MASTER IN COMPUTER SCIENCE

Segmentation and edge detection of medical images

De Jacquier De Rosée, Alexandre

Award date:
1998

FACULTÉS UNIVERSITAIRES NOTRE-DAME DE LA PAIX, NAMUR

INSTITUT D'INFORMATIQUE

RUE GRANDGAGNAGE, 21, B-5000 NAMUR (BELGIUM)

Segmentation and Edge Detection of Medical Images.
Segmentation et Détection de Contours dans les Images Médicales.

Alexandre de Jacquier de Rosée

Mémoire présenté en vue de l'obtention du grade de

Licencié en Informatique

Co-Promoteurs:
Professeurs Jean Fichefet et Jean-Paul Leclercq

Année Académique 1997 - 1998

1.1 Abstract (English)

The objective of this final-year thesis (mémoire) is to create a synthesis of the classical, and some more recent, methods of segmentation and edge detection, with a bias towards automating the procedure. We have a particular interest in segmentation and edge detection in the field of medical imagery, as a required step in the process of image coregistration.
The methods are presented in order of increasing complexity, and each is explained step-by-step.

Some of the methods have been implemented for use on three-dimensional images (most of the references are concerned with two-dimensional images), and their results are presented.
Judgement of segmentation and edge detection methods from their results is a very controversial subject in image processing, and there is no widely used method for it. A thorough evaluation of the methods from the results of the implementations that were made has not been attempted, because

-the author has not enough knowledge in human anatomy to judge by himself a segmentation or edge detection in an image from the specialised field of medicine
-no "ground truth" results were available for each image, to compare results with
-the organisation of evaluation sessions with a number of judges ([1]) was not feasible.

Thus all estimations of performance in terms of quality of the output are conclusions from related work, or obvious deductions from the mechanism of the algorithm.

Also included alongside this document is the source code behind the implementations, and a set of Internet bookmarks with links to the references from the Internet, and other sites of interest on the subject.

1.2 Résumé (Français)

L'objectif de ce Mémoire de dernière année est de créer une synthèse des méthodes classiques, et certaines plus récentes, de segmentation et de détection de contours dans les images, avec une orientation vers l'automatisation des procédés. Nous avons un intérêt particulier pour la segmentation et détection de contours dans le domaine de l'imagerie médicale, comme une étape nécessaire de la coregistration d'images médicales.
Les méthodes sont présentées par complexité croissante, et chacune est expliquée étape par étape.

Certaines des méthodes ont été implémentées pour application sur des images tridimensionnelles (la plupart des références concernant les images bidimensionnelles), et leurs résultats sont présentés.
L'évaluation des méthodes de segmentation et de détection de contours à partir de leurs résultats est un sujet très controversé dans le domaine du traitement d'images, et il n'existe pas de méthode largement répandue pour cela. Une évaluation complète des méthodes implémentées à partir de leurs résultats n'a pas été tentée, car

-l'auteur a pas assez de connaissances en anatomie humaine pour juger par lui-même le résultat d'une segmentation ou d'une détection de contours dans une image d'un domaine spécialisé tel que la médecine
-aucuns résultats reconnus de segmentation ou détection de contours n'étaient disponibles, pour comparaison avec chaque résultat obtenu dans ce travail
-l'organisation de sessions d'évaluation par des juges ([1]) n'était pas possible.

Ainsi toutes les estimations de performances qualitatives sont des conclusions d'autres travaux, ou des déductions évidentes à partir du mécanisme de l'algorithme.

Annexé au texte, on trouvera le code source des implémentations réalisées, et un ensemble de signets Internet contenant des liens aux sites Internet servant de références et autres sites d'intérêt sur le sujet.

2. Table of contents

3. List of Figures

4. Preliminaries

4.1 Medical images

The images used in modern medicine are increasingly three-dimensional images. These are bitmap images, and an element of the data array is called a voxel ("volume pixel"); the elements making up a two-dimensional image are pixels ("picture element").
Medical images are grey-level images, so the value at a voxel is an intensity. In some cases, the grey levels may be replaced by colours; since these colours are not related to the physical colour of the object being imaged they are known as pseudo-colours.
The images most used in medical coregistration (the larger project that this work fits into; see section 4.2) are images of the brain, since it moves very little, and its shape remains the same; this means that there will be, relative to other body parts, little change from one image of the head to the next; these are ideal properties for coregistration.
The possible sources for these images are: PET (Positron Emission Tomography), MRI (Magnetic Resonance Image), SPECT (Single Photon Emission Computed Tomography), CT (Computed Tomography), MRS (Magnetic Resonance Spectroscopy), ultrasound, or other; images of the same source are said to be of the same modality.
The real-world dimensions of a voxel are different; for a 3D image, often the scanning creates layers of 2D images, where the distance between layers is different from the width and height of the inter-layer pixels. The real-world size of a voxel is usually between 1 and 4 mm (PET: 2.0 * 2.0 * 3.125, MR: 1.88 * 1.88 * 1.3, SPECT: 4.4 * 4.4 * 4.4). This must be taken into account for most image transformations (such as rotation).

It is important to know that different modalities show anatomical structures very differently (some especially detect extrinsic objects such as injected fluids or solid markers).
Thus, scans of different modalities from a single subject will look quite different from one another. For example, white matter structures will appear bright in MRI and dark in PET, whereas grey matter will be darker than white matter in MRI and brighter than white matter in PET [2].
Also, different anatomical structures may appear with similar voxel intensities (e.g. the scalp and white matter in MR images).

"SPECT, PET, and MRS provide functional information, but delineate anatomy poorly, whereas MRI, ultrasound, and X-ray imaging (including CT) depict aspects of anatomy, but provide little functional information" [3].

The figures below show horizontal (transverse) slices of the head from three modalities: PET, MR and SPECT.

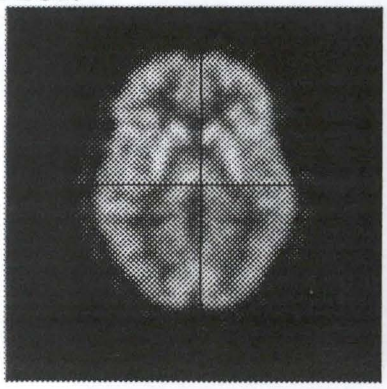
Figure 1: PET image

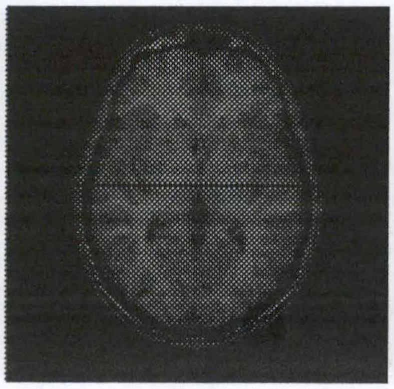
Figure 2: MR image

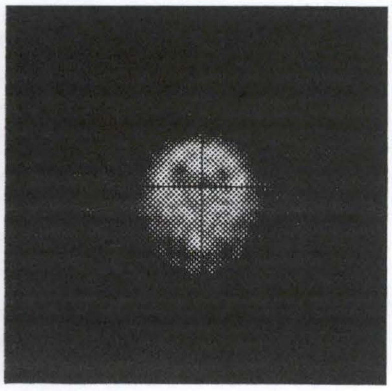
Figure 3: SPECT image

4.2 Medical Image Matching (Coregistration)

To perform Medical Image Matching, or coregistration ([2], [3], [4]), is to create an accurate map of data between two different images into a common space; this technique can provide unique insights not readily apparent when examining the images separately.
In many instances, it would be desirable to integrate the information obtained from two or more studies of different modalities of the same patient. Also useful is registration of monomodal images, whether a study of the same patient over a period of time to detect any changes, or a study of a group of patients.

In order to perform the difficult task of medical image matching, one must take into account important factors such as the different imaging modalities' distinct physical properties, and the differences in patient positioning.

A general coregistration technique methodology was presented by Gerlot-Chiron and Bizais:

(1) extraction of features in each image,
(2) pairing of these features,
(3) choice of a geometric transformation and estimation of its parameters,
(4) effectuation of this transformation.

It is in step (1), the extraction of features in each image, that segmentation and edge detection come into play, as they themselves are feature-extraction techniques. Whether one uses segmentation or edge detection (or both) for feature extraction depends on the type of feature looked for, but also, if knowledgeable, on the types of images to be coregistered.

5. Segmentation

"Image segmentation is a process which partitions the spatial domain of an image into mutually exclusive subsets. Each subset is uniform and homogeneous" ([5]).
These subsets are known as regions, and the purpose of segmentation in medical imaging is to have regions which accurately represent objects of interest in the image.
Regions are homogeneous if they obey a predetermined homogeneity rule; homogeneity ensures that voxels in a region are sufficiently statistically similar to the other voxels in the same region, more than to those in any other region. There is no mathematical definition for region homogeneity, and it often varies from one method to the next; one possible homogeneity requirement is that the difference between voxel intensity and region mean value must be less than a threshold.
It is not required that a region be connective; it is said that there is region connectivity if any two voxels belonging to a region can be joined by a path of voxels belonging to that region.
The output of a segmentation algorithm is an image where voxel values are labels, indicating region number. In the case of a single object and background, the output will be a binary image.
It is then easy to create an image containing only the object(s) of interest, by subtracting any unwanted regions.

Current clinical tools allow radiologists to "hand-segment" objects in images by drawing contours around the objects of interest on a slice-by-slice basis. This is slow and tedious, and sometimes not very accurate because prone to human error.

5.1 Threshold Segmentation

Threshold segmentation works by comparing voxel intensity with one or more threshold values, such that all voxels with intensities above one threshold will belong to a different region than those below that threshold.
Since the pixels or voxels of a same object in an image tend to have the same intensity values, an accurate set of thresholds will segment the image into meaningful regions.
If a single object is of interest in the image, a single threshold is necessary to distinguish the object from the background.

The major advantage of threshold segmentation is speed; once the threshold(s) is/are known, there is a single comparison for each voxel. As will be shown shortly, for automatic segmentation, calculating the threshold itself can be more complicated.

Because thresholding is performed only on a voxel basis, and no neighbourhood information is taken into account, threshold segmentation does not guarantee region connectedness (a region R is said to be connected if any two voxels belonging to R can be connected by a path consisting entirely of voxels belonging to R).
Also, threshold segmentation will be hazardous if different actual objects, or the object and background, share some of the same intensities (i.e. their intensity spectres overlap).

To avoid the first problem, if it is known that the object of interest is the largest, connectivity can be used to remove smaller unconnected objects in the image; there may still be some unwanted objects connected though, the misclassification being due to intensity overlap between the objects.

Below are some methods for calculating the threshold.

5.1.1 User Choice

The user must choose the threshold; this will be an iterative procedure by the user, as the optimal threshold is not likely to be guessed.

5.1.2 Mean Value

Using the mean grey level in the image as a threshold is a simple procedure, but rarely good, because the segmentation causes about half of the pixels to become black and about half to become white. This is only appropriate if it is known that the object takes about half the image volume ([6]).

5.1.3 Threshold from Histogram

If necessary, please refer to the annex section on histograms.

Using the histogram to select a threshold ([6], [7]) is done very often for finding thresholds.
Objects (including the background) in an image form the bulk of the image's volume, the rest being areas of transition between the objects. Also, each object will tend to have a similar intensity throughout.
Therefore, when a threshold is obvious, it occurs at the local minimum between two peaks in the histogram.

If the image contains one object and one background having homogeneous intensity, it usually possesses a bimodal histogram (the histogram has two peaks), and this selection for the threshold would appear to be a good one.

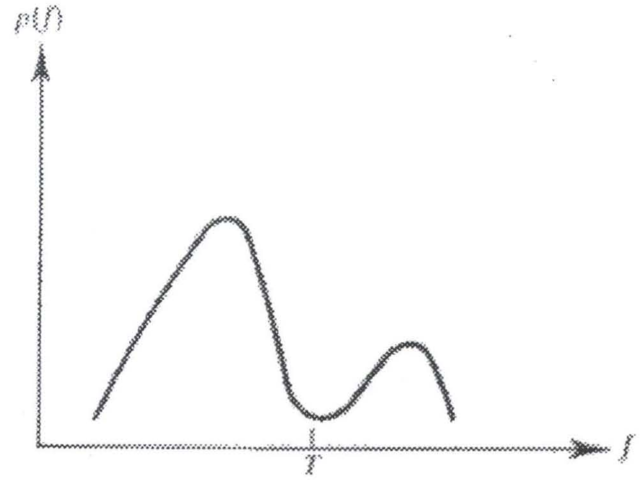

Figure 4: selecting a threshold at a minimum between two maximums

It is possible, however, that the transitions between two regions contribute much, making it difficult to distinguish the histogram maximums representing distinct objects.

To avoid this, a suggested method is to remove these edge voxels from the image histogram by performing edge detection and excluding the detected edge voxels from the histogram, when building it.

Another approach, is to build a histogram where the contribution from each voxel depends on a function of an edge detector output.

In a normal histogram, the contribution from each voxel to its intensity value is 1.
In this case, it should be a monotonically decreasing function such as $\dfrac{1}{1+|e(x,y,z)|}$

(where e is the edge detector function)

The resulting histogram tends to have well-defined peaks and valleys.

Finding the first peak in the histogram is simple: it is the largest value. However, the second largest maximum is probably not the second peak (this is usually right next to the peak), but the peak of another histogram lobe. Thus, locating the second peak is harder than it appears at first.
There are a large number of suggested methods to automatically find the two maxima and minimum:

5.1.3.1 Top-Hat transformation

The local histogram maxima can be detected using the Top-Hat transformation ([7]), which transforms a signal such that it emphasises peaks (for example, it has been used to detect stars in astronomical images).
The top-hat transformation is the original signal minus its opening (please refer to sections on mathematical morphology pages 17 and 40).
Because the opening operation will destroy peaks, subtracting it from the original signal will highlight peaks.
If the signal is noisy, either a larger structuring element should be used, or a smoothing operation should be performed.

