way back towards its previous value. This has the effect of reducing the size of the marked region. If the region is still too big then the threshold is increased some more. If the marked region is now too small then the threshold is decreased.

Figure 27 shows the method used for phase two of the search as pseudo code.

```
mix = GMM (previously configured and trained);
descriptors = descriptors of all blocks. (previously generated hash).

previous_threshold = previous_threshold set by phase one.
current_threshold = current_threshold set by phase one.
previous_percent_correct = precent_correct from phase one.

timeout = 10
iterations_run = 1

while(iterations_run < timeout)

// find half of the difference bewteen the previous threshold and the current
threshold.
        adjustment = (absolute(previous_threshold - current_threshold)) / 2

        previous_threshold = current_threshold;

        clear segmentation set.

        foreach this_block in all_blocks

                this_descriptor = descriptors[this_block]
                prob = gmmprob(mix, d)

                 if(prob > current_threshold) then
                        add block to segmentation set.
                else
                        do not add block to segmentation set.
                end if
        end for

// analyse results to calculate percent_correct, percent_incorrect_zeros and
percent_incorrect_ones.
        analyse_results()

        if(previous_precent_correct > precent_correct) then
                exit while loop
        else
                if(percent_incorrect_zeros > percent_incorrect_ones) then
                        current_threshold = current_threshold + adjustment.
                else
                        current_threshold = current_threshold - adjustment.
                end if
        end if
end while
```

Figure 27: pseudo code representation of phase two of the search.

Once the threshold has been established then a number of test sets can be passed through the process and the results generated, and stored. This whole process was then repeated using different methods of analysing the blocks in step (2), allowing the effectiveness of different techniques to be evaluated against each other.
The process that has been outlined above was used as described, with the initial aim being to investigate the performance of the various descriptors being considered (based on DCT Long, Haar Long, and Haar Short descriptors). The effect of varying the block size (and number of dimensions) was also considered at this stage.

However, the results obtained were seen to be extremely noisy, and so it was quickly realised that the process would need to be improved, before the results being obtained could have meaningful conclusions draw from them.

4.0 Adding post processing to improve the results

It was realised early on that the nature of this technique (having no prior knowledge of the anatomical structure being segmented, and being applied on a block by block basis) was to generate inherently noisy results. However, it was also seen that all six of the different transforms offered a reasonable distinction between tissues of different types. Therefore some post processing was added to the process (note the addition of step 5 to the process, as shown in figure 28). This post processing used techniques that rely on the positional relationship between blocks, and was intended to improve the quality of results such that they could be used in some simple segmentation tasks.

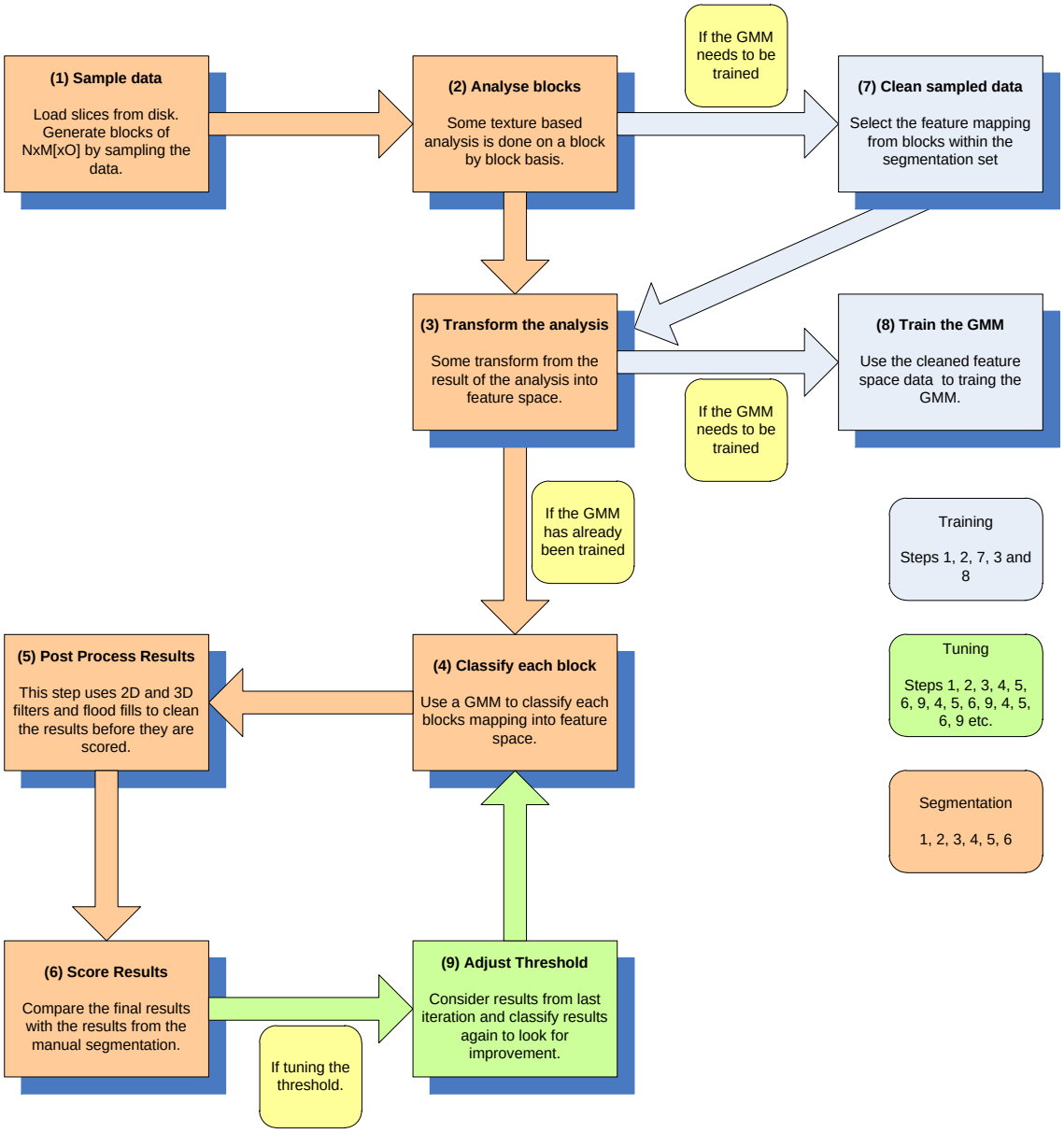


Figure 28: Flow chart showing the training, tuning and segmentation process that was used.

As previously stated, it was realised early on that the results generated by this approach will be inherently noisy. It was also noticed during informal reviews of initial results, that there was another problem. Even if the segmentation set had been reasonably well identified, it was common that remote and separate regions of similar texture were also highlighted as being part of the segmentation set. To try to combat both of these issues two post processing techniques were considered and then evaluated. These were a basic

implementation of a low pass filter (to reduce noise), and a region growing technique (to remove remote locations from the segmentation set).

4.1 Region Growing Process

Region growing approaches are well understood and well used in image segmentation. A region growing approach typically starts with one or more seed locations that are specified to be included within the region. Next, any neighbouring locations of the region are evaluated based on some pre-defined criteria for joining the region. If they satisfy the joining criteria then these locations are also added to the region and the region grows. Each time a location is added to the region any new neighbouring locations are also evaluated. The process continues in this iterative fashion until no more locations are waiting to be evaluated, and at this point the region stops growing.

The motivation for using a region growing approach is to allow remote regions to be removed from the segmentation set. It is suggested that with suitable seed locations being selected, the region would be grown until only local members of the segmentation set are included within the region, and remote members of the set can then be removed.

As mentioned earlier, there are many variations on the region growing approach, including techniques for dealing with multiple seed locations and/or multiple regions being grown simultaneously [53]. Also there is much research into efficient algorithms for implementing region growing approaches and various ways of controlling the growth of the region [22].

In this case two basic region growing approaches were implemented; strong and weak. Both the strong and the weak region growing process used a single seed location, with the assumption that the seed location was known to be a member of the segmentation set. Locations were assumed to be equally spaced on a grid system (either 2D or 3D). The criteria that new locations had to meet in order to join the region was that they had to be a neighbour of a location that already within the region, and they had to be a member of the segmentation set.

The only difference between the strong and the weak process was how neighbouring locations were defined. In the weak process a locations neighbours were defined to be all the adjacent locations. In the case of a 2D grid this would include adjacent locations in the cardinal locations (to the north, east, south and west) as well as adjacent locations in the diagonal locations (to the north-east, south-east, south-west and north-west). This definition is extended if a 3D grid is being used, so that 9 locations above and 9 locations below would also be considered neighbours. An example of a weak region growing process (using a 2D grid system) being used to remove remote locations is shown in Figure 29.

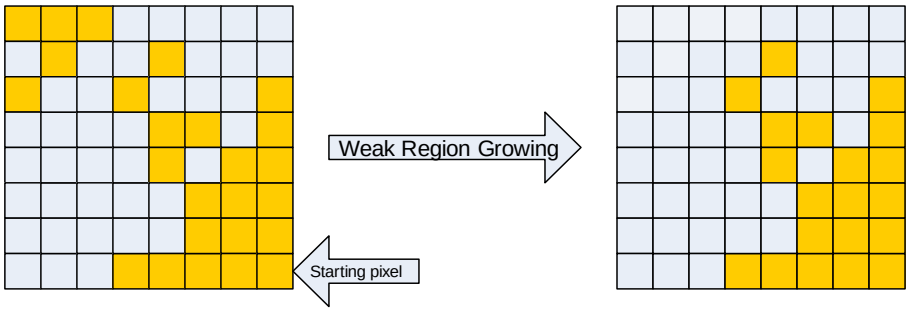


Figure 29: example of a week region growing process with the seed location marked.

In the strong process a locations neighbours were defined to be the adjacent locations in the cardinal directions only. In the case of a 2D grid this would only include adjacent locations to the north, east, south and west. Adjacent locations in the diagonal directions would not be included. This definition is extended if a 3D grid is being used, so that location directly above and the location directly below would also be considered neighbours. An example of a strong region growing process (using a 3D grid system) being used to remove remote locations is shown in Figure 30.

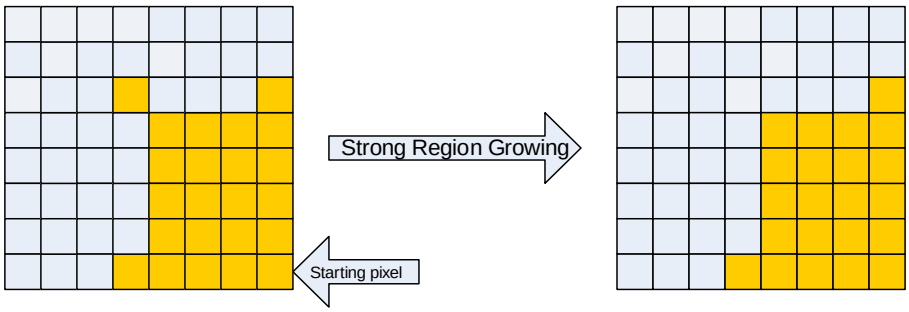


Figure 30: Example of a strong region growing process.

Note that the implementation allowed for the region growing to be carried out at the pixel or voxel level, but also at the block or sub-block level as required.

Comparing the behaviour of the weak and strong region growing processes shown in figures 29 and 30, it can be seen that the strong region growing process tends to remove single pixels (or sub-blocks or blocks if being used at these levels) at the periphery of the region. The weak region growing process will only ever result in the loss of pixels that are already truly isolated and remote form the region. It is therefore more likely that noise around the periphery of the region could result in the loss of pixels that should be retained when using a strong region growing process, than if the weak process had been used.

4.2 Low Pass Filter

A simple discrete low pass filter was also implemented with the aim of removing noise from the segmentation set. The use of a low pass filter to achieve this relied on the assumption that a location within the segmentation set was likely to be close to other locations that were also within the segmentation set. This was considered a reasonably safe assumption given that the images that were being processed were medical images, and similar types of tissues tend to be grouped together.

The implementation of the filter was simplistic, and based on adjusting the state of a location (if it was a member of the segmentation set or not) based on the state of its neighbours. Two types of low pass filter were implemented, a symmetric and an asymmetric. The basic implementation of the low pass filter was such that a location had its state changed if the number of adjacent locations that were of the opposite state was greater than a threshold. If the

threshold was the same when considering locations that were currently in the segmentation set compared against considering locations that were not in the segmentation set then the filter was termed symmetric. If this was not the case then it was termed asymmetric. There were two subtypes of asymmetric filters; those which require more energy (that is neighbouring pixels in the opposite state) to remove a location from the segmentation set than to add a location to the set, and those which do not. In this research it was decided to only consider the use of an asymmetric filter which did require more energy to remove a location from the segmentation set, than to add it. This was done to ensure useful information about the segmentation was not lost, but the noise was still removed. An example of the different output produced from a symmetric and asymmetric low pass filter is shown in figure 31.

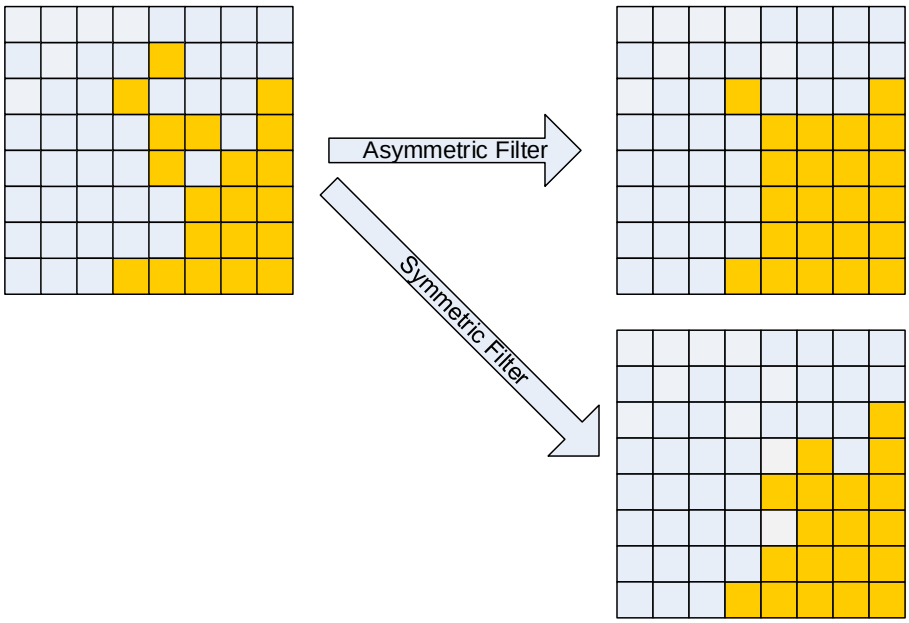


Figure 31: Example of an asymmetric and symmetric low pass filter.

Again note that the implementation of the filters allowed them to be used at various resolutions, from pixel or voxel, through sub-block to block level.

The thresholds that were used in the implementation of the filter are shown in table 11. These values were chosen after a brief investigation and some initial experimental work, however a detailed study of the effect of choosing other values was not undertaken, and may make for interesting further work.

Filter	Number of adjacent locations	Threshold to grow the segmentation set.	Threshold to shrink the segmentation set.
Symmetric 2D	8	5	5
Symmetric 3D	26	18	18
Asymmetric 2D	8	4	6
Asymmetric 3D	26	14	20

Table 11: thresholds used in low pass filters.

It can be seen in figure 31, that the filters have a smoothing effect on the original data. By informal review of the results, the filters both appear to reduce noise, and favour generating a continuous region. As might be expected, the asymmetric filter has removed fewer of the original pixels, and the symmetric filter has eroded the region more, although no more than a strong region growing process would have done. In fact, the symmetric filter has increased the size of the region, which has helped to generate a smoother boundary.

The input data used in Figure 31 had already been passed through the weak region growing process, to remove isolated pixels. It is noted that using either of the filters when there are isolated pixels in the vicinity of the main region could lead to bridges being formed to the isolated pixels.

4.3 Discussion of initial results.

The segmentation was initially tested using 20 slices from the “Upper Arm Bone” set. 20 slices were used for training and validation purposes and 20 for test. As can be seen from the data in table 6, the slices are 201x201 pixels in size, and although the slices are available in full colour, they were converted to monochrome before being used.

Initially the framework did not include any filtering and the only post processing carried out was a single strong region growing process, used to remove remote members of the segmentation set. The initial results were used to allow 2D and 3D, DCT and Haar-based approaches, of differing sampling resolutions to be compared.to be compared with each other.

Figures 32, 33 and 34 show the results of a slice that had been segmented by a 2D DCT analysis with a sampling resolutions of 2x2x1, 4x4x1 and 8x8x1.

Figures 35, 36 and 37 show the same slice as it was segmented using a 2D Haar analysis with a block size of 2x2x1, 4x4x1 and 8x8x1.

Looking at the 2D results, it can be seen that a reasonable segmentation was achieved with all of them, although there is some disruption at the top and at the bottom of the segmentation on all the figures.

Using a sampling size of 8x8 on both the DCT and the Haar based techniques does not give such a good subjective result as using a smaller sampling size. However by considering the calculated results from the segmentation (shown in table 12) it can be seen that the results are reasonably consistent, although at larger sampling sizes the percentage of incorrect voxels is increased. Note that the results shown in table 10 are the mean results from performing the segmentation over the 15 slices.

Analysis	**Sampling Size**	**Number of voxels per slice**	**Correct ones %**	**Correct zeros %**	**Correct voxels %**
DCT 2D	2x2x1	40000	84.9	98.8	95.7
DCT 2D	4x4x1	40000	86. 8	98.1	95.6
DCT 2D	8x8x1	40000	83.0	98.9	95.4
Haar 2D	2x2x1	40000	85.7	98.7	95.8
Haar 2D	4x4x1	40401	88.7	97.6	95.6
Haar 2D	8x8x1	40401	87.6	97.2	95.0

Table 12: Comparison of results between different transforms.

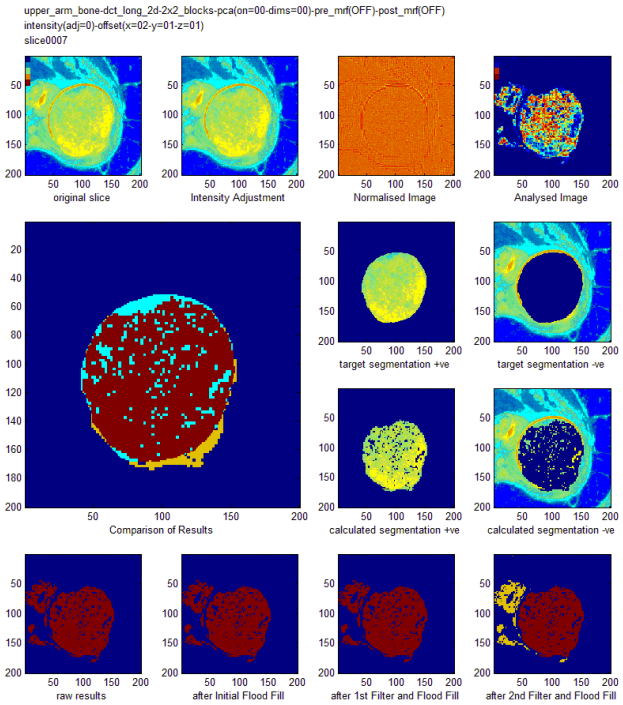


Figure 32: Results using 2D 2x2x1 DCT.

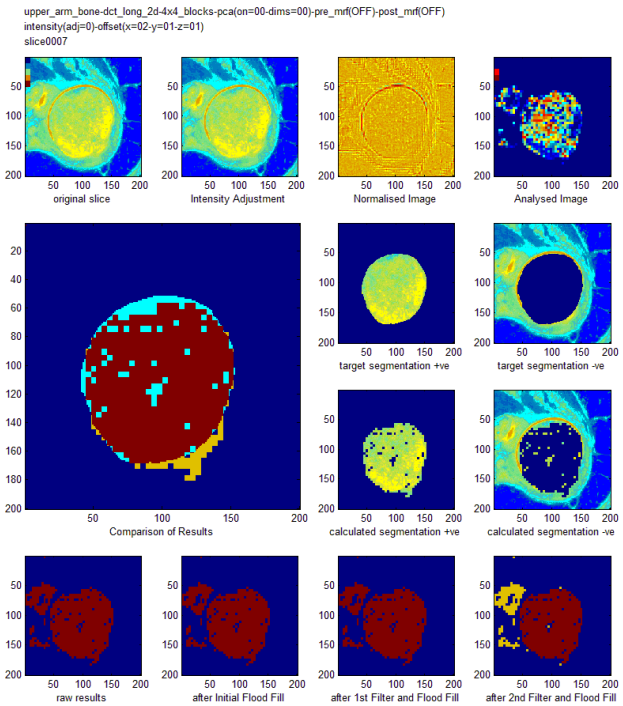


Figure 33: Results using 2D 4x4x1 DCT.

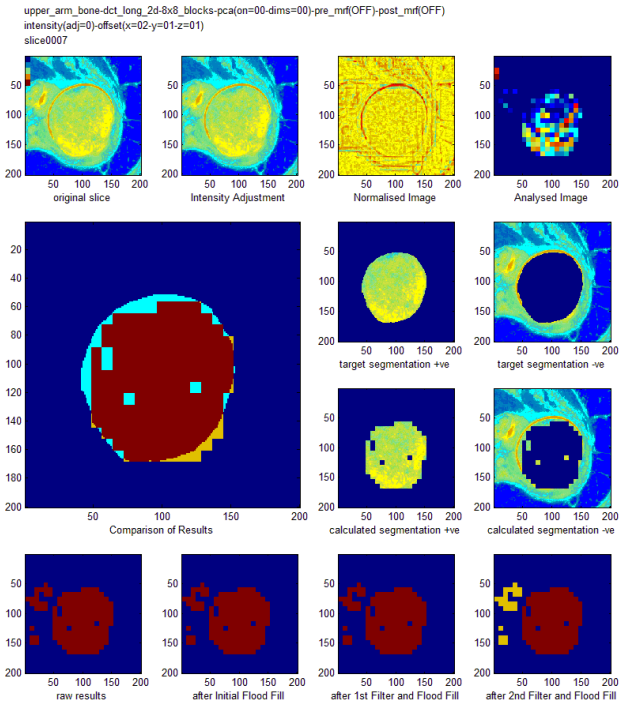


Figure 34: Results using 2D 8x8x1 DCT.

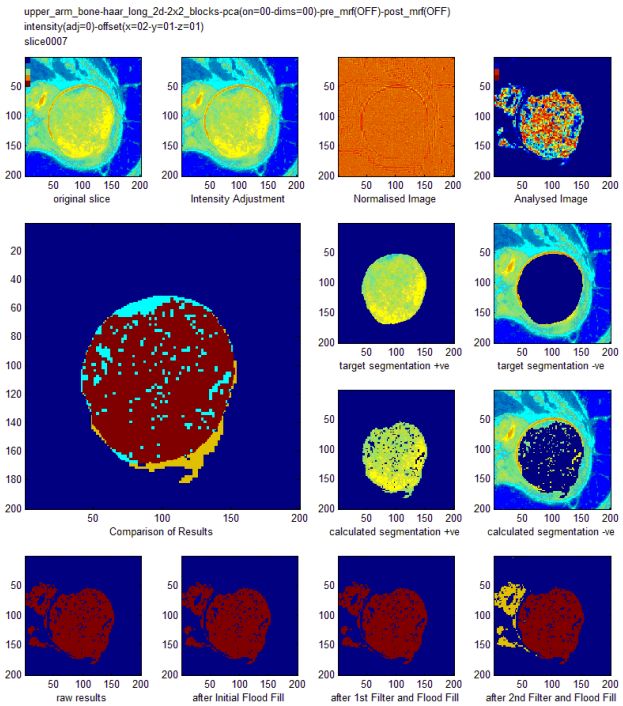


Figure 35: Results using 2D 2x2x1 Haar

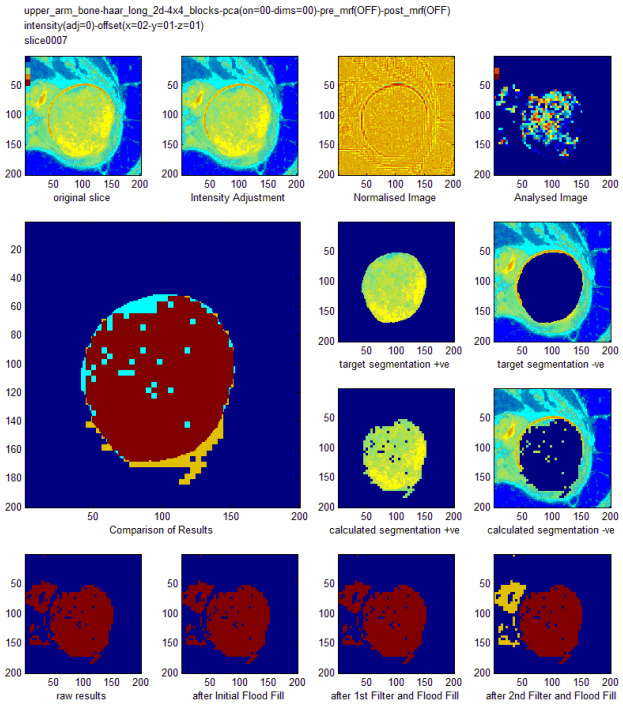


Figure 36: Results using 2D 4x4x1 Haar.

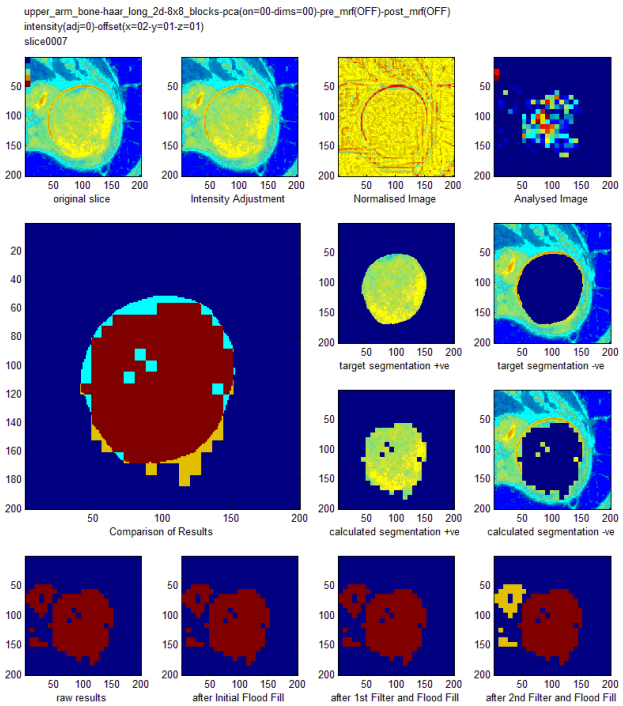


Figure 37: Results using 2D 8x8x1 Haar

Next, the results of using the same DCT and Haar analysis (but this time using a 3D transform) were collected. The sampling resolutions that were used were 4x4x2, 4x4x4, and 8x8x2. A single slice from the segmentation is again shown in Figures 38 to 43.

Considering the results of the initial work on the 3D analysis, it can be seen that again the larger sampling size of 8x8x2 was subjectively the worst performer. The lowest sampling size of 4x4x2 gave subjectively good, albeit, noisy results. And noise was significantly reduced when using a 4x4x4 sampling size (Around 10% of voxels within the segmentation set were incorrectly classified when using a block size of 4x4x2, as against around 0.5% of voxels when using a block size of 4x4x4). The calculated results are shown in table 13. A sampling size of 8x8x2 was shown to be the least effective, and at this stage the 3D results were generally worse than the 2D results.

Analysis	Sampling Size	Number of voxels per volume	Correct ones %	Correct zeros %	Correct %
DCT 3D	4x4x2	400000	83.0	98.6	95.1
DCT 3D	4x4x4	400000	82.0	98.8	95.4
DCT 3D	8x8x2	400000	68.5	99.4	92.5
Haar 3D	4x4x2	400000	83.2	98.5	95.1
Haar 3D	4x4x4	400000	79.6	97.5	94.0
Haar 3D	8x8x2	400000	73.9	99.4	93.7

Table 13: Comparison of sampling size against transform

From the initial results it seemed that there might be a correlation between using a high resolution and getting good results. However, it remained to be seen what throughput could be achieved when using a high sampling resolution. On the one hand, the small block size meant that the feature space was much reduced. If the feature space still contained enough useful information to do an effective classification then the computational complexity of such a classification would be

reduced. Using a 2x2x1 block size only produces 4 dimensions in feature space. Whereas using an 8x8x1 block size produces 64 dimensions in feature space. On the other hand, there would be 16 times as many features to be processed in a 2x2x1 setup over an 8x8x1 setup. Table 14 shows some measurements of run time that were captured while collecting the initial results.

Block Dimensions	Feature dimensions	Analysis	Total pixels	Total blocks	Training time (seconds)	Classification time (seconds)
2x2x1	4	DCT	400000	100000	46.8	84.3
4x4x1	16	DCT	400000	25000	11.4	30.7
8x8x1	64	DCT	400000	6250	3.7	13.5
2x2x1	4	HAAR	400000	100000	43.6	80.5
4x4x1	16	HAAR	400000	25000	11.4	31.1
8x8x1	64	HAAR	400000	6250	4.0	15.4
4x4x2	32	DCT	400000	12500	13.3	52.5
8x8x2	128	DCT	400000	3125	8.4	53.2
4x4x4	64	DCT	320000	5000	5.6	33.3
4x4x2	32	HAAR	400000	12500	10.7	44.0
8x8x2	128	HAAR	400000	3125	8.4	43.4
4x4x4	64	HAAR	320000	5000	5.0	28.8

Table 14: Comparison of sampling size against feature space dimensions and resulting run time.

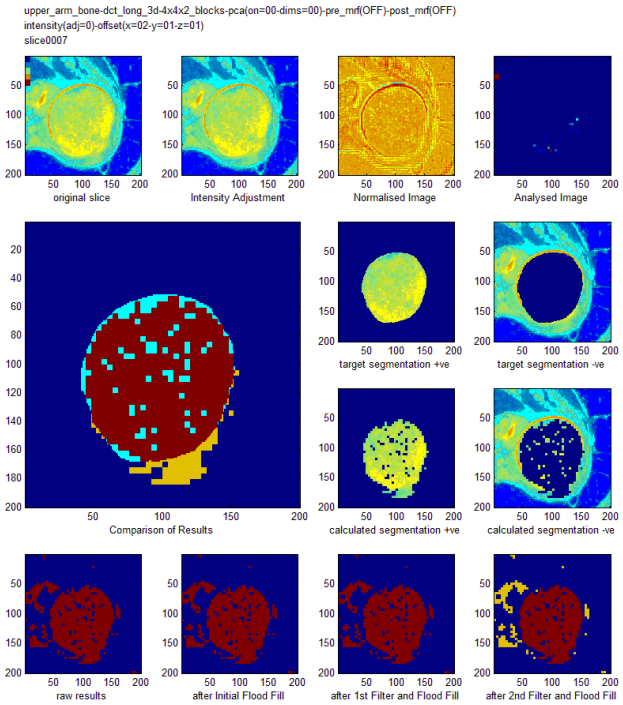


Figure 38: results using 3D 4x4x2 DCT.

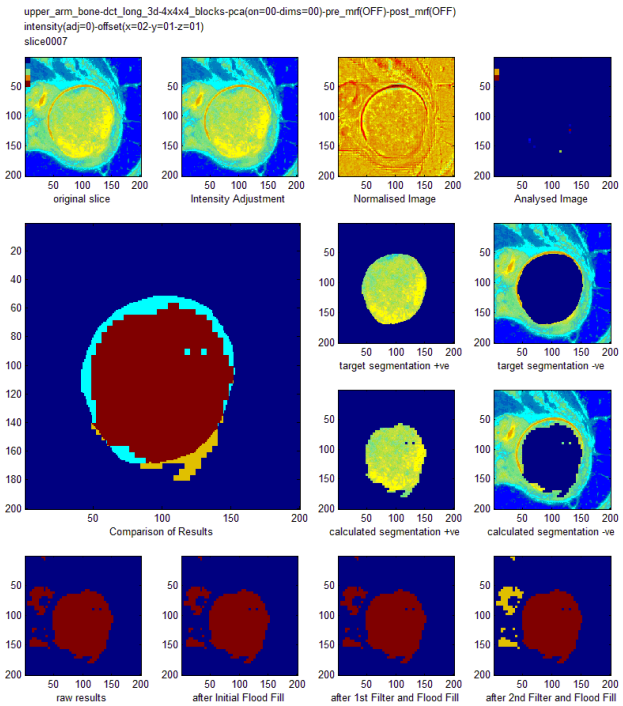


Figure 39: Results using 3D 4x4x4 DCT.

