NTNU
Department of Computer and Information Science

HELSE BERGEN
Haukeland Universitetssykehus
Department of Clinical Engineering

SEGMENTATION OF KIDNEYS FROM MR-IMAGES

Eirik Roald Ree
Bergen 16. June 2005

PREFACE

This is the final thesis of my education at the Norwegian University of Science and Technology (NTNU). The education has lasted 5 years and results in a Master of Science degree.

This work has been made from January 2005 until June 2005 in cooperation with the Department of Clinical Engineering at Haukeland University Hospital in Bergen, Norway.

I would like to thank Erling Andersen and Stig Frode Samnøy at Haukeland University Hospital for an interesting and challenging assignment, a desk in their office, and guidance during the project.

I would also like to thank Richard Blake at the Norwegian University of Science and Technology for his guidance.

Finally, I would like to thank Berit Hellan and Bård Kjos at the Norwegian University of Science and Technology for their assistance when doing the project away from the university.

ABSTRACT

Haukeland University Hospital are currently running a project for developing renal examinations using MR imaging. For many of the tasks they wish to do, i.e. visualization and volume estimation, a good segmentation of the kidneys is absolutely necessary. Most methods available today are time-consuming and labor intensive.

In this paper, we have studied several possible methods for automatic or semi-automatic segmentation of the kidneys. A method has been implemented by combining watershed segmentation with active contours.

The watershed algorithm is a watershed-from-markers using an Image Foresting Transform as described in [20][21]. This is simple to initialize by just setting a few marker points inside and outside of the desired region. We have simplified this initialization further, so the user only has to draw a rectangle around the kidney.

The results from the watershed algorithm are used for initializing the active contours. Because watershed may result in several regions in each image, while the desired result from the active contours should be a single region, the contour is initialized as a rectangle that bounds the watershed result. This contour then shrinks until it is a smooth outline that contains the result from the watershed.

The implemented algorithm has been tested with real MR images, and the results show that the method gives a good solution to the problem.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

The use of image technology plays a vital role in modern medicine. Methods such as magnetic resonance imaging (MRI) allow for the acquisition of 3-dimensional images of the body, from which a radiologist can inspect the interior anatomy and function without surgery. Medical imaging is used in all steps of a medical treatment [1]:

- For diagnosis; medical images can be used to see cancer tumors, misaligned joints, blood clots, internal bleeding, damaged tissue and a number of other factors that can assist in making a faster and better diagnosis.
- For treatment; medical images can be used to guide minimum-invasive methods such as peephole surgery where the operation is performed using only a thin tube that penetrates the body. This means that it is not necessary to open the patient with a scalpel, which greatly reduces the risk of the operation and makes recovery faster and less troublesome.
- For evaluation of a treatment; medical images can be used for monitoring the effect of the treatment. It is desirable to discover as early as possible if a treatment is working or not, so that it can be changed if necessary to improve the effect.

At Haukeland University Hospital a project has been started where they want to measure kidney function using MR imaging. This can be used for diagnosis, planning of treatment and monitoring of kidney disease as well as other diseases that influence kidney function [2].

The responsibility of the kidneys is to filter our blood. Excess water, salt and waste materials is extracted through the urine too keep the amount of fluid and salt in the body, as well as the blood pressure, at a constant level. At the same time, all plasma proteins should remain in the blood.

A general sign of kidney disease is that the filtration is reduced, and that proteins leak out into the urine. This can be measured to some extent by evaluation of urine and blood samples, but the sensitivity is low and it is impossible to separate the two kidneys. There are some methods today that measure kidney function using medical imaging, but these require radioactive isotopes to be injected in the blood and/or radiation in the form of x-rays, which is dangerous to the patient, especially with repeated examinations.

1.1 MR-IMAGING

Magnetic Resonance Imaging was developed in the 1970s, and was put into active work in 1984. Compared to all other imaging techniques, such as x-ray, computed tomography, nuclear medicine and ultrasound, it provides superior anatomical images, especially of soft tissue. Unlike most of the other methods, it is also believed not to be harmful to the patient [3].

Figure 1-1: The 3T MR machine at Haukeland University Hospital

The human body is made of billions of atoms. The nuclei of these spin on it's axis. To create an MR image, the patient is placed inside an extremely powerful magnetic field which forces most of the atoms to line up with it's direction. When a pulse of a certain radio frequency is sent through the patient, it makes some of the atoms absorb the energy and turn away from this alignment. When the pulse is finished the atoms fall back into alignment, and their energy is echoed out of the body as radio waves and measured. This data can be used to generate an image of the internal anatomy of the patient.

The kinds of atoms that are influenced by the radio pulse, and thus what is shown on the image, is determined by the frequency. Hydrogen is most commonly used, but by changing the frequency it is possible to view most atoms. The reason why hydrogen is used is that it is a major component in both water and fat, which are both common in human tissue and provide useful information. In fact, 63% of the body consists of hydrogen atoms [4]. The fact that hydrogen atoms have a single proton in it's nucleus also makes them align better with the magnetic field.

The big advantage of MR imaging compared to other methods, besides the fact that it is not harmful to the patient, is that it can give very good contrast even between different tissues of similar density. It is very easy to see blood vessels, which with other kinds of imaging are only visible if a contrast media is injected in the blood stream. The resolution is also very good, and unlike CT, which can only create images in the transverse (horizontal) plane, it can create images as slices in any direction
.