The figures below show the top-hat transformation applied to a sample histogram. The structuring elements used is

1	1	1	1	1	1	1

The largest peak is at intensity 500 and is off the scale; the second peak can be easily seen, but all other peaks are due to noise; since these are rather large, smoothing would have been recommended.

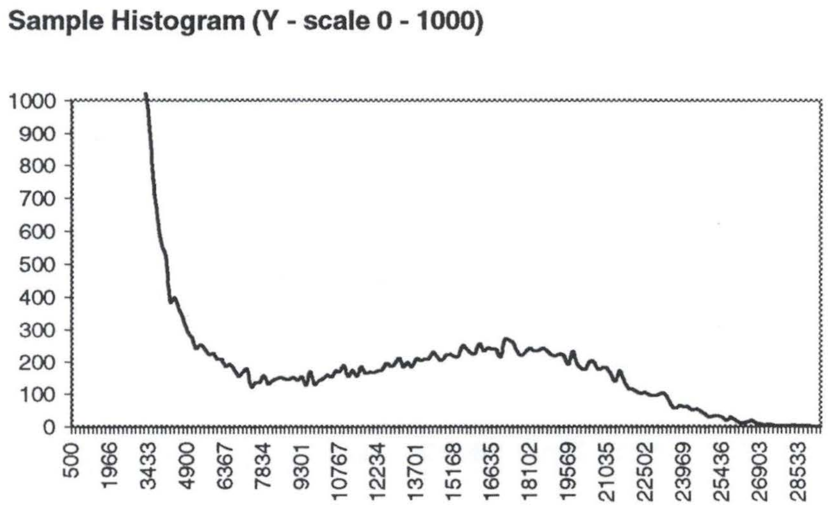

Figure 5: histogram of PET brain image

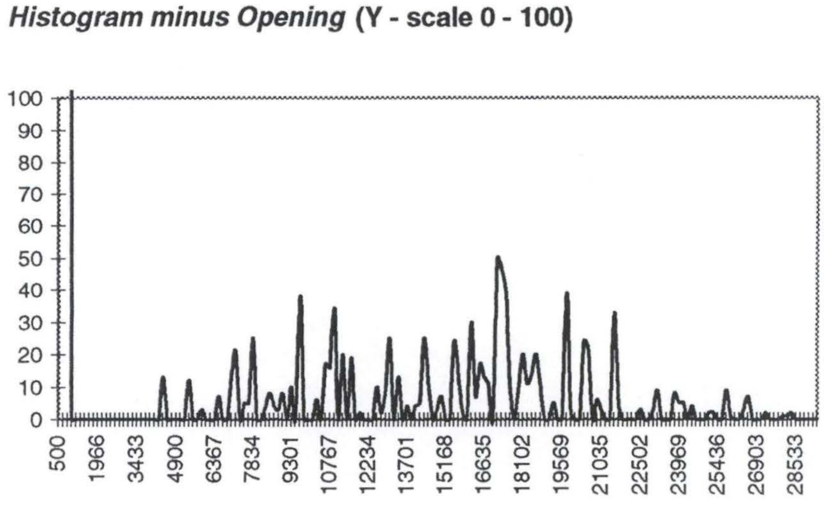

Figure 6: Top-Hat Transformation

5.1.3.2 Approximation

A simple trick that frequently works well enough is to look for the second peak by multiplying the histogram values by the square of the distance from the first peak. This gives preference to peaks that are not close to the maximum.

My method for finding the minimum was to find the longest run of minimum, and setting the threshold to the middle.

This is the method which was implemented for automatic threshold segmentation.

5.1.3.3 Gaussian Curves and Least-Squares Method

The idea behind this method ([6]) is to approximate the peaks by Gaussian curves. It is necessary to know the number of peaks, so it is assumed that the histogram is bimodal.
A bimodal histogram can be expressed as the sum of two Gaussians; to "fit" two Gaussians to the histogram, an intensity is iteratively selected such that if each side of the curve is assumed to be a Gaussian curve, the sum of the squared residuals will be minimised (least squares method):

$$\text{Residual} = \sum_{i=0}^{I-1} (G_1(i)+G_2(i)-F(i))^2$$

where I is the maximum intensity (x-axis on the histogram)

The threshold selected is the intersection point of the two Gaussians.

The figures below show the result of automatic threshold segmentation (implemented as part of this project), the threshold being calculated from the image histogram using approximation 5.1.3.2 seen above.
In figure 7, it is difficult to see that apart for the brain object, there is much data variation in the background. This has been equalised (see annex on histograms for histogram equalisation) to show this, in figure 8. The same has been done for figure 11, which would otherwise look almost entirely black. Equalisation is not part of the threshold segmentation process, it is for illustration purposes only.

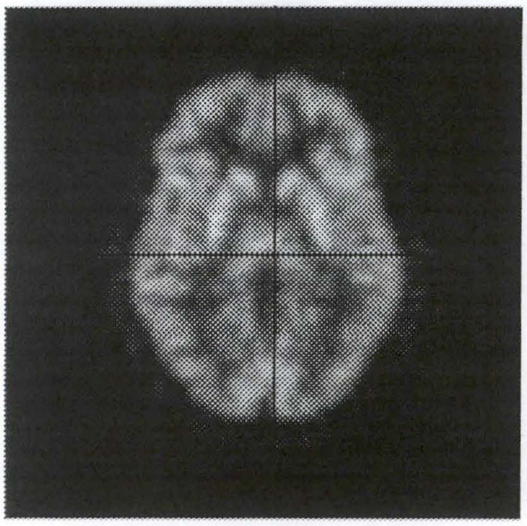
Figure 7: PET image

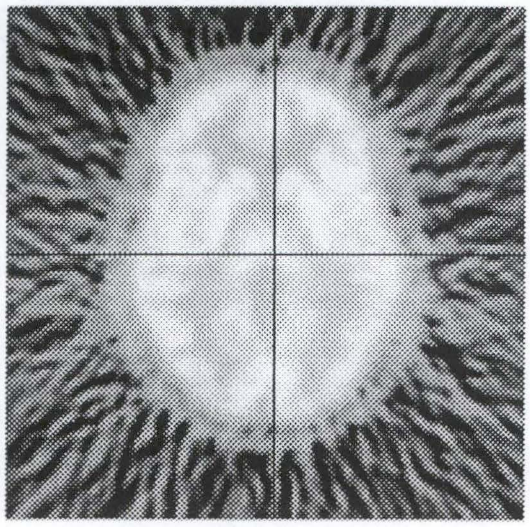
Figure 8: equalised PET image

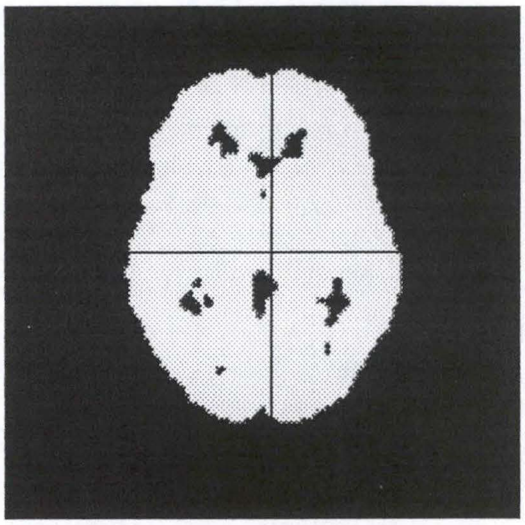
Figure 9: binary image result of segmentation

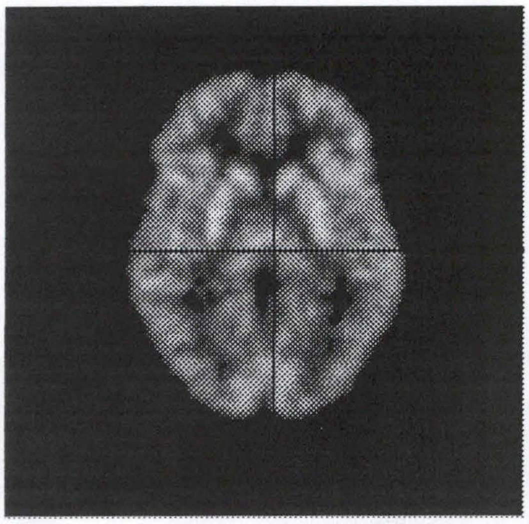
Figure 10: object, background removed

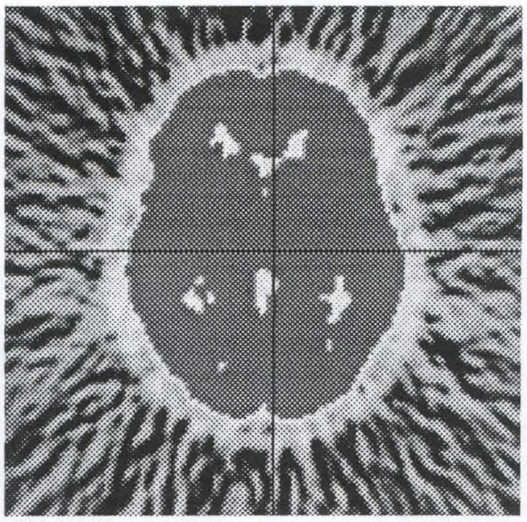
Figure 11: background, object removed (equalised)

5.1.4 Using edge pixels (Weszka)

It has been suggested earlier to build the histogram without the edge voxels, to better distinguish the histogram maximums representing distinct objects.
Weszka ([6]) suggested a similar idea: find the threshold from a histogram built considering only voxels with large Laplacians (edges will be at zero-crossings of the Laplacian - refer to annex on the Laplacian).
First the Laplacian of the image is computed, by convolution (see annex on convolution) with the mask:

0	1	0
1	-4	1
0	1	0

Now a histogram of the original image is found considering only those pixels having large Laplacians; it is suggested to consider those having Laplacians greater than 85% of the others (i.e. those in the 85th percentile), and exclude all the others.
Then the threshold is calculated from this histogram.

Using a better approximation to the Laplacian should give better results, but in many cases this simple procedure will show an improvement over the previous methods.

The figures below show the results of Weszka's edge pixel segmentation on a MR image, using a percentile of 0.9 (implemented as part of this project). The result is very different from the normal threshold segmentation.
Tests on the MR image gave better results with the percentile of 0.9 instead of the recommended 0.85, the latter including too much background noise in the segmented object.
In MR images different actual objects have the same intensities, so the object is unconnected, as said previously this is a problem of threshold segmentation.

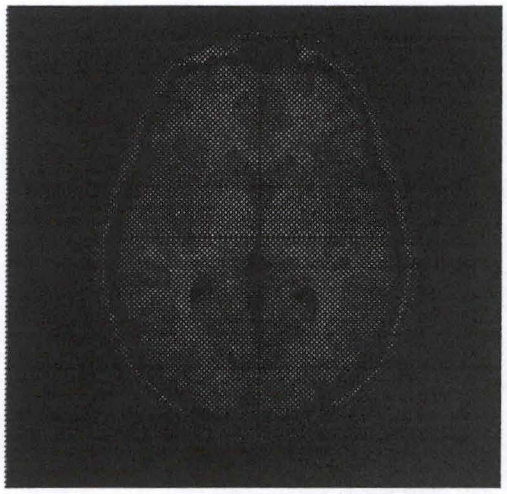
Figure 12: Original MR image

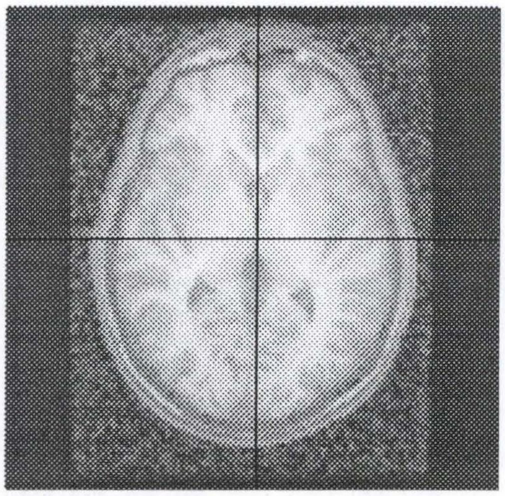
Figure 13: equalised MR image

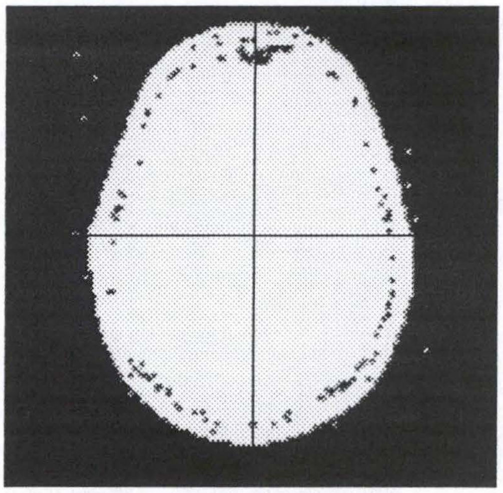
Figure 14: binary image result of segmentation, percentile 0.9

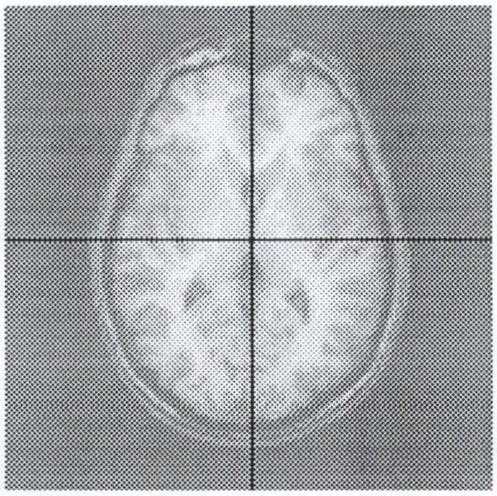
Figure 15: object with background removed (equalised), percentile 0.9