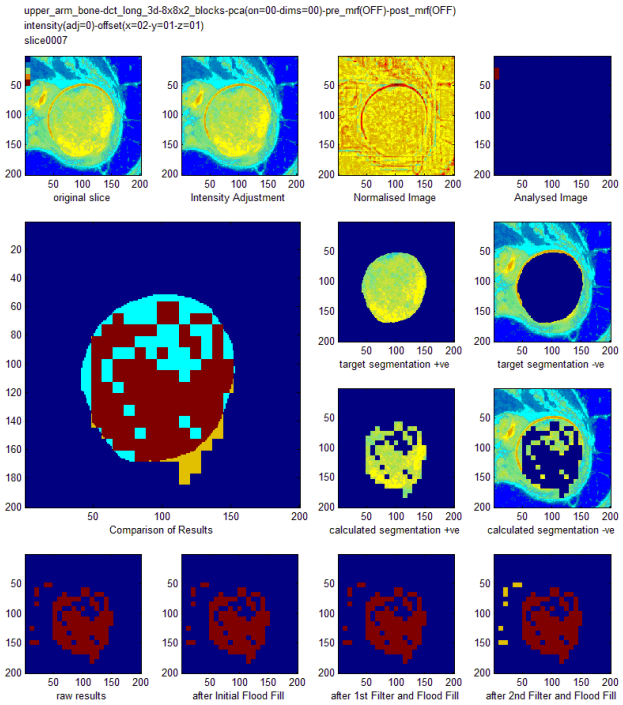


Figure 40: Results using 3D 8x8x2 DCT.

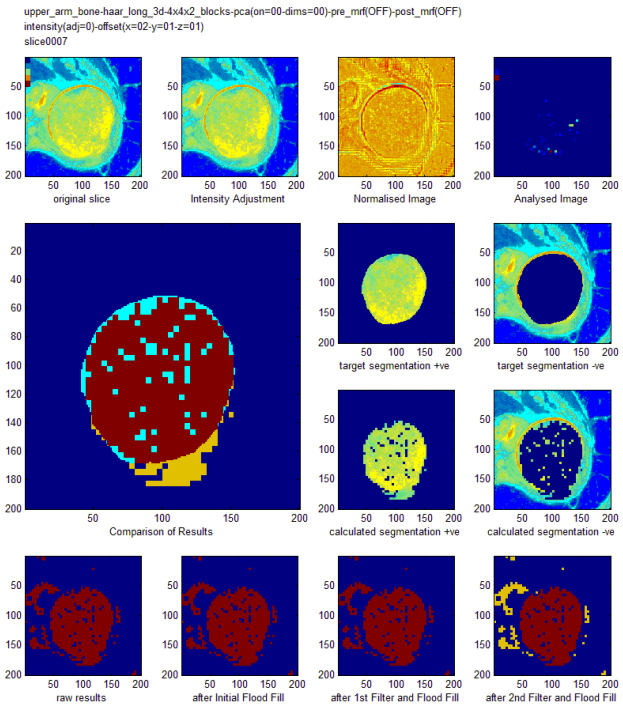


Figure 41: Results using 3D 4x4x2 Haar.

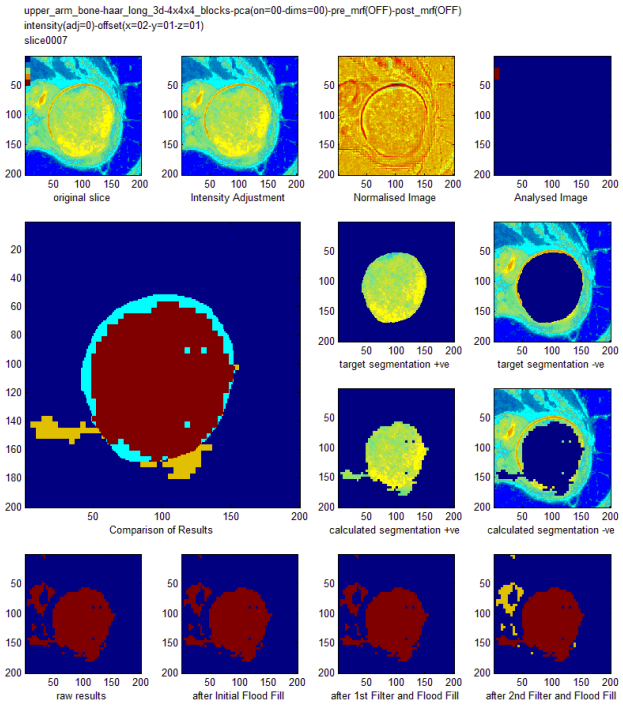


Figure 42: Results using 3D 4x4x4 Haar.

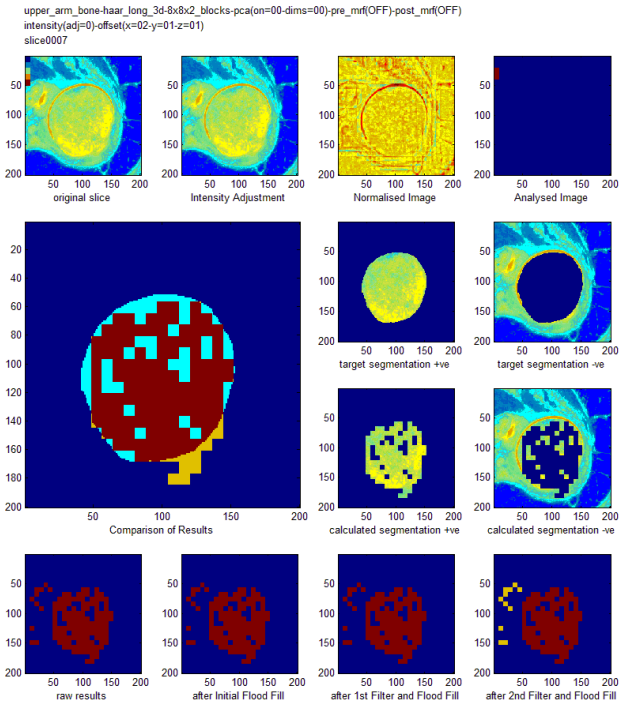


Figure 43: Results using a 3D 8x8x2 Haar

Table 14 shows the amount of time spent training the model and the amount of time taken to use the trained model to analyse and classify some new test dataset. The times were generated by using the tic and toc functions in Matlab on a Windows Vista based machine, and as such should be taken as a guide only. This is because it was difficult to reliably know what percentage of the time the machine had spent executing the foreground task rather than the background task. Also the background task loading may have been dynamically changing throughout the time the training and classification was taking place.

However, it can be seen from Table 14 that there was reasonable agreement in the trends shown between the Haar and DCT-based routines for the length of time spent in classification and training. This suggests that the choice between using a Haar or DCT transform is likely to have little impact on overall run time.

In a real life application, the training process will be a one-off process, so the length of time that this part of the process takes is perhaps less relevant than the classification process, which will be run many times over new data sets. However, it was not fully understood why the training process appeared to complete more quickly than the classification process. This is an area that would be interesting to investigate in the future.

Considering the initial 2D results it was seen that the number of blocks being processed created the biggest influence over the classification time. Therefore using a block size of 8x8x1 produced much shorter run times than seen when using a block size of 4x4x1 or 2x2x1. For 2D data, slice-by-slice analysis it seemed that was therefore a direct trade-off between throughput, and quality, as the most accurate results were obtained with the smallest block size, but this had by far the greatest run time.

Considering the initial 3D results, it was observed that the link between the number of blocks and the length of time it took to classify data was not so clear. One of the factors here was the amount of time it took to do a multi-dimensional Haar or DCT transform. If it is considered that carrying out a 1D Haar or DCT transform is a single operation, then the number of operations that it takes (which can be taken as to a guide for the amount of time that the analysis takes)

can be calculated, as shown in equations 4.1 and 4.2 and calculated for the various block sizes (for reference) in table 15.

$$N_{2D} = xy \tag{4.1}$$

Where:N_{2D} is the number of operations required for a 2D block.

x is the sample width of the block.

y is the sample height of the block.

$$N_{3D} = xy + yz + xz \tag{4.2}$$

Where:N_{3D} is the number of operations required for a 3D block.

x is the sample width of the block.

y is the sample height of the block.

z is the sample depth of the block.

Block dimensions	Number of operations to complete a Haar or DCT transform.
2x2x1	4
4x4x1	8
8x8x1	16
4x4x2	32
4x4x4	48
8x8x2	96

Table 15: Summary of number of operations required to carry out different sized transforms.

From the results it could be seen that using a 4x4x4 block size would result in a significantly reduced run time over a block size of 4x4x2 or 8x8x2 (between 35% and 38% reduction).

At this point some work was carried out into developing the Haar short transform (a novel transform which has been described in more detail in section 3.5).

Using the Haar short transform, the block size being used remains larger, but once the Haar transform has been carried out over the whole block, the block is then broken down into sub-blocks (of either 2x2x2 or 2x2x1 voxels in size) and each sub-block is given its own individual vector into feature space. The features for the sub-block are constructed by taking only a subset of the coefficients from the Haar transform of the whole block, as described by equations 3.16 to 3.19 (for a 4x4 block size). This has the side effect of reducing the dimensionality of the feature space being used. The motivation for developing this transform was to take advantage of the higher quality of results that had been achieved when using smaller block sizes, without increasing the number of Haar transforms that were required.

For example, consider different ways of analysing 16 voxels in a 4x4x1 block. It takes 8 Haar transforms to completely transform a 4x4x1 block. If this same number of voxels was to be analysed using a 2x2x1 block size, then each of the four smaller blocks would require 4 Haar transforms. This leads to 16 transforms overall. However, using the Haar short transform, the Haar transform is performed on the whole 4x4x1 block, and the results from this are used to generate four separate vectors into feature space for each of the 4 sub-blocks.

The results achieved with the Haar short transform for both 2D and 3D were found to be as shown in table 16. Generally the results show a lower quality segmentation was achieved when compared against a segmentation based on a straightforward DCT or Haar transform.

Analysis Haar Short	**Sampling Size**	**Number of voxels per volume**	**Correct ones %**	**Correct zeros %**	**Correct voxels %**
2D	2x2x1	400000	46.1	90.0	76.3
2D	4x4x1	400000	80.7	91.7	88.2
2D	8x8x1	400000	70.9	98.8	90.1
3D	4x4x2	400000	83.8	93.3	90.4
3D	4x4x4	400000	78.6	97.7	92.6
3D	8x8x2	320000	81.0	98.4	92.7

Table 16: Summary of results using the Haar Short transform (2D and 3D).

There is also an overhead with using the Haar Short transform, as the transformation from the result of the Haar transform into feature space is no longer a simple operation. Therefore the performance (as shown in table 17) was seen to be lower than using the Haar long and DCT long transforms that were previously used.

Block Dimensions	Feature dimensions	Analysis	Total pixels	Total blocks	Training time	Classification time
2x2x1	4	HAAR	760000	190000	165.6	169.8
4x4x1	9	HAAR	760000	47500	208.6	74.2
8x8x1	16	HAAR	760000	11875	290.6	44.4
4x4x2	18	HAAR	720000	22500	99.3	71.0
4x4x4	27	HAAR	640000	10000	84.1	49.9
8x8x2	32	HAAR	720000	5625	130.3	49.4

Table 17: Observed run times from various Haar Short transforms.

On an initial viewing of the results this Haar short transform could seem to underperform the DCT long or Haar long transform in every respect. However, when considering the slices once they have been processed by the Haar short transform, it can be seen that there may be some subjective benefits to this approach. Two slices from the upper arm bone data set are shown in figure 44. It can be seen that generally the level of noise is lower than has been seen with the DCT and Haar long transform, and in fact some slices are segmented with a very good degree of accuracy. However it was found that the overall results were held back by the segmentation of a few slices which suffered from 'leakage'. This is where the segmentation is almost successful, however a small bridge was formed between the required segmentation and a close by region of similar texture. If the leakage issue could be resolved then this technique might prove to provide good results.

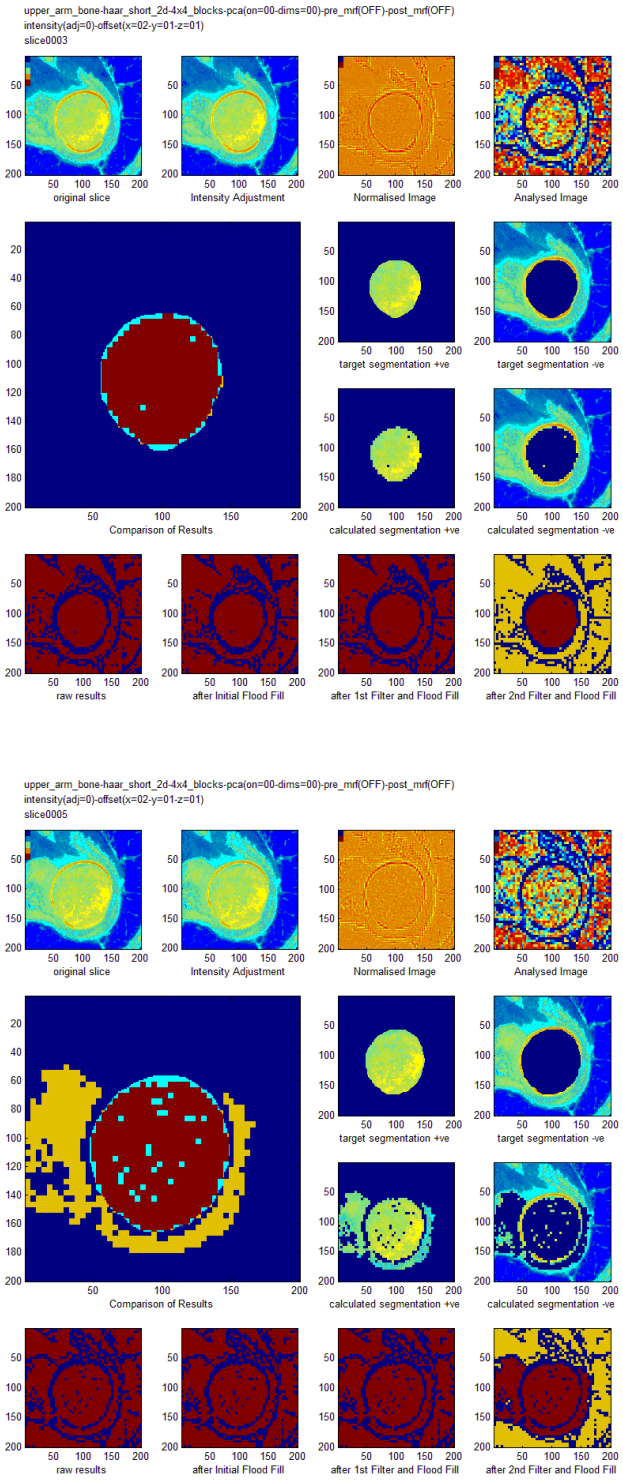


Figure 44: Two slices processed with 2D 4x4 Haar Short transform.

It could be seen from the results presented so far that noise and leakage cause the main issues with the segmentations that had been carried out, and so the research next focussed on ways of overcoming these limitations. Firstly the

issue of noise was considered. So far the only post processing that had been applied to the results was the use of a strong region growing process. The effectiveness of also performing a low pass filter on the raw results (before the region growing was carried out) was investigated.

4.4 Investigation reducing noise by use of a low pass filter.

As described in section 4.2, there were a number of possible ways of applying the low pass filter that had been implemented. The filter that had been implemented was initially developed to run at the pixel or voxel level (considering each pixel or voxel as a separate location that could either be included or excluded from the segmentation set as required). However, it was realised that it might be useful to be able to apply the filter to blocks. To do this the filter was implemented so that it would consider blocks to be atomic. Each location considered by the filter was actually a block, made up of a group of pixels. All pixels or voxels within a block are considered to have the same state, and the state of the whole block can be changed by the filter depending on the state of the neighbouring blocks. The filter was also able to operate at the sub-block level.

Therefore, the first question that had to be answered was what resolution should the low pass filter be run at.

Secondly the filter could be used in a 2D mode (i.e. on each slice independently) or in 3D mode (i.e. considering the state of the locations on adjacent slices as well as on the local slice). There will be different computational overheads based on using a 2D filter on each slice against using a 3D filter on the whole data set.

To investigate these effects and the amount of improvement that could be achieved a number of results were collected. Initially the research focused on removing noise from the segmentation. To do this successfully a filter run at the block level resolution was used. Results were collected for each of the 6 transforms, with either no filter, a 2D filter or a 3D filter being used. The results are shown in table 18.

Transform	Block Size	No Filter % Correct	2D LPF % Correct	3D LPF % Correct	2D LPF % increase	3D LPF % increase
DCT Long 2D	2x2x1	93.8	95.3	94.5	1.4	0.7
DCT Long 2D	4x4x1	95.4	95.7	96.2	0.3	0.8
DCT Long 2D	8x8x1	94.7	95.0	95.5	0.3	0.7
DCT Long 3D	4x4x2	90.3	95.6	95.6	5.3	5.3
DCT Long 3D	8x8x2	94.0	93.9	95.2	-0.2	1.1
DCT Long 3D	4x4x4	92.6	94.5	93.3	1.9	0.8
HAAR Long 2D	2x2x1	93.8	95.6	94.5	1.8	0.7
HAAR Long 2D	4x4x1	95.4	95.7	96.2	0.4	0.9
HAAR Long 2D	8x8x1	94.2	94.8	94.9	0.6	0.7
HAAR Long 3D	4x4x2	90.3	95.5	95.6	5.3	5.3
HAAR Long 3D	8x8x2	94.2	93.9	95.1	-0.3	0.9
HAAR Long 3D	4x4x4	92.3	94.8	93.3	2.5	1.0
HAAR Short 2D	2x2x1	76.3	92.3	93.8	16.0	17.6
HAAR Short 2D	4x4x1	88.2	89.0	93.4	0.8	5.1
HAAR Short 2D	8x8x1	90.1	80.8	87.7	-9.3	-2.4
HAAR Short 3D	4x4x2	90.4	95.5	95.5	5.0	5.1
HAAR Short 3D	8x8x2	92.6	94.0	94.6	1.3	2.0
HAAR Short 3D	4x4x4	92.7	94.2	93.5	1.6	0.8

Table 18: Observations when different filters are used.

The experiments carried out to obtain the results displayed in table 18 again used training, validation and test data from the “Upper Arm Bone” set. The final strong region growing process was still used. Each experiment was carried out three times. Once as before, once including a 2D low pass filter run at the block level resolution, and once including a 3D low pass filter run at the block level.

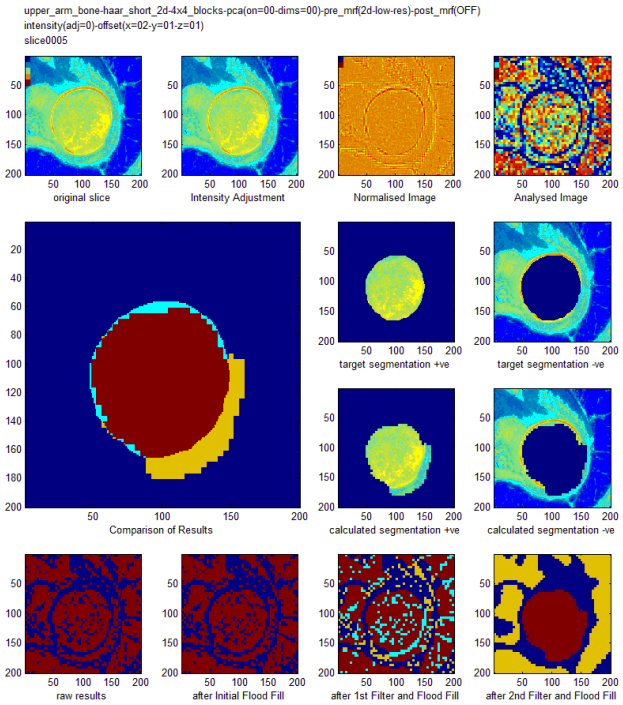
It was seen from the data presented in table 18 that both a 2D and a 3D block level low pass filter applied to the segmentation generally improves the results, in some cases by over 17%.

It is also important to consider the subjective quality of the results once the filters have been applied to the segmentation. In figure 45 two slices can be seen. The segmentation results of the first slice shown in the figure have been de-noised well. The filter was able to de-noise the segmentation, without creating any connecting bridges to local areas of similar texture. The second slice again shows that the filter was able to remove the noise successfully. However, in this

case it can also be seen that the filter has caused a bridge to a local area of similar texture, which has meant that the final segmentation of this slice is poor.

Now consider figure 46. This shows the same two slices as in figure 45. This time a 3D low pass filter had been applied to the segmentation, and at first glance it seems that the results are much better. The second slice in particular has not suffered from the bridging that the 2D low pass filter generated. This is because the low pass filter that has been applied is now working over the whole 3D dataset, and the voxels on the second slice are now filtered taking account of not just the neighbouring voxels on the same slice, but also the voxels on the sliced above and below. In this case the bridging did not occur on the slices above and below the slice in question, and so it is easily removed by using a 3D filter.

However, using a 3D filter means that the resolution of the results is greatly reduced, particularly along the z-axis. This is because the z resolution is poor compared to the x and y-axis resolution, but if a block level 3D filter is used then we are constraining each group of 4 adjacent slices to have the same segmentation.



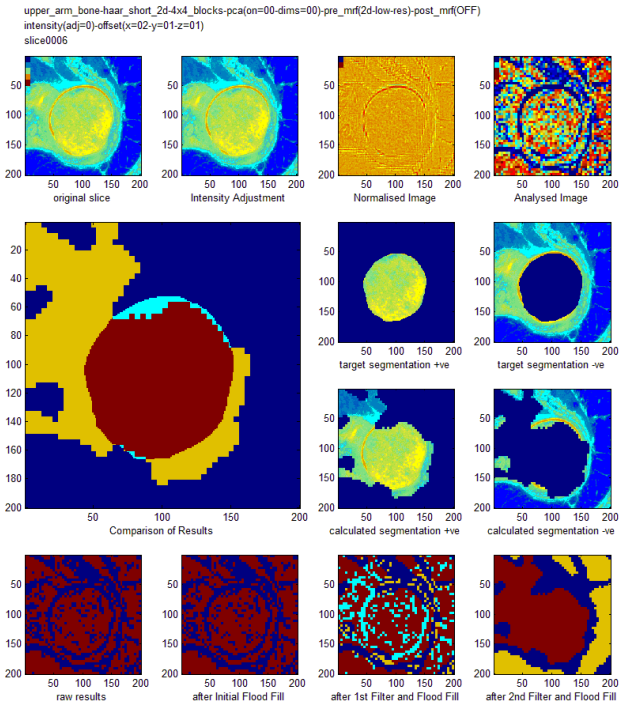
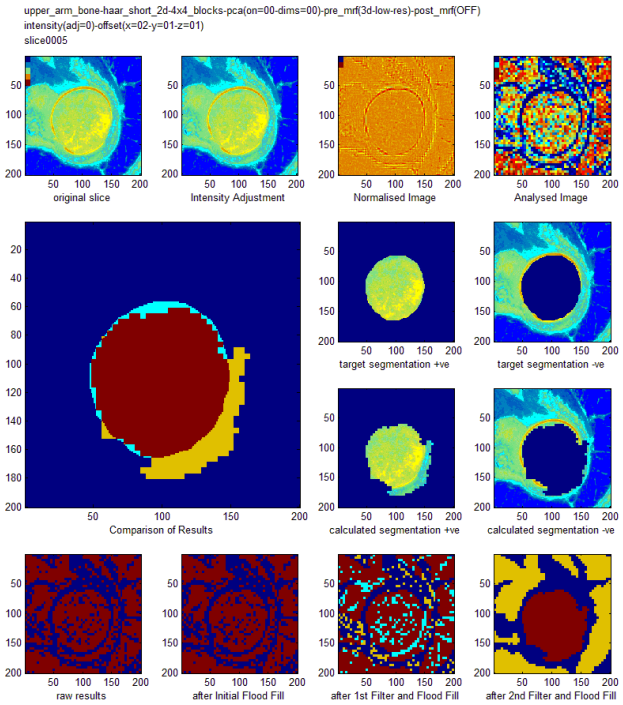


Figure 45: Two slices processed with 2D 4x4 Haar short transform, with a 2D low pass filter applied to raw segmentation results, before a strong region growing process was applied.



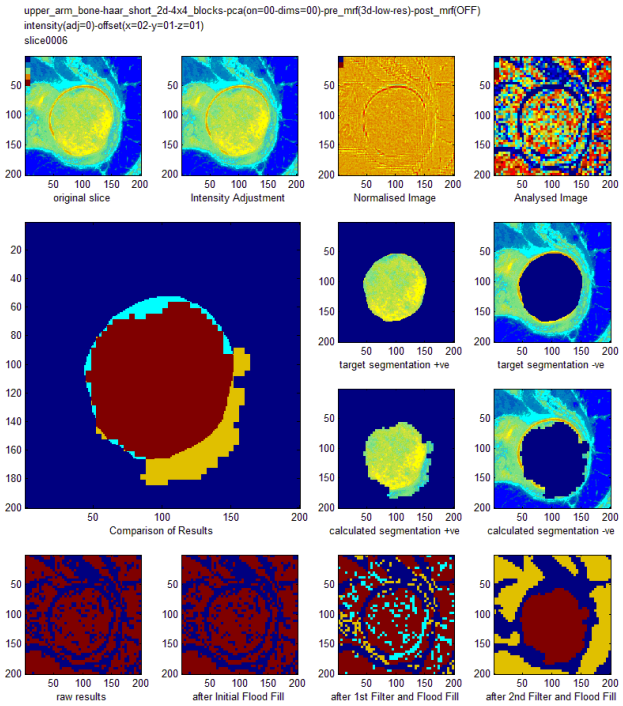


Figure 46: Two slices processed with 2D 4x4 Haar short transform, with a 3D low pass filter applied to raw segmentation results, before a strong region growing process was applied.

From the results obtained above, it seems clear that to address the noise problem, a filter using a block level resolution is effective. However, this can also increase instances of leakage, which is another issue that needs to be addressed. If a 3D filter is used then the leakage is improved. However, the results become quantised to the 3D resolution of the low pass filter, which reduces accuracy.

These results directed the research to consider that a combination of a 2D and 3D filter might give better results. Therefore another set of segmentations was performed. However, this time a combination of filters was applied during the post processing phase. First a 2D block level low pass filter was applied, followed by a region growing process (to remove any remote members of the segmentation set). Next a voxel level resolution 3D low pass filter was applied. Finally a strong region growing process would be carried out (as has always been the case). The aim here was to get the benefits of the noise reduction by using the 2D filter, while also reducing the effects of leakage, without reducing the resolution of the final results.

The results that were obtained are shown in table 19. It was clear that this approach had improved the success of the segmentation, but it is again worth reviewing some of the slices individually to ensure that the combination of the filters has had the predicted effect. Figures 47, 48, 49 and 50 show slices 4 through 7 respectively. Leakage was previously visible on slice 6 without the use of a 3D high resolution low pass filter. It can be seen that again, slice 6 is the only slice where the use of a 2D block level resolution filter resulted in a bridge being formed to a remote region (towards the top left of the slice). However, this is resolved by the subsequent application of the 3D low pass filter. It can also be seen that as the 3D low pass filter is now being used at a high resolution, each adjacent slice is now able to have its own segmentation.

Transform	Block Size	No Filter % correct	2D % correct	2D + 3D % correct	% increase over 2D	% overall increase
DCT Long 2D	2x2x1	93.8	95.3	94.6	-0.7	0.8
DCT Long 2D	4x4x1	95.4	95.7	96.2	0.3	0.8
DCT Long 2D	8x8x1	94.7	95.0	95.6	0.3	0.9
DCT Long 3D	4x4x2	90.3	95.5	95.6	5.2	5.3
DCT Long 3D	8x8x2	94.0	93.9	94.9	1.0	0.9
DCT Long 3D	4x4x4	92.6	94.5	93.4	-1.1	0.8
HAAR Long 2D	2x2x1	93.8	95.6	94.7	-0.9	0.9
HAAR Long 2D	4x4x1	95.4	95.7	96.2	0.5	0.8
HAAR Long 2D	8x8x1	94.2	94.8	95.1	0.3	0.9
HAAR Long 3D	4x4x2	90.3	95.5	95.6	0.1	5.3
HAAR Long 3D	8x8x2	94.2	93.9	95.0	1.1	0.8
HAAR Long 3D	4x4x4	92.3	94.8	93.3	-1.5	1.0
HAAR Short 2D	2x2x1	76.3	92.3	94.0	1.7	17.7
HAAR Short 2D	4x4x1	88.2	89.0	93.5	4.4	5.3
HAAR Short 2D	8x8x1	90.1	80.8	88.6	7.8	-1.5

HAAR Short 3D	4x4x2	90.4	95.5	95.5	0.0	5.1
HAAR Short 3D	8x8x2	92.6	94.0	94.6	0.7	2.0
HAAR Short 3D	4x4x4	92.7	94.2	93.5	-0.8	0.8

Table 19: The results of using a 2D low pass filter against using both a 2D and 3D low pass filter.

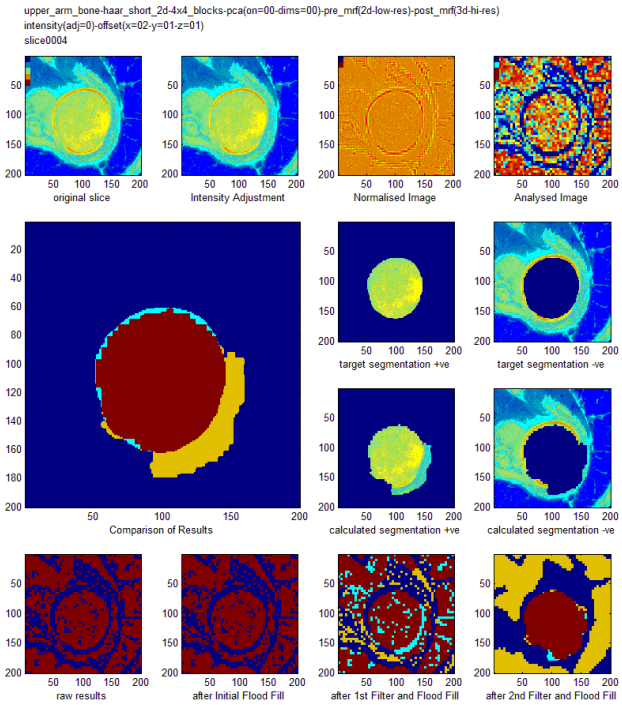


Figure 47: Slice 4, when testing a combination of 2D and 3D low pass filter.

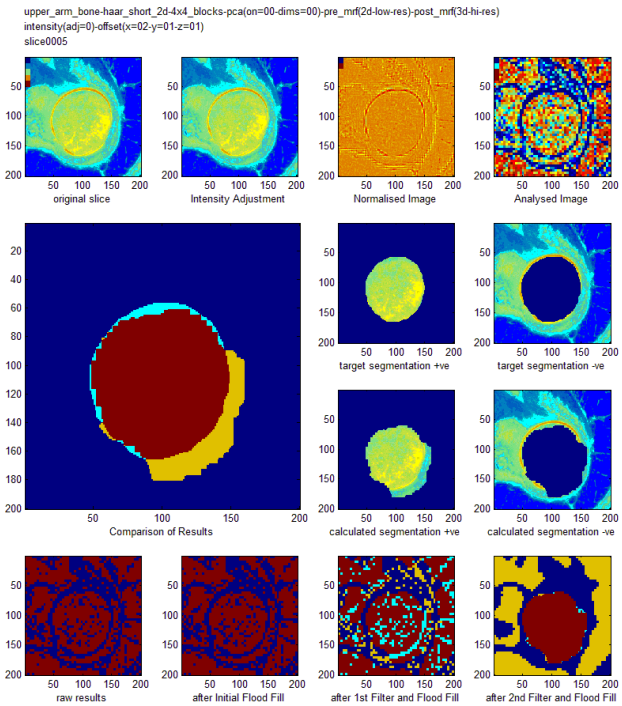


Figure 48: Slice 5, when testing a combination of 2D and 3D low pass filter.

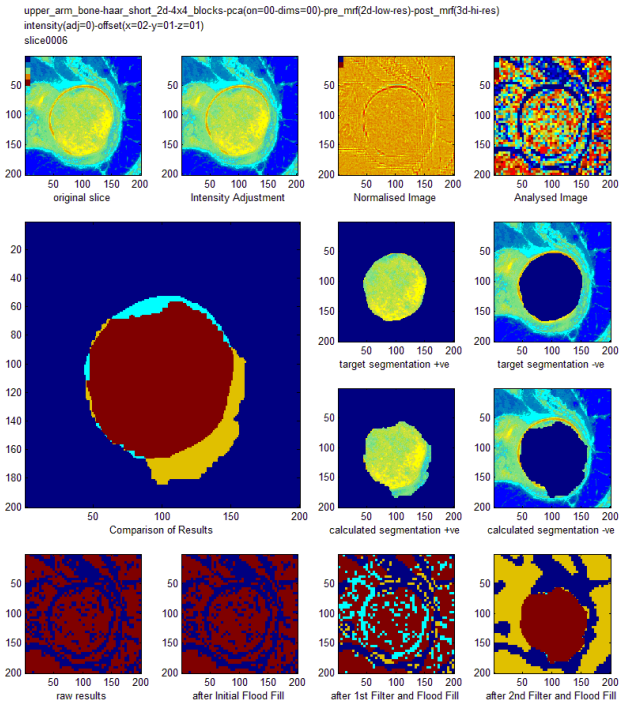


Figure 49: Slice 6, when testing a combination of 2D and 3D low pass filter.