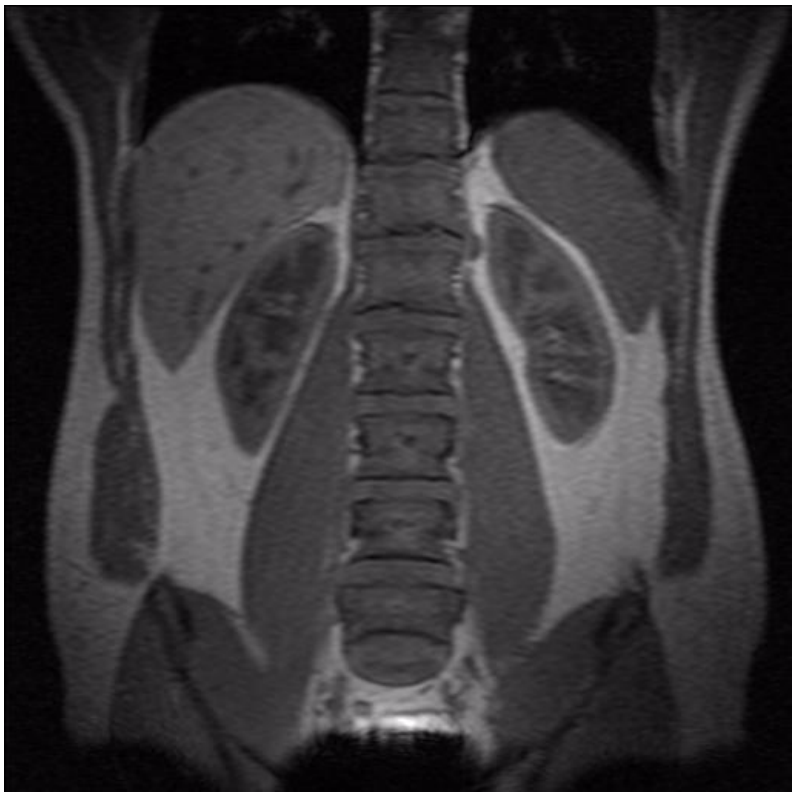


Figure 1-2: MR image

Figure 1-2 shows an anatomical MR image of the abdomen. The spatial resolution of this image is 0.74mm. It is easy to see the liver in the upper left-hand corner. On the other side of the spline, the spleen is visible, and the two kidneys are positioned below these. The contrast is quite good, and it is easy for the human eye to separate different tissue.

Most MR machines that are used today have a magnetic field of 1.5 Tesla, but at Haukeland University Hospital they just installed a new 3 Tesla machine, which is shown in Figure 1-1. This stronger magnet can give better resolution and signal-to-noise ratio, and improve the image in other ways.

By using different kinds of contrast fluids that are either swallowed or injected into the blood stream, it is possible to visualize the function of some organs. For example, it can be seen how the contrast follows the blood to the kidneys, and how much of it is filtered out into the urine.

1.2 PROBLEM DESCRIPTION

The task of this project is to segment the kidneys from MR volumes. This should be used to estimate volumes, and to guide image registration of different volumes and segmentation of different tissues inside the kidney, such as the cortex, medulla and renal pelvis. This can again be used to evaluate the function of the kidney by measuring filtration of different contrast fluids.

Different segmentation methods will be evaluated, and the best method implemented and tested. The resulting algorithm should be semi-automatic and require as little user interaction as possible.

The implementation should be done using Java and Java Advanced Imaging, and independent of a certain GUI, because it will be used as part of a larger workstation that will be developed at Haukeland University Hospital, using these technologies.

1.3 ORGANIZATION OF THIS DOCUMENT

Chapter 1 gives a brief introduction to the problem, and describes the relevance of the research.
Chapter 2 gives a description of some existing segmentation algorithms.
Chapter 3 evaluates the segmentation algorithms according to the problem at hand, and the algorithm for implementation is selected.
Chapter 4 describes the implementation in detail.
Chapter 5 describes and evaluates the results of the implemented algorithm for different problems.
Chapter 6 draws conclusions about the methods, and provides ideas for future work.
Appendix A provides background information on the anatomy and function of the kidneys.

CHAPTER 2: PREVIOUS WORK

Several algorithms exist for the segmentation of images in 2D as well as 3D. This chapter will give a description of algorithms that may be relevant for the 3-dimensional segmentation of kidneys.

2.1 MANUAL DRAWING

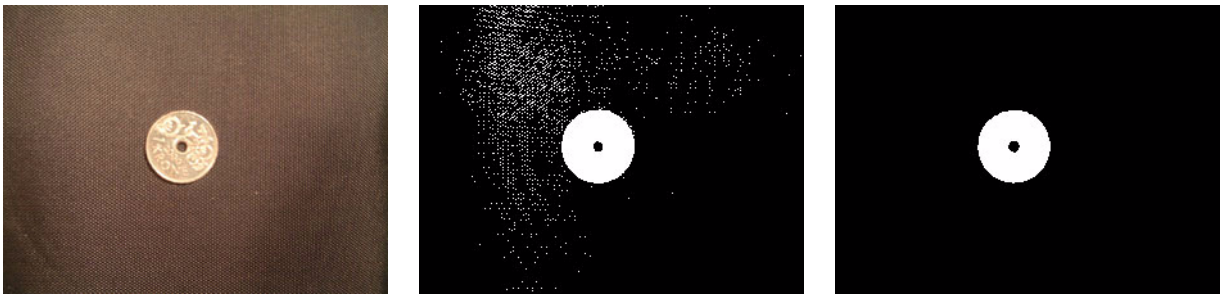
The method that is most commonly used for segmentation of medical images today is manual drawing. An expert user, most commonly a radiologist, draws the contour around the desired structure in each of the image slices. As of today, this method gives accurate results, and it is the "gold standard" that automatic segmentation methods are compared against.

The biggest problem with manual segmentation is that it is very time consuming. Manual segmentation of the kidneys from an MRI volume requires approximately 2-3 hours [5]. The result will also generally be different every time because the user is not able to accurately copy his or her own work.