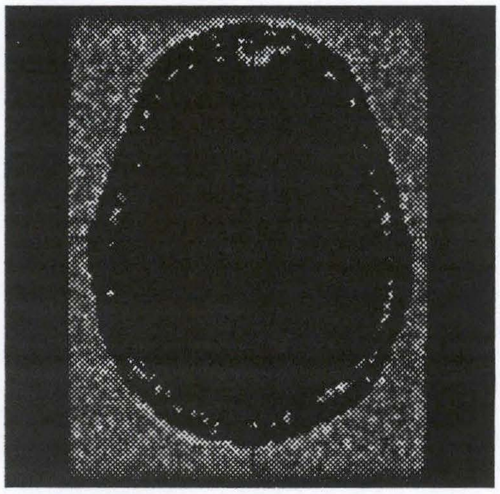
Figure 16: background, object removed, percentile 0.9

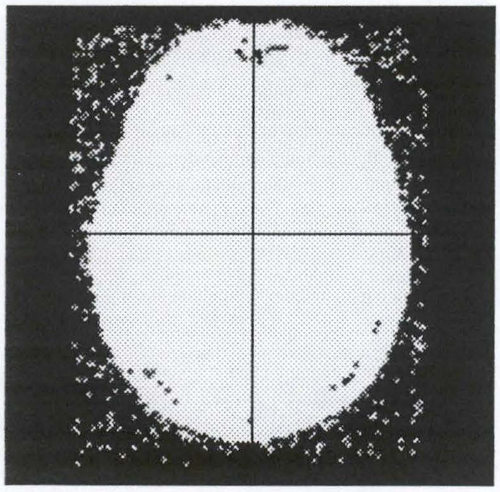
Figure 17: binary image result of segmentation, percentile 0.85

5.1.5 Iterative selection

In this method, the threshold is iteratively calculated until there is no change in the last iteration. The initial guess for the threshold value is the mean grey level in the image.
For each current threshold, the mean grey value of each segmented region is calculated; the threshold for the next iteration is

$T_{k+1} = (\text{mean1}_k + \text{mean2}_k)/2$

When $T_{k+1} = T_k$, the algorithm stops.

The histogram can be used to quickly calculate the mean values.

5.1.6 Using Entropy

"Entropy is a measure of information content".

The total entropy of an image is

$$H_T = -\sum_{i=0}^{I-1} p_i \log(p_i)$$

where p_i is the probability of grey level I
and I is the largest intensity.

If the image is segmented by a threshold, the entropy for each region can be calculated.
This method is to search for the threshold that will maximise the sum of the entropies of both segmented regions:

$$H = H_1 + H_2$$

5.2 Binary Mathematical Morphology

Binary mathematical morphology ([6], [11]) provides a set of methods to erode and dilate a binary image.
It is not a segmentation algorithm in itself, but it can be used to deal with misclassification due to threshold segmentation (or other intensity classification) of images, by incorporating neighbourhood information. If used for this purpose, the original image must be one in which different objects share intensities, such as MR images.
Binary mathematical morphology is used in more advanced segmentation algorithms such as that presented in [11].

The application of the mathematical morphology operators is by convolution of an image with a mask (known in this case as a structuring element). The masks can be spherical or cube-shaped, where the elements are set to one.

The possible operations are (for an image I, structuring element S):

5.2.1 Erosion

changes the value of pixel *i* in I from 1 to 0, if the result of convolving S with I, centred at *i*, is less than some predetermined value (this value can be the number of pixels in S).
S is also known as the *erosion kernel.*

5.2.2 Dilation

changes the value of pixel *i* in I from 0 to 1, if the result of convolving S with I, centred at *i*, is more than some predetermined value (this value can be 0).
S is also known as the *dilation kernel.*

5.2.3 Conditional Dilation

this is a dilation operation with an added condition: only pixels that are 1 in another binary image, I_c, will be changed to 1 by the dilation process. It is equivalent to masking the results of the dilation by the image I_c (or doing an AND operation between their pixels).

5.2.4 Opening

an erosion followed by a dilation with the same structuring element. The name opening describes the observation that this tends to open small gaps between the object. Opening also tends to remove noise in the background.

5.2.5 Closing

an erosion followed by a dilation with the same structuring element. A closing operation will fill, or close, the gaps between objects in the image, tends to remove noise in the objects.

These operations can be used to 'shave off' the misclassified objects in a segmentation output, then use connectivity to find the largest connected component in the image. The sequence of operations is as follows:

- Perform an erosion operation on the input with a spherical structuring element with radius corresponding to the thickness of the connector between wanted and unwanted misclassified object. The erosion should remove this connector.
- Use connectivity to find the largest connected object in the image.
- Perform conditional dilation, with a structuring element slightly larger than the one used in erosion, conditioned on the original input image.

The figures below show this sequence of events on an MR image which is the output of an intensity-based segmentation:

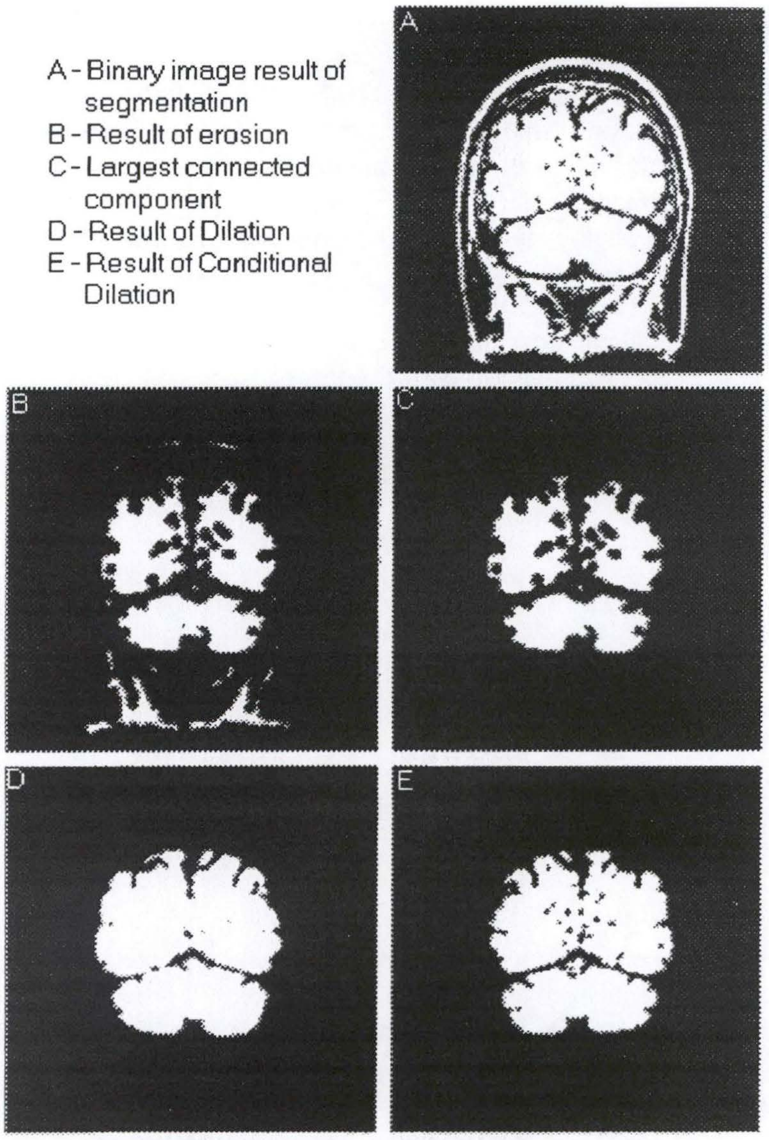
A - Binary image result of segmentation
B - Result of erosion
C - Largest connected component
D - Result of Dilation
E - Result of Conditional Dilation
A
B
C
D
E

Figure 18: Binary mathematical morphology for shedding unwanted connected objects

In the tests conducted in [11], this method successfully isolated the brain tissue from the cranium for about 90% of MR images.

Note that this method also removes background noise, if the structuring element is larger than the noise elements, but not noise in the object.

5.3 Region Growing and Merging

Segmentation algorithms based solely on voxel intensity fail to use the fact that voxels in the same neighbourhood tend to have similar statistical properties and belong in the same image regions. To produce homogeneous connected image regions, segmentation algorithms must therefore incorporate both proximity and homogeneity.
The region growing approach is to select some voxels representing distinct image regions, and grow these, until they cover the entire image. The voxels from which the regions grow are called seeds. The seeds are easily chosen interactively by the user.
The growth mechanism is:

- for each border voxel in the image (a border voxel is one which has been assigned to a region and has at least one neighbouring voxel which has not been assigned), regardless of region, check if there are any unclassified neighbour voxels;
- if so, to assign the neighbour voxels to the region, check if their sequential incorporation into the region will maintain region homogeneity.

  A possible homogeneity test is to see that the difference between voxel intensity and proposed region mean value must be less than a threshold.

  $$|f(k,l) - m_i| \leq T$$

- Repeat the above mechanism until every voxel has been assigned to a region, or until no changes were made in the last iteration.

The quality of the region growing algorithm depends heavily on the voxels chosen as seeds. A badly placed seed may, for example, grow from a transition area between two objects.

The image histogram, again, can be used to choose the seeds: image pixels whose intensity correspond to histogram peaks can be seeds.

If there's more than one initial seed per actual region, adjacent regions in the output with similar statistical properties must be merged.
A possible test to decide merging: if two adjacent regions have means close to each other, the regions are allowed to merge:

$|m_1 - m_2| < k\sigma_i$ I=1,2 σ is standard deviation.

Region merging can be incorporated in the growing mechanism, in order to construct a combined merging/growing segmentation algorithm, in which initial seeds are not necessary.
The image is scanned in a top-down, row-wise manner (for a two-dimensional image). Each pixel is therefore adjacent to at least one region (except the first scanned pixel). The pixel is considered for merging with all its neighbouring regions, and the proposed region which is most homogeneous is chosen. If the current pixel cannot be assigned to a region, it becomes a new region.
The quality of the combined region merging/growing depends on the choice of the merging rule.

One advantage of region-growing algorithms for three-dimensional images is that the seeds can often be determined in a single two-dimensional slice; these can therefore be found much more quickly, especially for their interactive selection.

5.4 Split-and-Merge

As its name indicates, the region splitting algorithms split regions until they are homogeneous (as opposed to region growing).
The splitting mechanism starts by assuming the entire image is one homogeneous region. Then,

- If the region is homogeneous, break.
- If the region is not homogeneous, split it into four (for a two-dimensional image) equally-sized regions.
- For each region, repeat the mechanism.

Therefore, when the algorithm stops, all regions will be homogeneous.

A major disadvantage of pure splitting techniques is that they create regions that, because they have been split from different regions, may be adjacent and homogeneous together, and should be merged.

Thus the split-and-merge algorithm includes merging and splitting at each iteration:

- For all regions, if a region is inhomogeneous, it is split into four new sub-regions.
- If two adjacent regions are homogeneous, and merging them would produce a new homogeneous region, they are merged. If more than one merge is possible, the best merge is chosen. Repeat this step until no merging is possible.
- Repeat the mechanism, until no further splitting or merging is possible.

The split-and-merge method will produce less, and smoother, regions than the basic splitting algorithm.

A major problem of splitting algorithms is that they produce regions with sharp square edges, since regions are split into squares.
This is because the homogeneity check is made over large regions; for example, if the region is large, a small inhomogeneous sub-region within it would not change global region statistics much. Checking if each pixel really belongs to a region would avoid this problem, but would be much more computationally expensive.

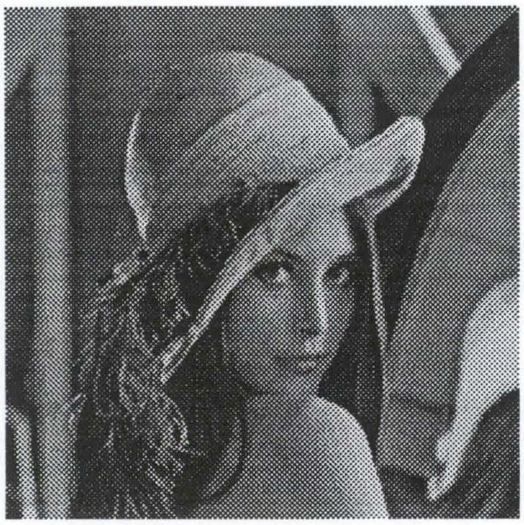
Figure 19: Original image

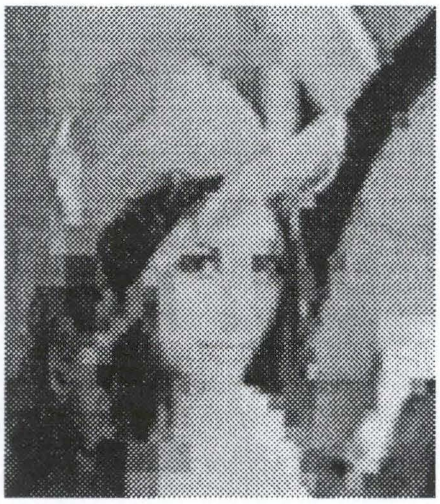
Figure 20: Segmentation by Region Splitting

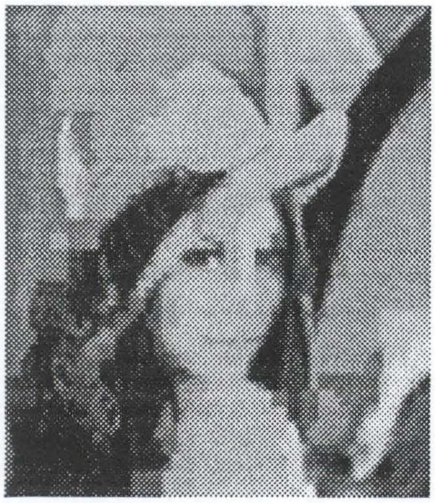
Figure 21: Segmentation by Split-and-Merge

5.5 Competitive Region Growing

The normal region growing algorithms have the disadvantage that they allocate voxels to a region without considering if that voxel would be better allocated to another region. If the scanning algorithm searches border voxels from top to bottom, left to right (in a two-dimensional image), regions nearer the top-left corner will be privileged; and it is even possible that most of the image will be allocated to the first region in a single pass.