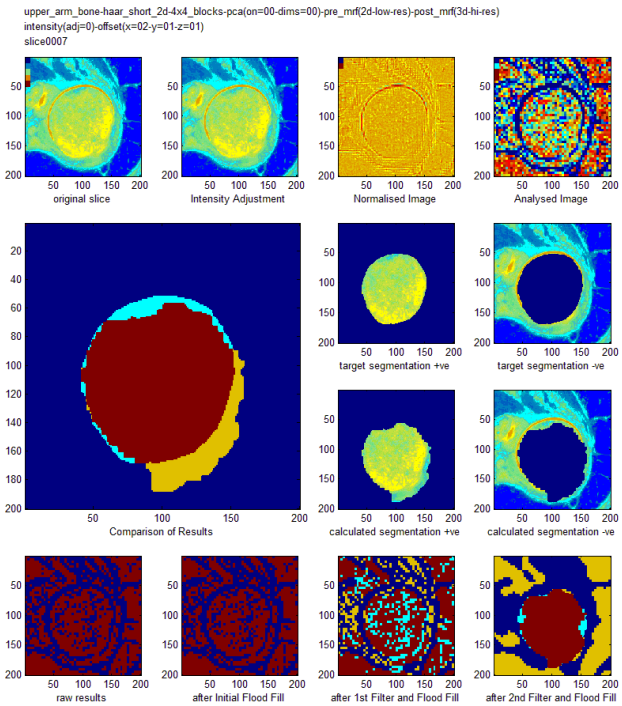


Figure 50: Slice 7, when testing a combination of 2D and 3D low pass filter.

5.0 Adding Robustness against intensity variation

A specific issue with medical image processing is the variation in brightness of the image both spatially and between different images. This intensity inhomogeneity (see section X) can make the images difficult to interpret manually or automatically. Note that this variation in brightness is due to the measurement and capture of the image, rather than due to the tissues being imaged. The intensity information caused by the nature of the different tissues being imaged is still present in the image, but the brightness and contrast of the image will not be consistent throughout the image or between images. This means that any process that is being used to interpret the images will benefit from being invariant to this type of variation in intensity within and between images.

There are two general types of adjustments that can be made to the intensity of pixels within an image; adjustments to brightness and adjustments to contrast. A measure such as the mean intensity of all pixels within an image could be considered a good measure of the overall brightness of the image. The brightness can be increased or decreased by adjusting all of the intensity values within the image by the same (constant) amount (as shown in equation 5.1. Note that the absolute difference between the intensity of any two pixels within the image will stay constant when adjusting the image brightness using this technique.

$$O_{im} = I_{im} \pm B \tag{5.1}$$

Where: O_{im} represents the intensity values of the pixels within the new image.

I_{im} represents the intensity values of the pixels within the old image.

B represents a constant adjustment that is being made to the intensity of each pixel within I_{im}.

The contrast of an image is defined as the range of intensity values used within that image. Adjusting this range is done by adding a gain factor, as shown in equation 5.2. In this case, when the contrast of an image is changed, the

absolute difference between any two pixels will not be consistent across the whole image.

$$O_{im} = I_{im}C \tag{5.2}$$

Where: O_{im} represents the intensity values of the pixels within the new image.

I_{im} represents the intensity values of the pixels within the old image.

C represents a gain factor that is being applied to the intensity of each pixel within I_{im}.

Considering previous work that has been done to combat intra-scan and inter-scan intensity inhomogeneity's (particularly in MRI scans) [23, 24, 25] it is apparent that a sensible model for this intensity variation is to adjust the contrast of the image using a gain factor that slowly varies over the volume of the image.

However, the technique that was chosen to model intensity variation was that of making fixed adjustments to brightness. At the time this was assumed to be a good model for the variations found in medical images. It was considered that this assumption was supported by the combination of using relatively small block sizes and the fact that the intensity variations that medical images are often subject to vary slowly over distance. Therefore, the variation applied to the intensity of individual pixels or voxels within any small and localised portion of the image would be likely to be quite consistent, and therefore testing the approaches on images with a constant brightness shift would be sufficient.

However, and with the benefit of further reading and hindsight this seems unlikely to be a good model for the intensity variation. Interesting future work would therefore include an investigation into the effectiveness of the techniques presented in this chapter to combat intensity variation if tested on images with a variable gain model of the intensity.

As stated, the techniques developed here to combat intensity variations were tested by adjusting the brightness of the datasets used for testing. The training and tuning was carried out as normal, but then the process was tested on a

dataset where the intensity of each pixel within the dataset had been increased or decreased by a constant value. In the tests that were carried out the adjustments made were -20, -16, -12, -8, -4, +4, +8, +12, +16, and +20 (note that the images are based on 8-bit unsigned integer intensity values, and therefore pixels intensities ranged in value from 0 to 255. Therefore the variation was approximately 16% of the maximum possible value).

The results were then collected and presented in tables 20, 22, 24, and 26. Figures 51 to 55 present results of segmenting a specific slice over a range of intensity variations. It can be seen from the results that the process is very sensitive to variations in intensity. To try and resolve this issue, the underlying analysis techniques being used were considered. Both the Haar and the DCT transform contain a single DC coefficient and a number of AC components. The DC coefficient is a measure of the brightness of the input data being transformed. The AC components are not affected by adjustments made to the brightness of the input data.

$$X_k = \sum_{n=0}^{N-1} x_n cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right] \tag{5.3}$$

Where:*N* represents the number of values in the input vector

$x = \{x_0, x_1, x_2, \ldots x_{N-1}\}$ and is the input data being transformed.

$X = \{X_0, X_1, X_2, \ldots X_{N-1}\}$ and is the output data after the transform has been completed.

This can be seen by considering equation 5.3 which shows how the coefficients of a DCT are calculated. Note that a DCT operates on data of a single dimension. Section 2.3.4 shows how DCTs can be used on data of a higher number of dimensions. However, the principle of the following discussion holds for any DCT performed over input data of any number of dimensions.

First let's consider the DC component. This is coefficient X_0 in the output data. When calculating this term, k is set to 0. This means evaluation of the cosine function always results in unity (regardless of the value of n), and X_0 is therefore simply the sum of all the input components. In practice this term is usually

scaled by a factor of $1/\sqrt{2}$. In anycase it can be seen that as X_0 is essentially a sum of the coefficients that make up x, it will always be sensitive to any adjustment of the brightness of the input data.

Now consider the calculation of the AC components. These are the coefficients for which k is non-zero, and as such evaluation of the cosine term will now no longer always yield unity as the result. However, if the cosine term is evaluated for a non-zero value of k and for each required value of n, it can be seen that the sum of these terms is zero. Hence it is the difference between the coefficients that make up x, rather than their absolute values that effects the generated output X. Therefore by adjusting the brightness of x, the AC components of X must remain unaffected.
Now, let's review what happens if the contrast is adjusted. In this case the DC coefficient X_0 (which is essentially the sum of the coefficients in x_0) will still see some adjustment. However, as the difference between the various members of x is no longer preserved, the AC coefficients will also be affected.

As the assumption had been made that adjusting the brightness of the test images provided a good model of the type of distortion commonly seen in a context of medical images, it was also decided that a sensible way to make sure that the process was invariant with respect to the variations in intensity was to only use the AC coefficients from the DCT and Haar transforms when mapping the blocks into feature space.

Taking this approach has the downside that less information about each block is being used when mapping the block into feature space, and so it was assumed that there may be a reduction in the quality of results. However, it was decided that the possible benefit of consistency across different intensities was worth investigating.

It is also important to note that because of the sampling approach taken to the data (splitting the data into blocks), once the process has been made invariant to variations in intensity, the process will work consistently with images where the whole image has a higher or lower intensity than the training image, or with images where the intensity is varied over regions within the image itself

(assuming that the block size is small enough that there is unlikely to be much intra-block intensity variation).

Figures 51 to 56 show a dataset that was trained on a specific image and then tested using images which have been intensity adjusted at levels of -20, -12, -4, +4, +12 and +20. It can be seen that outside of the central band of -4 to +4 the quality of the segmentation falls dramatically.

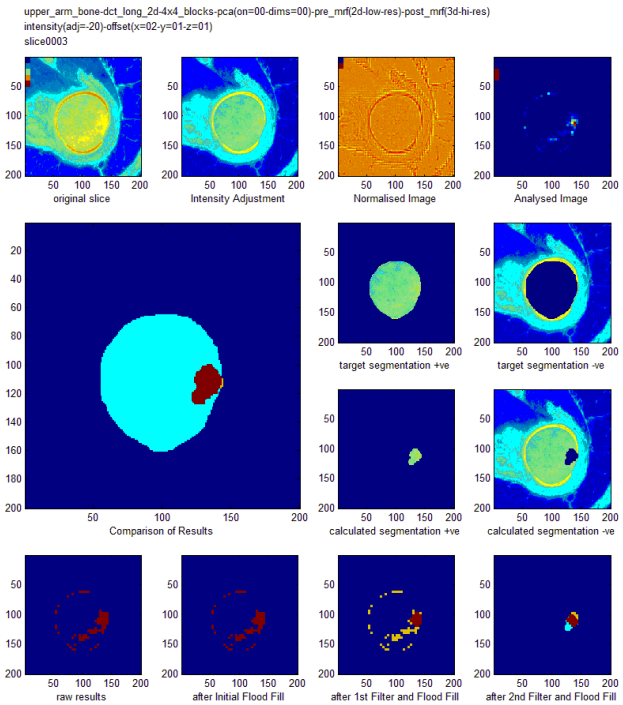


Figure 51: Results when input image has had its intensity decreased by 20.

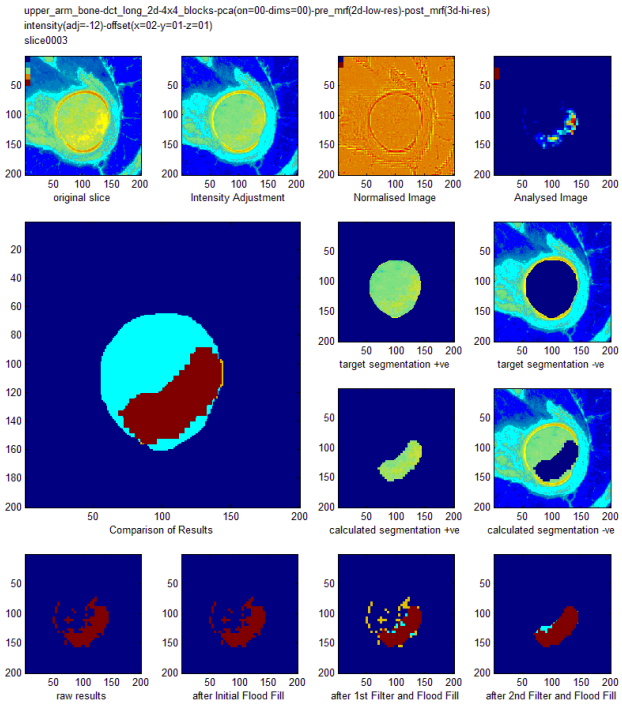


Figure 52: Results when input image has had its intensity decreased by -12.

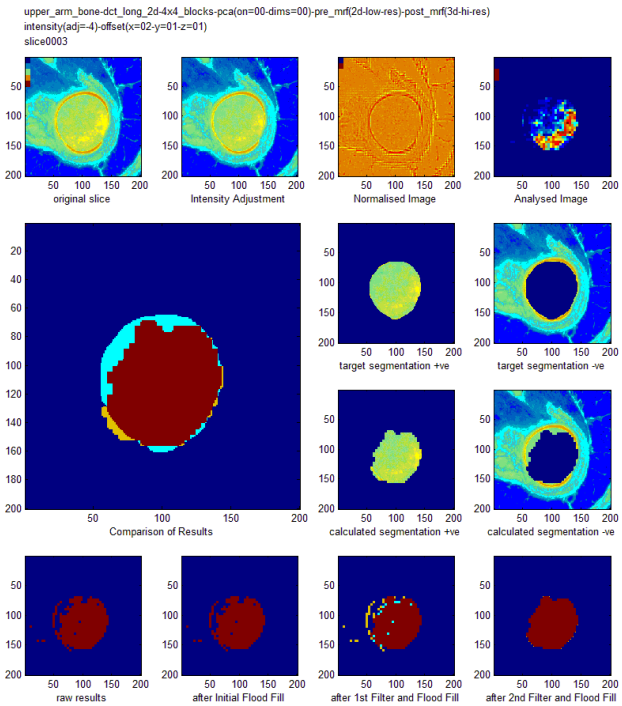


Figure 53: Results when input image has had its intensity decreased by 4.

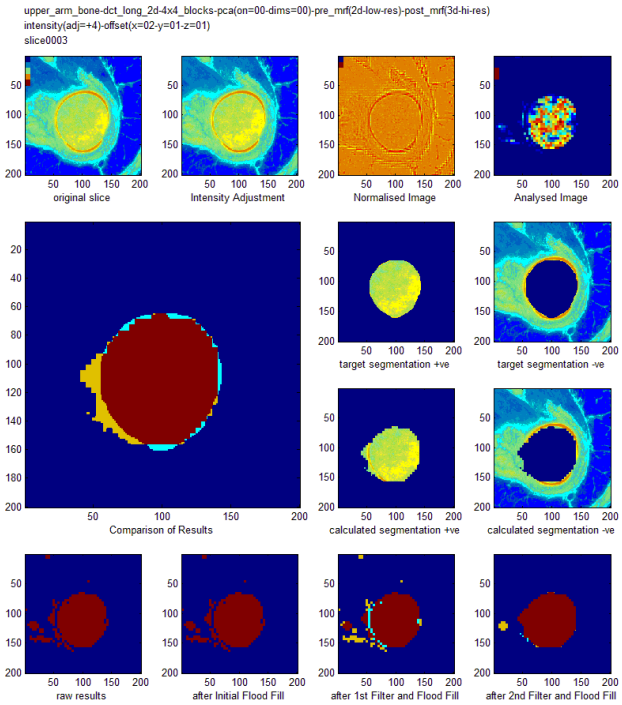


Figure 54: Results when input image has had its intensity increased by 4.

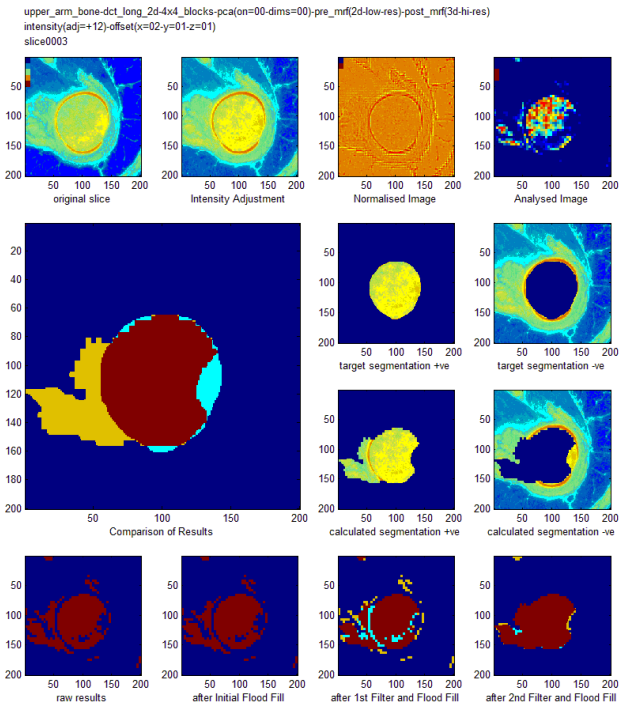


Figure 55: Results when input image has had its intensity increased by 12.

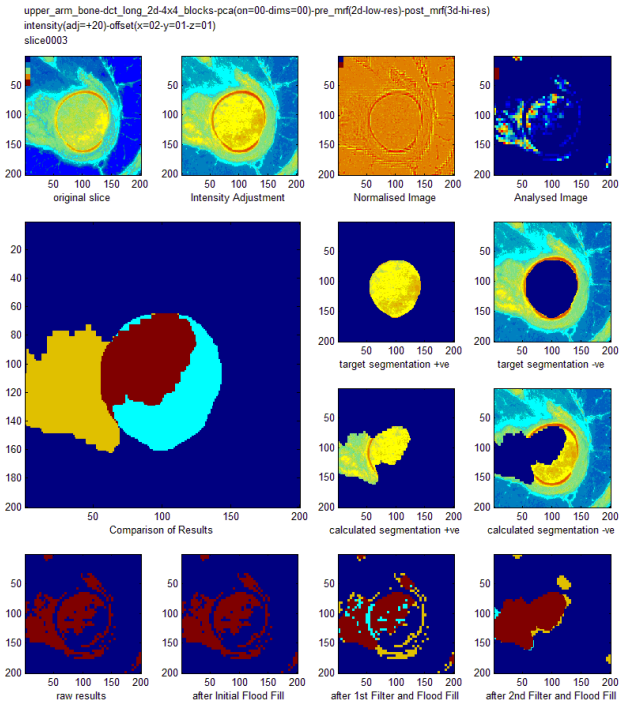


Figure 56: Results when input image has had its intensity increased by 20.

The same tests where then run again, this time without including the DC coefficients from the DCT and the Haar transforms in any part of the process, and the results collated (see tables 21, 23, 25 and 27). It can be seen that when the DC coefficient is not used, the results are consistent across intensity variations. However, the best result using this technique is not as good as the best result when the DC coefficient is taken into account.

Transform	Block Size	Intensity Adjustment	% Ones Correct	% Zeros Correct	% Correct
DCT LONG 2D	4x4x1	-20	9.618	99.984	71.76
DCT LONG 2D	4x4x1	-16	36.664	99.868	80.128
DCT LONG 2D	4x4x1	-12	60.946	99.65	87.561
DCT LONG 2D	4x4x1	-8	83.424	99.254	94.31
DCT LONG 2D	4x4x1	-4	88.495	98.929	95.67
DCT LONG 2D	4x4x1	0	91.121	98.557	96.235
DCT LONG 2D	4x4x1	4	92.218	96.972	95.487
DCT LONG 2D	4x4x1	8	90.537	94.844	93.499
DCT LONG 2D	4x4x1	12	80.791	93.427	89.48
DCT LONG 2D	4x4x1	16	50.665	94.373	80.722
DCT LONG 2D	4x4x1	20	26.306	94.046	72.889

Table 20: Results for DCT LONG 2D Transform over a 40 point variation in intensity, when the process uses the DC coefficient from the DCT transform.

Transform	Block Size	Intensity Adjustment	% Ones Correct	% Zeros Correct	% Correct
DCT LONG 2D	4x4x1	-20	85.137	96.428	92.901
DCT LONG 2D	4x4x1	-16	85.137	96.428	92.901
DCT LONG 2D	4x4x1	-12	85.137	96.428	92.901
DCT LONG 2D	4x4x1	-8	85.137	96.428	92.901
DCT LONG 2D	4x4x1	-4	85.137	96.428	92.901
DCT LONG 2D	4x4x1	0	85.137	96.428	92.901
DCT LONG 2D	4x4x1	4	85.137	96.428	92.901
DCT LONG 2D	4x4x1	8	85.137	96.428	92.901
DCT LONG 2D	4x4x1	12	85.137	96.428	92.901
DCT LONG 2D	4x4x1	16	85.137	96.428	92.901
DCT LONG 2D	4x4x1	20	85.137	96.428	92.901

Table 21: Results for DCT LONG 2D Transform over a 40 point variation in intensity, when the process does not use the DC coefficient from the DCT transform.

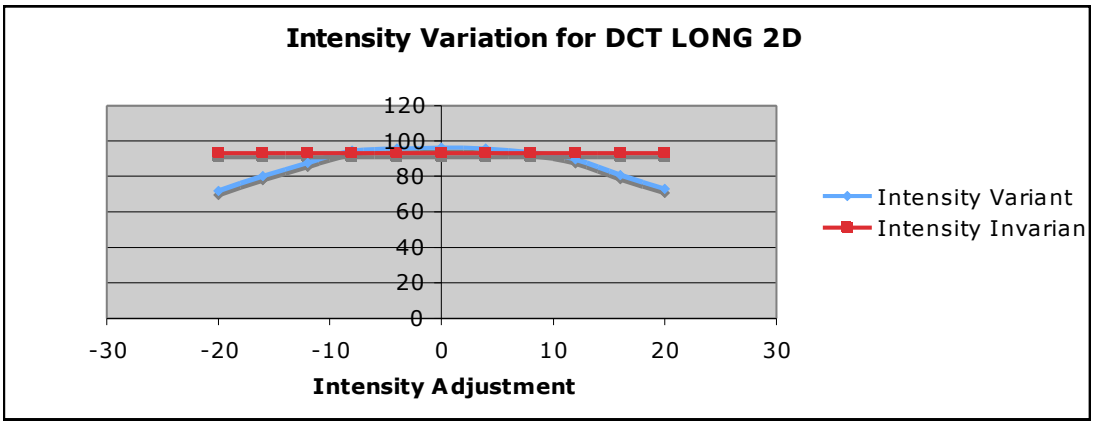


Figure 57: Plot showing the comparison of results between the Intensity Variant and Intensity Invariant methods for the DCT LONG 2D transform.

Transform	Block Size	Intensity Adjustment	% Ones Correct	% Zeros Correct	% Correct
DCT LONG 3D	4x4x2	-20	77.763	99.6	92.94
DCT LONG 3D	4x4x2	-16	85.489	99.399	95.157
DCT LONG 3D	4x4x2	-12	86.97	99.165	95.446
DCT LONG 3D	4x4x2	-8	87.812	99.013	95.597
DCT LONG 3D	4x4x2	-4	88.478	98.879	95.707
DCT LONG 3D	4x4x2	0	88.465	98.751	95.614
DCT LONG 3D	4x4x2	4	88.735	94.464	92.717
DCT LONG 3D	4x4x2	8	88.353	93.763	92.113

DCT LONG 3D	4x4x2	12	85.988	92.758	90.693
DCT LONG 3D	4x4x2	16	72.93	92.735	86.695
DCT LONG 3D	4x4x2	20	27.796	100	77.979

Table 22: Results for DCT LONG 3D Transform over a 40 point variation in intensity, when the process uses the DC coefficient from the DCT transform.

Transform	Block Size	Intensity Adjustment	% Ones Correct	% Zeros Correct	% Correct
DCT LONG 3D	4x4x2	-20	85.513	99.071	94.936
DCT LONG 3D	4x4x2	-16	85.513	99.071	94.936
DCT LONG 3D	4x4x2	-12	85.513	99.071	94.936
DCT LONG 3D	4x4x2	-8	85.513	99.071	94.936
DCT LONG 3D	4x4x2	-4	85.513	99.071	94.936
DCT LONG 3D	4x4x2	0	85.513	99.071	94.936
DCT LONG 3D	4x4x2	4	85.513	99.071	94.936
DCT LONG 3D	4x4x2	8	85.513	99.071	94.936
DCT LONG 3D	4x4x2	12	85.513	99.071	94.936

DCT LONG 3D	4x4x2	16	85.513	99.071	94.936
DCT LONG 3D	4x4x2	20	85.513	99.071	94.936

Table 23: Results for DCT LONG 3D Transform over a 40 point variation in intensity, when the process does not use the DC coefficient from the DCT transform.

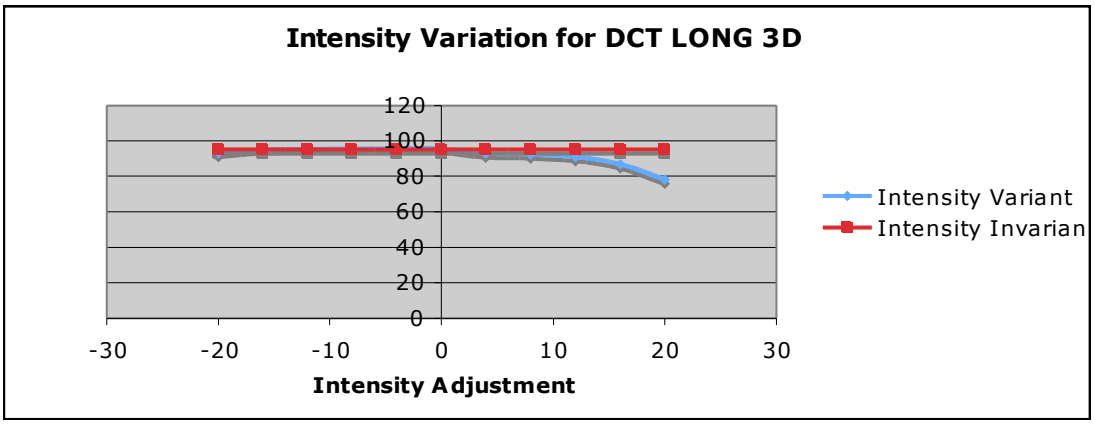


Figure 58: Plot showing the comparison of results between the Intensity Variant and Intensity Invariant methods for the DCT LONG 3D transform.

Transform	Block Size	Intensity Adjustment	% Ones Correct	% Zeros Correct	% Correct
HAAR LONG 2D	4x4x1	-20	8.312	99.954	71.332
HAAR LONG 2D	4x4x1	-16	40.814	99.83	81.398
HAAR LONG 2D	4x4x1	-12	64.569	99.586	88.649
HAAR LONG 2D	4x4x1	-8	85.244	99.157	94.812

HAAR LONG 2D	4x4x1	-4	89.001	98.85	95.774
HAAR LONG 2D	4x4x1	0	91.492	97.96	95.94
HAAR LONG 2D	4x4x1	4	92.269	96.5	95.179
HAAR LONG 2D	4x4x1	8	89.553	94.247	92.781
HAAR LONG 2D	4x4x1	12	74.542	93.875	87.837
HAAR LONG 2D	4x4x1	16	43.323	94.077	78.225
HAAR LONG 2D	4x4x1	20	21.301	93.869	71.204

Table 24: Results for HAAR LONG 2D Transform over a 40 point variation in intensity, when the process uses the DC coefficient from the HAAR transform.

Transform	Block Size	Intensity Adjustment	% Ones Correct	% Zeros Correct	% Correct
HAAR LONG 2D	4x4x1	-20	83.088	98.828	93.912
HAAR LONG 2D	4x4x1	-16	83.088	98.828	93.912
HAAR LONG 2D	4x4x1	-12	83.088	98.828	93.912
HAAR LONG 2D	4x4x1	-8	83.088	98.828	93.912
HAAR LONG 2D	4x4x1	-4	83.088	98.828	93.912
HAAR LONG 2D	4x4x1	0	83.088	98.828	93.912

HAAR LONG 2D	4x4x1	4	83.088	98.828	93.912
HAAR LONG 2D	4x4x1	8	83.088	98.828	93.912
HAAR LONG 2D	4x4x1	12	83.088	98.828	93.912
HAAR LONG 2D	4x4x1	16	83.088	98.828	93.912
HAAR LONG 2D	4x4x1	20	83.088	98.828	93.912

Table 25: Results for HAAR LONG 2D Transform over a 40 point variation in intensity, when the process does not use the DC coefficient from the HAAR transform.

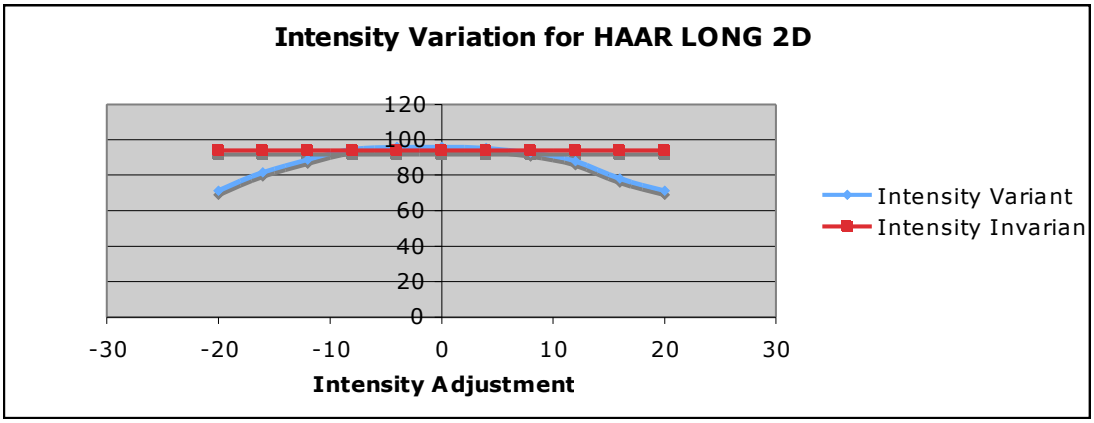


Figure 59: Plot showing the comparison of results between the Intensity Variant and Intensity Invariant methods for the HAAR LONG 2D transform.

Transform	Block Size	Intensity Adjustment	% Ones Correct	% Zeros Correct	% Correct
HAAR LONG 3D	4x4x2	-20	77.763	99.6	92.94
HAAR LONG 3D	4x4x2	-16	85.489	99.399	95.157
HAAR LONG 3D	4x4x2	-12	86.97	99.165	95.446
HAAR LONG 3D	4x4x2	-8	87.812	99.013	95.597
HAAR LONG 3D	4x4x2	-4	88.478	98.879	95.707
HAAR LONG 3D	4x4x2	0	88.465	98.751	95.614
HAAR LONG 3D	4x4x2	4	88.735	94.464	92.717
HAAR LONG 3D	4x4x2	8	88.353	93.763	92.113
HAAR LONG 3D	4x4x2	12	85.988	92.758	90.693
HAAR LONG 3D	4x4x2	16	72.93	92.735	86.695
HAAR LONG 3D	4x4x2	20	27.796	100	77.979

Table 26: Results for HAAR LONG 3D Transform over a 40 point variation in intensity, when the process uses the DC coefficient from the HAAR transform.

Transform	Block Size	Intensity Adjustment	% Ones Correct	% Zeros Correct	% Correct
HAAR LONG 3D	4x4x2	-20	85.484	99.005	94.882
HAAR LONG 3D	4x4x2	-16	85.484	99.005	94.882
HAAR LONG 3D	4x4x2	-12	85.484	99.005	94.882
HAAR LONG 3D	4x4x2	-8	85.484	99.005	94.882
HAAR LONG 3D	4x4x2	-4	85.484	99.005	94.882
HAAR LONG 3D	4x4x2	0	85.484	99.005	94.882
HAAR LONG 3D	4x4x2	4	85.484	99.005	94.882
HAAR LONG 3D	4x4x2	8	85.484	99.005	94.882
HAAR LONG 3D	4x4x2	12	85.484	99.005	94.882
HAAR LONG 3D	4x4x2	16	85.484	99.005	94.882
HAAR LONG 3D	4x4x2	20	85.484	99.005	94.882

Table 27: Results for HAAR LONG 3D Transform over a 40 point variation in intensity, when the process does not use the DC coefficient from the HAAR transform.

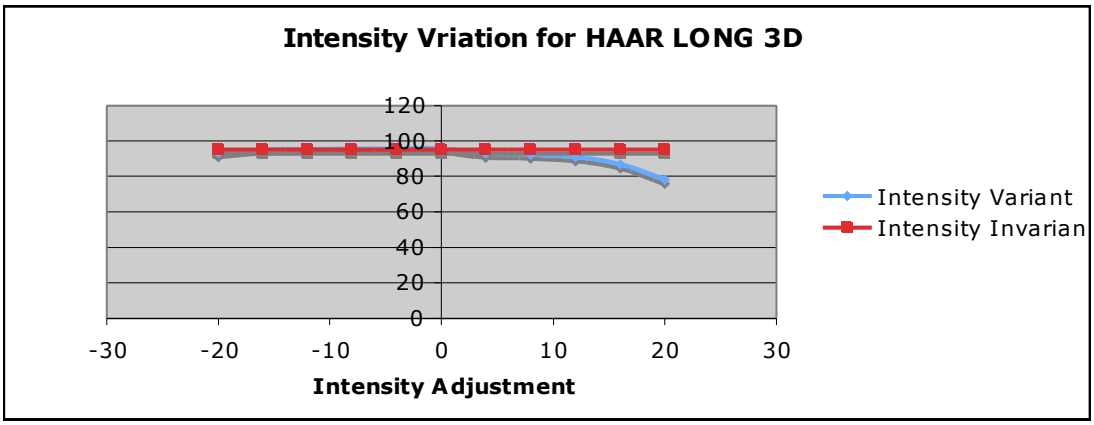


Figure 60: Plot showing the comparison of results between the Intensity Variant and Intensity Invariant methods for the HAAR LONG 3D transform.

The results presented in tables 20 to 27 and in figures 57 to 60 show that removing the use of the DC coefficient from the process does reduce the quality of the segmentation, but also makes the process invariant to variations in intensity. It is also worth reviewing the quality of the results obtained using the invariant process from a subjective point of view.

The results shown in figures 63 to 66 show the same slice after having being segmented using a DCT Long 2D process, with an intensity adjustment of -20, -8, 8 and 20 respectively. There are a few interesting things to note in these plots.