If there is little need for accuracy, it is possible to simplify the manual drawing by using rectangles, ellipses or other simple shapes. This is very fast, but the results are so inaccurate that it is of little use except as a pre-processing step for placement of a deformable model, or to reduce the data size for automatic algorithms.

2.2 THRESHOLDING

Thresholding is the simplest automatic segmentation algorithm, and it is extremely fast. A threshold value T_{min} is set, and every pixel (or voxel) with a gray level above this value are considered part of the region of interest (ROI). If the object has a medium gray level, with background pixels being both darker and brighter, a maximum threshold T_{max} can be added so that only pixels with a gray level between T_{min} and T_{max} are included in the ROI. This has little effect on the speed of the algorithm, but provides a lot of extra flexibility.



a) Coin on dark background

b) Thresholding with T_{min} too low

c) Good thresholding

Figure 2-1: Segmentation by simple thresholding

Figure 2-1 shows the result of thresholding of a coin on a dark background using two different values for T_{min}. In b), the threshold is set too low, causing part of the background to be included in the ROI, but c) provides a perfect result with a correctly set threshold value.

One of the problems with thresholding is that of setting the correct threshold values. Except from in extremely controlled environments, the threshold will vary depending on the lighting or other factors that influence the image acquisition. Setting the values manually can be quite labor intensive, thus removing much of the advantage of the algorithm. Several methods are developed for setting the threshold values automatically. Most of these work by analyzing the histogram of the gray scale values of the image [6].

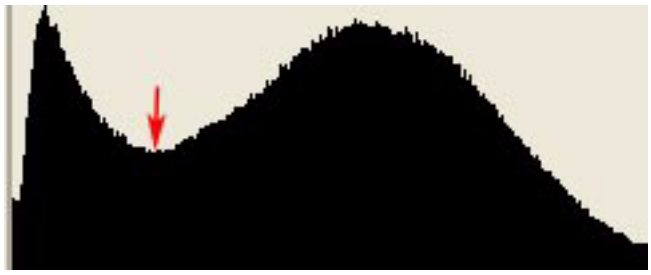


Figure 2-2: Bi-modal histogram with suggested threshold

In some cases, i.e. when segmenting text from a sheet of paper, it can be known that the object covers approximately a certain percentage of the image. It is then simple to split the histogram according to this percentage to find the best threshold value.

Other methods analyze the shape of the histogram. If the desired object has an approximate gray-level, and the background has another, the histogram will be bi-modal, meaning that it has two peaks. It then makes intuitive sense to set the threshold at the gray-level that has the minimum histogram value between these two peaks. This is illustrated in Figure 2-2. If the histogram is multi-modal (with more than two peaks), a threshold value can be set between each of the peaks to separate several objects.

With optimal thresholding, it is approximated that the histogram is created by a weighted sum of two or more probability densities with normal distribution. The threshold is set to get the minimum error segmentation according to this approximation. This will usually give a threshold that is close to, but not the same as, the conventional histogram analysis. The optimal threshold value can be calculated iteratively in less than ten iterations as described in [6].

In many cases it is impossible to get a good segmentation using the same threshold level on the entire image. This happens because of uneven illumination and other imaging factors. Adaptive thresholding tries to correct this by dividing the image into several smaller parts. A separate threshold value is then selected for each part of the image [7].

In most real cases, it is very difficult to get a good segmentation using thresholding. Usually, the segmented region will have holes in it, or areas outside of the desired structure will be incorrectly labelled. With multi-spectral images, results are usually better because it is less likely that other structures will satisfy the threshold levels for all the bands.

2.3 EDGE-BASED SEGMENTATION

Edge-based segmentation algorithms attempt to locate the desired structures in images by locating and linking discontinuities in gray level or color that represent the borders around the structures.

2.3.1 Edge Detection

Several methods exist for detecting edges in the image, and thus borders around the desired structures.

Convolution With Edge Detecting Operators

As edges are defined by changes in the gray levels of the image, they can be found by locating pixels where the gradient (the first derivative) of the image gray levels is large. Several convolution masks have been created for this purpose, but the most common are the Roberts, Prewitt and Sobel masks, which are shown in Table 2-1.

Operator	Mask 1	Mask 2
Roberts	$h = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$	$h = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$
Prewitt	$h = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$	$h = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$
Sobel	$h = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$	$h = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$

Table 2-1: First-order derivative gradient operators

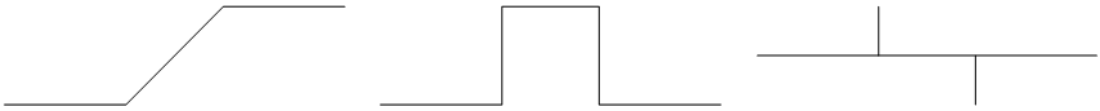
The simplest way of implementing a first-order derivative of an image is to use the Roberts operators, but because it is a 2x2 mask, it has no clearly defined center point, which makes it awkward to use. The Prewitt operator solves this by using a 3x3 mask, where the pixels on one side of the mask are subtracted from the pixels on the other side. By giving more importance to the center point, the Sobel operator achieves some smoothing, which gives it slightly better noise-suppression characteristics than the Prewitt operator.

The Prewitt and Sobel masks shown in Table 2-1 work best with horizontal and vertical edges, but they can be rotated to also find diagonal edges. Rotated Prewitt and Sobel masks are provided in Table 2-2.