Competitive region growing ([10], [12]) solves this problem: it doesn't artificially prioritise any regions, because all the regions are grown simultaneously, and it chooses the best pixel allocation from all possible competing allocations, so it is order independent.

Like in the normal region growing, an initial set of seeds must be specified, from which regions will grow.

Competitive region growing works by first building a set of all possible pixel-pairs where, where one pixel in the pair is allocated to a region, and its paired neighbour is not. For each pair a measure of homogeneity between the two pixels is calculated. Then,

- It selects the pair with the best calculated homogeneity value of this set. If the set is empty, then the entire image has been segmented.
- If the chosen pixel-pair's homogeneity value is higher than the limit, the segmentation is completed because further region growing would render the regions inhomogeneous.
- Otherwise, the pixel in the pair not belonging to any regions is allocated to the region of its pair.
- The chosen pixel-pair is removed from the set, which is updated with pixel-pairs including the newly added region pixel.
- Repeat the steps until no allocation is possible.

The major implementation difference between this and the normal region growing will be the memory requirements: to lessen the execution speed difference, the implementation should store the results of dissimilarity between pairs of pixels, to avoid recalculations.

The homogeneity measure calculated for each pair can be a measure of homogeneity of the proposed region; in this case some recalculation is required every cycle, as one region grows per chosen pixel-pair.

In the referenced article [12], the test image used was a synthetic, 256 by 256, 65536 grey-level (16 bits per pixel) image, with two homogeneous regions. Two tests were completed with different amount of noise. The segmentation only took 0.3 seconds on a Pentium-class computer. It was found to "work exceptionally well with noisy images".

The authors suggest their "algorithm should be used in automated and interactive segmentation where the definition of the segmentation limit causes problems or there is clear need for better image segmentation quality"

5.6 Relaxation algorithms

This is a similar algorithm to competitive region growing, in that each voxel is checked for best allocation with each region, until no changes are made in the last iteration. In this case, however, the segmentation method is not deterministic: for each voxel is kept a confidence vector that contains the probability of belonging to each region (so if there are three regions in the image, confidence vectors with three probabilities are stored for each voxel). The probabilities are also known as confidence weights ([7]).

Producing deterministic segmentation from confidence vectors is easy: voxels are allocated to the region it is most likely to belong to.

Probabilities must satisfy the relations, for each vector *k*, confidence weight *p*, *N* regions:

$$0 \leq p_k(i) \leq 1$$

$$\sum_{i=1}^{N} p_k(i) = 1$$

An initial estimate of the confidence vectors can be found from the histogram:

again, each histogram peak is taken as a distinct region.
The initial confidence weights are:

$$p_k^{(0)}(i) = \frac{1/d_i}{\sum_{i=1}^{N} 1/d_i}$$

d_i is the distance from the *i*th peak in the histogram. The probability is inversely proportional to this distance.

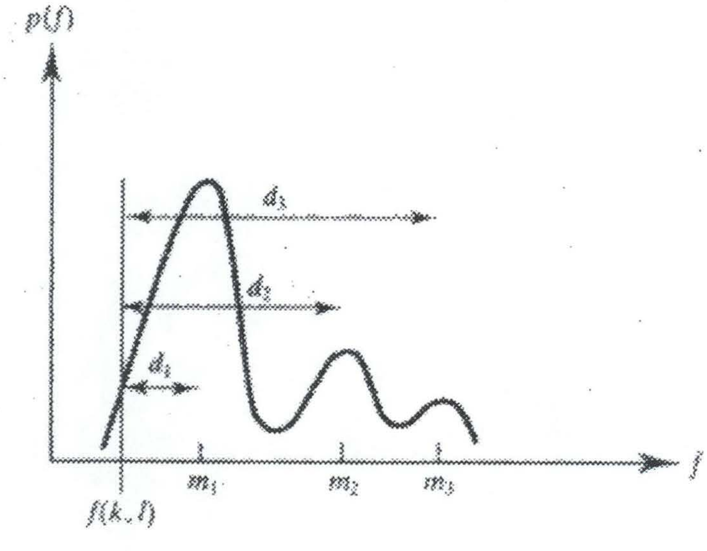

Figure 22: Distance of *f(k,l)* from the maxima

At this point, notice that if a deterministic segmentation was applied, like thresholding there could be no region connectedness, because confidence vectors are found only from intensity.

From the initial estimates, an iterative competition and co-operation between neighbouring pixels algorithm is applied until it reaches a steady state.
For this, a new function is introduced, called the probability function: this determines the odds that two regions could be found adjacent. The compatibility function *r* must have the range $-1 \le r(i,j) \le 1$.

If $r(i,j) < 0$ (Regions R_i, R_j are said to be incompatible, unlikely to be adjacent)
$= 0$ (Regions R_i, R_j are said to be independent)
> 0 (Regions R_i, R_j are said to be compatible, likely to become adjacent)

A method to estimate the compatibility function from the initial confidence weights is described in [13]. It can also be known a priori.
If the compatibility function is not known, I suggest to use it to specify which regions are definitely not adjacent (for example, any small regions can be classified as noise and therefore set to be unlikely to be adjacent to any other region).

In each iteration, each pixel receives confidence contributions from its pixel neighbourhood. The change in confidence weight for pixel *x* at step *n* is:

$$\Delta p_k^{(n)}(i) = \sum_j r(k,l)\, p_l^{(n)}(j) \qquad \text{where } j \text{ are the neighbouring pixels.}$$

The updated probabilities for each pixel at step *n* are given by:

$$p_k^{n+1}(i) = \frac{p_k^n(i)\left[1 + \Delta p_k^n(i)\right]}{\sum_{i=1}^{N} p_k^n(i)\left[1 + \Delta p_k^n(i)\right]}$$

The iterations stop when convergence is achieved. This method should remove small noisy regions within larger regions.

5.7 Watershed

The principle behind the watershed segmentation method ([14], [15]) is to consider a two-dimensional gradient image as a topographic surface, where the numerical value of each pixel stands for the elevation at this point.
A few definitions must be made in order to explain the watershed segmentation method:
A minimum in this gradient image is a set of connected pixels at the same height (a plateau), from which it is impossible to reach a lower point of altitude without having to climb.
These minima are subsets of distinct image regions, separated by lines of high altitude (high gradient values, thus edges), later defined as watershed lines.
A catchment basin associated with a minimum M is the set of pixels p such that a drop of water falling on p will slide down along the steepest slope into M.
The lines dividing catchment basins are known as the watershed lines. Just like in geography, the watershed lines are the lines the side of which determines the catchment basin that a drop of water would fall into.

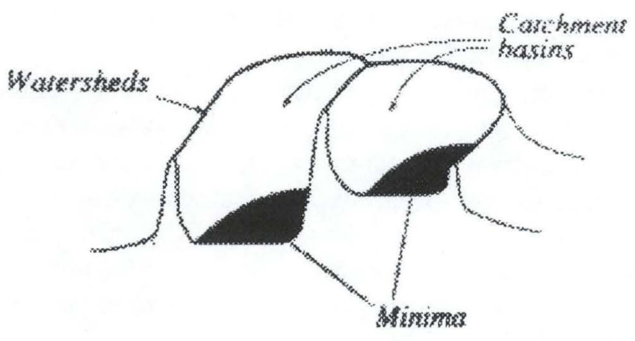

Figure 23: catchment basins, watersheds

The idea is to partition the gradient image into catchment basins, where these are distinct image regions, and the watershed lines are the edges separating them. At the end of the process the entire image will be segmented.

A problem with the rain analogy is that it is difficult to calculate the real flow path that a drop of water would slide down along, because of the discrete nature of the pixel grid (for example, it's difficult to calculate the gradient between four pixels, and there are only eight possible directions to a neighbouring pixel).

So the analogy used in implementations is that of flooding: the mechanism is to "slowly immerse the surface into a lake. Starting from the minima of lowest altitude, the water will progressively fill up the different catchment basins of the surface".
As the water rises, the catchment basins will meet where the watershed lines are lowest; the merging of waters coming from different sources is prevented (the analogy is to build dams where the waters meet) so that regions don't 'grow' into each other.
An animated image in [14] shows this very well.

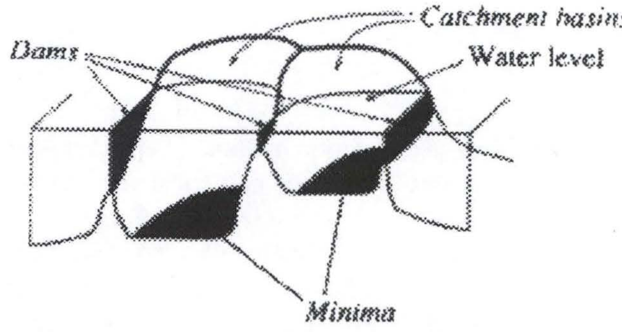

Figure 24: rising water level, dams

The algorithm is the following:

1. Calculate the image gradient.
   For each pixel value, from minimum to maximum:
2. find all connected subsets of pixels with current pixel value or less
3. compare each new subset with the last iteration; there are three possible cases:
   - if the subset encompasses a single subset from the last iteration, they correspond to the same catchment basin which has grown
   - if the subset encompasses no previous subset, it is a new catchment basin
   - if the subset encompasses two or more previous subsets, then the waters from these two catchment basins have merged; to determine where the watershed lines separates them, the geodesic influence zone of each previous subset is found.

Because there is always noise and other intensity variations in the image, straight watershed segmentation will result in over-segmentation (even if the gradient image is first blurred).
There are two possible ways to avoid this:
- remove irrelevant watershed lines (for example, by ignoring low image gradients)
- use predetermined markers representing distinct catchment basins; the algorithm is the same except that newly-flooded minima will no longer be considered as new catchment basins. At the end there as many regions as original markers. An animated image in [14] shows well how this works.

The watershed segmentation is considered to be powerful, but the L. Vincent and P. Soille article ([15]) suggest many improvements in the algorithm to speed up computations greatly.

5.8 Ridge Flow Models

This method by D. Eberly and S. M. Pizer [16] creates a hierarchy of many small primitive regions, and requires user interaction to select those that form the desired object.

A ridge is a line of junction of two surfaces sloping upwards toward each other. In the field of imagery, as can be expected, there are numerous definitions for ridges, but it can be said to be a line of local intensity maximums.

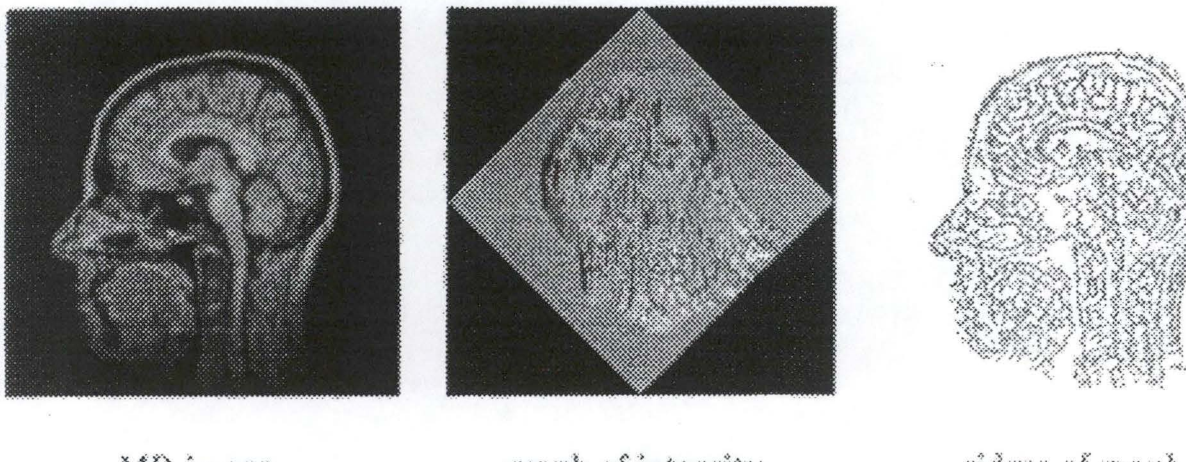

Figure 25: Ridges

The ridges are segmented and labelled into curvilinear segments, so that each labelled pixel has a maximum of two neighbouring pixels with the same label. This necessitates thin ridge lines, so a thinning algorithm is used to make the ridges one-pixel thick.

Then, the construction of primitive regions is analogous to the watershed segmentation method, in that at each pixel in the image the flow line is followed, and terminates at a ridge segment; all pixels on the flow line will be assigned to the label of the terminating ridge.

Segmentation like this of the original image gives a myriad of segmented regions, so a hierarchy of segmented images is made, where levels (known as scales) in the hierarchy have different amounts of blurring on them: the inner scale has no blurring, and the outer scale has intense blurring.

The blurring is done by convolution of a Gaussian kernel. The difference between levels is the standard deviation σ of the Gaussian. Blurring has the effect of annihilating detail, including ridges; as the scale σ tends to infinity, the blurred image tends to a constant.

The nodes in the hierarchical tree are the distinct regions; the links between the nodes are determined by how regions at one scale correspond to regions at the next scale.
A region of scale σ_n is considered a parent in the tree of any region(s) of scale σ_{n-1} that it overlaps more than any other. Higher scales will have less ridge segments, therefore less regions, so the outer scale will have very few regions in the segmentation.

Normally, some nodes in the tree will have only one child, and some will have many. However, the discretised process may introduce regions whose corresponding tree nodes have no children. These are not essential to the hierarchy and are removed from the tree.

To identify nodes in the tree, these are numbered, from the inner to the outer scales; a node label of -1 indicates that the node has a single child.