As expected, they all achieve identical results. The segmentation of this slice has been achieved reasonably well, with an anomaly towards the lower right hand side of the segmentation. This is highlighted in figure 61.

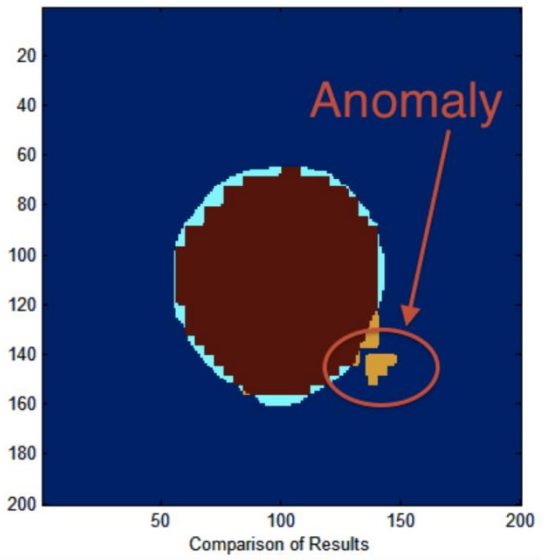


Figure 61: Although the results are now consistent over intensity variation, there are still anomalies present.

This is a result of the 2D low pass filter forming a bridge to a remote region that has been then included in the segmentation set. The interesting thing here is that the 3D high resolution low pass filter has actually removed some of this bridge, and this has then left a remote region within the segmentation set.

Removing this remote region from the segmentation set would improve the segmentation results for this slice, and would be particularly easy by applying a second region growing operation.

Another way of removing this anomaly from the results would be to try to stop the bridge forming in the first place. This could be done either by applying a region growing operation earlier in the process, before the 2D low pass filter is applied, or by adding a maximum threshold to the classification process. This may prove to be effective as can been seen from looking at the analysed image shown in figure 62. It is annotated with a white oval approximately surrounding the desired segmentation. It can be seen that the desired segmentation set returns a distinct response from the classifier (shown as mainly yellow, green and light blue). Dark blue surrounds the set and shows this area gave a lower response. However, there are other regions outside of the oval that have returned a stronger response from the classification. These regions are shown in predominantly bright red.

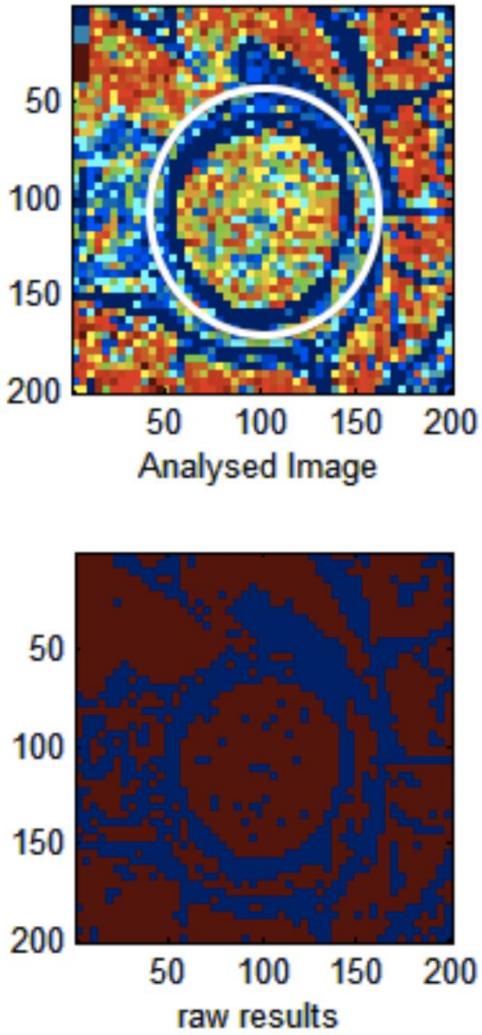


Figure 62: Locations with any response over the threshold are included in the segmentation set.

Considering the raw results that are shown in figure 62 it can be seen that both the regions within the desired segmentation set and from the bright red area outside of the desired segmentation set are included. A better quality of raw results could be obtained by selecting only those pixels whose results from the classifier that were within a suitable range, in this case excluding the weaker dark blue pixels and the stronger bright red pixels

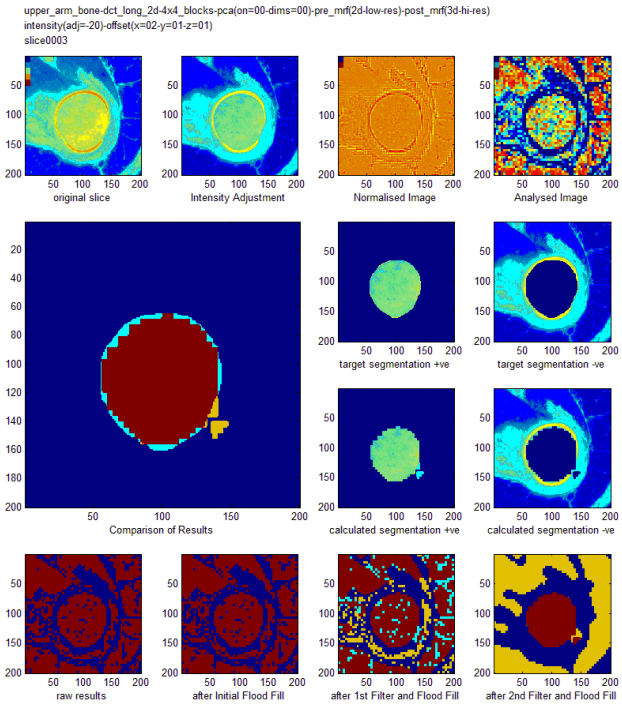


Figure 63: Results when input image has had its intensity decreased by 20.

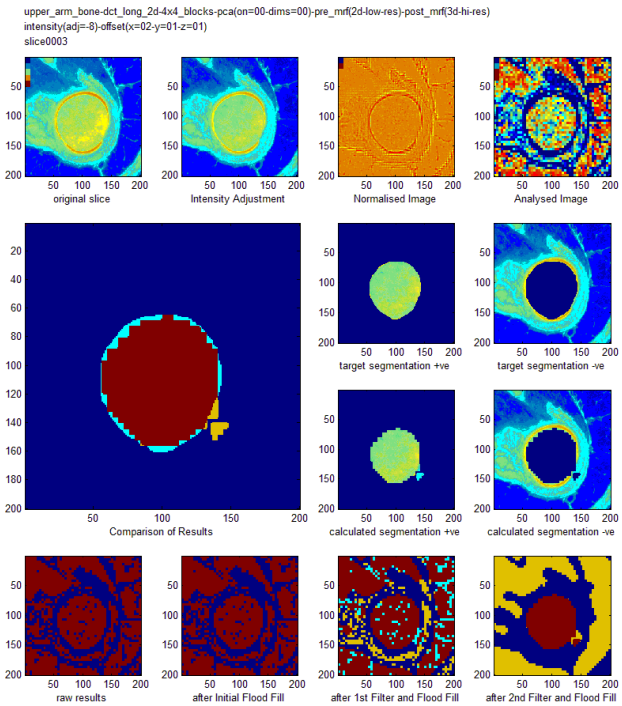


Figure 64: Results when input image has had its intensity decreased by 8.

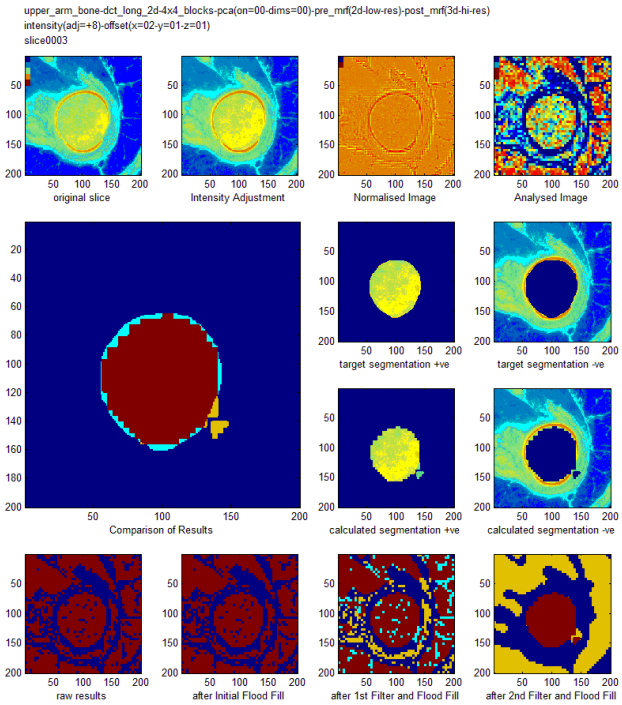


Figure 65: Results when input image has had its intensity increased by 8.

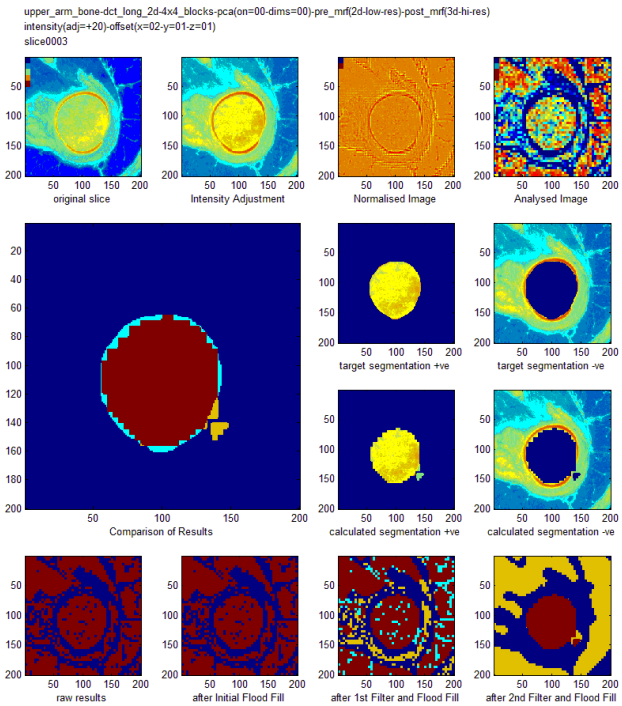


Figure 66: Results when input image has had its intensity increased by 20.

A new test set was developed to enable this anomaly to be investigated. The image used was a simple one, and only consisted of one slice. The slice consists of two different textures taking up half of the slice each. The desired segmentation only includes a single texture in one half of the slice.

The results from a segmentation based on this slice are shown in figure 68. It shows that the texture that is outside of the segmentation set has a stronger response to the classification than the texture within the segmentation set. This is one of the problems with the anomaly from the previous example.

A maximum threshold was then used to attempt to solve this problem, as described by the pseudo code shown in figure 67.

```
Empty the segmentation set.

For (each pixel in the slice)
        If(pixels response to classifier > lower threshold)
                If(pixels response to classifier < upper_threshold)
                        Add pixel to the segmentation set.
                End if
        End if
End for
```

Figure 67: Pseudo code showing operation of minimum and maximum threshold.

The threshold that was chosen was selected in the training process. The value that was chosen was slightly lower than the highest value that a member of the desired segmentation set generated. The reason for doing this is that although some members of the segmentation set are then excluded from the raw results, these will mostly be reinstated by the filtering and post-processing. By using a slightly lower threshold, a better separation from pixels outside of the segmentation set (but which have a high response to the classification) is obtained.

The slice in figure 68 shows that although the textures of each half of the slice are markedly different (and as such the analysed image shows a very marked separation between the two halves), the segmentation fails.

Once a maximum threshold is used, the segmentation is much better, as shown by figure 69.

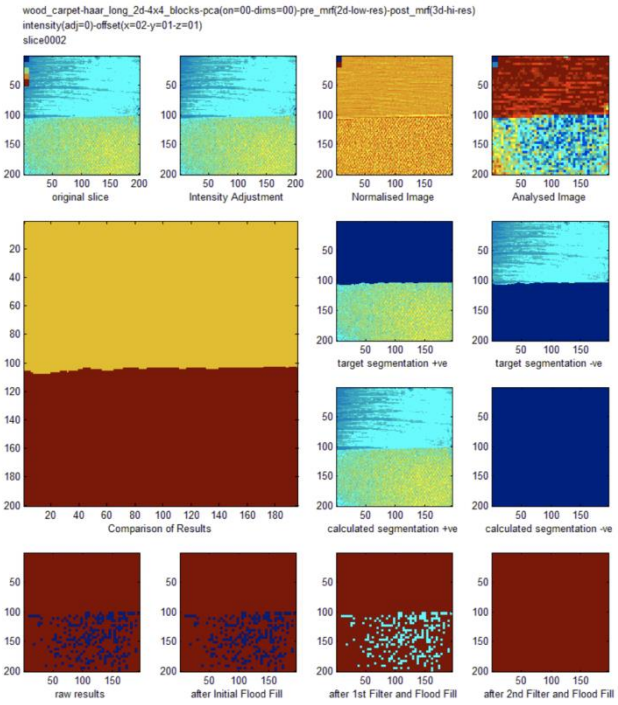


Figure 68: Example of segmentation where lack of a maximum threshold causes the segmentation to fail completely.

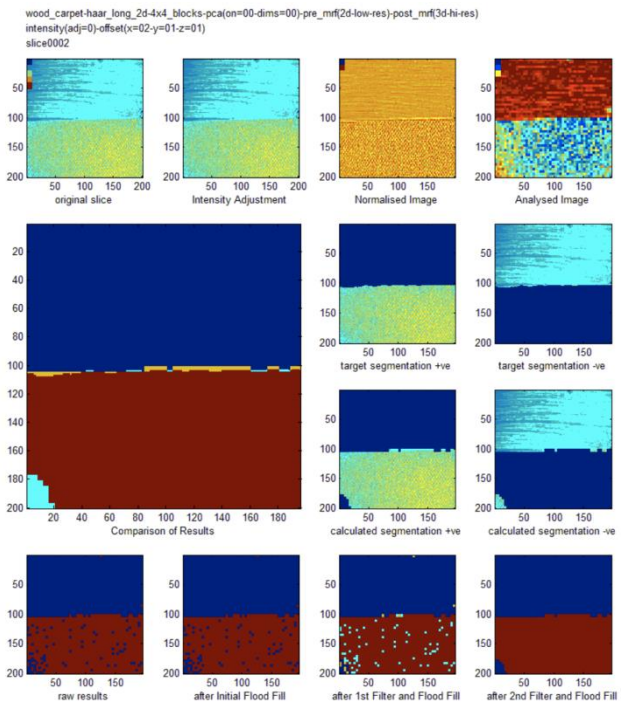


Figure 69: Example of segmentation where lack of a maximum threshold causes the segmentation to fail completely.

Once the maximum threshold had been shown to be implemented successfully, it was then applied to the real data to show the improvements there. The results are shown in figure 70.

Although using a maximum threshold was seen to improve the reduction in quality of results attributed to this anomaly, the root cause was never fully investigated or understood.

However, it is thought that the situation arises from the way the GMM was trained, specifically the way in which the GMM was only ever presented with descriptors associated with members of the segmentation set, and no descriptors that were not members of the set. A better training method may have been to weight the positive and negative examples of descriptors accordingly. Although no experimentation was carried out to prove or disprove this theory, anecdotally it seems likely that if the training process is presented with more information about descriptors that should be included within the segmentation set or not, it would be able to create a better model.

With hindsight, this would have been an easy adjustment to make to the process, and it would have been interesting to investigate the effect on the quality of results (and this issue in particular) of using both positive and negative examples in the GMM training process.

Using a maximum threshold was seen to improve the results, but on review a new anomaly was observed, on the right hand side of the image.

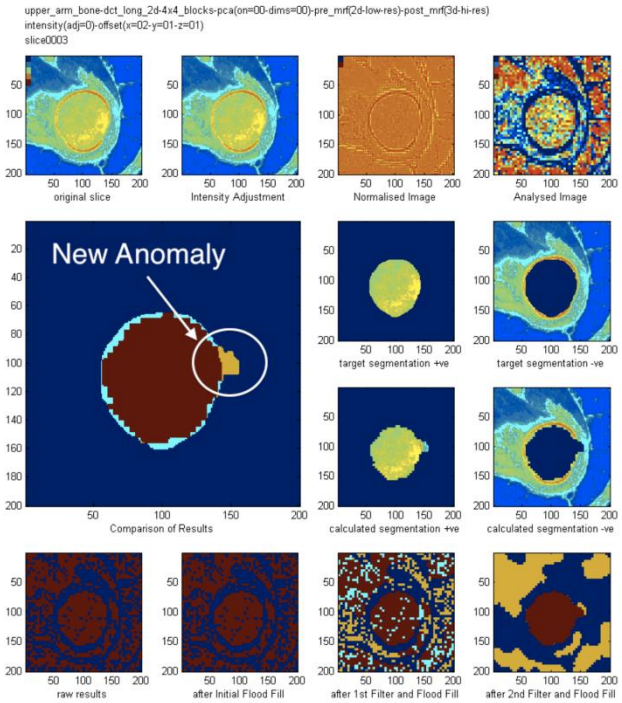


Figure 70: Although the previous anomaly has gone, a new anomaly has appeared.

The effect of adding the maximum threshold can be seen in the raw results, especially when compared to the previously obtained raw results. This comparison is shown in figure 71. It is apparent that the density of voxels that have been placed within the segmentation set correctly (within the white oval on the right hand image) is approximately the same before and after a maximum threshold has been used.

However, in regions that have been erroneously classified (within the raw results at least) as being in the segmentation set, the density has been reduced. This makes it easier for the post processing techniques to achieve better results.

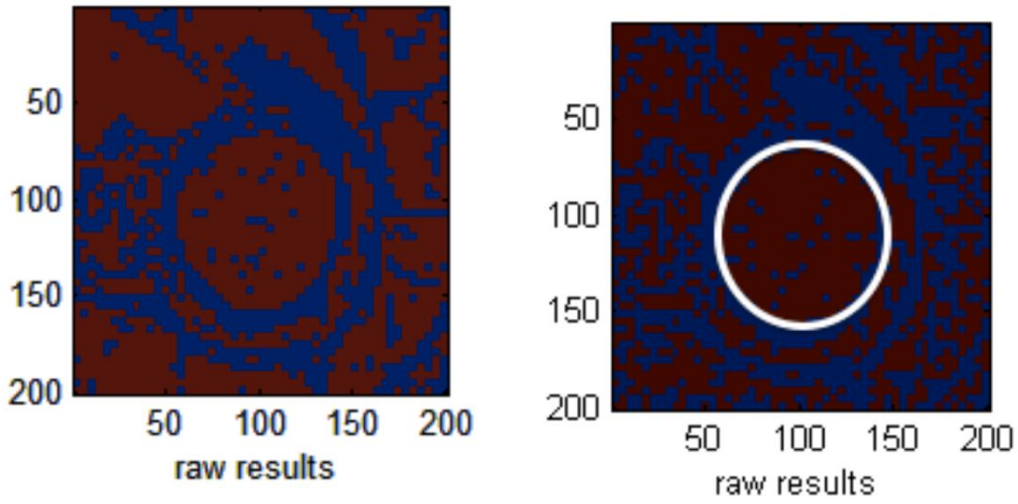


Figure 71: Left hand image shows the raw results before a maximum threshold was used. Right hand image shows the raw results once a maximum threshold has been used. The white oval annotation on the right hand image shows the approximate location of the desired segmentation.

It should be noted that although (in the last example) the GMM was able to distinguish textures that should be within the segmentation set, the highest probability was assigned to regions that should not have been assigned to the set. Although this was not thoroughly investigated at the time, there are a few possible reasons that this could have occurred.

The GMM was trained using positive examples only. That is, blocks that were from the segmentation set were extracted from the training images, and the texture of these blocks was analysed. The resulting mapping to feature space was passed to the GMM as training data. At no point were any negative examples passed to the GMM (i.e. textures that should be excluded from the segmentation set). Possibly if this had been done, then the GMM would have behaved in a more expected way, and the higher probabilities would have been reserved for textures that should have been included within the segmentation set, and a maximum threshold would not have been required.

Another possibility is the way the GMM was configured. Generally a GMM can be used in various different ways. In this case the configuration was set early on in the investigation, and the choices made were not reviewed at any subsequent

point as the results being achieved seemed to be reasonable. The configuration of the GMM included:

Number of centres: 10
Type:Spherical

It is possible that the dimensionality of the feature space, and the number of distinct textures found within the segmentation set could not be suitably modelled with the GMM setup as described above. Carrying out some investigation into the effect of varying the parameters of the GMM may have revealed this, and may also have meant that the maximum threshold was not required.

At this point reasonable results have been obtained using each of the 6 methods. These results are outlined in the following figures (72 to 77), which display the results of a segmentation carried out on 18 slices of the “Upper Arm Bone” dataset. The segmentation has been performed using the following post processing:

1) Block resolution 2D low pass filter.
2) Region growing operation.
3) Pixel or Voxel resolution 3D low pass filter.
4) Final region growing operation.

Each figure shows 18 slices (in order from left to right, and then top to bottom). The image of each slice contains four colours:

1) Dark Blue (pixels or voxels correctly excluded from the segmentation set).
2) Light Blue (pixels or voxels incorrectly excluded from the segmentation set).
3) Red (pixels or voxels correctly included within the segmentation set).
4) Yellow (pixels or voxels incorrectly included within the segmentation set).

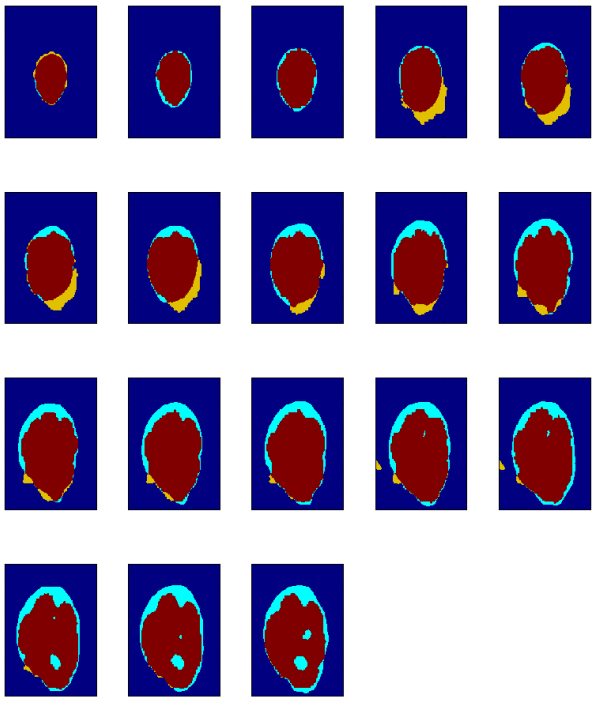


Figure 72: DCT Long 2D based segmentation results.

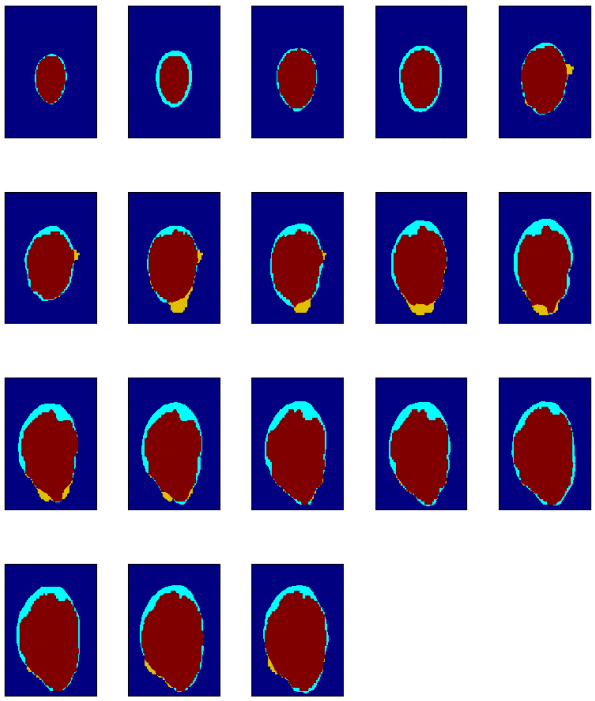


Figure 73: DCT Long 3D based segmentation results.

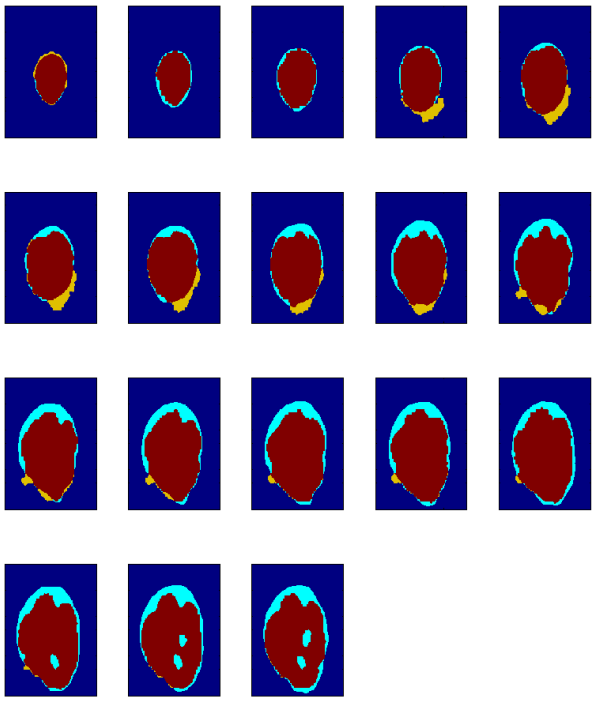


Figure 74: Haar Long 2D based segmentation results.

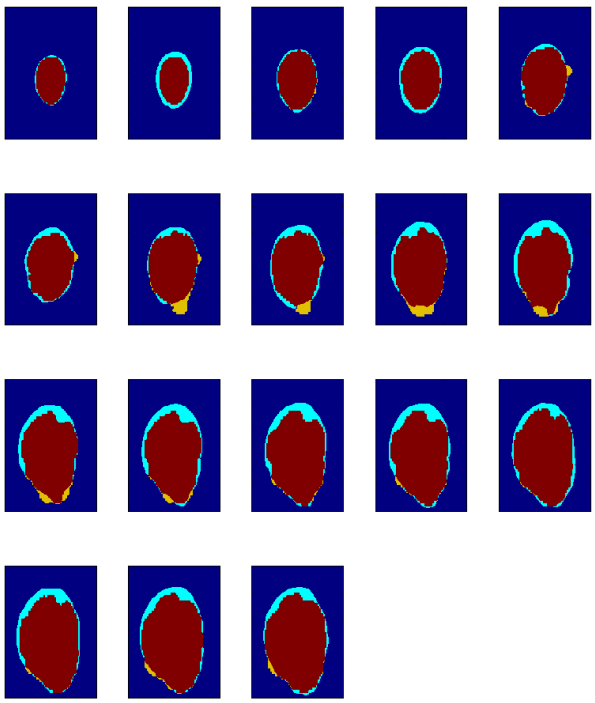


Figure 75: Haar Long 3D based segmentation results.

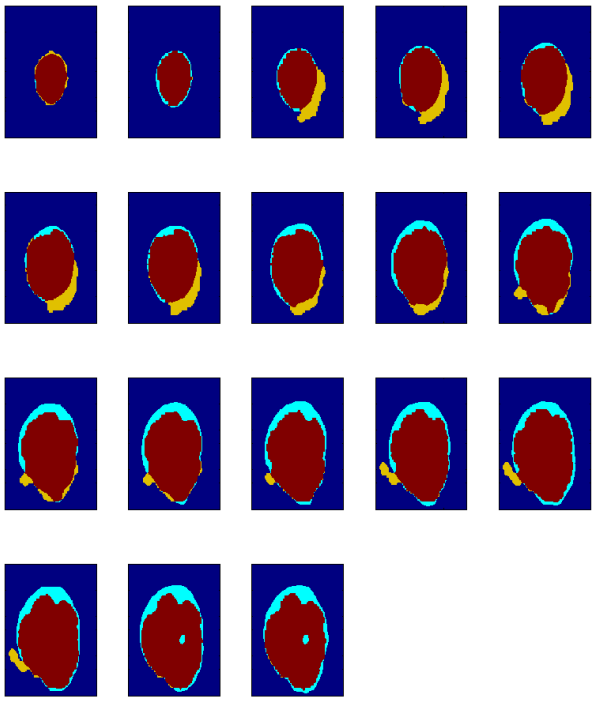


Figure 76: Haar Short 2D based segmentation results.

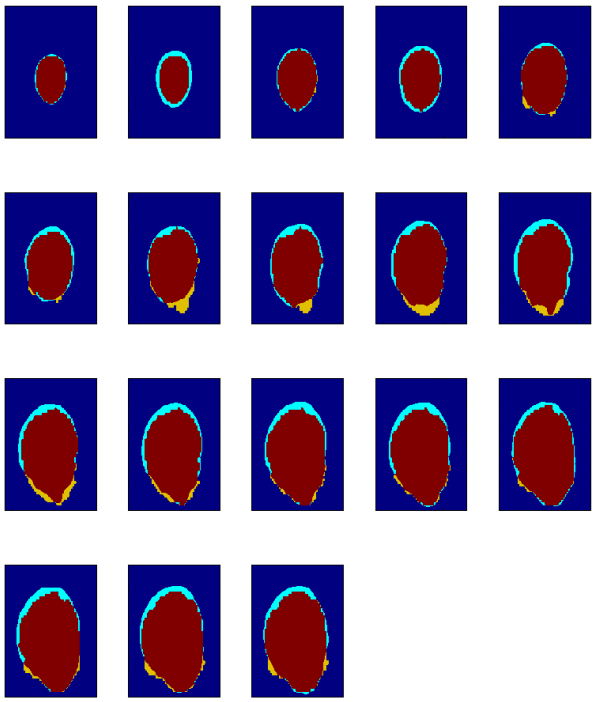


Figure 77: Haar Short 3D based segmentation results.

6.0 Testing the process on a more complex segmentation

The next phase of the research was to attempt a more complex segmentation. It was seen however, that when a single segment contained a number of different textures, then the quality of the resulting segmentation tended to be reduced. This was attributed to the fact that once many different textures are being treated as being within the segmentation set, it becomes more likely that a region outside of the desired segmentation set will contain at least one of those textures, and it therefore becomes harder to get good results.

The segmentation that was attempted was that of a bone in the upper leg. The "Upper Right Leg Bone" data set was used for this. A summary of the details of the dataset can be found in Table 7, but it is useful to note that the slices are 201 pixels x 201 pixels in size, and that the dataset is made up of 51 slices. The bone is visible in all the slices within the dataset.

A slice typical of those that were used from this dataset is shown in figure 78. The bone is visible as a lighter blue region towards the top of the slice. Note that once the DC component of the image was been removed (on a block-by-block basis) the texture of the central part of the bone is quite distinct from the outer border. Note also that the surrounding tissue has a similar texture to the outer border. It was decided to try and segment the outer border of the bone. This is roughly the region between the white ovals in figure 79.

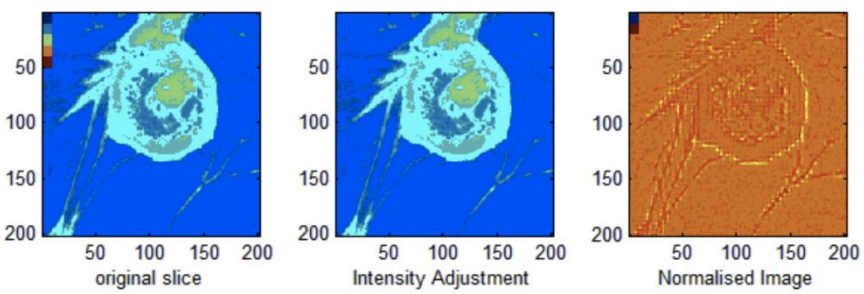


Figure 78: Upper Leg Bone.

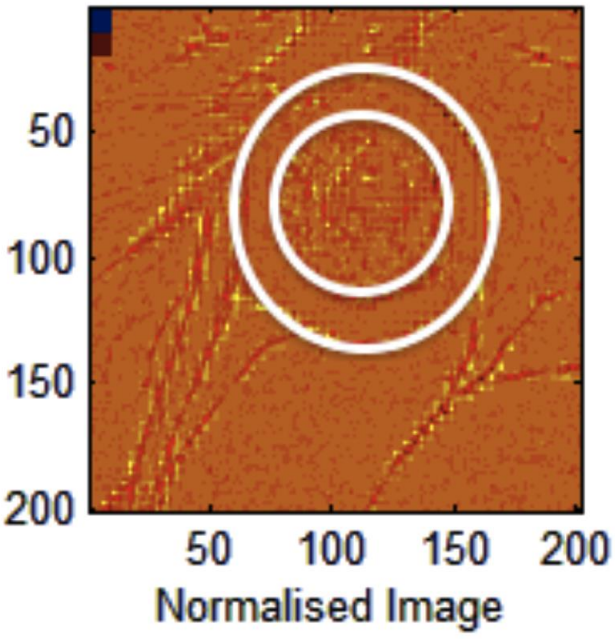


Figure 79: The region between the two white ovals is roughly the outer border of a bone in the upper leg.