Operator	Diagonal mask 1	Diagonal mask 2
Prewitt	$h = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$	$h = \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$
Sobel	$h = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$	$h = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$

Table 2-2: Diagonal Prewitt and Sobel operators

As can be seen in Figure 2-3, application of gradient operators result in thick edges where the precise location of the edge is difficult or impossible to determine. It also provides no information about which side of the border an edge point is located. By using the second-order derivatives, double edges are created instead with negative values on the bright side of the border, and positive values on the dark side. By placing edges where the double derivative crosses zero, a thin border with a precisely defined position is created.



a) Gray-level profile of edge b) First-order derivative c) Second-order derivative

Figure 2-3: Edge, with first- and second-order derivatives

The second-order derivative of an image can be estimated using the Laplace operator shown in Table 2-3.

Operator	4-neighborhood	8-neighborhood
Laplace	$\nabla^2 f = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\nabla^2 f = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

Table 2-3: Second-order derivative Laplace operator

These operators are not commonly used because the second-order derivatives are extremely sensitive to noise and give no information about edge direction. But the Laplace operators are valuable for estimating precise edge location by finding the zero-crossings of the result. To reduce the effect of noise, it is combined with a blurring mask resulting in the Laplacian of Gaussian (LoG) operator. This operator can be made of different sizes depending on how much blurring is applied. The 5x5 LoG is shown in Table 2-4.

Operator	5x5 mask
Laplacian of Gaussian	$\nabla^2 f = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$

Table 2-4: Laplacian of Gaussian

Morphological Edge Detection

Morphology has two basic operations; erosion and dilation [8]. Both use a structuring element, which is simply a small set of pixels, and for each pixel x of the image, a new value is calculated depending only on the pixels covered by the structuring element centered at x.

Erosion is defined by

$$[\varepsilon_B(f)](x) = \underbrace{\min_{b \in B}} f(x+b)$$

where $f(x)$ is the original image, and B is the structuring element. This means that each pixel, x, in the eroded image is assigned the smallest value of the pixels of the image covered by B when B is centered on x.

Dilation is defined by

$$[\delta_B(f)](x) = \underbrace{\max_{b \in B}} f(x+b)$$

which means that each pixel, x, in the dilated image is assigned the value of the largest pixel of the image covered by B when B is centered on x.

Three combinations of these operations are used for finding three different gradients of an image [8]:

1. Arithmetic difference between the dilation and the erosion, called the Beucher gradient
2. Arithmetic difference between the dilation and the original image, called the external gradient
3. Arithmetic difference between the original image and the erosion, called the internal gradient

The Beucher gradient is the basic morphological gradient, but it results in edges that are two pixels thick. Therefore the other alternatives are often used. The internal gradient enhances internal boundaries of bright objects, and external boundaries of dark objects, and the external gradient does the opposite.

Normally the morphological gradients provide no information about edge direction, but edges of a certain direction can be found by using a structuring element for the erosion and dilation that is a line perpendicular to the desired edge.

2.3.2 Boundary Detection

To find the desired structures of the image, unique borders must be found that surround these structures. Ideally, the methods described in the previous section should provide these borders, and nothing else, but this is almost never the case. Image noise, non-uniform illumination and a number of other factors result in random undesired edges as well as gaps in the desired borders. Numerous methods are used for linking the edges into likely borders.

The simplest methods look at a small neighborhood around each pixel with a significant edge. If a pixel with similar magnitude and direction of it's gradient is found, a border is created between them. If not, the edge is removed. In [9], Canny describes a method for thresholding edges based on their magnitude and neighboring pixels.

Hough Transform

If the desired structure has a known shape that can be described by a mathematical formula, like a line or a circle, the Hough transform [10] can find the positions in an edge image where this structure is most likely to be located.

Straight lines can be described in their normal representation as

$$x\cos\theta + y\sin\theta = \rho$$

To find these lines, a 2-dimensional Hough image, where the axes represent θ and ρ, is created. Then the edge image is scanned for every pixel (x, y). When an edge is found, all the corresponding pixels (θ, ρ) in the Hough image are increased. The brightest pixels in the Hough image will then correspond to the most likely lines in the original image.

A similar approach can be used for locating circles. These are represented by

$$(x-a)^2 + (y-b)^2 = r^2$$

where r is the radius of the circle and (a, b) is the location of the circle center. In this case, a 3-dimensional Hough volume must be created with axes representing a, b and r. Then the process continues in the same way. Because there are three unknowns in the equation, the processing time and memory required can be quite substantial, but this can be reduced if the radius or position is known exactly or approximately.

Graph Theoretic Methods

If a start and end pixel of the border is known, it is possible to use graph theory to find the best border between these points. A large graph is created with a node for every pixel, and costs are created between each node. This cost can be a combination of several functions [6]:

- It is likely that a border consist of pixels with high gradient values. Thus the cost $c_i = (max\{\nabla(x_k)\}) - \nabla(x_i)$ can be used for adding pixel x_i to the border, where the magnitude of the pixel in the gradient image is subtracted from the maximum gradient value.
- In many cases, borders with small curvature are preferred, so that a cost of $c_{i,j} = abs[\phi(x_i) - \phi(x_j)]$, where $\phi(x_r)$ is the gradient direction of pixel x_r, can be used for creating a border between pixels x_i and x_j.
- Sometimes there can be an estimate of where the border should be. In these cases the distance from the pixel to the estimated border can be used as a cost.
- Other cost functions may be used as well when other higher level information about the desired structure is known. A number of different cost functions can be found in [11].

When the graph is created, finding the minimum-cost path is still a complex memory- and computationally demanding task because of the size of the graph. Several methods exist for making the method useful. Some of these are described in [6].
When 3-dimensional data exist in the form of several image slices, the complexity of the graph search is further increased. Not only is the graph bigger, but the search must be for a surface instead of a border. In [12], Frank describes a surface growing algorithm that is computationally efficient and fast, but it does not guarantee optimal results.