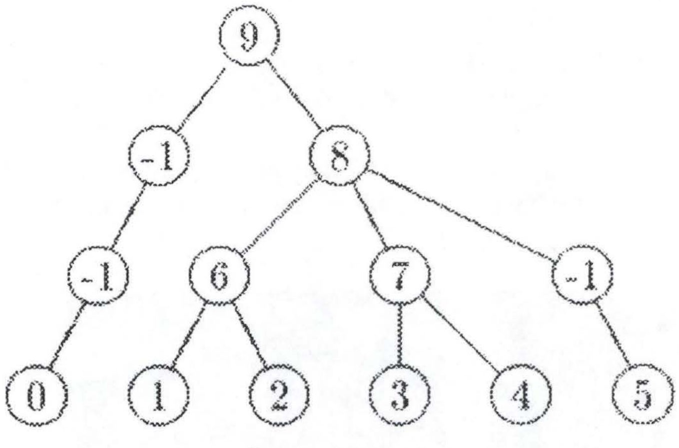

Figure 26: Scale Hierarchy

The purpose of this segmentation method is to allow the user to select a primitive region of the inner scale, then easily select any regions belonging to the desired sub-tree.

For this a graphical user interface tool was created which allows the user to traverse the hierarchy and quickly identify objects of interest.
The operations available to the user are:

select a region
select all regions which are children of the currently selected node ("add more" operation)
select all regions which are children of multiple selected nodes ("add more all" operation)

All these operations are available for deselecting a nodes too. Pixels of user-selected regions can be shown in a different shade to easily recognise them on a screen.

In the reference article, the example used was a 20*256*256 grey-level image; there were 31679 primitive regions at scale $\sigma=1$.

For images with good contrast, the method can be used to successfully identify objects. Unfortunately, if the object-to-background contrast is poor, primitive regions at higher scale tend to overlap with some of the background, requiring tedious selections by the user to deselect these. The method needs a better blurring method that will not average across boundaries.

If the objects in an image are precisely located and represented by an abstract structure such as a tree or a graph, then the objects can be identified by matching their representations against an atlas of representations stored as a database of previously segmented objects. Object-specific measurements can be made, including deviation from normality.

5.9 Active Contour Models

Deformable models ([11], [17], [18], [19], [20], [21], [22], [23]) are also known as elastic models, and active contour models.

They are used to represent classes of objects which vary in shape from one instance to the other, or objects which change shape over time.
For example, everyone will have a brain shape different from the other, but some features of the brain are in common for all; thus, it is possible to store a representation of a brain object, such that a deformation of it will accurately represent any individual's brain.
Deformable models may be 3D surfaces, like a balloon which can be squeezed out of shape, 3D space curves, which we bend to form figures, or 2D contours (splines) which are used to draw profiles.

In segmentation in medical imaging, a very often used deformable model is the Snake (first proposed by Kass, Witkin and Terzopoulos [21]).
This is an energy minimising spline; from a given starting point it deforms itself to conform with the nearest salient contour. Snakes do not detect contours; the initial location must be provided either by other processing or by higher level knowledge.

The snake is an energy function, a weighted combination of internal and external forces. To obtain the best fit between the snake and the object, the energy is minimised.
Snakes require manual placement, but this is often done easily, by positioning "snaxels" (snake pixels) using a pointing device.

Specifically, a snake is defined as

$$E_{snake} = \int_0^1 E_{internal}\, v(s) + E_{image}\, v(s) + E_{constraint}\, v(s).ds$$

$E_{internal}$ is the internal spline energy caused by stretching and bending.

E_{image} is a measure of the attraction of image features such as contours.

$E_{constraint}$ is a measure of external constraints e.g. higher level understanding of the general shape or user-applied energy (from points through which the spline must pass).

v(s) = (x(s),y(s),z(s)) or (x(s),y(s)) is the parametric representation of the contour in 3D or 2D space.

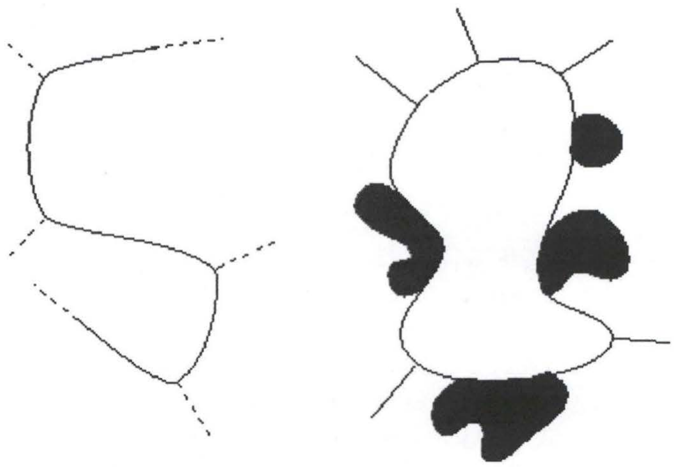
Figure 27: External and image forces which determine the snake

The internal energy provides a smoothness constraint. This can be further defined as

$$\mathrm{E}_{\mathrm{internal}} = \alpha(s)\left|\frac{dv}{ds}\right|^2 + \beta(s)\left|\frac{dv^2}{ds^2}\right|^2$$

α(s) is a measure of the elasticity of the snake.

β(s) is a measure of the stiffness of the snake.

The first order term makes the snake act like a membrane; the constant controls the tension along the spine (stretching a balloon or elastic band). The second order term makes the snake act like a thin plate; the constant controls the rigidity of the spine (bending a thin plate or wire). If β(s) = 0 then the function is discontinuous in its tangent, i.e. it may develop a corner at that point. If α(s) = β(s) = 0 then this also allows a break in the contour, a positional discontinuity.

The image energy is derived from the image data. Considering a two dimensional image, the snake may be attracted to lines, edges or terminations.

$$E_{image} = \omega_{line} E_{line} + \omega_{edge} E_{edge} + \omega_{term} E_{term}$$

where ω_i is an appropriate weighting function:

Commonly, the line energy E_{line} is defined simply by the image function $f(x, y)$; so if ω_{line} is a large positive, the spline is attracted to light lines (or areas) and if a large negative then it is attracted to dark lines (or areas).

If ω_{edge} is large, the spline is attracted to large image gradients.

If ω_{lterm} is large, the spline is attracted to line terminations or corners.

Definitions of internal energy:

- Smoothness
- Similarity to a predefined shape
- Balloons

Definitions of external energy:

- Edge gradient
- Intensity
- Force field

[19] presents interesting animations showing iterations in energy minimisation, in MPEG format (the images below are the start and end frames from one of these animations).

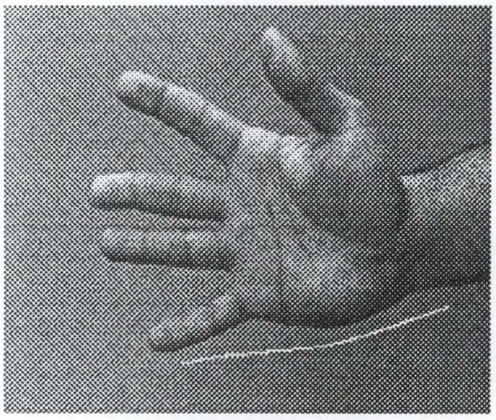
Figure 28: Initial Snake spline placement

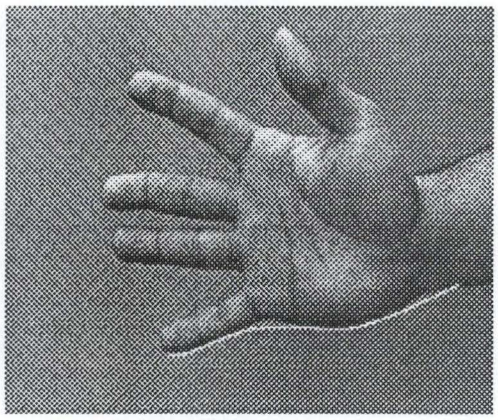
Figure 29: Snake with energy minimised

5.10 Statistical Analysis - Textures

Texture ([6], [7], [24]) in an image can be said to be a repetitive arrangement of a basic pattern over an area, though there is no mathematical definition for it.
Statistical analysis attempts to describe texture in terms of the size, shape, colour, and orientation of the elements of the pattern.

Sets of adjacent voxels sharing similar texture characteristics can then be grouped in to a region.

In medical imaging, if certain imaging modalities were known to highlight certain textures, one could look for a region with texture properties similar to it.

5.11 Segmentation Methods Summary

Methods	Advantages	Disadvantages	Output Quality
Threshold Segmentation	Fastest once thresholds known.	No region connectedness. Ignores neighbourhood information. Some methods for calculating thresholds complicated, not necessarily better.	Good if objects use different intensity spectres. May be problems if numerous transition voxels.
Binary Mathematical Morphology	Can shave off unwanted connected objects. Removes some background noise.		Successfully isolates brain tissue from cranium for 90% of MR images.
Region Growing	Uses neighbourhood information. Seeds found automatically, or easily chosen interactively by user.	Some adjacent regions must be merged. Voxel allocation not optimal (region prioritisation).	Bad if seeds badly chosen.
Region Growing and Merging	Uses neighbourhood information.	Voxel allocation not optimal (region prioritisation).	Bad if seeds badly chosen. Depends on merging rule.
Region Splitting	Can build tree representation.	Rectangular regions. Some adjacent regions must be merged.	
Split and Merge	Less and smoother regions than pure splitting.	Regions with square edges.	
Competitive Region Growing	No region prioritisation.	Higher memory requirements or slower than region growing.	Better than region growing.
Relaxation Algorithm	No region prioritisation. Not deterministic. Small noisy regions are removed.	Compatibility function requires knowledge about region.	
Watershed Segmentation	Considered powerful. Many suggested improvements.	Over-segmentation in pure watershed method.	
Ridge Flow Models	Tree representation.	Requires user interaction. Needs better blurring method.	
Active Contour Models		Requires user interaction (though simple). Complex algorithms.	
Statistical Analysis			

6. Edge Detection

Edge or boundary detection is a process which detects the pixels or voxels that separate two neighbouring areas with different statistical properties in an image. An edge is defined as a local variation of image intensity; its position is considered to be the centre of the range of pixels across which there is a change in intensity; it should be a thin line, one pixel wide. Any pixel with a gradient value above a certain threshold, can be considered to be an edge pixel ([7], [25]).

The image gradient is

$\partial f/\partial x, \partial f/\partial y$

The magnitude is

$$\sqrt{(\partial f/\partial x)^2 + (\partial f/\partial y)^2}$$

And its implementation can be used as an edge detector, where the magnitude of the gradient above a certain threshold can be said to be an edge.

If we approximate

$$\Delta x = f(x+1,y) - f(x-1,y)$$

and

$$\Delta y = f(x,y+1) - f(x,y-1)$$

then the gradient can be represented by

$$\sqrt{\Delta x^2 + \Delta y^2}$$

Having the gradients along the different axes allows the calculation of an estimate of edge direction as well as edge magnitude.

6.1 Mask or Templates operators

Many methods approximate a derivative operator by using a small discrete template or mask as a model of an edge. Template operators easily implemented (refer to annex on convolution of an image with a template).

For example, the above equations for calculating Δx and Δy can be implemented using the masks

Δx:

0	0	0
-1	0	1
0	0	0

Δy:

0	-1	0
0	0	0
0	1	0

Templates can also be used for line and point detection.

6.1.1 Sobel

The Sobel operator is a classic edge detector and is still used extensively in comparisons with newer algorithms.

The masks used for the Sobel edge detector are

Vertical edges:

-1	0	1
-2	0	2
-1	0	1

and

Horizontal edges:

1	2	1
0	0	0
-1	-2	-1

A disadvantage of the Sobel edge detector is that, like all gradient operators, it produces thick edges; the response to a step edge is present over either two or three pixels width, depending on the precise position of the step relative to the pixel grid.
This suggests the use of post-processing in the form of edge "thinning" and thresholding.

Also, the Sobel operator is susceptible to noise. Better quality in the presence of noise can be achieved by using larger masks at the expense of computational effort, and thicker edges.

The figures below show results of the Sobel implementation which was carried out in this project. The gradient thresholds were chosen interactively, refined from an initial guess.

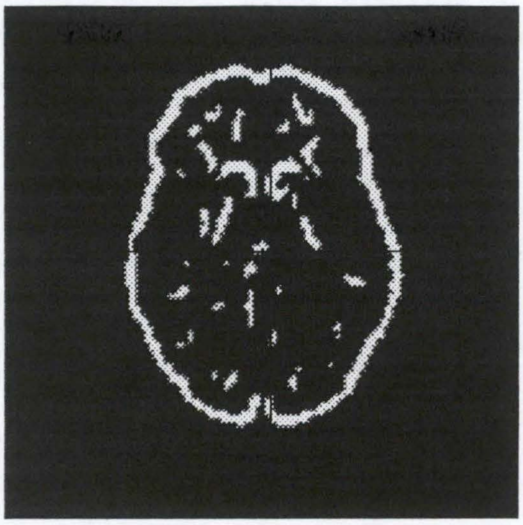
Figure 30: Sobel segmentation of PET image (gradient threshold 30000)

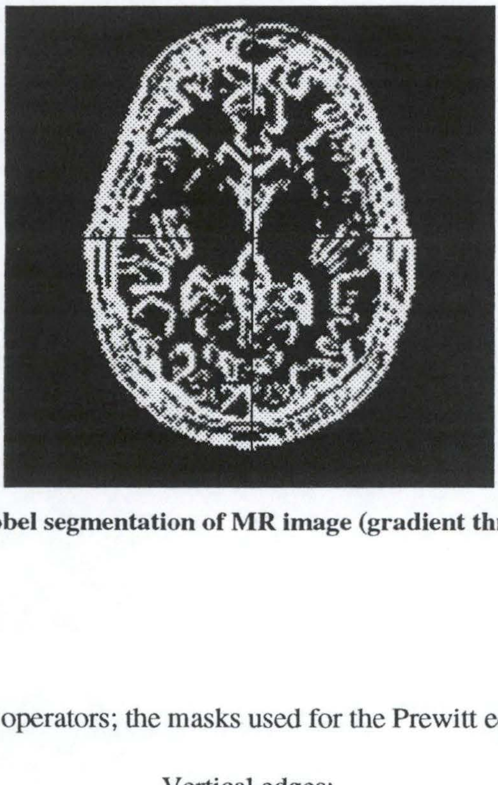
Figure 31: Sobel segmentation of MR image (gradient threshold 5500)

6.1.2 Prewitt

This is another set of template operators; the masks used for the Prewitt edge detector are

Vertical edges:

-1	0	1
-1	0	1
-1	0	1

Horizontal edges:

1	1	1
0	0	0
-1	-1	-1

6.1.3 Other Edge Templates

Many other types of edge templates exist, and can be used to detect edges along different directions. For a two-dimensional image, 3*3 edge template masks can detect edges only at four different directions (0, 45, 90, and 135 degrees). Templates of higher size can be more sensitive to edge orientation.