The region that the segmentation is targeting is shown in figure 79. Notice that although the texture within the central oval is quite different from that between the ovals, the texture between the ovals is quite similar to many regions outside of the ovals.

The post processing that was used initially to carry out this more complex segmentation is shown in table 28.

Post processing step number	**Description**
1	Symmetrical 2D block level low pass filter
2	Symmetrical 3D pixel/voxel level low pass filter
3	Strong region growing process

Table 28. Post processing that was carried out during initial segmentations of the more complex dataset.

Initial results from this new, more complex segmentation (as shown in table 29), were poor for both the DCT and Haar Long transform based segmentations. The

Haar Short transform based segmentations performed reasonably well though, due perhaps to the effectively higher resolution provided by use of the sub-blocks.

As previously discussed, although the Haar Short transform based segmentation requires no extra Haar transforms when compared to the other techniques, the transform process is more computationally demanding and results in a larger number of sub-blocks that have to be classified. If similarly good results could be achieved by using the DCT or Haar Long transform based segmentation, this could still provide a more computationally efficient solution.

Transform	Block Size	% Ones Correct	% Zeros Correct	% Correct
DCT LONG 2D	4x4x1	66.2	56.5	57.6
DCT LONG 3D	4x4x2	17.7	97.4	88.7
HAAR LONG 2D	4x4x1	76.3	50.0	52.8
HAAR LONG 3D	4x4x2	8.7	99.6	89.7
HAAR SHORT 2D	4x4x1	69.0	98.6	95.4
HAAR SHORT 3D	4x4x2	52.0	97.0	92.1

Table 29: Results on more complex segmentation.

Reviewing the results from table 26 more closely, the results of the Haar Long and DCT Long process seemed to exhibit a distinct split between the use of 2D and 3D blocks. Although using 2D blocks resulted in a lower percentage of correctly classified voxels, using 3D blocks resulted in an extremely poor percentage of correctly classified members of the segmentation set. This split in results suggested that there were two separate issues that were contributing to the poor results, and this can clearly be seen in figures 80 to 83.

First, the results from the processes based on 3D blocks were reviewed in detail. It is obvious from figures 80 and 81 that the classification of the transformed data was not a complete failure. The shape of the desired segmentation is visible in the raw results, but it is only a few blocks thick, and also suffers from noise around the edges allowing it to be quickly eroded by the low pass filters that were used.

When comparing against the equivalent 2D results (that can be seen in figures 82 and 83) it was seen that the raw results are generally stronger – although this gives rise to other issues such as the forming of bridges to areas that should remain outside of the segmentation set.

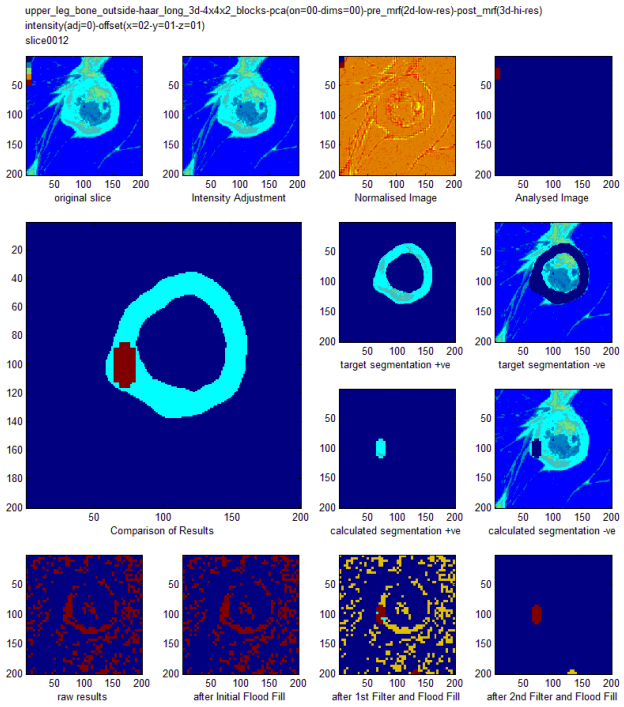


Figure 80: More complex segmentation, attempted using 3D Haar Long transforms, and post processing as described in table 28.

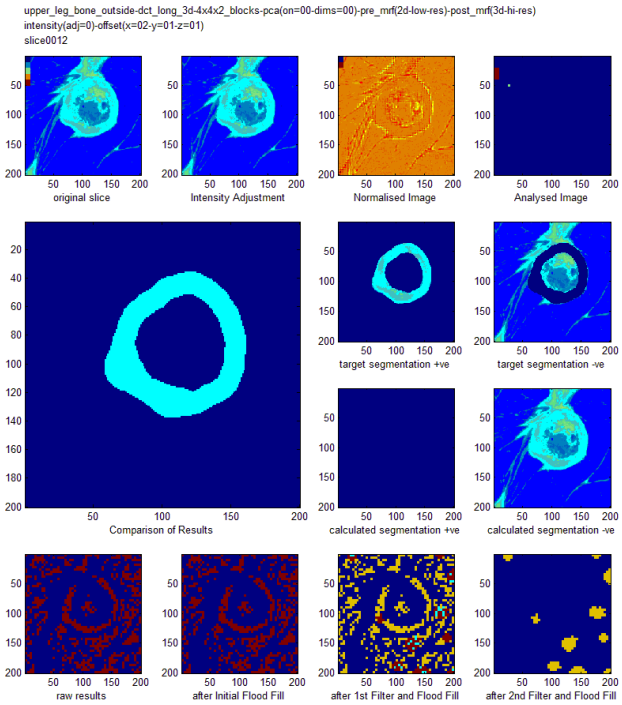


Figure 81: More complex segmentation, attempted using 3D DCT Long transforms, and post processing as described in table 28.

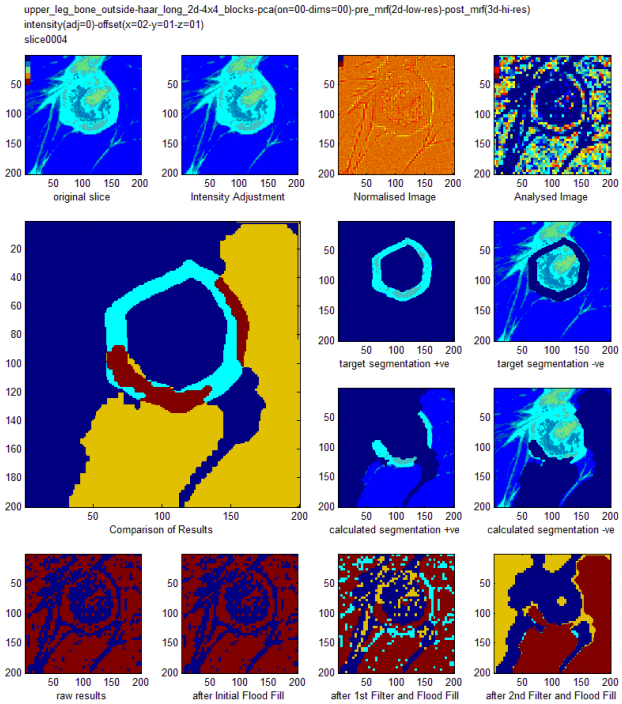


Figure 82: More complex segmentation, attempted using 2D Haar Long transforms, and post processing as described in table 28.

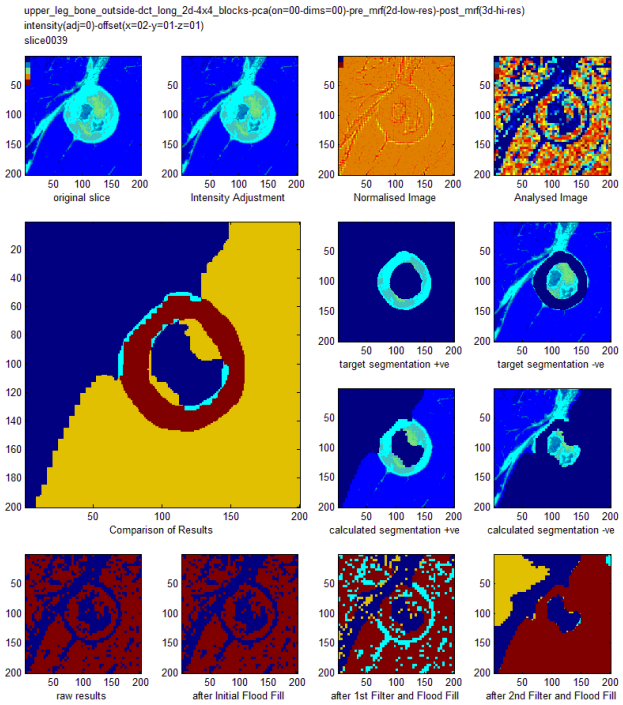


Figure 83: More complex segmentation, attempted using 2D DCT Long transforms, and post processing as described in table 28.

After review it seemed that the quality of the results based on 2D transforms was reduced because the texture on the outside of the segmentation set (once the DC component has been removed) appears quite similar to the texture within the segmentation set, which tends to result in bridges being more easily formed between the two.

It was noted that in many of the slices a high level of separation between pixels within and outside of the segmentation set was achieved. However, this separation was not perfect, and either bridges were part of the raw results, or formed as a result of filters being used in the post-processing steps.

To try and improve the results, an initial region growing process was used. This region growing was performed on the raw results, from a starting point that was known to be within the segmentation set (as with the other region growing that was already being used).

The segmentation was performed again, and the results reviewed. Table 30 outlines the post processing that was used in this experiment. Figures 84 to 87 show the effect of adding this initial region growing, and the numerical results are shown in table 31.

Post processing step number	Description
1	Strong region growing process
2	Symmetrical 2D block level low pass filter
3	Symmetrical 3D pixel/voxel level low pass filter
4	Strong region growing process

Table 30: outline of post processing used in the updated segmentation.

Transform	Block Size	% Ones Correct	% Zeros Correct	% Correct
DCT LONG 2D	4x4x1	70.1	97.4	94.4
HAAR LONG 2D	4x4x1	78.9	95.8	94.1
HAAR SHORT 2D	4x4x1	81.1	93.5	92.3

Table 31: Results seen when using an initial region growing process.

From looking at the slices shown in figures 84 to 87 it was seen that while the segmentation of some slices where bridges had previously been formed to remote regions had been improved, the addition of the initial region growing had also had a detrimental effect on some slices, where the segmentation set had been eroded too much.

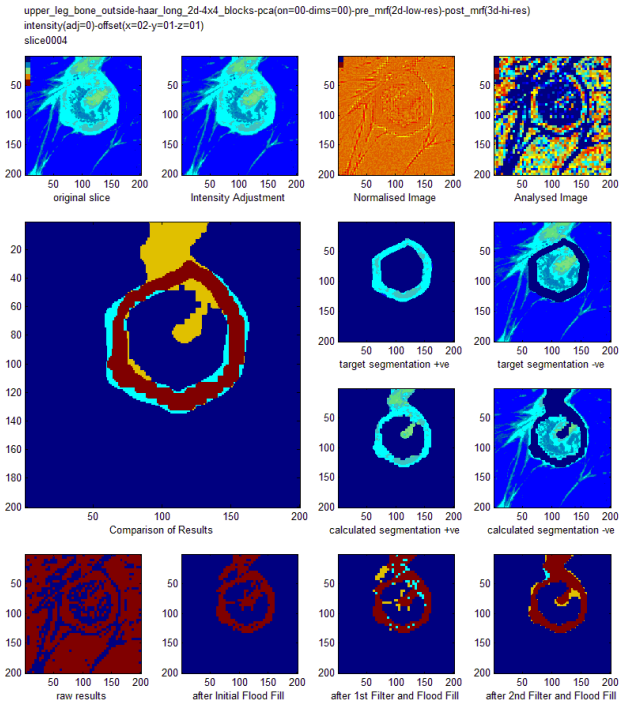


Figure 84: Segmentation preformed using 2D Haar Long transforms, and using post processing as outlined in table 30.

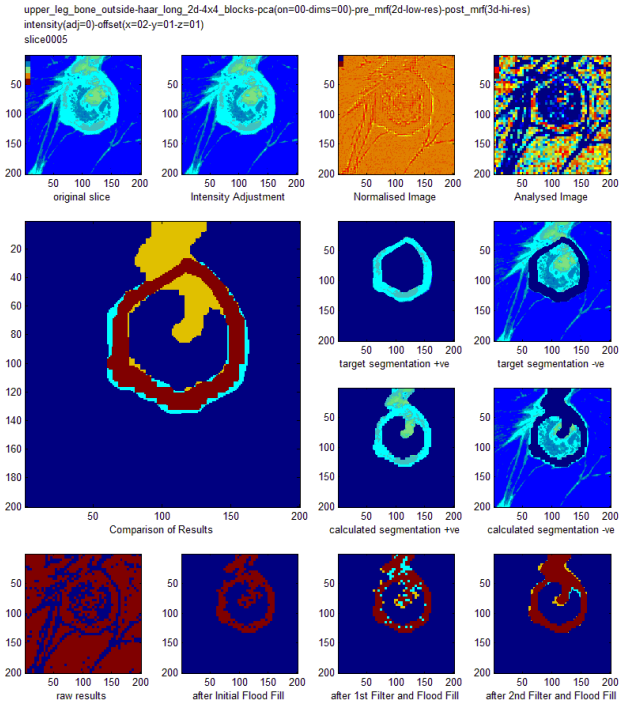


Figure 85: Segmentation preformed using 2D Haar Long transforms, and using post processing as outlined in table 30.

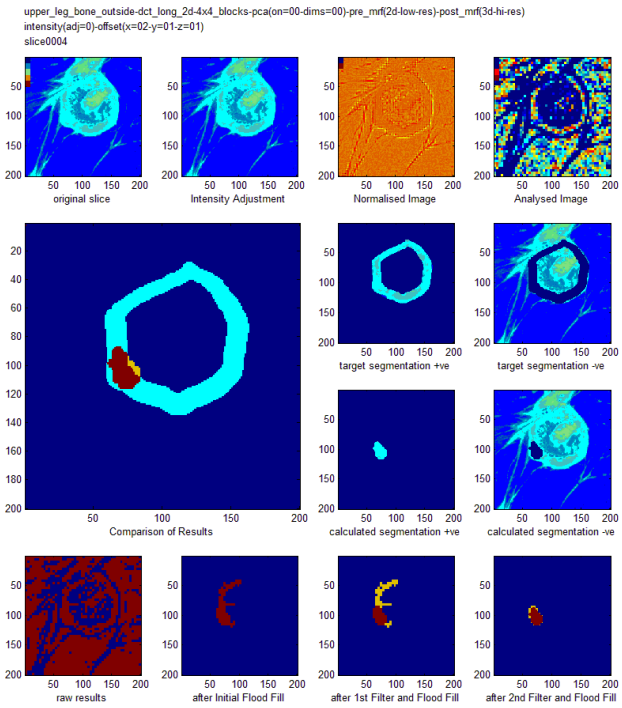


Figure 86: Segmentation preformed using 2D DCT Long transforms, and using post processing as outlined in table 30.

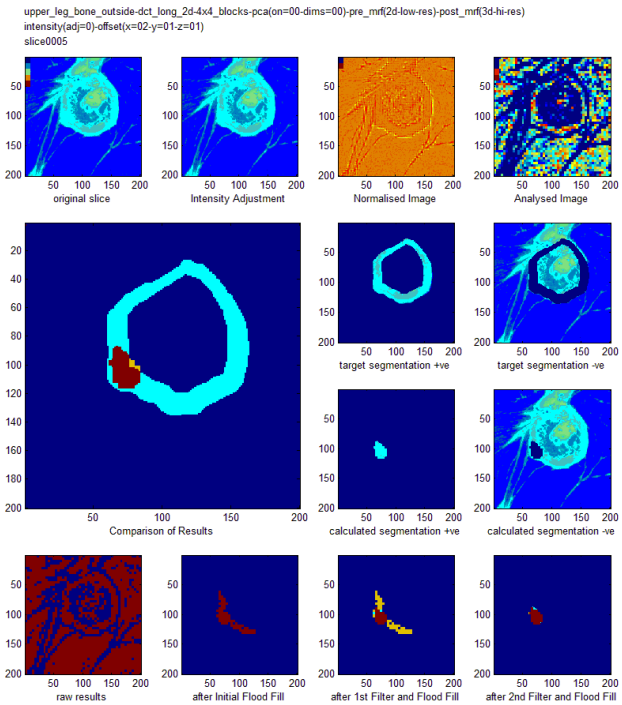


Figure 87: Segmentation preformed using 2D DCT Long transforms, and using post processing as outlined in table 30.

The region growing process was then weakened to try and find a good compromise between the benefits that it provided and the extra erosion that had been observed. The effect of weakening the region growing on an over eroded slice can be seen in figures 88 and 89. The experiments were carried out again, this time using post processing as outlined in Table 32.

Post processing step number	**Description**
1	Weak region growing process
2	Symmetrical 2D block level low pass filter
3	Symmetrical 3D pixel/voxel level low pass filter
4	Strong region growing process

Table 32: outline of post processing used in segmentation.

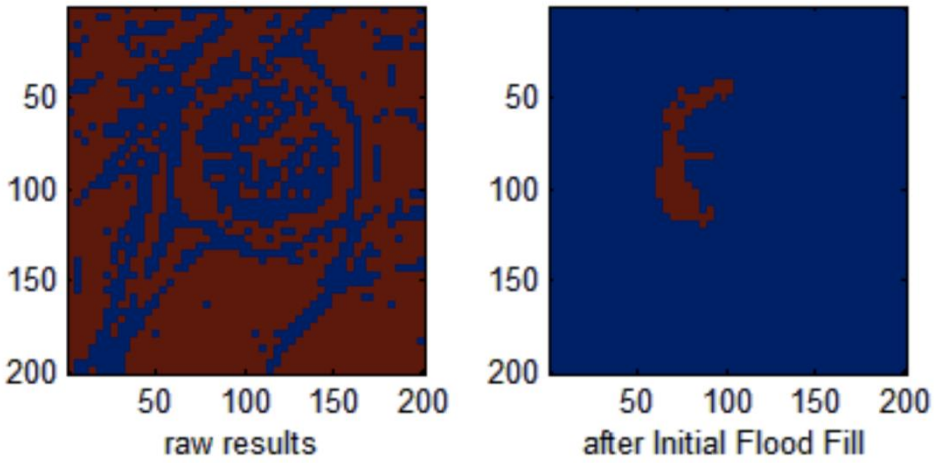


Figure 88: raw results (left) and results with after a strong region growing process had been applied (right).

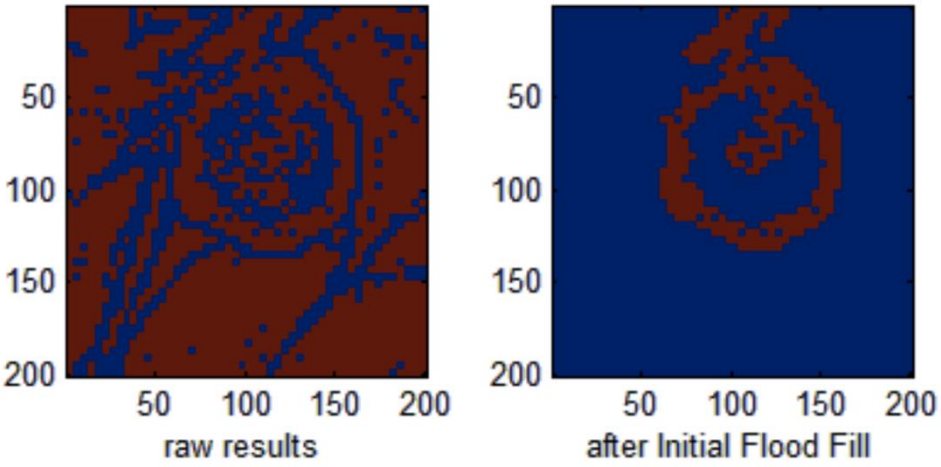


Figure 89: raw results (left) and results after a weak region growing process had been applied (right).

It was also important to make sure that weakening the initial region growing did not have significantly negative consequences on slices that had not suffered over erosion. Figure 90 and 91 shows that this is in fact the case, and the results are reasonably similar for both the strong and weak region growing.

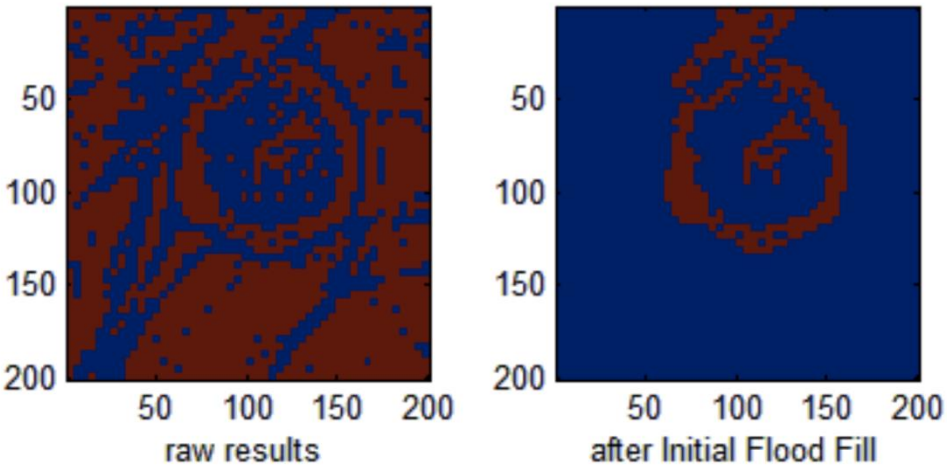


Figure 90: raw results (left) and results after a strong region growing process had been applied (right).

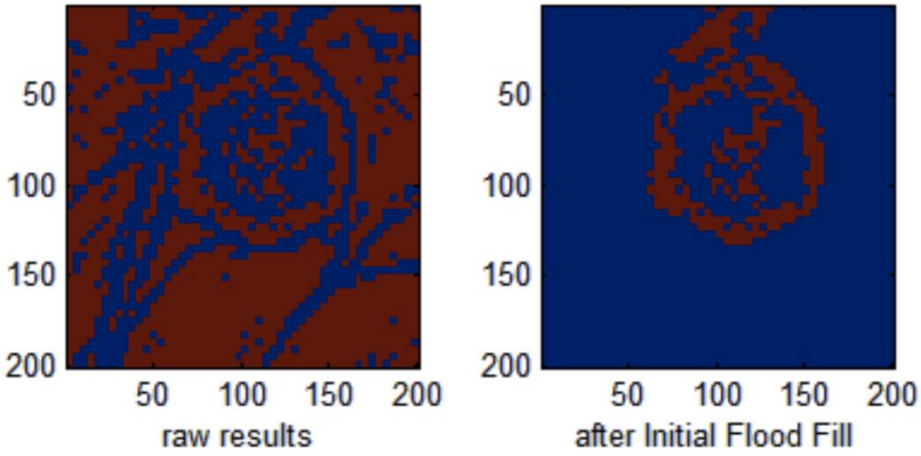


Figure 91: raw results (left) and results after a weak region growing process had been applied (right).

Whilst weakening the initial region growing can be seen to increase the quality of the interim results the final results were still found to be disappointing, and very similar to before. Figure 92 shows a typical segmentation that was generated as part of this latest experiment. It can be seen that the reason this slice has a poor segmentation is that the initial low pass filter is also responsible for over erosion of the segmentation set. This was in fact the same issue that was seen with the results of the processes based on 3D block sizes.

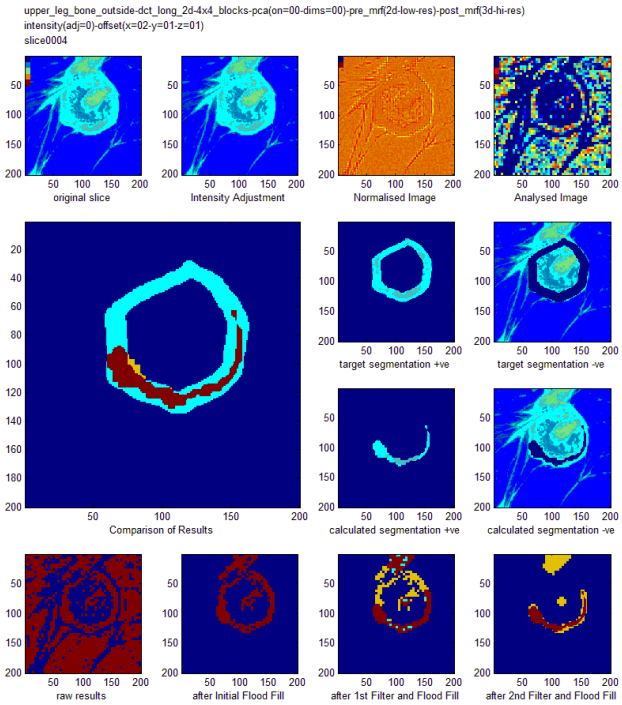


Figure 92: The initial low pass filter was responsible for over eroding the results.

The initial low pass filter being used was a symmetric low pass filter. The implementation of the filter is such that if a pixel is initially in the segmentation set, but enough of its neighbouring pixels are excluded from the set, then the original pixel would also be excluded from the set. The opposite is also true, in that a pixel outside the segmentation set would be included within the set, if enough of its neighbours are within the set.

It takes the same amount of energy (that is neighbouring pixels in the opposite state) to move a pixel into or out of the segmentation set. This is the symmetry of the filter. This symmetry was considered to be responsible for the over erosion of the segmentation set. The low pass filter was changed to be asymmetric, meaning that more energy would be needed to remove a pixel from the segmentation set, than to add a pixel to the set.

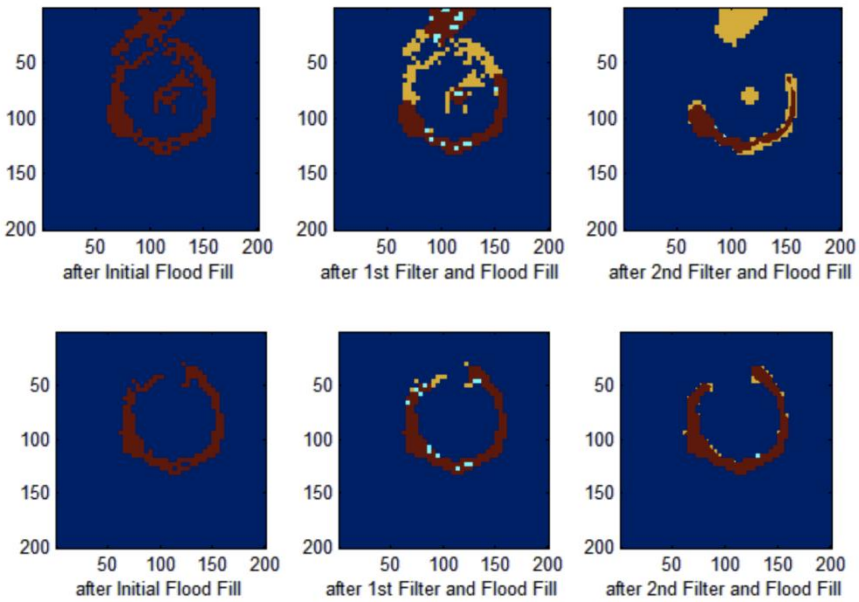


Figure 93: Using a symmetric low pass filter (top) and an asymmetric low pass filter (bottom).

The segmentations were performed once again, this time using post processing steps detailed in table 33.

Post processing step number	Description
1	Weak region growing process
2	Asymmetrical 2D block level low pass filter
3	Symmetrical 3D pixel/voxel level low pass filter
4	Strong region growing process

Table 33: Post processing steps that were carried out to investigate the use of different 2D transforms.

The results that were achieved, although subjectively better and more consistent than before, are objectively lower (especially in the % of ones that are seen to be correct). This data is shown in table 34.

Transform	Block Size	% Ones Correct	% Zeros Correct	% Correct
DCT LONG 2D	4x4x1	70.2	98.2	95.2
HAAR LONG 2D	4x4x1	69.8	96.3	93.5
HAAR SHORT 2D	4x4x1	68.5	97.9	94.7

Table 34: Comparison of results using different 2D transforms.

The final optimisation to the process was to add one more region growing process between the application of the two low pass filters. This was done as it was observed that in some cases remote regions had been formed at this stage, and they could be easily removed.

At this point the 2D and 3D results were generated again, using all of the post processing steps described in table 35. The results that were collected are shown in table 36.

Post processing step number	Description
1	Weak region growing process
2	Symmetrical 2D block level low pass filter
3	Weak region growing process
4	Asymmetrical 3D pixel/voxel level low pass filter
5	Strong region growing process

Table 35: summary of post processing steps used in the generation of the final results.

Transform	Block Size	% Ones Correct	% Zeros Correct	% Correct
DCT LONG 2D	4x4x1	71.0	97.9	95.0
DCT LONG 3D	4x4x2	36.9	100.0	93.1
HAAR LONG 2D	4x4x1	69.2	98.1	94.9
HAAR LONG 3D	4x4x2	55.2	99.3	94.5
HAAR SHORT 2D	4x4x1	72.2	97.6	94.8
HAAR SHORT 3D	4x4x2	61.5	98.1	94.1

Table 36: Final results for the complex segmentation set, using post processing as described in table 35.

In summary, although the results are showing an improvement with the full set of post processing in place, they are still not of a useful quality, so at this point the direction of the research was reviewed and new techniques, which could be used in conjunction with some of the methods previously developed, were considered.

7.0 Moving to a model-based approach

The results obtained in previous chapters have shown that DCT and HAAR transforms (in various configurations) can give a good measure of the texture of different tissues seen in medical images. However, as a measure they do not by themselves provide good enough results to be useful in a real segmentation application. Something more is required.

In this chapter a model-based approach is considered. In a model-based approach some prior knowledge of the object being segmented is included in the technique. The approach that was chosen was to use an Active Appearance Model or AAM. AAMs have been used in the field of image segmentation before, typically in the field of face and facial expression recognition. The AAM is described in more detail in section 2.3.8.

There are some aspects of the AAM that make it a particularly appropriate choice for use with medical images. Firstly it works well where there is a common structure to the images being considered. For example figure 94 shows some images from the Cohn-Kanade dataset. Although the individual faces and expressions in each image vary, all the faces have a recognisable structure in common (made up of the features of the face), are positioned in a consistent manner, and are free from occlusion. All of these things are important to allow a useful model to be constructed, and can also be said to be true when considering medical images of specific anatomical features.

The appearance model that is created during use of an AAM has two components; shape and texture. The shape component is based on the landmarks which have been annotated onto the training images. In the case of using an AAM to identify faces, landmarks might include locations such as the corners of the eyes, or the corners of the mouth. The texture model is based on some measure of the texture of the image, but most commonly the simplest measure of texture is used, that of pixel intensity.

Figure 94: Sample of images from the Cohn-Kanade dataset.

This provides good results with a low cost (i.e. low computational time and energy required to use the AAM to segment an image). However, as previously stated, medical imaging techniques are commonly affected by unpredictable variations in intensity across the image, and so using intensity as a simple measure of texture is unlikely to give good results.

Work has been done to apply more complex measures of texture to build the texture model, but studies have shown that while results are of a high quality this has been traded off against a computationally expensive process, resulting in an equally impractical solution.

The initial investigation has shown that using an mDCT (modified DCT, a normal DCT with the DC coefficient removed) can provide an effective intensity invariant description of different textures as found in the data sets so far.

So the investigation was carried forward into a second phase, to investigate if a novel combination of a simple measure of texture robust to variations in intensity (such as mDCT) could be used to build the texture model of an AAM that could

then provide good quality segmentations of medical images within a reasonable computational budget.

An obvious measure of this is for the technique to be comparable to a manual segmentation, both in terms of quality of results, but also in terms of the time required to perform the segmentation considering use of a sensible hardware specification.

To carry out this second phase of the research a larger dataset was required. AAMs have been used in both 2D and 3D applications. In the case of a 3D application, the AAM needs a number of complete example 3D segmentations for training purposes, and then a reasonable amount of test data is also required.

The eye was chosen as a suitable anatomical structure for experimentation. This was for several reasons:

1) The Female Visible Human Dataset has 3 sets of high resolution images resulting in 3 left eyes and 3 right eyes.
2) Eyes can be manually segmented by someone who is not an expert, giving a reasonable quality result (for training and scoring purposes).
3) There is internal structure, which makes them ideal for use with an AAM.

Six new eye datasets were therefore constructed by extracting six volumes from the three sets of images from the Female visible human dataset. Two volumes of size 150x150x43 slices were extracted from each, one containing the left eye and one containing the right eye. Each of the eyes was manually segmented in a similar fashion as previously carried out on the other data sets.
The resulting six eye datasets were numbered from 1 to 6 for identification purposes. It was also decided that this point that each of the 6 datasets should either be used for training or for test purposes. This is summarised in table 37.

Dataset Number	Female Visible Human dataset	Left or Right eye	Used for training or test
1	A	Left	Training
2	B	Left	Training
3	C	Left	Training
4	A	Right	Test
5	B	Right	Test
6	C	Right	Test

Table 37: Summary of datasets that were created.