2.4 REGION-BASED SEGMENTATION

Because it is difficult to detect edges properly, especially in noisy images, it is common to create regions directly instead. The basic idea of region-based segmentation is to divide the image into regions of maximum homogeneity. When the algorithm finishes, the result should be a set of regions so that each region complies with a homogeneity criterion, which could be an average gray level, certain multi-spectral properties, texture, etc. It should also not be possible to merge two adjacent regions without violating the homogeneity criterion. Four basic methods of region-based segmentation exist.

Region Merging

With region merging, each pixel of the image is initially considered as a separate region. A region merging criterion is created based on the homogeneity criterion. Adjacent regions are then iteratively merged if they satisfy the merging criterion. The algorithm finishes when it is impossible to merge any adjacent regions without violating homogeneity.

The result of region merging may differ depending on the order in which regions are merged. Consider the case where three regions, R1, R2 and R3 exist, where R1 may be merged with both R2 and R3. If R1 and R2 are merged creating (R1/R2), it may then be impossible to merge this region with R3, so that the end result is (R1/R2) and R3. But if R1 is initially merged with R3, the end result may be (R1/R3) and R2 instead.

Region Growing

Region growing is very similar to region merging, but instead of initially considering each pixel to be a separate region, one or more seed-regions are created. These regions are allowed to grow by adding neighboring pixels as long as the homogeneity criterion is not violated.

Region Splitting

With region splitting, the entire image is initially considered to be a single region. This will in most cases not satisfy the homogeneity criterion. Regions are then iteratively split until each region satisfies the criterion. A quadtree is usually used for keeping track of all the regions (or an oct-tree in a 3-dimensional implementation).

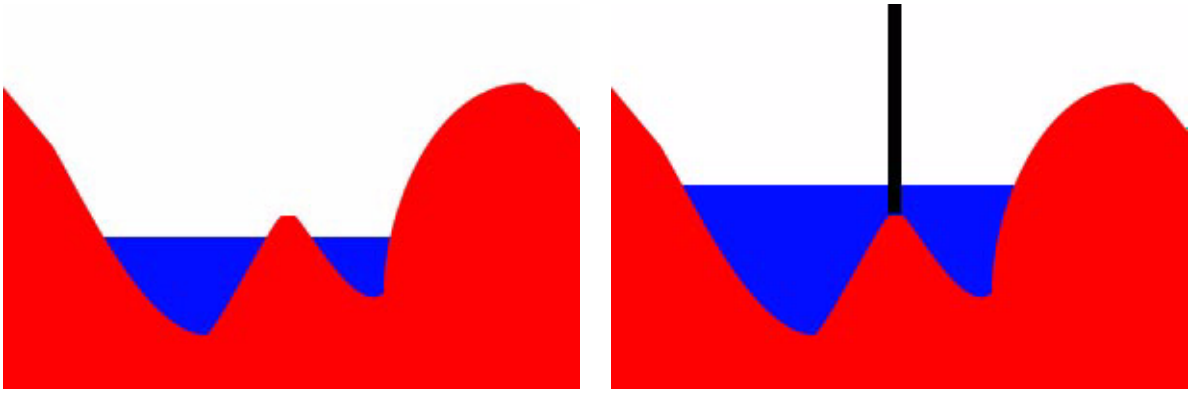
The problem with region splitting is that parts of the image that should have been one region is often split into many because the dividing lines of the quadtree cut right through them. The result will usually be heavily over-segmented.

Region Split-And-Merge

Split-and-merge combines the best properties of splitting and merging [13]. A region splitting is first performed creating an over-segmented image. Then region merging is performed to merge together homogenous regions that have been split into more parts. The results are usually better than with region merging because the initial regions for the merging process are larger than single pixels, so that noise has less effect on the result.

2.5 WATERSHED SEGMENTATION

Watershed segmentation is a method that comes from mathematical morphology [8]. Images are thought of as 3-dimensional topographic maps (with x and y corresponding to east and north, and gray level corresponding to altitude). If a drop of water is placed at a pixel, it will flow downstream to a local minimum of the image surface.The set of pixels that all end up in the same minimum is called a catchment basin, and the borders between catchment basins are called watersheds. When used on a gradient image, the watersheds will form closed regions that surround areas of similar gray levels [14].



a) Watershed segmentation with flooding at an early stage.

b) At a later stage, the two catchment basins have met, and a watershed has been created.

Figure 2-4: Watershed Segmentation by immersion simulations

In some cases, where the topographic map has a flat plateau, the flow direction at a given pixel is not determined. Because of this, and for better performance, a slightly different analogy is used when implementing watershed segmentation. Instead of dropping water from above, the model is immersed in water so that each catchment basin is flooded from it's local minima [15]. When the water from two catchment basins meet, a dam is created to separate the basins. At the end of the flooding process, each local minima will be surrounded by dams representing the watersheds.

Figure 2-4 illustrates the flooding process at two different stages. In a), the flooding process has just begun, and water is filled from the bottom into two separated catchment basins. In b), the water is so high that the two catchment basins meet, and a watershed has been created separating them. This watershed will represent the border between the two segmented regions.

In more formal terms, the flooding process proceeds as follows [8]:

The lowest value of the gray scale image f is denoted by h_{min}, and it's highest value is h_{max}. The catchment basin that is associated with a local minimum M is known as $CB(M)$, and the points of this catchment basin with an altitude that is less than or equal to h are denoted by

$$CB_h(M) = \{p \in CB(M) | f((p) \leq h)\} = CB(M) \cap T_{t \leq h}(f)$$

where $T_{t \leq h}(f)$ indicates thresholding of the image f with a maximum threshold level of h.