0 degrees

-1	0	1
-1	0	1
-1	0	1

90 degrees

1	1	1
0	0	0
-1	-1	-1

135 degrees

0	1	1
-1	0	1
-1	-1	0

and

45 degrees

1	1	0
1	0	-1
0	-1	-1

At each pixel the selected result is that of the template that produces the maximal output.
If the result of the convolution is close to zero, then there is no edge present at that pixel location. The direction of the edge cannot be reliably found if the result of the convolution at a pixel is similar for all masks.

6.1.4 Laplace operator

The first-order derivatives of the image function have local maxima or minima at edge locations due to large local intensity changes, and there the second-order derivatives have zero-crossings (please refer to annex on Laplacian).

Convolving the image with the Laplace operator ([26]) and detecting the zero-crossings will yield the edges.
Unfortunately, the Laplace operator on its own creates many false edges, because all variations in intensity are detected (including noise).

Although this approach to edge detection is straightforward, there is often need for post-processing to remove zero crossings corresponding to weak edges (where the intensity gradient is small) and other false edges (such as positive minima and negative maxima of the first derivative).

6.2 Marr-Hildreth

Marr and Hildreth were concerned with modelling the early stages of human visual perception, and designed their edge detection algorithm ([6], [25]) to consist of three steps:

- convolve the image with a Gaussian function (to smooth the image, and remove noise)
- complete the Laplacian of the convolved image
- edge pixels are those where there is a zero crossing.

The first two steps are associative, so the Laplacian of the Gaussian (LoG) can be computed (see annex on Laplacian).

Using this method, the detected edge sometimes differs from the actual edge, and the edges are not always thin; it is however better than the previous ones for images with much noise.

The figures below show the result of Marr-Hildreth edge detection from my implementation, on the test MR image.
The result of the Laplace convolution has been equalised to show detail, the original being too dark to distinguish any features.

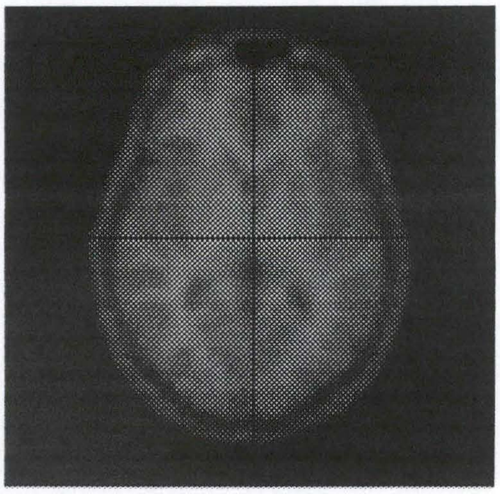

Figure 32: MR image, with Gaussian Blur (σ=1.4)

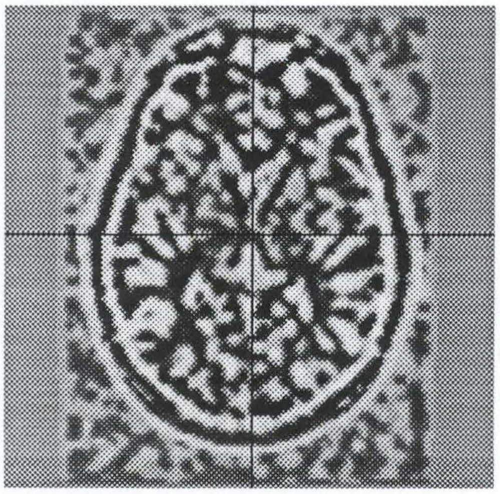

Figure 33: above image, after the Laplace convolution (equalised)

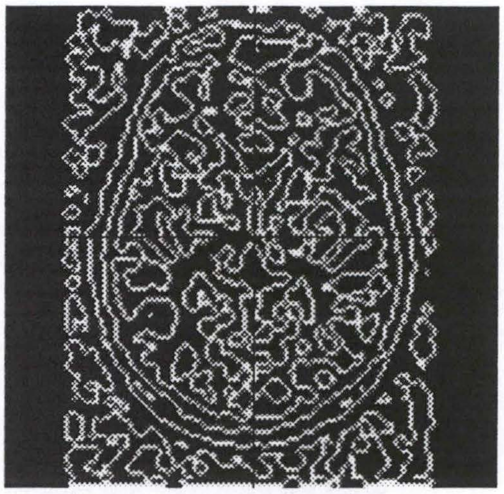

Figure 34: edges from Marr-Hildreth edge detection

6.3 Canny

The Canny edge detector ([6], [25], [27]) arises from the earlier work of Marr and Hildreth. In designing his edge detector, John Canny assumed that the image was corrupted by a Gaussian noise process. In practice this is not an exact model but it represents an approximation to the effects of sensor noise, sampling and quantisation.

John Canny specified three goals for an edge detector:

- good error rate: edges should not be found where there are none and all real edges should be detected
- good localisation: the distance between the detected edge and the actual edge should be minimum
- multiple-response: only a single edge should be detected where a single edge exists.

The Canny edge detection algorithm is composed of these stages:

- Image Smoothing: convolve the image with one-dimensional Gaussian masks, in each direction; this is to smooth the image, removing the assumed Gaussian noise. Two one-dimensional Gaussians are used because two-dimensional convolution with large Gaussians is computationally much more expensive.
- Differentiation: differentiate the smoothed image with respect to each direction. The resulting gradient magnitude can be calculated from Pythagoras' theorem, and the gradient angle can be found too.
- Non-maximum Suppression: edges must now be placed at the points of maximum intensity gradient, where there is a peak in the gradient function. However, in this case we wish to suppress non-maxima perpendicular to the edge direction, rather than parallel to the edge direction, since we expect continuity of edge strength along an extended contour.

  This is the non-maximum suppression step: each pixel in turn forms the centre of a nine pixel neighbourhood. By interpolation of the surrounding discrete grid values, the gradient magnitudes are calculated at the neighbourhood boundary in both directions perpendicular to the centre pixel, as shown in the figure below. If the pixel under consideration is not greater than these two values, it is suppressed.

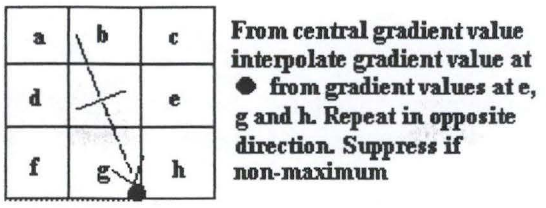
a
b
c
d
e
f
g
h
From central gradient value interpolate gradient value at ● from gradient values at e, g and h. Repeat in opposite direction. Suppress if non-maximum

Figure 35: non-maximum suppression

- Hysteresis Edge Thresholding: if a normal threshold limit was used to select edges, the edges would appear broken, because of fluctuation. The Canny edge detector uses hysteresis edge thresholding: an upper limit and lower limit are provided; any gradient value above the upper limit is accepted as an edge, and any gradient value below the lower limit is rejected; any gradient between the two limits that is adjacent to an accepted edge pixel is accepted as an edge pixel.

The first derivative of the image function convolved with a Gaussian, is equivalent to the image function convolved with the first derivative of a Gaussian. Therefore, it is possible to combine the smoothing and detection stages into a single convolution in one dimension, convolving with the first derivative of the Gaussian and looking for peaks.

This edge detector fares better than the detectors seen above, though results depend on the parameters used (but the best set of parameters for a particular image is not known).

6.4 Shen-Castan

Shen and Castan ([6]) also suggest a smoothing convolution, prior to the search for edges. Instead of applying a Gaussian smoothing, they suggest that the function which minimises

$$C_N^2 = \frac{4\int_0^\infty f^2(x)dx.\int_0^\infty f'^2(x)dx}{f^4(0)}$$ (in one dimension)

is the optimal smoothing filter for an edge detector.

They found that the optimal function to minimise C_N is the Infinite Symmetric Exponential Filter (ISEF):

$$f(x) = \frac{p}{2}e^{-p|x|}$$

In two dimensions, the ISEF is:

$$f(x,y) = ae^{-p(|x|+|y|)}$$

But, like in the Canny edge detector, these filters can be applied separately in the x and the y directions.

Then the image is convoluted with a Laplacian operator.

Before zero crossings are detected, there is a false zero-crossings suppression:
Zero-crossings where there are negative maximum gradients, and positive minimum gradients are suppressed (not considered an edge); in other words we will allow positive zero crossings (where the second derivative changes sign from positive to negative) to have a positive gradient, and negative zero crossings (where the second derivative changes sign from negative to positive) to have a negative gradient.

To deal with images that are very noisy, Shen and Castan suggest an adaptive gradient method:
For each proposed edge pixel not suppressed in the previous steps, a window with fixed width W is centred on it.
If this is indeed an edge pixel, then the window will contain two regions of differing grey level separated by an edge.
The best estimate of the gradient at that pixel should be the difference in level between the two regions, where one region corresponds to the zero pixels in the binary Laplacian image (BLI) and the other corresponds to the one-valued pixels.

Finally a hysteresis thresholding is applied to the edges found in the previous step.

6.5 Grey-Level Mathematical Morphology

The binary morphological operators seen above can be extended to greyscale images ([7], [28]).
For dilation, for each possible position of the structuring element, the results of the addition of the grey level of the source image and structuring element are compared. Then the greater of all the values is written to the output image.
For erosion, it is the minimum value which is withheld.

The changes which occur to an image from dilation and erosion are greatest near edges.
Therefore, the difference between eroded and original image gives large values at edges, and this can be used as an edge detector (the "difference" is simply a subtraction voxel-by-voxel between the images, with absolute values).

The figures below show the effects of greyscale morphology on an image well-known by image processing specialists. The erosion and dilation were done using a Java program in [28], and the difference between eroded and original image was done in Paint Shop Pro.

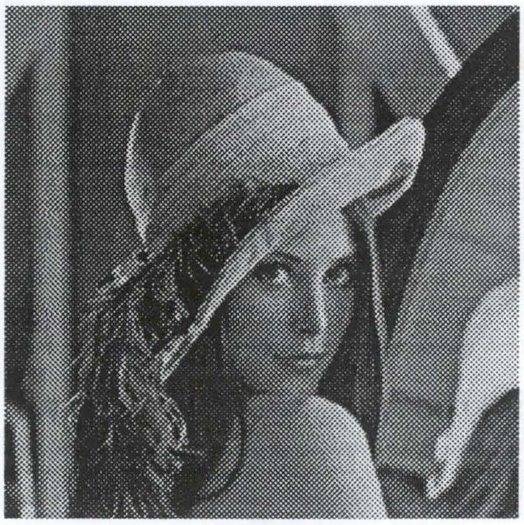

Figure 36: "Lenna"

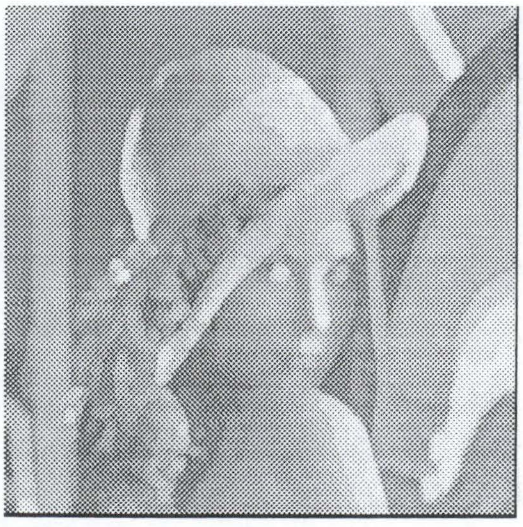

Figure 37: dilated Lenna (using structuring element below)

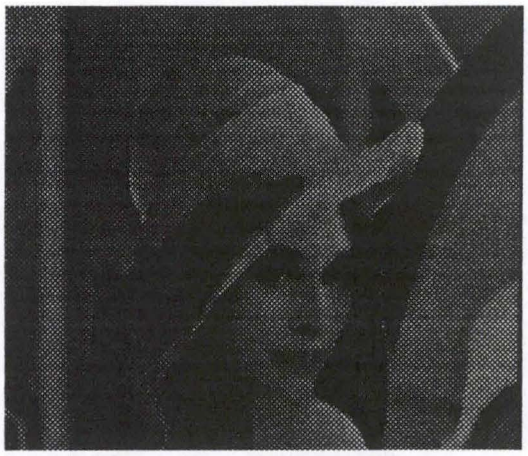
Figure 38: eroded Lenna (using structuring element below)

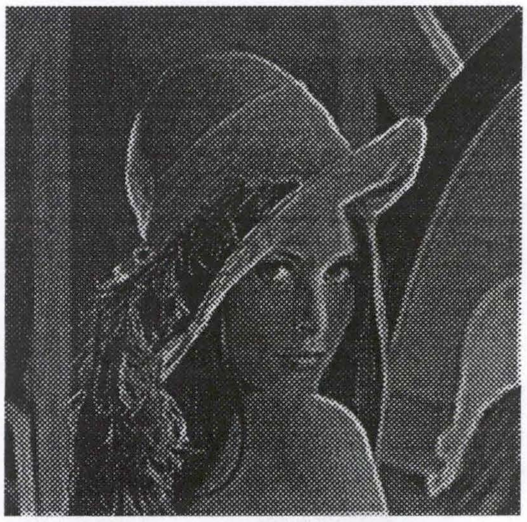
Figure 39: Difference between original and eroded image

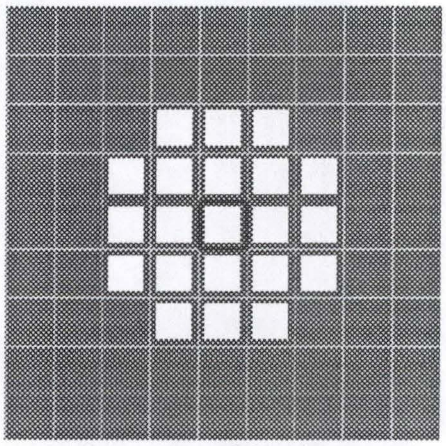
Figure 40: structuring element

6.6 Comparison of the edge detectors

According to Parker ([6]), Shen-Castan seems to have the advantage as noise becomes greater. Overall, the Shen-Castan (ISEF) edge detector is ranked first by a slight margin over Canny, which is second. Marr-Hildreth is third, followed by Kirsh (not seen in this work), and Sobel, in that order. The comparison between Canny and Shen-Castan does depend on the parameters selected in each case, and it is likely that evaluations can be found that use a better choice of parameters. The best set of parameters is not known, and so ultimately the user is left to judge the methods.