In each eye dataset, around 30 slices actually contained parts of the eye's segmentation set, and so needed marking up and segmenting. Marking these up took around 4 hours, with an average time of about 80 seconds per slice, and around 40 minutes per eye. This information is included to provide some level of reference to compare the AAM run times against.

Once the manual segmentations were completed, the results were then rescaled to 60%, 70%, 80%, 90% and 100% of original size. The rescaling was done to increase the size of the test set, and also to increase the amount of data available to validate the approach with. The technique used to rescale the results was based on removing a proportion of the pixels rather than using a more complex method such as interpolation.

To increase the size of the test set still further, the eyes were then distorted, by applying constrained random and known intensity variations to the 3 colour channels of the images. A brief description of the distortions that were applied is given in table 38, with more detail in the following sections. The number of samples available for each of the 6 eye datasets is listed in table 39.

Type of distortion.	Description.
No Distortion.	Intensity of each of the 3 colour channels and for each voxel in the images is left unmodified.
Flat reduction in intensity across all voxels.	Intensity value for each colour channel of each voxel of the images is reduced by a constant offset.
Flat increase in intensity across all voxels.	Intensity value for each colour channel of each voxel of the images is increased by a constant offset.
Constrained random distortion of intensity on each of the 3 colour channels.	Intensity value for each colour channel of each voxel of the images is increased by an amount based on the location of the voxel within the image.

Table 38: Summary of techniques used to distort image data.

	Scale of dataset.				
Type of distortion.	**60%**	**70%**	**80%**	**90%**	**100%**
No Distortion.	1	1	1	1	1
Flat reduction in intensity across all voxels	1	1	1	1	1
Flat increase in intensity across all voxels	1	1	1	1	1
Constrained random distortion of intensity on each of the 3 colour channels.	20	20	20	20	20

Table 39: Summary of test datasets that were created.

7.1 Detailed description of distortions applied to datasets.

The following sections more formally describes the distortion that was applied to the image datasets to increase the amount of data available for testing of the techniques that were developed.

7.1.1 No Distortion

When no distortion is applied to the input data all three colour channels of the input data remain unaffected. This is described by equation 7.1.

$$D_{c,x,y,z} = O_{c,x,y,z} \tag{7.1}$$

Where:$D_{c,x,y,z}$ is the new intensity value for colour channel c of the voxel at location x, y, z within the dataset.

$O_{c,x,y,z}$ is the original intensity value for colour channel c of the voxel at location x, y, z within the dataset.

7.1.2 Flat reduction in intensity across all voxels

When a flat reduction in intensity across all voxels is applied, a constant reduction of the intensity of all colour channels is applied to all voxels irrespective of their location within the dataset. This is described by equation 7.2.

$$D_{c,x,y,z} = O_{c,x,y,z} - R \tag{7.2}$$

Where:$D_{c,x,y,z}$ is the new intensity value for colour channel c of the voxel at location x, y, z within the dataset.

$O_{c,x,y,z}$ is the original intensity value for colour channel c of the voxel at location x, y, z within the dataset.

R is a constant and is the amount by which the intensity value is reduced.

7.1.3 Flat increase in intensity across all voxels

When a flat increase in intensity across all voxels is applied, a constant increase of the intensity of all colour channels is applied to all voxels irrespective of their location within the dataset. This is described by equation 7.3.

$$D_{c,x,y,z} = O_{c,x,y,z} - I \tag{7.3}$$

Where: $D_{c,x,y,z}$ is the new intensity value for colour channel c of the voxel at location x, y, z within the dataset.

$O_{c,x,y,z}$ is the original intensity value for colour channel c of the voxel at location x, y, z within the dataset.

I is a constant and is the amount by which the intensity value is increased.

7.1.4 Constrained random distortion of intensity

In reality the intensity variations that are often seen in medical images are not of the constant shift style described in the previous sections. Typically the type of intensity variation that is seen is a progressive variation, although location, gradient and size is not predictable, and is influenced by many factors including the nature of the medical images, the part of the body being imaged, the age of the patient, the exact equipment being used to gather the images (one MRI scanner can perform differently to another for example).

Rather than try and model this complex intensity variation directly, a random distortion was applied to the datasets. This random distortion was used to test the approaches robustness against intensity variation in general, rather than to test against a specific model and/or intensity variation that a particular medical imaging process might be susceptible to.

Defining the correct distortion here was critical to the usefulness of the testing of the technique. Clearly if a large enough independently random distortion had been applied to each of the colour channels of each voxel within the image, all meaningful data from the image will be lost. At the same time, if the magnitude of the distortion is too small then the testing tells us much less about the robustness of the technique against this unpredictable intensity variation.

Therefore a constrained random approach was chosen. This is where the randomness applied to the intensity variations are controlled, just enough to stop the useful information in the image from being lost, but not so much that the

testing becomes less useful. Using this technique a progressive distortion was applied, based on the sin function. Each colour channel of the dataset was modified independently, although using the same method. A small constraint was added to the randomisation to avoid very high frequency distortion being applied to the images (which would not model realistic variations in intensity seen in medical images).

The distortion is described by equation 7.4.

$$F_{c,x,y,z} = O_{c,x,y,z} + M_c \left(\sin\left(\frac{2\pi(x+O_x)}{s_x}\right) + sin\left(\frac{2\pi(y+O_y)}{s_y}\right) + \sin\left(\frac{2\pi(z+O_z)}{s_z}\right)\right) \quad (7.4)$$

Where:$D_{c,x,y,z}$ is the new intensity value for colour channel c of the voxel at location x, y, z within the dataset.

$O_{c,x,y,z}$ is the original intensity value for colour channel c of the voxel at location x, y, z within the dataset.

$S_x = R_x + F_{limit}$

$S_y = R_y + F_{limit}$

$S_z = R_z + F_{limit}$

R_x, R_y, R_z represents randomly generated numbers between 1 and X_n, Y_n, Z_n.

X_n, Y_n, Z_n represents the dimensions of the image in the X, Y and Z directions.

O_x, O_y, O_z represents randomly generated offsets (between 1 and X_n, Y_n, Z_n).

F_{limit} represents a constraint to avoid very high frequency distortions being added to the images. A value of 25 was chosen for F_{limit}.

M_c represents a scaling factor for colour channel c. The value for M_c was randomly selected in the range of 0 to 50% of the maximum range of intensity values. Suitable measures were taken in the implementation to avoid issues with clipping.

7.2 Initial work using a 2D AAM

The use of an AAM was trialled using a 2D AAM. Slices were taken from the data set, which showed horizontal cross sections through the central part of the

eye ball. As such they were spatially related, and all showed similar internal structure, and so could be used to construct a sensible appearance model.

Examples of some of the slices that were used are shown in figure 95, along with the associated segmentations in figure 96.

Figure 95: Various slices used to investigate a 2D AAM

Figure 96: Manual segmentation of slices shown in figure 87

The common structure can easily be seen. The AAM was built using Matlab code derived from functions which were originally written by D. Kroon from the University of Twente (March 2011) [20]. Kroon's original examples showed the AAM working on a set of images of hands, each with a set of landmarks defined. The AAM used a texture model based on pixel colour, therefore each pixel had a three dimensional representation of its texture.

Kroon's Matlab code includes all the functions required to create and train an AAM, and also to search previously unseen images using the AAM. The main features of the code are described below.

The first step (creating and training) requires the functions to be provided with a set of training data. The training data includes a set of images and associated landmark data. The landmark data is supplied for each image, and consists of an ordered list of co-ordinates for each individual landmark. It is important to note the ordered nature of the list, the order the landmarks are presented in must be maintained between different training images. Put another way, the first landmark detailed in each training image's list should provide the location of the same feature within the image.

The next major step is to normalise the shape of each of the input images. This is done in two parts. Firstly the mean location of all the landmarks is found. Secondly, the warping function is used to normalise the shape of the image. The warping function takes as inputs the image (which is a three colour image), the location of the landmarks for that image, and the mean location of the landmarks (as previously calculated). It uses this information to perform the warping of the image and returns this (again as a three colour image). An important point to note here is that the warping of each of the three colour channels (red, green and blue) is independent from the others. That is, varying the data held in the blue channel has no effect on the warping of the red or green channels.

At this point, there exists a shape normalised version of all the training images, and the mean locations of all the landmarks (and therefore the mean shape) is also known. The number of pixels held within the mean shape can now be found, and a vector is created for each shape normalised training image to hold the texture data for the shape normalised patch. In Kroon's code, the vector for each image is sized to be 3x the number of pixels within the mean shape. This provides enough storage for the red, green and blue colour channels for each pixel. The mean texture can then be calculated.

Kroon's code then uses the cov and eigs matlab functions to perform PCA over the mean shape and the mean texture data again to combine these two models into an appearance model.

Once the appearance model has been constructed the code then continues to investigate the relationship described in equation 2.29 as described in section 2.3.8.

Once this process is complete, the next step is to use the model to search a previously unseen image for the object of interest. Kroon also provides a number of functions to allow searching using the model. The search function requires the previously unseen image to be provided as an input, along with an initial estimate of the location of the object of interest. This location is provided as co-ordinates that specify the centre of the mean shape. The function then

follows a multi resolution approach (suggested by Cootes et al.) to search the image [18]. Using this approach leads to a more efficient run time, but does not affect the overall quality of results. Once the model has been searched the functions exist that can be used to back annotate the mean shape to the image under investigation. This allows the shape of the object in the image being investigated to be highlighted.

To start the investigation into the use of a 2D AMM, Kroon's code was used unmodified. This allowed the construction of an AAM (using a 3 channel intensity based texture model) which could be used to perform segmentation on undistorted and distorted eyes from the dataset.

Although, at this stage, Kroon's code did not have to be modified, the input data had to be created, gathered and formatted correctly so it could be passed to his functions. To train the model using the slices of the eye, sensible landmarks had to be provided for each slice. The eyes all had similar orientation, and manual segmentation data for each slice had already been created, hence it was possible to automate this process. To generate the landmarks for each of the eyes the central location within the segmentation set was found by taking the mean position of all pixels within the segmentation set. From this central point the boundary of the set directly north, and then at other bearings (working clockwise at a suitable angular interval) was found. This results in the landmarks being at the boundary of the segmentation region, and spaced at equal angles. Figure 97 shows two sets of landmarks that were generated using this technique. The position of landmarks relative to the segmented region can be seen. Note that it was decided to generate 25 landmarks on each slice.

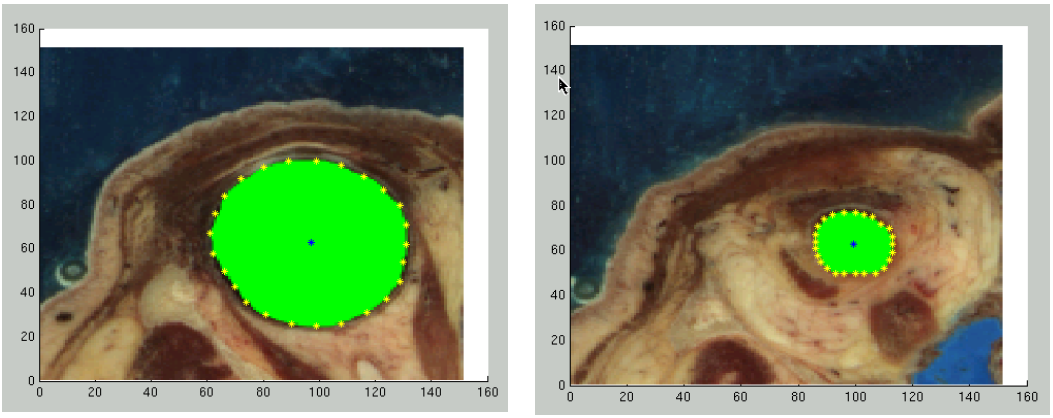


Figure 97. The manual segmentation of two slices, with the segmenation set shown in green. The blue star marks the calculated centre point of the sementation, and the yellow stars mark the calculated boundary points which are used as landmarks.

To allow landmarks to be specified with a higher level of accuracy, it was decided that a block-based approach would no longer be taken. In the previous work images were split into blocks, and each block's texture was analysed. One of the issues that had to be overcome was the associated reduction in resolution and the blocky nature of the raw results. To avoid such issues, each pixel or voxel in the image would now have its own to texture analysed based on the intensity of it and some of its neighbours. However, this means that many more DCT or wavelet transforms need to be carried out per image, which will have an effect on the practicality of the process. To counteract this, it was decided that only small transforms (of size 2x2 or 2x2x2) would be done. A side effect of this decision was that using these sizes the DCT and Haar transform are equivalent. Therefore the remainder of the work was carried out based on DCTs.

Once this had been done, Kroon's unmodified functions were successfully used to obtain some initial results. However the unmodified code was limited to using exactly three channels to describe the texture of each pixel. This restriction was removed in order to allow the functions to be used in the investigation of a number of different texture descriptions, some of which required more or less data to describe the texture of each pixel. The texture descriptions under consideration are detailed in Table 40.

Texture Description	Channels required.	Comments
Monochrome intensity.	1	Each pixel is described by a single intensity value.
Full colour intensity.	3	Each pixel is described by three intensity values. One for each of red, green and blue intensity.
mDCT on monochrome image.	3	The mDCT transforms 4 pixels (from a block of 2x2 pixels) of a monochrome image into 3 values describing the texture of that 2x2 block.
mDCT on full colour image.	9	When used on a colour image, the mDCT is performed independently on the red, green and blue intensity values. This generates 3 sets of 3 values, describing the texture of the block in terms of each colour separately. These 9 values can be combined into a single texture description describing the texture of the full colour bock.

Table 40. List of texture descriptions under investigation.

The unmodified code assumed that the training images supplied, and the images that would be searched all had 3 channels, and that this would also be the number of channels required to describe the texture of the pixel. This relationship was true because the texture data and the intensity data in the image were assumed to be equivalent. Rather than break this assumption, it was decided the simplest way to allow Kroon's code to work with a range of difference texture descriptions was to convert the images into a texture description before passing them to Kroon's functions. In this way most of Kroon's code could continue to operate with no adjustment, and only a few

minor adjustments to generalise the number of channels that input images contained would be required. With this in mind, three specific updates were made to the functions.

The first of the updates was to enable images with a general number of dimensions to be warped. The unmodified code assumed that the images supplied to the warping function would have exactly three channels. This was a trivial update, as the warping of each channel is actually entirely independent, and so can be run on more or less channels as required.

The second update was to the creation of the texture vectors used in building the appearance models. The code assumed the vectors used for storing the texture data would always have a size of exactly 3 times the number of pixels within the mean shape. However, it was relatively easy to generalise the creation of the vectors, and associated code, allowing the number of channels being used to be specified as required.

The final change was to the code which allowed previously unseen images to be searched by the AAM. Part of this code displayed the image, augmented with the suspected shape and location of the object being searched for. However, although the function only needed minor changes to allow it to search images with a generalised number of channels, when it came to displaying the results (specifically the augmented image) displaying an image based on an arbitrary number of channels did not make any sense. To resolve this issue, the original (3 channel) image being searched was stored, and used for the last part of the process.

Another modification to the example code was to automate the selection of a starting point. In the real world a user might be asked to select a starting point near the centre of the eye. This could be done using a graphical interface and a single click by the user. When generating the results taking this approach would be tiresome and would result in a lack of repeatability in the recorded results. Therefore the manual segmentation data was used to find the centre location of the eye, and then an offset was applied. This was done to simulate an inaccurate selection of the centre point of the eye by the user. It was decided

that a consistent offset would be used to help keep the results comparable. During informal testing it was noted that generally speaking the size of the offset is inversely proportional to the quality of the results.

To choose the size of the offset an experiment was carried out. A simple game was created where circles of random sizes and centre positions were drawn and a user was asked to click at the centre of the circle. The time taken to complete the game was recorded and users were encouraged to aim for a low overall time as well as being as accurate as possible. The results were then used to help ascertain the accuracy with which a typical operator would be able to click at the centre of a structure. The results showed that it was possible to carry out this task with an accuracy level of better than 2% (the user could consistently click less than 2% away from the centre of the circle, with respect to the diameter of the circle) spending an average of around 2 seconds per circle. As the eye is not perfectly circular, and it is harder to see the boundaries, an offset considerably larger than this was chosen, roughly an X and Y offset of 10% of the diameter of the eye, or around 14% distant from the centre of the eye with respect to the diameter.

7.3 Results of 2D AAM-based segmentation

This initial use of an AAM provided promising results. The training data that was used to generate these results was taken from data set 1 (visible human female, data set A, left eye). The test set used was based on data set 5 (visible human female, data set B, right eye). The amount of test data available was increased by applying various distortions to the testsets, as detailed in table 41.

Appearance Model Used	Test Set
Intensity	Undistorted
Intensity	Constrained random distortion
mDCT	Undistorted
mDCT	Constrained random distortion

Table 41: description of test sets used to evaluate effectiveness of a 2D AAM.

The results achieved are summarized in the following tables (tables 42 to 45), each result being specified as a percentage of pixels whose classification is in

agreement between the manual and automated segmentation. Note that a simple colour coding is used:

- GREEN: 99% agreement between manual and automatic segmentation.
- AMBER: 90% to 98% agreement between manual and automatic segmentation.
- RED: below 90% agreement between manual and automatic segmentation.

	Scale (as percentage of original image)				
Slice No.	**60**	**70**	**80**	**90**	**100**
1139	99	99	99	99	99
1138	99	99	99	99	99
1137	99	99	99	99	99
1133	99	99	99	99	99
1132	99	99	99	99	99
1135	99	99	99	99	99
1134	99	99	99	99	99
1136	99	99	99	99	99
1131	99	99	99	99	99

Table 42: Results when using intensity-based appearance model, and undistorted test set.

	Scale (as percentage of original image)				
Slice No.	**60**	**70**	**80**	**90**	**100**
1139	89	93	96	98	91
1138	95	99	92	99	90
1137	92	97	91	97	96
1133	99	99	99	95	94
1132	99	89	96	98	92
1135	95	94	98	99	95
1136	93	97	99	98	97
1134	97	98	99	98	97
1131	92	96	90	89	99

Table 43: Results when using intensity-based appearance model, and constrained randomly distorted test set.

	Scale (as percentage of original image)				
Slice No.	**60%**	**70%**	**80%**	**90%**	**100%**
1139	81	92	99	99	99
1138	72	79	99	99	99
1137	81	90	99	99	99
1133	80	91	99	99	99
1132	81	88	99	99	99
1135	83	89	99	99	99
1136	79	79	99	99	99
1134	83	87	99	99	99
1131	85	93	99	99	99

Table 44: Results when using mDCT-based appearance model, and undistorted test set.

	Scale (as percentage of original image)				
Slice No.	**60**	**70**	**80**	**90**	**100**
1139	79	92	99	99	99
1138	75	86	94	99	99
1137	78	89	99	99	99
1133	83	92	99	99	99
1132	82	88	94	99	99
1135	80	92	99	99	99
1136	84	83	99	99	99
1134	84	88	99	99	99
1131	82	83	99	99	99

Table 45: Results when using mDCT-based appearance model, and constrained randomly distorted test set.

As can be seen from these initial results there are advantages and disadvantages to using intensity or mDCT based appearance models. The intensity model is more robust to variations in scale, but fails to cope with distortions imposed in the test set almost completely. Conversely the mDCT model is mostly successful with distortions imposed in the test set, but fails to cope well with variations in scale.

This is maybe not so surprising, as the technique used to rescale the datasets involved removing data from the samples, rather than interpolation. The idea behind this was to limit the effect of the rescale on the frequency domain, but there will always be some effect if pixels are being removed.

At this point it is interesting to consider the subjective quality of the results achieved and the modes of failure when the process is not successful. Slice 1134 is an example of a slice which has only been subjected to a mild amount of distortion in the constrained randomly distorted test set, yet when using the intensity model the results are poor. However, the mDCT has generated good results. The figures below show a comparison between the manual and automated segmentations using each model. The plots shown in the following pages are made of 9 sub plots showing:

1) The original image (top left)
2) A representation of the appearance model (which is either the original image or the output of an mDCT performed on 2x2 blocks over the image). (top centre)
3) A comparison between the automated and manual segmentation. (top right)

4,5 and 6) A summary of the manual segmentation including the binary segmentation data, a +ve and –ve segmented view of the image.

7, 8 and 9) A summary of the automatic segmentation including the binary segmentation data, a +ve and –ve segmented view of the image.

Figure 98 shows the effect a small level of distortion has on the results generated when using an intensity based appearance model. Note that although the location of the eye has been found, the boundary of the eye is not in the right place.

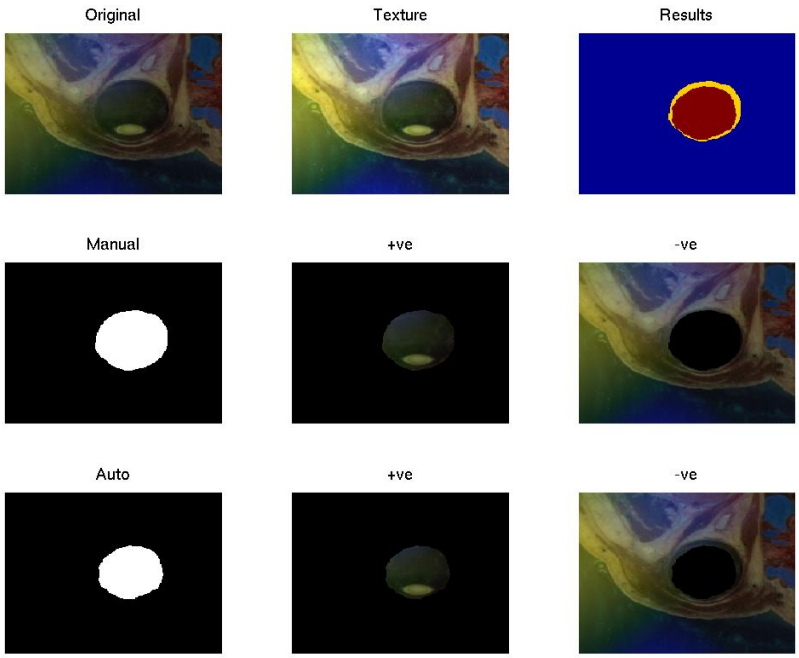


Figure 98: Segmentation generated when a small level of distortion applied to the test dataset, and an intensity based AAM is used.

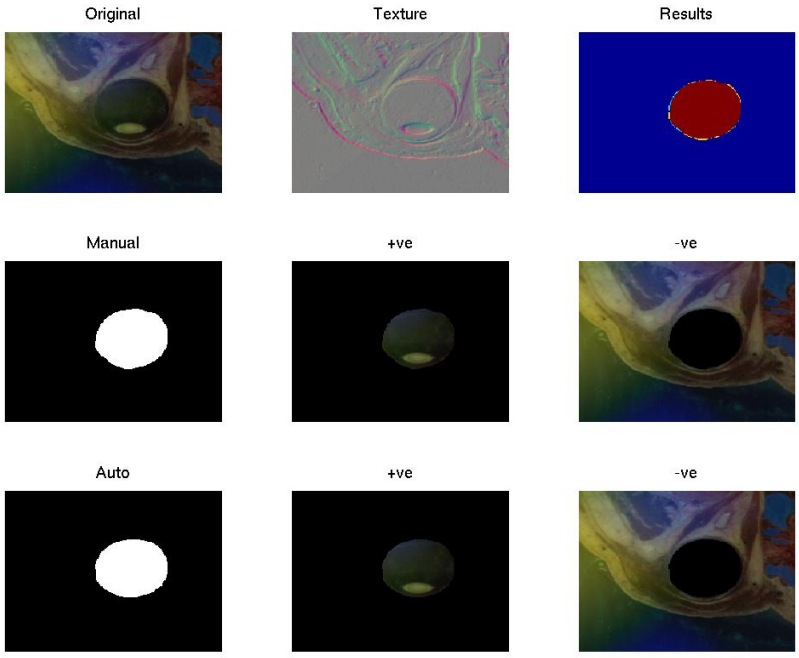


Figure 99: Segmentation generated when a small level of distortion applied to the test dataset, and a mDCT based AAM is used.

When an mDCT based AAM is used however, the results are not adversely affected by the small level of distortion in the input data. This is shown in figure 99.

Next, consider the case of a slice which has been exposed to heavier levels of distortion. A higher level of distortion on all three colour channels has led to a complete failure as shown in figure 100.

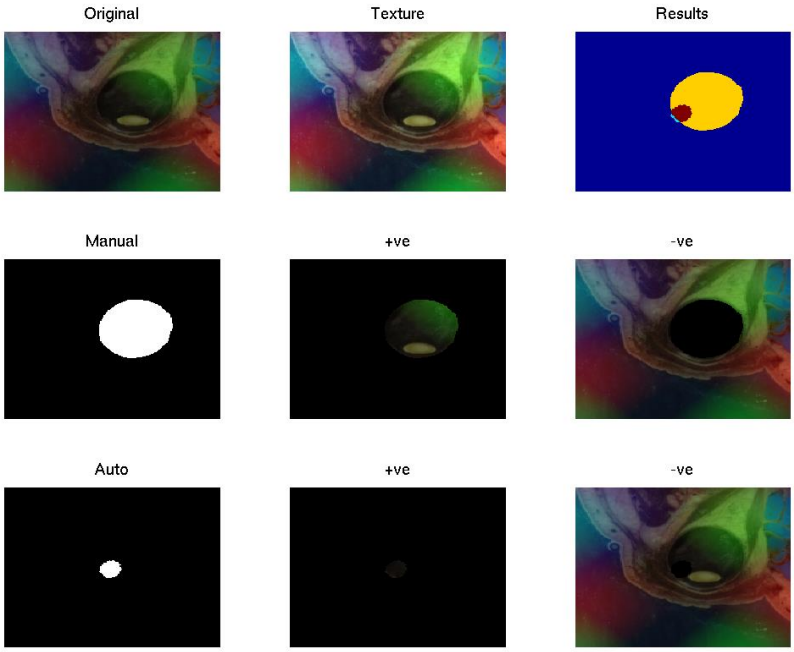


Figure 100: Slice 1139 when segmented using intensity based appearance model.

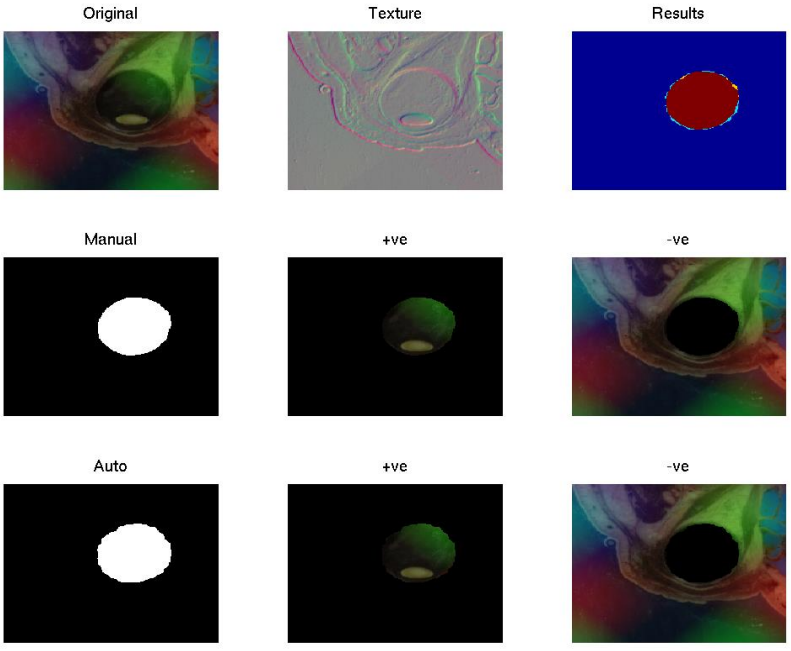


Figure 101: Slice 1139 when segmented using an mDCT based appearance model.

Once again, using the mDCT based appearance model has proved successful, despite the heavier levels of distortion, and the boundary has been correctly identified. This is shown in figure 101.

When comparing the mDCT based texture models of the undistorted and distorted versions of slice 1137 (see figure 102), it can be seen that they are very similar (which of course is the point of using the mDCT model). The RMSE between the output of the two mDCTs is just 1.75. To help put this into context it is useful to know something about the variance of the associated data. The variance of the field used when applying distortion to the original slice was calculated as 100.8. The variances of the output of the mDCTs for the undistorted and distorted slices were calculated as 121.9 and 127.9 respectively. These results show that the output of the mDCT texture analysis is largely unaffected by the distortion applied to slice 1137. Therefore an mDCT based AAM is a good candidate for performing successful 2D segmentations, even in the presence of substantial intensity variation, and so the next part of the research focused on building a 3D AAM and using it to perform a segmentation of an entire eye.

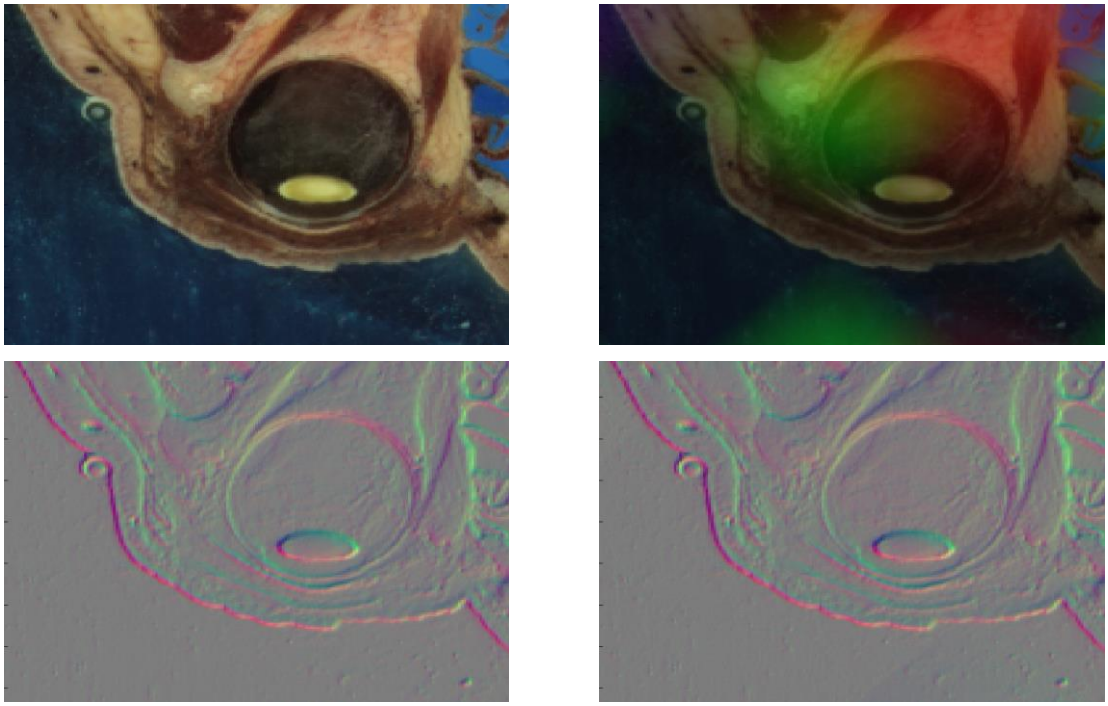


Figure 102: Undistorted slice 1137 (top left). Distorted slice 1137 (top right). Output of mDCT texture analysis of undistorted slice 1137 (bottom left). Output of mDCT texture analysis of distorted slice 1137 (bottom right).

7.4 Automated Segmentation of Full Eye using a 3D AAM

After seeing the success of the 2D model it was decided to continue to investigate if a similar quality of results could be achieved when using a 3D AAM to segment the whole eye.

The convention of training the AAM using data from the left eyes, and testing being carried out with data from the right eyes was maintained. Note that the slices containing the right eye were flipped such that they then had the same orientation as the left eye.

Kroon's example code also included a set of functions that could be used to train and use a 3D AAM. There were also functions supplied demonstrating their use, in constructing an appearance model of a jaw bone.

Again Kroon's functions assumed that the 3D image data was of a fixed number of channels. As the functions associated with the 3D AMM were very similar in structure to those associated with the 2D AMM, a similar set of modifications was carried out to allow the number of channels in the image data to be generalised.

Again the approach was taken to transform the input images from intensity based images into texture description based images before passing them to Kroon's functions, and this worked well.

When considering the supplied example, it was noted that the definition of landmarks was now a little more complex. Rather than just supplying an ordered list of landmarks in three dimensional space, a surface model was created by also defining triangular faces, each face having vertices at specific landmarks. It was the resulting wireframe model that was analogous to the shape created when working with the 2D AMM.

Some new code was written to automatically create a suitable surface model of each eye from the manual segmentations. The first part of creating the surface

model involved a similar approach as taken previously, automatically defining landmarks on each 2D slice. Once the landmarks had been defined for each slice, the slices were combined into a 3D space. The surface model was then built by listing a series of triangular faces, each face described by listing the landmarks that supply the position of its vertices.