The subset of all catchment basins with a gray scale value less than or equal to h is

$$X_h = \bigcup_i CB_h(M_i)$$

The set of points belonging to the local minima of elevation h are denoted by $RMIN_h(f)$.

When the flooding process is running, the first points to be reached are the regional minima at level h_{min}, which is equivalent to $X_{h_{min}}$. The algorithm proceeds recursively by flooding the catchment basins to level $h+1$. This is done by creating one or more connected components Y of $T_{t \leq h+1}(f)$. Each of these connected components can have one of three relations to X_h:

1. If $Y \cap X_h = \phi$, Y must be a new regional minimum, so Y is added to $RMIN_h(f)$ and a $CB(Y)$ is created.
2. If $Y \cap X_h$ is a single connected region, Y is an expansion of an existing catchment basin M, and $CB_{h+1}(M)$ set equal to Y.
3. If $Y \cap X_h$ is two or more unconnected regions, Y is an expansion of two or more catchment basins. Watersheds are created in the middle between these basins, and the basins expand so that they combined fill Y.

When $h = h_{max}$, all catchment basins are filled, and the segmentation is finished.

Watershed With Markers

One problem with watershed segmentation is that the results are usually over-segmented because of spurious local minima in the image. Several methods can help this. The simplest is to use smoothing of the image before processing to reduce the number of local minima, but this rarely gives good enough results. It may also weaken desired edges between regions. In [16], an over-segmenting watershed segmentation is followed by region merging. But the most common way of solving the problem of over-segmentation is to use watershed-by-markers where minima are imposed in the image for each desired region, thus removing all undesired local minima [17].

Minima imposition is a two step process. All gray levels in the original image are increased by one, and a marker image is created where the imposed minima are 0 and all other pixels have the maximum gray level. The point-wise minima of these two images is calculated. The second step is a morphological reconstruction by erosion of this result from the marker image [8][18].

In [19], Meyer proposed an algorithm that combines minima imposition and watershed segmentation based on a priority queue. The flooding process is initiated from each minima by inserting into the queue the neighboring pixels of the minima. Pixels with the smallest gray level in the queue are removed and marked, and it's neighbors are added to the queue. This was further improved with the Image Foresting Transform in [20][21].

2.6 MODEL-BASED SEGMENTATION

Model-based segmentation methods are used to find the contour of structures in images based on image information such as the gradient. Higher level information about the desired structures is used to create closed contours that fall along the most likely borders of the sought structures.

2.6.1 Traditional Snakes

Snakes, or active contour models, are energy-minimizing splines that are controlled by external and internal energy [22][23]. The snake behaves according to image features, and tries to lock on to nearby edges while keeping a regular shape.

The snake is usually represented parametrically by $v(s) = [x(s),y(s)]$. The energy function that controls the movement of the snake is

$$\begin{aligned} E_{snake} &= \int_0^1 E_{snake}(v(s))ds \\ &= \int_0^1 (E_{int}(v(s)) + E_{ext}(v(s)))ds \\ &= \int_0^1 (E_{int}(v(s)) + E_{img}(v(s)) + E_{con}(v(s)))ds \end{aligned}$$

where E_{int} represents the internal energy of the spline due to bending and stretching, and E_{img} is the image forces that pull the spline towards edges. E_{con} gives external constraint forces which can be provided by the user interactively, but these are normally not used with traditional snakes [22].

The energy of the internal forces in a point $v(s)$ is given by

$$E_{int}(v) = \frac{1}{2}\left(\alpha(s)\left|\frac{\partial v}{\partial s}\right|^2 + \beta(s)\left|\frac{\partial^2 v}{\partial s^2}\right|^2\right)$$

where $|v|$ is the magnitude of vector v, $\alpha(s)$ controls the elasticity/stretching of the snake, and $\beta(s)$ controls rigidity/bending of the snake. If $\beta(s)$ is set to 0, the snake can be second-order discontinuous (have sharp corners), but with larger values it will try to maintain a smooth outline.

E_{img} can be provided by many different functions, but some of the most common are

$$\begin{aligned} E_{img} &= I \\ E_{img} &= -I \\ E_{img} &= -|\nabla I|^2 \\ E_{img} &= -|G_\sigma * \nabla I|^2 \\ E_{img} &= -|G_\sigma * \nabla^2 I|^2 \end{aligned}$$

where I is the original image, ∇I is the gradient of the image, and G_σ implies blurring with a gaussian filter of size σ.

The snake will move around the image while changing shape and size, until it comes to rest at a local minima of the energy function.

Snakes have many advantages compared to other segmentation methods. Even if edge information in the image varies, or is completely missing at some points, the snake will automatically fill the gaps to make a single continuos contour. By setting $\beta(s)$ high, the snake can be forced to prefer smooth outlines which is often desired, i.e. in medical imaging because the human body has few sharp corners. But by setting $\beta(s)$ to a lower value, sharp corners can also be accommodated by the same algorithm. By setting $\alpha(s)$ to a high value, the snake can be set to prefer short paths. This helps to make an intuitive shape when edge information is missing.

A problem with active contours is that E_{img} usually has a short capture range. If the snake is not initialized close to the desired border, it will not lock on and be pulled towards the correct edge. Several methods are used to help this problem. The simplest is to use one of the last two versions of E_{img} described above, where the edge is smoothed. This increases the reach of the edge somewhat, but also reduces the accuracy because the precise location of the boundary may be lost.

In [22], the authors developed a user interface where a user could insert “volcanoes” and “springs” that would push and pull the snake. This adds to E_{con} and can be used to move the snake towards a desired edge if the initialization is not close enough. The downside is that it requires user interaction.

2.6.2 Discrete Contour Models

The original definition of snakes is difficult to implement directly. When the snake is given as a continuos function $\nu(s) = [x(s),y(s)]$, there is no way to do pointwise local deformation according to external forces because the snake must be deformed as a whole.