Article [1] presents a thorough methodology for the evaluation of edge detectors (although the methodology can be used to evaluate any results).
Its authors say that obtaining ground truth for real (i.e. not synthetic) images is a practical impossibility; "even the definition of an intensity edge is debatable".
Therefore "the traditional technique for comparing low-level vision algorithms is to present image results, side by side, and to let the reader subjectively judge the quality". The problem is that this method is very subjective.
They propose an evaluation methodology, where a team of human judges rate the outputs, and using statistical techniques from psychology, the rating statistics are compiled to rate the edge detectors, and the judges' coherence. The Analysis of Variance (ANOVA) technique was used to "separate the dependencies among the variables and to ascertain the statistical significance of observed differences". The methodology is very thorough, even statistically evaluating the consistency of the judges' ratings. It is slow though, as testing took place across many days; there were 8 judges and 8 original real images, for a total of 288 output images to rate. The judges' ratings were from 1 (incoherent edges) to 7 (all edges detected, no false edges).
They tested their methodology on the results from four edge detectors: the Sobel, the Canny, the Nalwa-Binford, and the Sarkar-Boyer edge detectors (the last two are not presented in this work).
One interesting aspect of their study is that they tested the edge detectors with many (six to twelve) parameter sets, in two scenarios:

- fixed parameter set ("current practice"), where for each detector the selected parameter set is the one best for all images.
- adapted parameter set ("ideal practice"), where for each detector and image the best parameter set is chosen.

It was found that the difference between fixed and adapted parameters is greatest for the Sarkar-Boyer, followed by the Sobel, then the Canny, and the least for the Nalwa-Binford; this means that with the Nalwa-Binford edge detector, choosing a fixed parameter set gives more consistent results over a range of images, though the quality may be lower for some images.

The overall ranking of the edge detectors was found to be, from overall best to overall worst: Canny, Nalwa-Binford, Sarkar-Boyer, and Sobel.
However, it was found that (except the Sobel) all edge detectors were best for at least one of the test images, and "no one single edge detector was best overall: for any given image it is difficult to predict which edge detector will be best". This suggests that given image statistics it may be possible to select the best method and the best set of parameters, but more research needs to be done to develop strategies to do this.

6.7 Edge Detectors Summary

Methods	Advantages	Disadvantages	Output Quality
Template operators	Fast. Easy to implement. Can detect directional edges. Can detect lines and points.	Thick edges. Susceptible to noise.	Now outdated.
Marr-Hildreth	Approximations for Laplacian of Gaussian make this method fast.	Edges sometimes differ from actual edges. Edges not always thin.	Better than previous for noisy images.
Canny	Can be speeded up with approximations and merging and derivation steps. A fixed parameter set should give consistent results over a range of images.	More parameters.	Generally better than previous.
Shen-Castan			Better as noise increases.
Grey-Level Mathematical morphology			Only obvious edges detected.
			Note: most edge detectors are best for some images.

7. Conclusions

This has been a most fascinating project, providing insights into the worlds of medical image processing and computer vision.

Medical image comparisons have often been made by specialists to discover anomalies, but it is hoped that coregistration will help these make a much better (and faster) diagnosis, which will highlight problems which would otherwise be invisible to the naked eye.

Segmentation and edge detection methods are still in full development. They are progressing both in terms of quality, and execution speed, but their evaluation is still a problem. Usually a new method is presented with a few results and comparisons, and sometimes the test images used are synthetic (i.e. not real-world images); no doubt in some cases the test images for the evaluation of a new method were chosen to the advantage of the method, without stating this restriction.
It is recognised that thorough evaluation methods such as that presented by M. Heath, S. Sarkar, T. Sanocki and K. Bowyer ([1]) should be systematically used; unfortunately these are slow and still prone to human subjective reasoning (for example, it was noticed by the aforementioned researchers that humans tend to rate thick detected edges better than thin edges, while not at all necessarily better in edge detection).

It is clear that the simpler methods of segmentation and edge detection are inadequate on their own for the purpose of medical image coregistration, but the more advanced ones could not be fully evaluated for such a purpose within the framework of this project.

The Canny and Shen-Castan edge detectors are now widely recognised powerful, and are often used in comparison with other methods. For automatic segmentation, of all methods covered here, competitive region growing, the relaxation algorithm, and the watershed methods are best. If the segmentation can be interactive, using deformable models such as Snakes can be a very good edge detector, but a good visualisation tool should not be overlooked.

8. Acknowledgements

I would like to thank my co-promoters, Professor Jean Fichefet and Professor Jean-Paul Leclercq, for giving me the opportunity to work on this project, and for continuing interest in its advancement.

A very big thanks goes to PhD student Tri Vu Khac for introducing me into the world of medical image coregistration, for timely advice and the source of much research material.

For reading and providing helpful comments regarding the writing of this document, I want to thank again my promoter Professor Jean-Paul Leclercq and also my father Yves de Rosée.

Finally, appreciation to all other students working on their own projects alongside me at the university during the entire summer.

9. References

[1] M. Heath, S. Sarkar, T. Sanocki, K. Bowyer. Comparison of Edge Detectors: A Methodology and Initial Study.
[2] R.P. Woods, J.C. Mazziotta, S.R. Cherry. MRI-PET Registration with Automated Algorithm.
[3] P. A. Van den Elsen, E.-J. D. Pol, M. A. Viergever. Medical Image Matching - A Review With Classification.
[4] C. R. Maurer Jr., J. M. Fitzpatrick. A Review of Medical Image Registration.
[5] E. R. Dougherty. Digital Image Processing Methods.
[6] J. R. Parker. Algorithms for Image Processing and Computer Vision.
[7] I. Pitas. Digital Image Processing Algorithms.
[8] Ridler, Calvard. Picture Thresholding Using an Iterative Selection Method.
[9] Program: thresh.c. http://www.ittc.ukans.edu/~jgauch/kuim/src/thresh.c
[10] T. Vu Khac. Image Coregistration C++ Library.
[11] T. Kapur, W.E. Grimson, W.M. Wells III, R. Kikinis. Segmentation of Brain Tissue from Magnetic Resonance Images.
[12] J. Alakuijala, J. Laitinen, S. Sallinen; H. Helminen. New Efficient Image Segmentation Algorithm: Competitive Region Growing of Initial Regions. http://bme01.engr.latech.edu/cdrom/texts/337.html
[13] S. Peleg, A. Rosenfeld. Determining compatibility coefficients for curve enhancement relaxation processes.
[14] Image Segmentation and Mathematical Morphology. http://cmm.ensmp.fr/~beucher/wtshed.html
[15] L. Vincent, P. Soille. Watersheds in Digital Spaces: An Efficient Algorithm based on Immersion Simulations.
[16] D. Eberly and S. M. Pizer. Ridge Flow Models for image segmentation.
[17] Active or Flexible Contour Models.
http://www.icblhw.ac.uk/marble/vision/medium/snakes/snakes.htm
[18] G. Székely, A. Kelemen, C. Brechbühler, G. Gerig. Segmentation of 2-D and 3-D objects from MRI volume data using constrained elastic deformations of flexible Fourier contour and surface models.
[19] Snakes, Active Contours, and Deformable Models.
http://www.wpi.edu/~dima/ummed/presentation/index.html
[20] Active Contour Models. http://www.dcs.qmw.ac.uk/research/vision/msc-projects/active_contours/active_contours.html
[21] M. Kass, A. Witkin, D. Terzopoulos. "Snakes: Active Contour Models". First International Conference on Computer Vision, pp 259-268, 1987.
[22] Introduction to Active Contours. http://www.c3.lanl.gov/~jason/snakes.html
[23] Active Contour Models (Snakes).
http://www.cogs.susx.ac.uk/users/davidy/teachvision/vision7.html
[24] Z. Liang. Tissue Classification and Segmentation of MR Images.
[25] T. Vu Khac. Review and Comparison of Image Coregistration Methods.
[26] Laplacian Edge Detection.
http://www.owlnet.rice.edu/~elec539/Projects97/morphjrks/laplacian.html
[27] Edges: The Canny Edge Detector. http://www.icblhw.ac.uk/marble/vision/low/edges/canny.htm
[28] Introduction to Morphology. http://cobb.ee.psu.edu/users/greg/morphology.html
[29] Gaussian Smoothing. http://q127-3.coventry.ac.uk/~hipr/gsmooth.html
[30] B. Jähne. Digital Image Processing: Concepts, Algorithms, and Scientific Applications.
[31] Laplacian/Laplacian of Gaussian. http://q127-3.coventry.ac.uk/~hipr/log.html

[32] L. G. Brown. A Survey of Image Registration Techniques.
[33] Edges: Gradient Edge Detection. http://www.icblhw.ac.uk/marble/vision/low/edges/gradient.htm
[34] Canny Edge Detector. http://q127-3.coventry.ac.uk/~hipr/canny.html
[35] The KUIM Image Processing System. http://www.ittc.ukans.edu/~jgauch/kuim/kuim.html

[36] F. Fazio, G. Rizzo, M. C. Gilardi, G. Lucignani, A. Savi. Miltimodality Imaging: A Synergistic Approach.
[37] M. E. Noz, G. Q. Maguire Jr. Multimodality Image Correlation and Fusion.
[38] J. West, J. M. Fitzpatrick, M. Y. Wang B. M. Dawant, et al. Comparison and Evaluation of Retrospective Intermodality Image Registration Techniques.
[39] R. P. Woods, J. C. Mazziota, S. R. Cherry. Automated Image Registration.

10. Annexes

10.1 Mean, Standard Deviation, Variance

By definition mean can be considered as the one number in the set of numbers that best describes all the numbers in that set.

$$\overline{X} = \frac{\sum f(X)}{\sum f}$$

Variance is a statistical measure of variation or dispersion or scatter of set of values from their mean.

$$\text{Variance} = \frac{\sum x^2}{N}$$

Where:

x is the deviation of the value from the mean i.e. $X - \overline{X}$.

N is the total number of values.

Standard deviation is very closely related to the variance, it is just the square root of the variance. Standard deviation represented by σ.

$$\sigma = \sqrt{Variance}$$

10.2 Histograms

The histogram ([7]) is a very often used statistical analysis of an image: it is a graph showing the usage of each intensity of an image. The x-axis lists the intensity values, the y-axis shows the number of pixels (sometimes the percentage) which are at the corresponding intensity.
The image histogram carries important information about the image content. If its pixel values are concentrated in the low image intensities, the image is 'dark'. A 'bright' image has a histogram that is concentrated in the high image intensities.

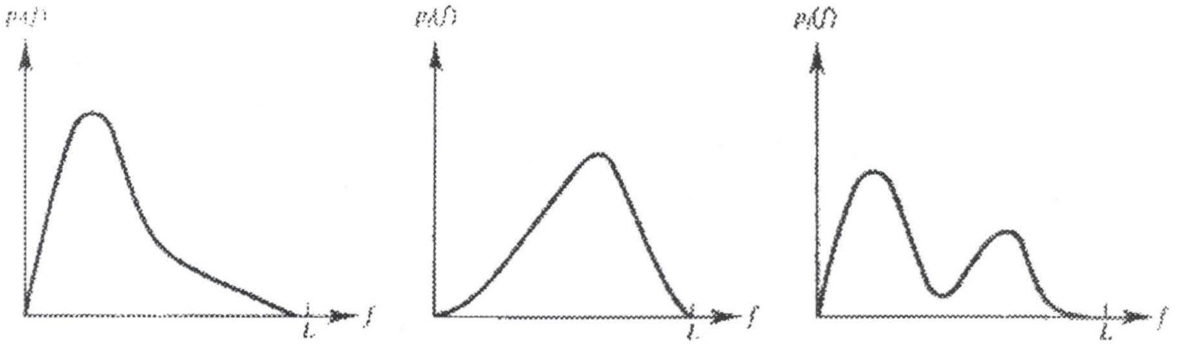

Figure 41: sample histograms

If the image histogram is concentrated on a small intensity region, the image contrast is poor and the subjective image quality is low.
In this case, image quality can be enhanced by a technique called histogram equalisation:
Build a new histogram where the value for each colour is the sum of the (percentage) values in the original histogram of all colours of smaller intensity, including itself. Thus, the new histogram value at the highest intensity will be 1. Then multiply all the new histogram values by the maximum possible intensity (given the number of bits per pixel). The result is a table which gives a new replacement intensity for each pixel in the image.
When the original image has a poor contrast, the equalised image has a much better one.