As each slice had the same number of landmarks, the algorithm used to specify each face was a simple one. Figure 103 shows a small group of land marks from two slices (slice S and slice S+1). To construct all the faces required to describe the surface between these two slices, initially a triangular face is defined with a single vertex from slice S+1 (landmark N) and with two vertices from slice S (landmark N and N+1). A second face is then described with two vertices from slice S+1 (landmark N and N+1) and a single vertex on slice S+1 (landmark N+1). This process is then repeated around all the landmarks, to provide the complete surface between these two slices, and the whole process is repeated for all pairs of adjacent slices.

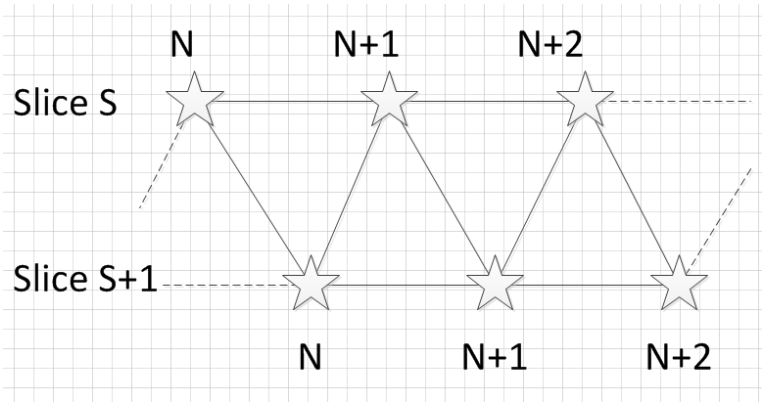


Figure 103: A view of some of the landmarks of two adjance slices. The verticies of each triangular face are landmarks (marked with stars).

All that remains in order to complete the surface model is to close the top and bottom of the model. To do this faces are constructed between each pair of adjacent landmarks on the top slice and the central point (as calculated earlier). This is then repeated for the bottom slice.

An example of a completed surface model is shown in figure 104. Iregularities arrising from the non-expert nature of the segmentation are apparent.

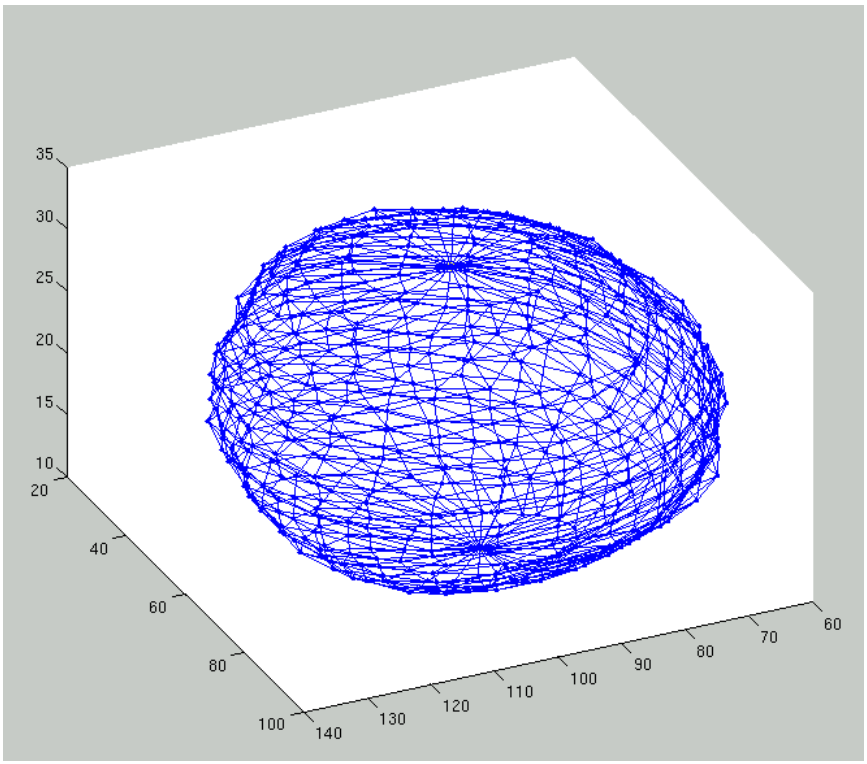


Figure 104: shows a wire frame representation of the surface model used in the 3D AAM.

In most medical imaging techniques there is a discrepancy between the resolution in the X-Y plane and the resolution in the Z-axis direction. For this reason two version of the mDCT were used in this part of the investigation; a 2D mDCT applied to a 2x2 block of data on the same slice, and a truly 3D mDCT applied to a 2x2x2 volume of data including data from adjacent slices. The 3D mDCT is much more computationally demanding than the 2D equivalent, and so it was interesting to compare the quality of results in the context of the cost of the segmentation (approximated to be proportional to run time of the process).

Appearance Model Used	Test Set	Example Number
Intensity	Undistorted	0
Intensity	Intensity consistently reduced	1
Intensity	Intensity consistently increased	2
Intensity	Constrained random distortion	3-22
2D mDCT	Undistorted	0
2D mDCT	Intensity consistently reduced	1
2D mDCT	Intensity consistently increased	2
2D mDCT	Constrained random distortion	3-22
3D mDCT	Undistorted	0
3D mDCT	Intensity consistently reduced	1
3D mDCT	Intensity consistently increased	2
3D mDCT	Constrained random distortion	3-22

Table 46: Summary of datasets used to evaluate the effectiveness of a 3D AAM approach with various appearance models.

The test datasets that were used are shown in table 46. The process was tested in two modes of operation. The test sets used are 3D volumes with each voxel being represented by 3 colour channels. One way of reducing the computational effort required to carry out the segmentation further is to reduce the number of colour channels, and therefore the number of dimensions being used in the associated appearance model. In the first set of results the input data was reduced to a single colour channel. In the second set of results the input data was not reduced to a single colour channel. Both sets of results are presented in tables 49 to 60 (which can be found in appendix A), using the same colour coding as used in the presentation of the 2D AMM results.

The results shown are given as two figures; the percentage of voxels correctly identified as being in or out of the segmentation set, and the run time of the process in seconds. It should be noted that the segmentation was carried out on a Linux farm that was being managed by LSF (Load Sharing Facility, which is a commercially available and commonly used tool which manages workload over CPUs and machines in a Linux farm). In this situation it is impossible to control

which workstations are carrying out the tasks, and the load on those workstations from running other unrelated tasks. Therefore the run times observed should only be taken as a rough guide to the cost of performing the segmentation. The times have also been rounded down to the nearest second. The mean times (t_{mean}) is also shown in each table.

The criterion for success is taken as 99% or more voxels being similarly classified between the manual segmentation (against which the process is being scored) and the segmentation generated by the AAM. This figure was decided upon by subjectively reviewing the results. For reference, figure 105 shows the results of such a successful segmentation. Note the results are shown on a single slice through the eye, even though the results are from a segmentation of the whole eye). It can be seen from the image that the automated process has done a good job of segmenting the eye. In fact, when compared with the manual segmentation, the boundary is smoother and subjectively could be regarded as a better result.

A summary of the results found in Appendix A can be seen in table 47.

Appearance Model	**Scale**	**Single-Channel datasets**		**Three-Channel datasets**	
		Mean time	**Number of tests @ 99%+**	**Mean time**	**Number of tests @ 99%+**
Intensity	100 %	28s	5/69 (7%)	42s	10/69 (14%)
2x2 mDCT		506s	38/69 (55%)	528s	51/69 (74%)
2x2x2 mDCT		1733s	11/69 (16%)	4343s	24/69 (35%)
Intensity	90 %	27s	6/69 (87%)	38s	4/69 (6%)
2x2 mDCT		381s	43/69 (62%)	407s	42/69 (61%)
2x2x2 mDCT		17/69	1495s	3295s	13/69

		(25%)			(19%)
Intensity	80 %	26s	2/69 (29%)	35s	2/69 (3%)
2x2 mDCT		293s	8/69 (12%)	315s	5/69 (7%)
2x2x2 mDCT		1397s	0/69 (0%)	2368s	0/69 (0%)
Intensity	70 %	25s	4/69 (58%)	33s	1/69 (1%)
2x2 mDCT		215s	0/69 (0%)	258s	0/69 (0%)
2x2x2 mDCT		698s	0/69 (0%)	1813s	0/69 (0%)
Intensity	60 %	23s	1/69 (1%)	32s	0/69 (0%)
2x2 mDCT		185s	0/69 (0%)	189s	0/69 (0%)
2x2x2 mDCT		546s	0/69 (0%)	2793s	0/69 (0%)

Table 47: Summary of results using a 3D AAM to segment previously unseen eyes at various scales ranging from 60% to 100%.

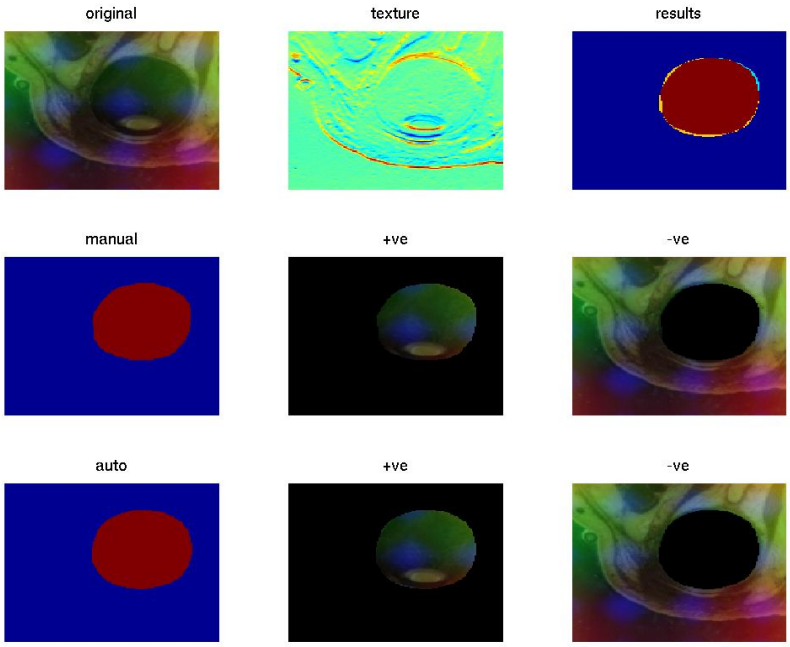


Figure 105: Shows a slice through a full 3D segmentation of an eye. In this case 99% of voxels were classified correctly (when compared to the manual segmentation).

One interesting trend seen was the relationship between the scale of the eye being segmented and the quality of results. While there was some success segmenting eyes that had their scale reduced to 90% of the original images size, there was little success below this point.

In Table 42 the results of a 2D AAM segemention using an intensity based texture model and an undistored test set are shown. The results show that sucessful segmentations are achieved for eyes at scales from 60% upto 100%. However, comparing this against the results of 2D AMM segmentations using an mDCT based texture model and an undistorted test set (as shown in table 44) it can be seen that no good segmentations are achieved when the eye has been scaled to 70% or below.

This suggests that the poor results seen when attempting the segmentation of eyes that have been reduced in size are not a feature of the methods used in rescaling of the test sets, but actually related to the mDCT based texture model being used.

However, to fully understand this limitation further work would be required, including using other scaling techniques (such as interpolation), or a larger and more varied dataset. The eyes used to make up the training and test datasets were taken from 3 individuals. The left eyes were used for training purposes, and the right eyes for test. Therefore the sizes of the eyes in training and test datasets were related. It would be interesting to use a larger dataset, which would presumably include a more representative cross section of typical sizes, to investigate more fully how this technique is sensitive to changes in scale.

Considering the results obtained when testing with eyes at a scale of 100%, several other trends can be seen. Table 48 shows a summary of these results.

Appearance Model	**Single-Channel datasets**		**Three-Channel datasets**	
	Mean time at 100% scale.	**Number of tests @ 99%+**	**Mean time at 100% scale.**	**Number of tests @ 99%+**
Intensity	28s	5/69 (7%)	42s	10/69 (14%)
2x2 mDCT	506s	38/69 (55%)	528s	51/69 (74%)
2x2x2 mDCT	1733s	11/69 (16%)	4343s	24/69 (35%)

Table 48: Summary of results using a 3D AAM to segment previously unseen eyes at a scale of 100%.

Firstly the results are of better quality when using full colour images over monochome images. However, the run time of a segmentation is greater when using full colour images.

It can also be seen that while the traditional AAM (using an intensity based appearance model) performs well on examples 0,1, and 2 (intensity not changed, intensity uniformly increased, and intensity uniformly decreased) in the other examples it performed poorly, showing the lack of tolerance to more complex variations in intensity.

When using an AAM with an appearance model based on an mDCT, the results show a much better tolerance of variations in intensity, however, the run times are greatly increased.

It is clear from the summary of the results that the best solution (in terms of the number of successful segmentations) is using a 2x2 mDCT transform, and also when using 3 colour channels. It is not surprising that the intensity-based appearance model did not perform well over the whole test set. It is also not surprising that the run time of the 2x2 mDCT based process is considerably better than that of the 2x2x2 mDCT based approach (there are many less DCTs to perform in the case of the former). It is not so obvious why the number of successful segmentations achieved by the 2x2 mDCT based approach is considerably superior to those achieved by the 2x2x2 mDCT based approach.

It is thought that this is actually due to the differences in resolution between the X-Y plane and the Z-axis of the images. The resolution in the direction of the Z-axis is in the region of 5 times lower than that of the X-Y plane, which would suggest that voxels which are adjacent but on different slices will have a weaker relationship than neighbouring voxels on the same slice. Hence the texture description gathered using a 2x2 mDCT holds a better quality of information about that specific location, over the more diluted information contained in a 2x2x2 mDCT.

Subjectively reviewing the quality of the successful segmentations it can be seen that in some cases the AAM generated a segmentation of perhaps higher quality than the original manual segmentation. This can be attributed to the way the AAM builds up the Shape Appearance model during training, by warping the training data to fit the models shape, and ultimately taking the mean of the appearance of each training examples to create the final combined model.

This approach means that small inaccuracies in the manual segmentation are smoothed out in the final model. However, it is thought that if further work was to be carried out it would be interesting to try and repeat the process using a larger, and expertly segmented dataset, and to see if this extra training effort led to a better quality of results overall.

Another interesting measure of the results is to compare the run time against the time it would take a human to do a similar segmentation. As stated earlier, it took in the region of 4 hours to manually segment the 6 undistorted eyes. This is around 40 minutes (or 2400 seconds per eye). Using the 2x2 mDCT and a 3 colour channel Appearance model took a mean time of 528 seconds, and a maximum time of 553 seconds (in both cases under 10 minutes), which shows an improvement over manual segmentation of around 30 minutes per eye.

8.0 Conclusions and Possible Further Work

The aim of this research has been to create and evaluate a practical (in terms of cost, energy efficiency, quality of results, and run time) segmentation process for use in the context of medical image processing, which is robust against unpredictable, variations in intensity (commonly found in such images).

The dataset chosen to carry out the investigation was from the Visible Human Project, and is described in section 2.4. This data set was primarily chosen due to its availability, and for the ease of identifying various structures within the cryosection data. Ideally the research would have been carried out on a database of MRI images, with some accompanying segmentation data provided for training and evaluation purposes (for example binary data showing the segmentation of a structure in the brain such as the hippocampus). However difficulties in obtaining such a dataset prevented this.

Several structures within the datasets (various bones, eyes etc.) were inexpertly segmented and used throughout the research, and were chosen based on the ease with which they could be segmented. Datasets were prepared, and used throughout the research for training and evaluation purposes, and a scoring mechanism was developed based on the number of voxels correctly classified as belonging to the segmentation region or not. This was used to compare various techniques.

DCT and Wavelet transforms are regularly used in the fields of image processing (particularly in the fields of compression, storage and transmission), and so, in chapter 3, initial research into the effectiveness of using various DCT and Wavelet based transforms (including the standard Haar transform, and a novel Haar Short transform which was developed) was reviewed. The transforms were used to create a texture descriptor, which was then classified by a GMM. It was found that while the transforms were able to provide some separation between the different textures present in the test data, the block based nature of the process was inherently noisy, and good results could not be readily achieved.

Chapter 4 reviewed the addition of some post processing to try and address the quality of the results. Strong and Weak region growing processes were used, in conjunction with low pass filters and although the filtering improved the quality, the results were still not good enough to be useful in the real world, and were found to be susceptible to relatively small variations in intensity.

In Chapter 5, the transforms being used were adapted to make them robust to variation in intensities (by removing the DC coefficients from the texture descriptors being generated), and it was seen that the effect of this was to slightly reduce the quality of the results, but to make the results consistent across various uniform intensity variations that were applied to the test data.

A more complex segmentation was attempted in Chapter 6, and it was found that (even after the post processing had been modified and improved) the results that could be obtained were too variable and generally not of a high enough standard, and so would need manual review and in most cases re-work. However, the results also suggested that DCT and Haar transforms could form the basis of an intensity invariant texture descriptor that could be used to create a practically useful segmentation process.

At this point a new technique was considered, and in Chapter 7 a model based approach was adopted, that of using an AAM. AAMs were initially introduced by Cootes et al. and once trained, were shown to be affective in registering previously unseen 2D images with a model based on not only appearance but also on shape.

Traditionally the appearance part of the shape appearance model used has been based on intensity alone. Initial experimental work carried out in Matlab and based on the code written by Kroon [20], showed (unsurprisingly) that an AAM performs poorly if the images it is being used to interpret have been subjected to variations in intensity.

Ya Su et al. have carried out research into using an AAM with an appearance model based on a more complex texture descriptor called the GLBP (a

combination of a Gabor Wavelet and a Local Binary Pattern) [19]. The motivation behind their research was to improve the model fitting performance of an AAM in situations where environmental factors (such as changes in illumination) can easily interfere with the quality of results if the appearance model is based only on intensity. In their research 2D images (such as photographic images etc.) were used to evaluate their technique, and they found that by using a GLBP based appearance model they were able to improve the model fitting in cases where illumination had been varied. However, this was at the expense of efficiency, and it was seen (via a simple comparison between run time of each technique) that using the AAM with an appearance model based on the GLBP texture description took over 10 times longer than using intensity alone, when using a 2D AAM.

In section 7.1 Kroon's code was updated to allow a mDCT transform to be used as the basis of building the appearance model, and an AAM was then used to test its performance against a traditional AAM.

The results given in section 7.2 show that the mDCT gave a good performance, even when the images had been subjected to intensity variation, and so in section 7.3 the research was carried forward to evaluating a 3D mDCT based AAM, against a traditional 3D AAM. It was found that there was again a marked improvement over the traditional AAM, the best results being seen when a 2D mDCT was used to build the appearance model. This was considered to be the case sue to the poorer resolution on the Z-axis of the data (as is often the case with medical imaging data) meaning that the relationship between neighbouring voxels on adjacent slices was not as strong as the relationship between similarly neighbouring voxels on the same slice.

Using a 2D mDCT as the texture descriptor is also beneficial as the segmentation is much less computationally demanding. A 2x2x2 mDCT requires12 separate 1x2 DCT calculations to be performed, and results in a 7-dimensional texture descriptor. A 2x2 mDCT requires only 4 separate 1x2 DCT calculations to be performed, and results in a 3-dimensional texture descriptor. This means (in the case of using the 2D mDCT) the appearance model is

calculated much quicker, and the AAM can also fit the image to the model much quicker.

Further work is required to investigate if this technique is equally effective when performing segmentations on images from more suitable imaging techniques such as MRI. Ideally a database of pre-segmented MRI images would be used to build and test a model, to review the suitability of this approach in a real life situation.

The database of MRI images should contain data over a range of patient groups (such as age, for example, which can have an effect on the types of intensity variation that can be observed). Also, if a large enough sample of images was obtained and used, then the effect of varying scale and shape could be better investigated.

References

1. J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, W. T. Freeman. Discovering objects and their location in images. Conference Proceedings of Tenth IEEE International Conference on Computer Vision (ICCV'05). IEEE. 2005. Beijing, Chain.
2. K. T. Mullen. The Contrast Sensitivity of Human Colour Vision to Red-Green and Blue-Yellow Chromatic Gratings.The Journal of Physiology. Vol 359, Issue 1. Pages 381 to 400. 1985.
3. J. S. Duncan, N. Ayache. Medical Image Analysis: Progress over Two Decades and the Challenges Ahead. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 22, Pages 85 to 106. 2000.
4. M. Pokric, N. A. Thacker, M. L. J. Scott, A. Jackson. Multi-dimensional Medical Image Segmentation with Partial Voluming. Conference Proceedings of Medical Image Understanding and Analysis. 2001. Birmingham, UK.
5. R. Gomes, M. Spain. Regocnizing Scenes: Can coarse geometry aid a bag-of-features. 2006.
6. D. L. Pham, C. Xu, J. L. Prince. A Survey of Current Methods in Medical Image Segmentation. Annu. Rev. Biomed. Eng. Vol 2. Pages 315 to 337. 2000.
7. M. Grundland, N. A. Dodgson. Decolorize: Fast, contrast enhancing color to grayscale conversion. Pattern Recognition. Vol 40, Issue 11. Pages 2891 to 2896. 2007.
8. M. Marszaek, C. Schmid. Spatial Weighting for a bag-of-features. Computer Vision and Pattern Recognition. Vol 2. Pages 2118 to 2125. 2006.
9. David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision. Vol 60, Issue 2. Pages 91 to 110. 2004.
10. I. T. Nabney. NETLAB: Algorithms for Pattern Recognition. Chapter 3, Density Modelling and Clustering. Pages 84 to 89. Editor, S. Singh. Publisher, Springer. 2002. London, UK.
11. T. Kitasaka, K. Mori, J. Hasegawa and J. Toriwaka. A Method for Extraction of Bronchus Regions from 3D Chest X-ray CT Images by

Analyzing Structural Features of the Bronchus. Forma. Vol 17, Issue 4. Pages 321-338. 2002.

12. V. Spitzer, M. J. Ackerman, A. L. Scherzinger, D. Whitlock. The Visible Human Male: A Technical Report. JAMIA. Vol 3, Issue 118. Pages 118 to 130. 1996.
13. G. K. Wallace. The JPEG still picture compression standard. Communications of the ACM. Vol 34, Issue 4. Pages 30 to 44. 1991.
14. E. Nowak, F. Jurie, and B. Triggs. Sampling Strategies for Bag-of-Features Image Classification. Conference Proceedings of the European Conference on Computer Vision. 2006. Graz, Austria.
15. R. Fergus, L. Fei-Fei, P. Perona, A. Zisserman. Learning Object Categories from Google's Image Search. Conference Proceedings of Tenth IEEE International Conference on Computer Vision (ICCV'05). IEEE. 2005. Beijing, Chain.
16. P. Gingins, P. Beylot, P. Kalra, N. M. Thalmann, W. Maurel, D. Thalmann, and J. Fasel. Modeling using the Visible Human Dataset. Conference Proceedings of Medical Informatics Europe. 1996. Copenhagen, Denmark.
17. *T. R. Paluska, M. J. Sise, D. I. Sack, C. B. Sise, M. C. Egan, M. Biondi. Incidental CT Findings in Trauma Patients: Incidence and* Implications for Care of the Injured.Trauma. Vol 61, Issue 1. Pages 157 to 161. 2007.
18. T.F. Cootes, G.J.Edwards, and C.J.Taylor. Active Appearance Models. IEEE PAMI. Vol 23, Issue 6. Pages 681 to 685. 2001.
19. Y. Su, X. Li, D. Tao, X. Gao. Texture Representation in AAM using Gabor Wavelet and Local Binary Patterns. Conference Proceedings of IEEE Systems Conference. 2009. Vancouver, Canada.
20. D. Kroon. Active Shape Model (ASM) and Active Appearance Model (AAM), Date Accessed: 6th May 2014. World Wide Web Page: http://www.mathworks.com/matlabcentral/fileexchange/26706-active-shape-model--asm--and-active-appearance-model--aam-
21. R. Adams, L. Bischof. Seeded Region Growing. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol 16, Issue 6. Pages 641 to 647. 1994.

22. S. A. Hojjatoleslami, J. Kittler. Region growing: a new approach. IEEE Translations on Image Processing. Vol 7, Issue 7. Pages 1079 to 1084. 1998.
23. M. N. Ahmed, S. M. Yamany, N. Mohamed, A. A. Farag, T. Moriarty. A modified Fuzzy C-means Algorithm for Bias Field Estimation and Segmentation of MRI Data. IEEE Transactions on Medical Imaging Vol 21, Issue 3. Pages 193 to 199. 2002.
24. B. Likar, M. A. Viergever, F. Pernus. Retrospective Correction of MR intensity inhomogeneity by information minimization. IEEE Transactions on Medical Imaging Vol 20, Issue 12. Pages 1398 to 1410. 2001.
25. W. M. Wells III, W. E. L. Grimson, R. Kikinis, F.A. Jolesz. Adaptive segmentation of MRI Data. IEEE Transactions on Medical Imaging Vol 15, Issue 4. Pages 429 to 442. 1996.
26. L. van den Branden, J. Christian, V. Olivier. Perceptual quality measure using a spatiotemporal model of the human visual system. Electronic Imaging: Science & Technology. Pages 450 to 461. 1996.
27. C. R. Jack Jr, R. C. Petersen, Y. C. Xu, P. C. O'Brien, G. E. Smith, R. J. Ivnik, B. F. Boeve, S. C. Waring, E. G. Tangalos, E. Kokmen. Prediction of AD with MRI-Based Hippocampal Volume in Mild Cognitive Impairment. Neurology. Vol 52, Issue 7. Pages 1397 to 1403. 1999.
28. R. J. Minns, R. Bibb, R. Banks, R. A. Sutton. The use of a reconstructed three-dimensional solid model from CT to aid the surgical management of a total knee arthroplasty: a case study. Medical engineering & Physics. Vol 25, Issue 6. Pages 523 to 526. 2003.
29. S. Ghetia, N. Gajjar, R. Gajjar. Implementation of 2-D Discrete Cosine Transform Algorithm on GPU. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering. Vol2, Issue 7. 2013.
30. B. Pollak. Experiences with Planography. CHEST journal. Vol 24, Issue 6. Pages 663 to 669. 1953.
31. L. W. Goldman. Principles of CT: Radiation Dose and Image Quality. Journal of Nuclear Medicine Technology. Vol 35, Issue 4. Pages 213 to 225. 2007.
32. H. Hu. Multi-slice Helical CT: scan and reconstruction. Medical Physics. Vol 26, Issue 1. Pages 5 to 18. 1999.

33. M. A. G. Ballester, A. P. Zisserman, M. Brady. Estimation of the Partial Volume Effect in MRI. Medical Image Analysis. Vol 6, Issue 4. Pages 389 to 405. 2002.
34. S. A. Roll, A. C. F. Colchester, P. E. Summers. L. D. Griffin. Intensity-Based Object Extraction from 3D Medical Images including a Correction of Partial Volume Errors. Conference Proceedings of the British Machine Vision Conference. 1994. York, UK.
35. A. Rahmim, H. Zaidi. PET versus SPECT: Strengths, limitations, challenges. Nuclear Medicine Communications. Vol 29, Issue 3. Pages 193 to 207. 2008.
36. A. J. Reader, H. Zaidi. Advanced in PET image reconstruction. PET Clinics. Vol 2, Issue 2. Pages 173 to 190. 2007.
37. G. E. Christensen, R. D. Rabbitt, M. I. Miller. 3D brain mapping using a deformable neuroanatomy. Physicals in medicine and biology. Vol 39, Issue 3. Pages 609 to 618. (1994).
38. R. Kiknis, M. E. Shenton, D. V. Iosifescu, R. W. McCarley, P. Saiviroonporn, H. H. Hokama, A. Robatino, D. Metcalf, C. G. Wible, C. M. Portas, R. M. Donnino, F. A. Jolesz. A Digital Brain Atlas for Surgical Planning, Model-Driven Segmentation, and Teaching. IEEE Transactions on Medical Imaging Vol 2, Issue 3. Pages 232 to 241. 1996.
39. N. Ahmed, T. Natarajan, K. R. Rao. Discrete Cosine Transform. IEEE Transactions on Computers. Vol 100, Issue 1. Pages 90 to 93. 1974.
40. V.K. Ananthashayana, K. S. Geetha. A Novel Recursive Multiplierless Algorithm for 2-D DCT. World Academy of Science, Enginerring and Technology. Vol 3. Pages 518 to 522. 2009.
41. H.Permuter, J. Francois, I. Jermyn. A study of Gaussian mixture models of colour and texture features for image classification and segmentation. Pattern Recognition. Vol 39, Issue 4. Pages 696 to 706. 2006.
42. A. Dempster, N. Laird, D. Rubin. Maximum likelihood from incomplete data via EM algorithm. Journal of the Royal Statistical Society, Series B. Vol 39, Issue 1. Page 1 to 38. 1977.
43. V. Namias. The fractional order Fourier transform and its application to quantum mechanics. IMA Journal of Applied Mathematics. Vol 25, Issue 3. Pages 241 to 265. 1980.

44. D. Griffin, J. S. Lim. Signal estimation from modified short-time Fourier transform. IEEE transactions on Acoustics, Speech and Signal Processing. Vol 32, Issue 2. Pages 236 to 243. 1984.
45. P. Busch. Heisenberg's Uncertainty Principle. Physics Reports. Vol 5, Issue 6. Pages 155 to 176. 2007.
46. A. Haar. On the theory of orthogonal function systems. Fundamental papers in wavelet theory. Pages 155 to 188. 1910.

Appendix A - Results from 3D AMM experiments

	Scale (as percentage of original image), Eye (dataset 4,5 or 6).								
Ex.	60%	60%	60%	70%	70%	70%	80%	80%	80%
	4	5	6	4	5	6	4	5	6
0	98% 22s	98% 30s	98% 29s	98% 21s	99% 29s	98% 30s	98% 21s	99% 31s	98% 30s
1	91% 20s	98% 29s	92% 22s	98% 21s	98% 29s	98% 23s	98% 21s	98% 30s	98% 30s
2	98% 20s	98% 29s	98% 23s	98% 21s	97% 29s	98% 23s	98% 21s	98% 31s	98% 29s
3	94% 20s	91% 29s	95% 23s	92% 21s	98% 29s	95% 23s	95% 22s	93% 30s	94% 30s
4	88% 20s	92% 29s	95% 22s	97% 21s	95% 29s	96% 23s	98% 21s	98% 29s	98% 30s
5	94% 20s	94% 29s	91% 22s	92% 21s	93% 28s	95% 23s	96% 21s	98% 30s	98% 29s
6	96% 20s	90% 29s	92% 22s	98% 20s	91% 30s	98% 23s	97% 21s	93% 30s	97% 30s
7	88% 20s	98% 29s	93% 23s	98% 21s	98% 29s	98% 30s	98% 21s	98% 30s	93% 29s
8	93% 20s	90% 28s	89% 22s	98% 21s	93% 29s	97% 23s	96% 21s	96% 30s	97% 30s
9	92% 20s	92% 29s	89% 22s	96% 20s	92% 29s	98% 23s	98% 21s	96% 30s	93% 30s
10	89% 20s	98% 29s	93% 22s	92% 20s	99% 29s	96% 22s	98% 21s	95% 30s	98% 30s
11	92% 20s	89% 29s	92% 22s	96% 20s	91% 29s	98% 23s	96% 21s	96% 30s	94% 29s
12	93% 20s	97% 29s	92% 23s	92% 20s	99% 29s	93% 23s	97% 21s	94% 30s	98% 30s

13	89% 20s	92% 28s	92% 23s	98% 21s	98% 31s	92% 30s	98% 21s	97% 30s	95% 29s
14	99% 20s	92% 28s	93% 23s	96% 22s	94% 31s	98% 29s	96% 21s	96% 29s	93% 30s
15	95% 20s	95% 28s	98% 22s	94% 21s	98% 30s	94% 31s	98% 21s	98% 29s	93% 30s
16	93% 20s	95% 28s	95% 23s	95% 20s	91% 30s	92% 23s	98% 22s	95% 29s	92% 30s
17	91% 20s	94% 28s	95% 23s	97% 21s	95% 29s	92% 23s	96% 21s	92% 30s	98% 30s
18	92% 20s	98% 29s	98% 22s	98% 21s	98% 30s	91% 22s	99% 21s	98% 29s	97% 22s
19	90% 21s	98% 28s	97% 23s	98% 21s	98% 30s	97% 29s	98% 20s	97% 30s	93% 23s
20	95% 20s	98% 28s	92% 22s	97% 21s	97% 29s	92% 30s	97% 21s	93% 30s	98% 30s
21	92% 20s	89% 29s	91% 23s	91% 21s	92% 29s	95% 23s	98% 21s	96% 29s	98% 23s
22	95% 20s	98% 28s	90% 23s	93% 21s	98% 29s	99% 23s	92% 21s	92% 30s	93% 29s
t_{mean}	20s	28s	22s	20s	29s	25s	21s	29s	28s

Table 49: Results when using an intensity based appearance model and a single colour channel training and test dataset (scale 60% - 80%).