A solution to this was suggested in [24] where the snake is defined as a set of vertices connected by edges. Each vertex has a cost that should be minimized. This is evaluated as

$$C_i(x, y, d, \alpha) = I(x, y) + D(x, y) + \Lambda(d) + \Theta(\alpha)$$

where $I(x, y)$ is an image term that uses simple thresholding to indicate if the current position is likely to be inside the desired structure, $D(x, y)$ is a potential field that causes the contour to either grow or shrink, $\Lambda(d)$ is a linear function that grows with the distance to the neighboring vertex, and $\Theta(\alpha)$ maintains the shape of the model by adding a cost if the angle between the two edges connected to the vertex is too far from $180°$. This model is discrete because, unlike with traditional snakes, the energy function is evaluated only at the vertices, and not for the connecting edges.

In [25], the discrete dynamic contour model was introduced. This uses the same kind of discrete model with vertices connected by edges, but the dynamic process that controls the contour is different. Instead of each individual vertex trying to minimize it's energy function, they are exposed to forces and force fields. Each vertex is given a mass and velocity, and it's movement is simulated according to the laws of physics.

At discrete steps, a force is applied to each vertex as a weighted sum of internal and external forces which will cause acceleration, which again will change the velocity of the vertex as it moves through the image.

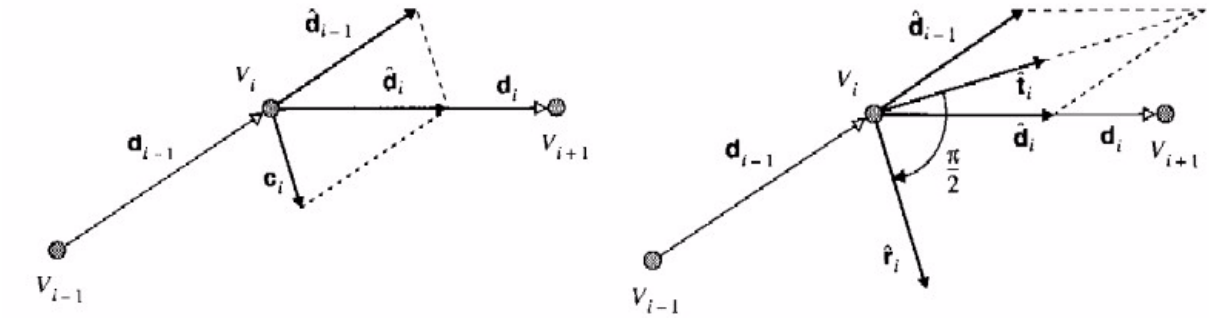
Internal Forces

The internal forces are used with discrete dynamic contour models to minimize local curvature. This will counterbalance the external forces that try to shape the model according to all the variations in the image feature landscape, as with traditional active contours.

Local curvature at a vertex V_i is defined as the difference in directions between the two edges that meet at V_i. This means that local curvature c_i at V_i is given by

$$c_i = \hat{d}_i - \hat{d}_{i-1}$$

where $\hat{d}_i$ is the unit vector of the edge segment d_i going from V_i to V_{i+1}. This is illustrated in Figure 2-5a).



a) **Local curvature.** b) Tangential and radial directions

Figure 2-5: Local curvature, tangent and radial direction at vertex V_i [17]

The tangential direction $\hat{t}_i$ at a given vertex V_i can be found as

$$\hat{t}_i = \frac{\hat{d}_i + \hat{d}_{i-1}}{\left\| \hat{d}_i + \hat{d}_{i-1} \right\|}$$

and radial direction $\hat{r}_i$ by rotating $\hat{t}_i$ by $\pi/2$ radians:

$$\hat{r}_i = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \hat{t}_i$$

The resulting vectors $\hat{r}_i$ and $\hat{t}_i$ are shown in Figure 2-5b). They represent a local coordinate system for vertex V_i which is used for evaluating both internal and external forces.

The local curvature vector c_i will always be pointing in the same, or opposite, direction of $\hat{r}_i$. By looking at c_i in the local (r,t) coordinate system, it can be given a positive or negative length along the r-axis as

$$c_i = (c_i \cdot \hat{r}_i)\hat{r}_i$$

Curvature vectors for some typical situations can be seen in Figure 2-6 for both Cartesian representation and in (r,t)-coordinates.

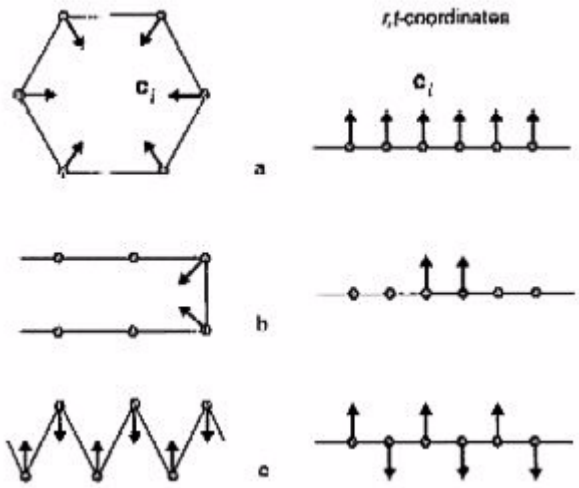


Figure 2-6: Curvature vectors in Cartesian representation and (r,t) coordinates [25].