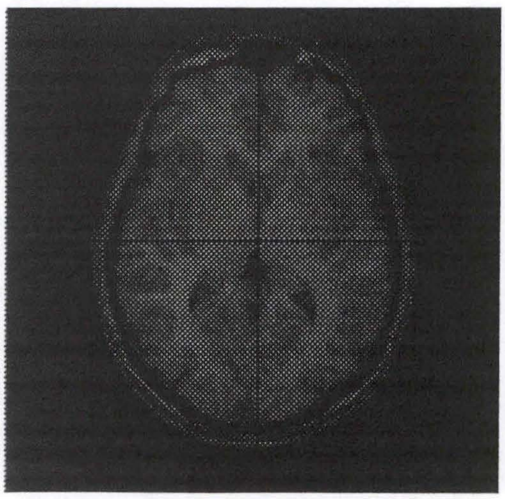

Figure 42: Original MR image

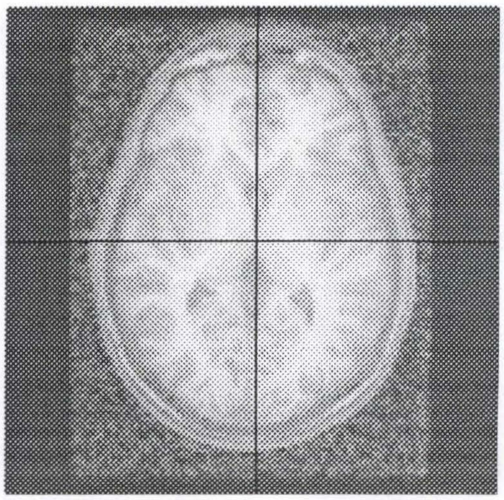

Figure 43: Equalised MR image

10.3 Zero Crossings

There is a zero crossing at a pixel when the two opposing neighbours in one direction have different signs.

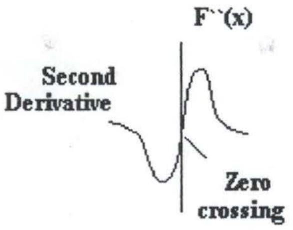

Figure 44: Zero Crossing

10.4 Convolution

A convolution is the application of a mask, or template to an image.
Applying a mask on a pixel (x,y) is easy:

for the mask

a	b	c
d	e	f
g	h	i

and an image function *f(x,y)*

the result at the pixel (x,y) will be

$$\left|\begin{array}{lll} a * f(x-1,y-1) & + b * f(x,y-1) & + c * f(x+1,y-1) \\ + d * f(x-1,y) & + e * f(x,y) & + f * f(x+1,y) \\ + g * f(x-1,y+1) & + h * f(x,y+1) & + i * f(x+1,y+1) \end{array}\right|$$

Masks are not necessarily of size 3*3.
The mask presented above is for two-dimensional images; in a three-dimensional image the mask would be cube-shaped.

10.5 Gaussian Blur

The Gaussian smoothing operator ([6], [27], [29]) is a 2-D convolution operator that is used to blur images and remove detail and noise. The degree of smoothing is determined by the standard deviation of the Gaussian.

The Gaussian gives more weight to each pixels' nearest neighbourhood, which provides gentler smoothing and preserves edges better than a similarly sized mean filter with a uniformly weighted average.

A Gaussian has the form

$$G(x) = e^{-\left(\frac{x^2}{2\sigma^2}\right)}$$

which looks like this:

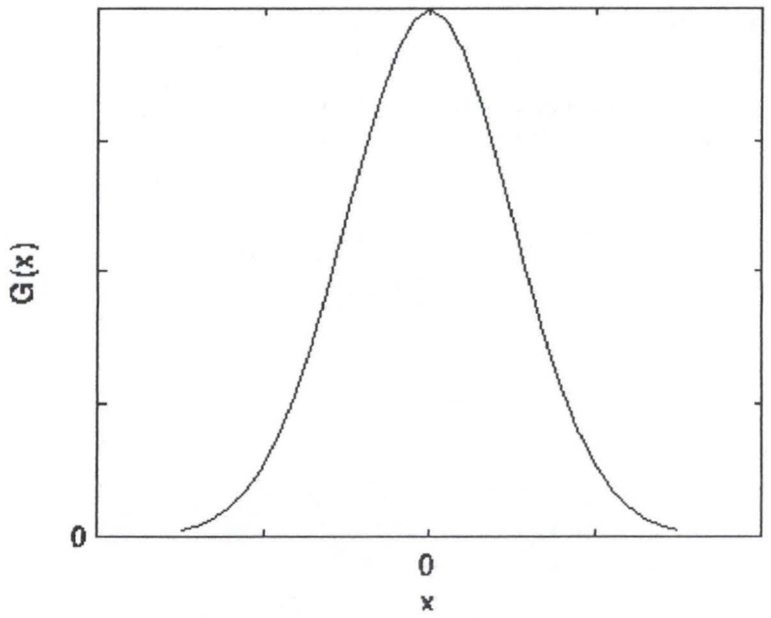

Figure 45: Graph of Gaussian

The derivative (used in the Canny edge detector) is therefore:

$$G'(x) = \left(-\frac{x}{\sigma^2}\right)e^{-\frac{x^2}{2\sigma^2}}$$

(clearly, the derivative is 0 at $x = 0$ and $x \to \infty$)

To show the smoothing effect, the following figure shows a step edge convolved with a Gaussian function:

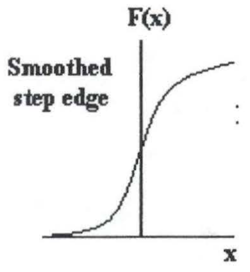

Figure 46: Gaussian-smoothed step edge

In two dimensions, a Gaussian is given by:

$$G(x,y) = e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}$$ and G has derivatives in both the x and y directions.

To perform Gaussian smoothing we would like to produce a convolution mask to apply to the image. Since the image is stored as a collection of discrete pixels we need to produce a discrete approximation to the Gaussian function before we can perform the convolution. In theory, the Gaussian distribution is non-zero everywhere, which would require an infinitely large convolution mask, but in practice it is effectively zero more than about three standard deviations from the mean, and so we can truncate the mask at this point.

The figure below shows a convolution mask that approximates a Gaussian with a σ of 1.4. The values in the mask are scaled so that the smoothing does not brighten or darken the image.

$$\frac{1}{115}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Figure 47: convolution mask of Gaussian (σ 1.4)

(Larger standard deviation Gaussians, of course, require larger convolution masks in order to be accurately represented.)

To reduce computation, the 2-D convolution can be performed by first convolving with a 1-D Gaussian in the x direction, and then convolving with another 1-D Gaussian in the y direction.

10.6 Laplacian Operator

If we want to detect edges, a filter operation is necessary which emphasises the changes in grey values and suppresses areas with constant grey values. Derivative operators are suitable for such an operation.

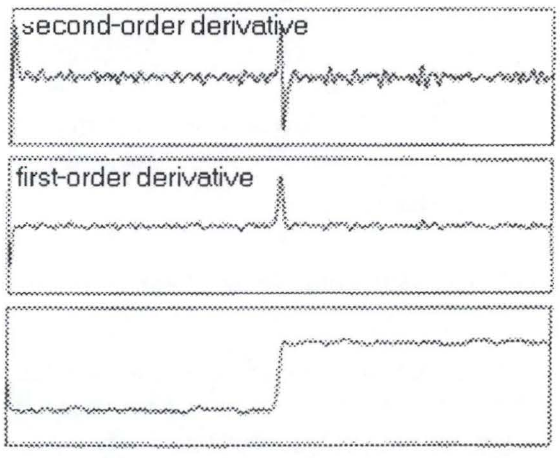

Figure 48: Graph of image intensity function, with first and second derivatives

The first derivative shows an extremum at the edge, while the second derivative crosses zero (this is known as a zero-crossing) where the edge has its steepest ascent. Both criteria can be used to detect edges.

The sum of the two second partial derivatives is called the Laplace operator ([6], [30], [31]) and is denoted by ∇.

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

An estimator of the Laplace operator is given by:

$$\nabla 2f(x,y) \approx f(x,y) - [\ f(x,y+1) + f(x,y-1) + f(x+1,y) + f(x-1,y)\ ]/4$$

which is equivalent to applying the mask

0	1	0
1	-4	1
0	1	0

A cubic mask can also be applied for a three-dimensional image, with a middle value of -6 instead of -4.

An approximation to the Laplacian can be obtained quickly by simply subtracting the original image from the smoother image. If the filtered image is S and the original is I, the resulting image B = S - I is the band-limited Laplacian of the image.

The binary Laplacian image BLI is obtained by setting all of the positive valued pixels in B to 1 and others to 0. Zero crossings are the boundaries if the regions in BLI.

Because these masks are approximating a second derivative measurement on the image, they are very sensitive to noise.

Therefore, the Laplacian is often applied to an image that has first been smoothed with a Gaussian smoothing filter in order to reduce its sensitivity to noise.
Fortunately, the convolutions are associative, so the Gaussian and Laplacian masks can be convolved together first, to produce a new mask: this is called the LoG (Laplacian of Gaussian) mask can be calculated in advance so only one convolution needs to be performed on the image. For example, this is used in the Marr-Hildreth algorithm.

A discrete mask that approximates the LoG (for a Gaussian σ of 1.4) is shown in the figure below:

0	0	3	2	2	2	3	0	0
0	2	3	5	5	5	3	2	0
3	3	5	3	0	3	5	3	3
2	5	3	-12	-23	-12	3	5	2
2	5	0	-23	-40	-23	0	5	2
2	5	3	-12	-23	-12	3	5	2
3	3	5	3	0	3	5	3	3
0	2	3	5	5	5	3	2	0
0	0	3	2	2	2	3	0	0

Figure 49: Laplacian of Gaussian (σ 1.4)

10.7 Program Functions

These are C functions, in this work implemented for the SUN Solaris 2.5.1 workstation. It should be portable without too much difficulty, the problem being the difference in bytesize of basic datatypes in different systems.

The input images must be in AIR format. Outputs will be saved in that format too.
All functions work for 65536 grey-level images (16 bit SIGNED-SHORT per byte), some for 256 grey-level images.
All functions will return error messages if used with voxel datatypes it does not recognise.

Typedef.h contains definitions and types required for these functions.

10.7.1 General functions:

char *str_concat(char *left, char *right, char *result)
String concatenation function. String result is concatenation of string left and string right.

int Image_Open(air_img_ptr img_ptr, char *filename)
Function to load an AIR-format image from disk, to an air_img_ptr structure
note: filename is without extension

int Image_Save(air_img_ptr img_ptr, char *filename)
Function to save an image to disk in AIR-format, from an air_img_ptr structure
note: filename is without extension

void print_image_info(air_img_ptr img)
Prints the image header information to screen.

air_img_ptr Convert_Image_UCHAR(air_img_ptr img_in_ptr)
Function to convert an AIR image to DT_UNSIGNED_CHAR datatype (256 grey levels).

void Free_Image(air_img_ptr img_ptr)
Release memory allocated to img_ptr.

air_img_ptr Image_Absolute(air_img_ptr img_in_ptr)
Transforms all negative voxel values in image to positive.

air_img_ptr Image_Difference(air_img_ptr img_1_ptr, air_img_ptr img_2_ptr)
Function returns image 1 minus image2.

10.7.2 Segmentation Functions:

long Threshold_Percentile(hist_ptr h_ptr, float percentile)
Function will calculate the threshold at which a voxel is greater than percentile of the other voxels.
Can be used for the Weszka thresholding method.

```
air_img_ptr Image_Keep_Regions(air_img_ptr img_orig_ptr,
                                air_img_ptr img_regions_ptr,
```

int numregionswtd,
int *regionswtd)

Function returns an image keeping only the regions in img_orig_ptr wanted; other voxels are set to 0.
numregionswtd is the number of regions to be kept;
regionswtd is an array with region numbers to be kept.

air_img_ptr SegmentThreshold(air_img_ptr img_in_ptr, int numthresholds, int *thresholds)
Function to perform threshold segmentation from given array of threshold values.
The thresholds must be sorted by increasing values.
The function returns an image where the voxel values are labels of region to which they belong.

10.7.3 Edge Detection Functions:

air_img_ptr Sobel(air_img_ptr img_in_ptr, long threshold)
Function to perform Sobel edge detection on the input image, using the 2-D mask seen in the text above.
Any gradient pixel above threshold is an edge.

10.7.4 Histogram Functions:

hist_ptr Build_Histogram(air_img_ptr img_in_ptr)
Function to build a histogram from grey-level image.

air_img_ptr Histogram_Equalization(air_img_ptr img_in_ptr)
Function performs histogram equalisation as seen above.

void Twopeaks(hist_ptr h_ptr, int *numthresh_ptr, int **thresholds_ptr)
Function to calculate a threshold from the histogram of a grey-level image.
Using approximation seen above.
Only one threshold will be calculated, so:
*numthresh_ptr will be 1
*thresholds_ptr will be an array of one integer.

hist_ptr Build_Histogram_Lap(air_img_ptr img_in_ptr,
air_img_ptr img_lap_ptr,
long threshold)

Like Build_Histogram, builds histogram of img_in_ptr but puts in histogram only those voxels which are larger than threshold in the img_lap_ptr image.

10.7.5 Gaussian, Laplacian, Zero-Crossings

air_img_ptr Gaussian_Blur(air_img_ptr img_in_ptr)
Function returns an image which is the Gaussian blurring of input image. The convolution mask used is the one seen above for σ=1.4.

air_img_ptr Image_Laplacian(air_img_ptr img_in_ptr)
Approximation of the laplacian by convolving img_in_ptr image with 3-D mask.

air_img_ptr Find_Zero_Crossings(air_img_ptr img_2nd_deriv_ptr)
Function to find zero-crossings in img_2nd_deriv_ptr.