	Scale (as percentage of original image), Eye (dataset 4,5 or 6).					
Ex.	90%	90%	90%	100%	100%	100%
	4	5	6	4	5	6
0	99% 22s	99% 30s	99% 30s	99% 23s	99% 31s	98% 31s
1	98% 22s	99% 30s	99% 30s	99% 23s	99% 31s	99% 32s
2	98% 22s	98% 30s	98% 31s	98% 23s	97% 33s	97% 32s
3	94% 22s	95% 30s	96% 31s	98% 23s	95% 32s	96% 32s
4	97% 22s	93% 31s	92% 30s	96% 24s	94% 31s	95% 32s
5	97% 23s	97% 30s	98% 31s	94% 23s	98% 32s	94% 32s
6	95% 22s	98% 30s	92% 31s	94% 23s	94% 32s	95% 32s
7	97% 22s	94% 30s	98% 31s	97% 23s	95% 32s	93% 33s
8	97% 22s	97% 31s	95% 31s	96% 23s	97% 32s	95% 33s
9	94% 22s	96% 30s	99% 31s	98% 23s	93% 32s	97% 32s
10	95% 22s	96% 29s	95% 30s	95% 23s	98% 32s	93% 31s
11	97% 22s	95% 30s	93% 30s	97% 23s	98% 31s	95% 31s
12	93% 21s	98% 30s	94% 30s	95% 23s	94% 31s	95% 32s
13	95% 22s	97% 29s	97% 31s	95% 23s	95% 31s	97% 32s
14	95% 22s	95% 31s	94% 30s	94% 23s	97% 32s	98% 31s

15	93% 22s	98% 30s	94% 31s	98% 23s	95% 32s	94% 31s
16	95% 23s	95% 30s	98% 31s	96% 24s	93% 32s	96% 31s
17	96% 22s	98% 31s	94% 30s	95% 23s	93% 31s	94% 32s
18	98% 23s	94% 30s	96% 31s	96% 23s	98% 31s	98% 31s
19	96% 22s	97% 31s	97% 31s	96% 23s	98% 32s	96% 31s
20	94% 22s	98% 31s	96% 31s	96% 23s	94% 32s	93% 32s
21	93% 22s	96% 31s	98% 31s	94% 23s	98% 32s	96% 32s
22	98% 22s	96% 29s	98% 30s	97% 23s	96% 32s	98% 32s
t_{mean}	22s	30s	30s	23s	31s	31s

Table 50: Results when using an intensity based appearance model and a single colour channel training and test dataset (scale 90% - 100%).

	Scale (as percentage of original image), Eye (dataset 4,5 or 6).								
Ex.	60%	60%	60%	70%	70%	70%	80%	80%	80%
	4	5	6	4	5	6	4	5	6
0	88% 115s	88% 212s	88% 210s	91% 150s	91% 365s	92% 149s	98% 289s	98% 290s	92% 291s
1	90% 113s	90% 211s	90% 210s	90% 256s	90% 152s	91% 252s	92% 289s	98% 289s	98% 290s
2	89% 113s	89% 212s	90% 211s	91% 147s	91% 244s	91% 250s	98% 288s	98% 290s	99% 289s
3	88% 113s	89% 211s	91% 212s	91% 147s	90% 245s	91% 250s	92% 288s	92% 290s	99% 294s
4	90% 114s	90% 212s	89% 211s	89% 149s	90% 254s	91% 253s	91% 291s	91% 292s	91% 299s
5	88% 114s	90% 211s	91% 211s	91% 148s	89% 245s	91% 252s	92% 289s	98% 293s	92% 299s
6	88% 113s	89% 212s	90% 211s	91% 148s	90% 255s	91% 255s	98% 292s	98% 290s	91% 299s
7	89% 114s	89% 211s	90% 210s	90% 148s	90% 253s	91% 247s	99% 291s	95% 293s	92% 297s
8	89% 114s	90% 211s	90% 211s	90% 149s	90% 253s	92% 245s	91% 289s	92% 293s	90% 299s
9	90% 113s	90% 211s	90% 212s	90% 147s	91% 245s	91% 252s	92% 291s	99% 291s	95% 291s
10	88% 134s	89% 218s	89% 228s	91% 147s	91% 245s	91% 251s	92% 290s	90% 292s	98% 299s
11	90% 114s	89% 215s	90% 232s	90% 147s	89% 245s	92% 250s	91% 290s	92% 290s	98% 292s
12	89% 119s	90% 224s	90% 241s	90% 147s	90% 245s	91% 251s	96% 293s	99% 292s	92% 297s
13	89% 121s	89% 221s	88% 222s	90% 147s	90% 245s	91% 251s	99% 291s	92% 291s	93% 298s
14	90% 119s	91% 224s	90% 225s	90% 147s	93% 244s	92% 251s	92% 291s	97% 301s	91% 293s
15	90%	89%	88%	90%	91%	91%	98%	98%	99%

	131s	221s	232s	149s	252s	245s	291s	292s	299s
16	89% 117s	90% 224s	89% 222s	90% 147s	90% 245s	91% 252s	92% 290s	91% 294s	92% 299s
17	90% 120s	89% 239s	88% 223s	90% 147s	91% 245s	91% 252s	98% 289s	97% 292s	91% 301s
18	90% 121s	90% 222s	88% 221s	91% 148s	91% 252s	92% 244s	92% 291s	98% 293s	92% 297s
19	90% 128s	90% 230s	90% 220s	90% 147s	95% 243s	91% 251s	91% 290s	99% 293s	91% 298s
20	89% 119s	90% 219s	90% 214s	91% 146s	91% 245s	91% 250s	98% 292s	97% 293s	92% 316s
21	89% 130s	89% 215s	89% 221s	90% 146s	90% 244s	90% 252s	91% 297s	91% 302s	90% 299s
22	88% 131s	90% 213s	91% 232s	89% 147s	90% 245s	91% 251s	98% 294s	93% 298s	92% 294s
t_{mean}	119s	217s	219s	152s	248s	245s	290s	292s	296s

Table 51: Results when using a 2x2 mDCT based appearance model and a single colour channel training and test dataset (scale 60% - 80%).

	Scale (as percentage of original image), Eye (dataset 4,5 or 6).					
Ex.	90%	90%	90%	100%	100%	100%
	4	5	6	4	5	6
0	99% 343s	99% 447s	96% 350s	99% 409s	99% 511s	99% 513s
1	99% 342s	99% 444s	99% 342s	96% 409s	99% 616s	96% 512s
2	92% 340s	96% 445s	92% 342s	99% 409s	99% 512s	99% 615s
3	92% 341s	96% 445s	95% 342s	95% 408s	99% 511s	99% 510s
4	99% 341s	99% 449s	96% 351s	95% 407s	96% 514s	98% 616s
5	96% 341s	99% 447s	99% 350s	96% 409s	95% 615s	92% 517s
6	99% 345s	99% 457s	98% 353s	99% 512s	99% 514s	94% 514s
7	99% 343s	99% 459s	99% 355s	95% 409s	99% 512s	96% 511s
8	98% 352s	99% 451s	98% 351s	95% 512s	97% 561s	96% 532s
9	91% 344s	99% 448s	99% 352s	99% 409s	99% 619s	99% 512s
10	99% 352s	99% 448s	99% 353s	99% 513s	99% 515s	96% 516s
11	99% 345s	99% 456s	92% 355s	96% 409s	99% 615s	96% 409s
12	96% 344s	99% 448s	95% 349s	95% 512s	95% 409s	97% 510s
13	99% 342s	99% 446s	99% 350s	99% 408s	99% 513s	99% 617s
14	97% 342s	99% 447s	99% 353s	99% 409s	96% 511s	99% 616s
15	98%	99%	97%	97%	99%	99%

	345s	457s	353s	512s	513s	515s
16	99% 342s	98% 447s	99% 351s	99% 513s	96% 515s	95% 514s
17	93% 344s	99% 447s	99% 353s	99% 408s	97% 513s	99% 616s
18	99% 345s	99% 448s	98% 359s	99% 514s	99% 514s	99% 511s
19	94% 352s	99% 449s	95% 351s	95% 513s	99% 516s	99% 523s
20	99% 345s	98% 455s	99% 353s	99% 512s	99% 409s	95% 617s
21	99% 344s	99% 448s	99% 351s	95% 512s	99% 517s	96% 516s
22	99% 342s	99% 447s	98% 353s	98% 513s	99% 515s	99% 516s
t_{mean}	344s	449s	350s	458s	524s	536s

Table 52: Results when using a 2x2 mDCT based appearance model and a single colour channel training and test dataset (scale 90% - 100%).

	Scale (as percentage of original image), Eye (dataset 4,5 or 6).								
Ex	60%	60%	60%	70%	70%	70%	80%	80%	80%
	4	5	6	4	5	6	4	5	6
0	90% 560s	91% 564s	90% 556s	91% 658s	91% 756s	91% 755s	90% 825s	91% 831s	91% 2480s
1	90% 549s	90% 547s	90% 557s	91% 647s	91% 748s	91% 750s	91% 822s	91% 861s	91% 2478s
2	90% 547s	90% 547s	91% 556s	91% 633s	90% 744s	90% 741s	90% 848s	91% 948s	91% 2396s
3	89% 552s	91% 557s	90% 548s	92% 641s	91% 741s	91% 750s	92% 852s	92% 949s	91% 2397s
4	88% 534s	91% 554s	90% 546s	91% 644s	90% 748s	91% 743s	91% 857s	94% 968s	91% 2391s
5	89% 534s	90% 543s	91% 560s	90% 645s	93% 749s	89% 736s	91% 852s	92% 959s	91% 2397s
6	91% 544s	91% 547s	90% 546s	90% 624s	90% 739s	91% 740s	93% 856s	92% 973s	90% 2404s
7	91% 549s	90% 540s	90% 562s	90% 644s	90% 739s	91% 745s	91% 869s	93% 971s	92% 2392s
8	90% 537s	90% 538s	90% 558s	90% 640s	91% 755s	91% 750s	92% 848s	92% 952s	91% 2398s
9	90% 535s	91% 556s	90% 546s	90% 647s	91% 750s	91% 753s	90% 854s	93% 957s	93% 2390s
10	89% 544s	90% 551s	90% 540s	90% 649s	91% 746s	90% 726s	92% 854s	92% 960s	91% 2401s
11	90% 545s	91% 535s	90% 552s	91% 644s	91% 741s	91% 750s	91% 865s	93% 967s	91% 2414s
12	89% 544s	90% 550s	90% 556s	91% 625s	89% 739s	90% 751s	91% 869s	91% 975s	93% 2509s
13	90% 534s	90% 543s	89% 555s	91% 648s	91% 752s	90% 753s	91% 868s	93% 972s	91% 2406s
14	90% 547s	90% 543s	90% 546s	91% 646s	91% 750s	91% 718s	91% 866s	92% 965s	91% 2411s
15	90%	91%	90%	92%	90%	90%	91%	93%	91%

	544s	532s	542s	610s	711s	623s	867s	963s	2383s
16	90% 544s	89% 549s	90% 541s	91% 612s	91% 720s	90% 724s	91% 839s	91% 954s	91% 2388s
17	90% 544s	90% 535s	89% 554s	90% 615s	90% 714s	90% 615s	92% 834s	92% 847s	91% 2481s
18	90% 536s	90% 553s	90% 545s	90% 612s	90% 709s	91% 732s	89% 839s	91% 843s	91% 2465s
19	90% 536s	90% 543s	90% 546s	91% 615s	91% 738s	91% 747s	92% 824s	91% 829s	93% 2465s
20	90% 532s	90% 543s	90% 542s	91% 642s	92% 737s	91% 738s	89% 817s	94% 821s	91% 2460s
21	91% 544s	90% 548s	90% 542s	91% 631s	90% 733s	89% 708s	91% 822s	91% 819s	91% 2471s
22	89% 537s	90% 559s	90% 557s	90% 624s	91% 721s	90% 593s	91% 825s	93% 830s	91% 2465s
t_{mean}	542s	546s	550s	634s	738s	723s	846s	918s	2427s

Table 53: Results when using a 2x2x2 mDCT based appearance model and a single colour channel training and test dataset (scale 60% - 80%).

	Scale (as percentage of original image), Eye (dataset 4,5 or 6).					
Ex	90%	90%	90%	100%	100%	100%
	4	5	6	4	5	6
0	98% 916s	98% 1013s	99% 2555s	96% 1152s	98% 1239s	99% 2779s
1	98% 913s	98% 1000s	99% 2556s	96% 1281s	98% 1167s	98% 2788s
2	98% 915s	98% 1010s	99% 2555s	99% 1152s	99% 1241s	99% 2793s
3	96% 905s	98% 1017s	98% 2546s	96% 1297s	98% 2805s	98% 1166s
4	98% 905s	97% 1005s	98% 2544s	96% 1161s	98% 1258s	98% 2788s
5	98% 915s	98% 1012s	99% 2545s	96% 1143s	98% 1263s	98% 2787s
6	97% 921s	97% 1021s	98% 2556s	96% 1158s	98% 1249s	98% 2793s
7	98% 913s	99% 1000s	99% 2553s	95% 1161s	98% 1248s	99% 2782s
8	98% 913s	98% 1011s	98% 2556s	95% 1154s	98% 1249s	98% 2783s
9	97% 1030s	98% 914s	99% 2538s	95% 1145s	98% 1259s	98% 2788s
10	98% 919s	97% 1007s	99% 2555s	96% 1157s	98% 1250s	98% 2800s
11	98% 913s	97% 1016s	98% 2556s	96% 1156s	98% 1245s	98% 2787s
12	92% 915s	98% 1009s	98% 2540s	95% 1153s	98% 1249s	98% 2784s
13	97% 903s	98% 1006s	99% 2542s	98% 1161s	98% 1256s	98% 2786s
14	97% 912s	99% 1001s	98% 2551s	95% 1141s	98% 1250s	98% 2794s
15	97%	98%	98%	96%	98%	99%

	912s	1009s	2543s	1154s	1249s	2784s
16	97% 902s	97% 1009s	99% 2539s	96% 1152s	99% 1264s	98% 2788s
17	98% 899s	98% 1007s	98% 2545s	95% 1156s	99% 1253s	98% 2792s
18	98% 900s	97% 996s	99% 2545s	95% 1159s	98% 1260s	98% 2799s
19	97% 898s	99% 1024s	98% 2758s	96% 1145s	98% 1256s	98% 2800s
20	96% 1029s	98% 1027s	99% 2585s	95% 1141s	99% 1249s	99% 2799s
21	92% 901s	99% 1008s	99% 2548s	95% 1156s	98% 1244s	98% 2785s
22	98% 915s	99% 1023s	98% 2562s	95% 1159s	98% 1257s	99% 2800s
t_{mean}	920s	1006s	2559s	1164s	1315s	2719s

Table 54: Results when using a 2x2x2 mDCT based appearance model and a single colour channel training and test dataset (scale 90% - 100%).

	Scale (as percentage of original image), Eye (dataset 4,5 or 6).								
Ex	60%	60%	60%	70%	70%	70%	80%	80%	80%
	4	5	6	4	5	6	4	5	6
0	98% 33s	98% 35s	98% 33s	98% 32s	98% 36s	98% 35s	98% 34s	98% 35s	98% 36s
1	98% 33s	98% 33s	98% 33s	98% 32s	98% 35s	98% 33s	98% 34s	98% 36s	98% 43s
2	98% 25s	98% 39s	98% 32s	98% 32s	98% 34s	98% 33s	98% 34s	98% 37s	98% 42s
3	92% 32s	93% 33s	90% 34s	92% 33s	98% 33s	95% 35s	92% 34s	97% 36s	98% 42s
4	93% 32s	93% 33s	98% 34s	98% 33s	94% 34s	96% 34s	98% 34s	98% 36s	98% 35s
5	89% 33s	91% 34s	92% 34s	93% 32s	91% 37s	97% 43s	98% 34s	98% 35s	98% 36s
6	90% 25s	90% 39s	96% 33s	98% 35s	96% 38s	96% 43s	96% 34s	95% 35s	94% 35s
7	94% 32s	98% 33s	93% 33s	98% 34s	98% 35s	91% 34s	99% 34s	98% 36s	96% 35s
8	93% 32s	93% 34s	95% 33s	97% 32s	94% 36s	98% 34s	96% 34s	99% 35s	95% 36s
9	94% 25s	93% 39s	91% 34s	98% 33s	92% 34s	97% 34s	98% 34s	96% 35s	95% 35s
10	90% 32s	98% 32s	93% 34s	94% 32s	98% 34s	98% 35s	98% 33s	96% 35s	98% 35s
11	92% 32s	89% 34s	93% 34s	97% 33s	93% 35s	98% 34s	94% 34s	97% 35s	95% 35s
12	90% 32s	94% 34s	92% 33s	94% 33s	98% 34s	95% 35s	96% 34s	93% 36s	97% 42s
13	95% 32s	92% 33s	91% 33s	90% 33s	97% 34s	95% 34s	98% 34s	97% 37s	95% 42s
14	90% 25s	95% 39s	93% 34s	95% 32s	93% 34s	94% 33s	97% 35s	96% 35s	95% 42s
15	97%	96%	97%	93%	95%	92%	97%	97%	92%

	32s	34s	34s	32s	34s	35s	34s	35s	36s
16	98% 32s	92% 33s	92% 33s	94% 33s	96% 34s	93% 35s	96% 34s	96% 35s	93% 36s
17	91% 25s	92% 40s	89% 34s	95% 32s	97% 35s	95% 34s	95% 33s	95% 36s	98% 36s
18	93% 32s	91% 34s	90% 34s	96% 32s	97% 35s	91% 34s	94% 34s	98% 35s	97% 36s
19	90% 25s	95% 40s	93% 33s	97% 32s	98% 35s	96% 34s	97% 34s	95% 35s	97% 35s
20	89% 25s	93% 32s	92% 40s	97% 33s	97% 33s	92% 34s	97% 34s	93% 36s	97% 35s
21	98% 33s	89% 33s	93% 33s	90% 32s	89% 34s	98% 34s	94% 34s	97% 36s	96% 42s
22	93% 25s	95% 39s	91% 33s	92% 32s	98% 34s	99% 34s	94% 34s	94% 36s	92% 35s
t_{mean}	29s	35s	33s	32s	34s	34s	33s	35s	37s

Table 55: Results when using an intensity based appearance model and a three colour channel training and test dataset (scale 60% - 80%).

	Scale (as percentage of original image), Eye (dataset 4,5 or 6).					
Ex	90%	90%	90%	100%	100%	100%
	4	5	6	4	5	6
0	98% 35s	99% 37s	98% 44s	99% 37s	99% 46s	99% 46s
1	98% 36s	99% 44s	98% 37s	99% 38s	99% 45s	99% 38s
2	98% 35s	99% 37s	98% 43s	99% 38s	99% 45s	99% 46s
3	94% 35s	94% 38s	93% 43s	98% 38s	97% 46s	94% 45s
4	98% 36s	96% 37s	93% 43s	96% 37s	95% 46s	95% 46s
5	96% 36s	96% 45s	98% 36s	93% 38s	96% 46s	95% 46s
6	96% 36s	97% 44s	94% 37s	94% 38s	98% 46s	94% 44s
7	97% 35s	94% 37s	97% 42s	97% 38s	95% 47s	93% 46s
8	96% 34s	95% 37s	96% 44s	97% 38s	94% 44s	93% 39s
9	95% 35s	95% 37s	99% 44s	96% 38s	94% 46s	99% 47s
10	98% 35s	97% 38s	97% 43s	95% 38s	97% 45s	96% 46s
11	97% 35s	96% 37s	93% 42s	98% 38s	97% 45s	96% 45s
12	94% 35s	98% 38s	93% 43s	95% 38s	98% 47s	93% 45s
13	98% 35s	94% 37s	98% 43s	94% 37s	94% 46s	94% 45s
14	96% 35s	94% 37s	95% 44s	93% 36s	97% 45s	98% 39s

15	93% 34s	98% 38s	95% 43s	97% 38s	96% 46s	94% 46s
16	98% 36s	95% 37s	98% 43s	97% 38s	93% 45s	95% 46s
17	94% 36s	98% 37s	95% 44s	94% 39s	93% 46s	95% 46s
18	94% 36s	95% 44s	96% 37s	97% 38s	96% 45s	96% 45s
19	95% 35s	97% 38s	98% 44s	97% 37s	95% 46s	94% 46s
20	95% 36s	94% 37s	96% 45s	98% 38s	98% 46s	93% 45s
21	95% 35s	95% 38s	96% 44s	94% 38s	96% 46s	96% 46s
22	98% 35s	97% 38s	97% 43s	96% 39s	93% 45s	97% 44s
t_{mean}	35s	38s	42s	37s	45s	44s

Table 56: Results when using an intensity based appearance model and a three colour channel training and test dataset (scale 90% - 100%).

	Scale (as percentage of original image), Eye (dataset 4,5 or 6).								
Ex	60%	60%	60%	70%	70%	70%	80%	80%	80%
	4	5	6	4	5	6	4	5	6
0	89% 129s	88% 224s	90% 226s	90% 153s	91% 362s	91% 261s	98% 306s	98% 308s	92% 410s
1	88% 121s	89% 231s	90% 225s	90% 152s	90% 256s	91% 360s	98% 296s	98% 303s	92% 314s
2	89% 119s	88% 221s	90% 224s	90% 257s	90% 259s	91% 256s	98% 296s	98% 314s	98% 310s
3	89% 119s	89% 233s	90% 228s	91% 152s	90% 360s	91% 259s	98% 307s	98% 323s	99% 414s
4	88% 121s	89% 224s	90% 222s	89% 153s	91% 359s	91% 260s	98% 296s	90% 313s	98% 305s
5	89% 119s	89% 225s	90% 224s	91% 153s	90% 256s	91% 360s	98% 307s	98% 309s	98% 409s
6	88% 120s	88% 223s	90% 224s	90% 258s	90% 260s	90% 256s	98% 305s	98% 310s	91% 411s
7	88% 118s	89% 221s	89% 225s	90% 259s	90% 258s	93% 255s	92% 305s	98% 309s	92% 306s
8	88% 121s	89% 231s	90% 228s	91% 153s	91% 361s	92% 261s	90% 308s	98% 303s	90% 302s
9	88% 128s	89% 225s	89% 223s	88% 257s	90% 256s	91% 260s	98% 304s	99% 308s	98% 308s
10	88% 119s	88% 222s	89% 222s	91% 259s	90% 256s	91% 261s	98% 307s	98% 305s	98% 303s
11	88% 117s	88% 220s	90% 216s	91% 258s	87% 258s	89% 255s	98% 304s	96% 305s	98% 304s
12	88% 118s	88% 223s	89% 222s	89% 256s	90% 257s	91% 256s	91% 295s	99% 302s	92% 302s
13	89% 120s	89% 222s	90% 220s	90% 256s	90% 258s	92% 257s	98% 295s	98% 302s	98% 304s
14	89% 119s	89% 224s	90% 223s	90% 153s	90% 360s	91% 260s	98% 296s	98% 304s	92% 304s
15	88%	89%	90%	90%	90%	91%	98%	98%	92%

	119s	221s	222s	258s	256s	258s	294s	315s	307s
16	89% 119s	89% 224s	89% 223s	89% 256s	89% 256s	91% 258s	98% 307s	98% 308s	90% 411s
17	89% 119s	89% 223s	90% 223s	90% 259s	89% 259s	92% 267s	98% 306s	99% 308s	90% 410s
18	89% 119s	89% 223s	90% 225s	90% 260s	91% 259s	91% 265s	98% 310s	91% 307s	99% 408s
19	89% 120s	89% 224s	90% 224s	90% 259s	91% 259s	91% 258s	92% 297s	98% 312s	98% 304s
20	89% 119s	88% 223s	89% 224s	90% 256s	90% 256s	91% 261s	98% 309s	98% 305s	98% 304s
21	88% 118s	90% 224s	90% 223s	90% 153s	88% 362s	91% 259s	91% 297s	98% 304s	98% 311s
22	88% 121s	88% 224s	89% 221s	90% 257s	90% 256s	91% 261s	98% 304s	98% 305s	92% 305s
t_{mean}	120s	224s	223s	221s	284s	268s	302s	307s	337s

Table 57: Results when using a 2x2 mDCT based appearance model and a three colour channel training and test dataset (scale 60% - 80%).

	Scale (as percentage of original image), Eye (dataset 4,5 or 6).					
Ex	90%	90%	90%	100%	100%	100%
	4	5	6	4	5	6
0	98% 355s	99% 453s	98% 367s	99% 535s	99% 540s	99% 527s
1	98% 357s	99% 459s	99% 354s	99% 520s	99% 543s	99% 523s
2	98% 357s	99% 465s	99% 467s	99% 529s	99% 529s	99% 528s
3	99% 370s	99% 480s	98% 471s	99% 537s	99% 537s	97% 529s
4	98% 359s	99% 478s	98% 466s	99% 544s	99% 533s	99% 544s
5	99% 353s	99% 451s	98% 364s	96% 548s	91% 546s	91% 546s
6	99% 358s	99% 462s	90% 461s	92% 533s	99% 530s	99% 553s
7	99% 361s	99% 457s	99% 465s	99% 530s	99% 530s	96% 528s
8	99% 361s	99% 459s	98% 356s	99% 518s	92% 528s	99% 529s
9	99% 357s	99% 467s	99% 472s	99% 528s	99% 521s	99% 527s
10	99% 358s	98% 460s	99% 358s	99% 530s	99% 518s	99% 527s
11	99% 369s	99% 464s	98% 463s	96% 515s	99% 525s	99% 530s
12	99% 353s	99% 451s	98% 356s	96% 531s	91% 530s	99% 541s
13	98% 344s	99% 458s	99% 355s	99% 529s	97% 527s	99% 528s
14	92% 345s	99% 458s	98% 353s	99% 519s	99% 527s	99% 524s
15	99%	99%	99%	92%	99%	99%

	352s	454s	365s	518s	531s	524s
16	91% 358s	99% 462s	98% 357s	99% 518s	99% 523s	99% 520s
17	91% 343s	91% 458s	99% 357s	98% 521s	99% 535s	99% 528s
18	98% 353s	98% 451s	98% 357s	95% 518s	96% 528s	99% 523s
19	91% 355s	99% 451s	99% 363s	98% 528s	99% 521s	99% 526s
20	99% 358s	99% 476s	99% 466s	99% 516s	99% 526s	96% 530s
21	99% 359s	98% 464s	99% 464s	96% 518s	99% 529s	99% 528s
22	99% 370s	97% 468s	98% 466s	99% 518s	99% 525s	99% 531s
t_{mean}	356s	461s	405s	526s	529s	530s

Table 58: Results when using a 2x2 mDCT based appearance model and a three colour channel training and test dataset (scale 90% - 100%).

	Scale (as percentage of original image), Eye (dataset 4,5 or 6).								
Ex	60%	60%	60%	70%	70%	70%	80%	80%	80%
	4	5	6	4	5	6	4	5	6
0	90% 899s	90% 1000s	90% 2542s	91% 1303s	91% 2833s	90% 1293s	93% 1657s	94% 3296s	94% 3292s
1	90% 902s	90% 1005s	90% 2546s	91% 1283s	91% 2831s	90% 1286s	91% 1653s	94% 3292s	94% 3297s
2	90% 903s	90% 1010s	90% 2555s	91% 1280s	91% 2823s	91% 1419s	93% 1650s	94% 3286s	93% 3291s
3	90% 915s	90% 1012s	90% 2541s	91% 1286s	91% 2826s	90% 1282s	92% 1657s	94% 3304s	91% 3296s
4	90% 901s	90% 1006s	91% 2541s	91% 1176s	91% 2990s	90% 1278s	93% 1657s	93% 3296s	91% 3296s
5	90% 915s	90% 998s	91% 2542s	88% 1175s	91% 2938s	91% 1317s	93% 1654s	91% 3324s	91% 3299s
6	90% 902s	91% 1007s	90% 2543s	91% 1303s	91% 2846s	91% 1323s	90% 1654s	91% 3280s	92% 3295s
7	90% 905s	90% 1017s	90% 2545s	86% 1304s	91% 2845s	91% 1309s	93% 1654s	93% 3290s	94% 3286s
8	90% 903s	90% 1018s	90% 2551s	91% 1301s	91% 2847s	90% 1299s	93% 1649s	94% 3265s	92% 1631s
9	90% 1027s	91% 1026s	90% 2557s	91% 1299s	91% 2847s	91% 1301s	93% 1625s	93% 3262s	93% 1626s
10	90% 920s	90% 1003s	89% 2560s	88% 1296s	91% 2841s	90% 1307s	93% 1619s	93% 3253s	93% 1629s
11	90% 904s	90% 1021s	90% 2558s	91% 1302s	91% 2844s	90% 1300s	91% 1620s	93% 3251s	91% 1627s
12	90% 921s	90% 1018s	89% 2560s	91% 1298s	91% 2838s	90% 1306s	92% 1604s	94% 3154s	94% 1723s
13	90% 1024s	90% 1042s	90% 2563s	91% 1297s	91% 2835s	90% 1300s	94% 1636s	91% 3260s	93% 1620s
14	90% 920s	90% 1019s	90% 2561s	89% 1298s	91% 2837s	91% 1305s	93% 1618s	94% 3253s	93% 1628s
15	90%	91%	90%	91%	91%	91%	94%	93%	94%

	921s	1008s	2557s	1287s	2842s	1296s	1609s	3259s	1619s
16	90% 904s	90% 1008s	90% 2559s	91% 1300s	91% 2846s	90% 1299s	94% 1620s	94% 3248s	91% 1621s
17	90% 919s	90% 1025s	90% 2543s	91% 1297s	91% 2837s	91% 1307s	94% 1618s	93% 3251s	92% 1620s
18	90% 920s	90% 1022s	90% 2672s	91% 1298s	91% 2822s	91% 1308s	93% 1617s	93% 3251s	91% 1619s
19	90% 1026s	90% 1027s	90% 2626s	89% 1289s	91% 2839s	91% 1287s	92% 1619s	93% 3253s	93% 1627s
20	90% 916s	90% 1024s	91% 2547s	90% 1288s	91% 2832s	90% 1297s	90% 1609s	93% 3148s	93% 1724s
21	90% 903s	90% 1009s	90% 2540s	91% 1298s	91% 2834s	91% 1310s	93% 1620s	93% 3252s	93% 1629s
22	90% 921s	90% 1016s	90% 2545s	91% 1301s	91% 2835s	91% 1297s	93% 1620s	91% 3247s	94% 1622s
t_{mean}	925s	1015s	2558s	1285s	2848s	1305s	1632s	3259s	2213s

Table 59: Results when using a 2x2x2 mDCT based appearance model and a three colour channel training and test dataset (scale 60% - 80%).

	Scale (as percentage of original image), Eye (dataset 4,5 or 6).					
Ex	90%	90%	90%	100%	100%	100%
	4	5	6	4	5	6
0	98% 2182s	98% 3836s	98% 3828s	97% 4403s	98% 4413s	99% 4503s
1	98% 2310s	99% 3834s	98% 3721s	95% 4416s	98% 4422s	99% 4514s
2	98% 2197s	99% 3835s	98% 3816s	95% 4416s	98% 4425s	99% 4508s
3	98% 2197s	98% 3828s	98% 3962s	97% 4420s	98% 4430s	99% 4517s
4	98% 2338s	98% 3857s	98% 3821s	96% 4406s	98% 4402s	99% 4504s
5	98% 2326s	98% 3858s	98% 3899s	95% 4388s	99% 4352s	98% 4348s
6	98% 2338s	98% 3862s	98% 3854s	95% 4354s	98% 4353s	98% 4243s
7	98% 2312s	98% 3872s	98% 3866s	95% 4249s	98% 4336s	99% 4347s
8	98% 2368s	99% 3857s	98% 4053s	95% 4369s	98% 4356s	98% 4331s
9	98% 2196s	97% 3828s	98% 3848s	95% 4245s	99% 4347s	99% 4322s
10	98% 2166s	99% 3794s	98% 3791s	95% 4248s	99% 4349s	99% 4322s
11	98% 2160s	98% 3799s	98% 3872s	98% 4241s	98% 4337s	99% 4343s
12	98% 2146s	99% 3829s	98% 3806s	95% 4228s	98% 4353s	98% 4348s
13	98% 2165s	98% 3838s	99% 3839s	95% 4229s	98% 4334s	99% 4326s
14	97%	99%	98%	96%	99%	99%

	2308s	3837s	3728s	4243s	4351s	4318s
15	98% 2183s	98% 3824s	98% 3809s	96% 4226s	98% 4323s	99% 4333s
16	99% 2182s	98% 3811s	98% 3819s	96% 4240s	98% 4356s	99% 4349s
17	98% 2176s	98% 3812s	98% 3812s	95% 4352s	99% 4229s	99% 4340s
18	98% 2166s	99% 3810s	98% 3806s	99% 4225s	98% 4339s	99% 4323s
19	98% 2165s	98% 3808s	98% 3807s	95% 4243s	98% 4354s	98% 4350s
20	99% 2169s	99% 3806s	97% 3819s	98% 4248s	99% 4353s	99% 4458s
21	98% 2179s	99% 3822s	98% 3809s	95% 4352s	98% 4360s	99% 4238s
22	98% 2169s	98% 3818s	99% 3812s	94% 4355s	98% 4358s	99% 4236s
t_{mean}	2221s	3829s	3834s	4308s	4357s	4366s

Table 60: Results when using a 2x2x2 mDCT based appearance model and a three colour channel training and test dataset (scale 90% - 100%).