Figure 2-6 shows that the curvature vectors can not be used directly as internal forces. If the contour is exposed to no external forces, it will form a circle of constant curvature (which is the best estimate of a smooth outline), but this circle will shrink until it disappears. Because c_i can be represented as a positive or negative length along the r-axis, this can be fixed by simple convolution. The internal force $f_{in}(V_i)$ for V_i is evaluated as

$$f_{in}(V_i) = ((c_i \cdot \hat{r}_i) \otimes k_i)\hat{r}_i$$

where

$$k_i = \left\{ \ldots, 0, 0, -\frac{1}{2}, 1, -\frac{1}{2}, 0, 0, \ldots \right\}$$

The value 1 applies to position i, and the values -1/2 to positions $i-1$ and $i+1$.

The resulting internal forces are shown in Figure 2-7. It is obvious that these will try to create a region of constant curvature, but have no effect on the contour when this is accomplished.

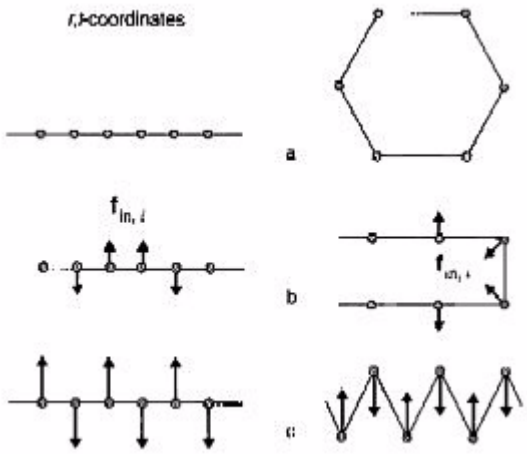


Figure 2-7: Internal forces calculated from curvature vectors in Figure 2-6 [25].

When the contour models are used on 3-dimensional data, each vertex may have edges to more than two vertices. All of these edges should be included when calculating internal forces to get the best possible result.

External Forces

Any of the energy functions described in section 2.6.1 can be used to produce the energy distribution E_{img}. The user is also allowed to add external energy E_{user} by for example setting points that attract or reject the contour. The external energy is

$$E_{ex} = E_{img} + E_{user}$$

The force expression for external energy for vertex V_i is

$$f_{ex}(V_i) = -\nabla E_{ex}$$

This causes problems because the force is free to act in any direction; tangential as well as radial. This may cause vertices to cluster in local minima of the image because a force along the contour can push the vertices towards each other. The problem can be solved by decomposing the force into a locally radial and a locally tangential component, and using only the radial component. The resulting force is

$$f_{ex,r}(V_i) = (f(V_i)_{ex} \cdot \hat{r}_i)\hat{r}_i$$

Deformation

The force $f(V_i)$ acting on a vertex V_i is a weighted combination of external and internal forces. A damping force is also used to help the contour come to rest. The resulting force is

$$f_i = f(V_i) = w_{ex}f_{ex,r}(V_i) + w_{in}f_{in}(V_i) + f_{damp}(V_i)$$

$$f_{damp}(V_i) = w_{damp}v_i$$

where v_i is the current velocity of the vertex. The weighting factor w_{damp} is negative and determines the amount of damping.

The model is deformed according to the rules of physics. Each vertex has a mass m_i. At discrete points in time, spaced Δt apart, the position p_i of vertex V_i is updated according to the following scheme:

$$p_i(t + \Delta t) = p_i(t) + v_i(t)\Delta t$$

$$v_i(t + \Delta t) = v_i(t) + a_i(t)\Delta t$$

$$a_i(t + \Delta t) = \frac{1}{m_i}f_i(t + \Delta t)$$

Resampling

If two neighboring vertices move too far apart, the resolution of the result will suffer. If many vertices are spaced very close, processing time is wasted because all of these vertices give a higher resolution than what is of practical use. A resampling scheme is therefore used to control the resolution of the model, where a control parameter l_{des} governs the length of each edge segment. This is used to calculate

$$l_{min} = \frac{1}{2} l_{des}$$
$$l_{max} = \frac{3}{2} l_{des}$$

The resampling is implemented as a two-pass process:

1. The entire contour is examined to see if any edge segment has become shorter than the minimum length l_{min}. If this is the case, the edge segment is removed by replacing the two vertices on each side of the edge with one single vertex positioned between the replaced vertices.
2. The entire contour is examined again to see if any edge segment has become longer than the maximum length l_{max}. If this is the case, the edge segment is divided into two shorter segments by adding a vertex in the middle.

2.6.3 Additional External Forces

The discrete dynamic contour model has proven to give very good results in many situations [25], but like the traditional active contours it requires a close initialization. This is obvious because the discrete model is only an implementation of the traditional snake principles introduced in [22]. An advantage is that it is easier to add new external forces that can be used to help the problem.

Balloon Forces

Balloon forces [26][27][28] are external forces that either expand (inflating a balloon) or shrink the snake (deflating a balloon). This will make the contour change it's size until it is stopped by the image forces when it has locked on to a sufficiently strong edge. With an inflating force, it is absolutely essential that the initial contour is entirely within the desired structure, and similarly with an deflating force, it must be initialized outside the structure. If this is not the case, the shape may expand forever or shrink into nothingness.

Setting the strength of the force correctly is critically important. If the force is too weak, the contour may lock on to weaker incorrect edges before it reaches the desired border, and if the force is too strong it may move past it.

Another problem occurs when a piece of the edge is missing or too weak. The resulting shape will have a dent at those positions because the balloon forces will push it outwards or inwards, and the internal forces can not prevent this completely.

Gradient Vector Flow

Gradient Vector Flow (GVF) was introduced by Xu and Price in [29][30] as a static external force, used as the image force, that helps the problem with the capture range, as well as a problem with poor convergence to boundary concavities.

The GVF is computed as a diffusion of the gradient vectors of an edge-map derived from the image. The gradient vector flow field is defined to be the vector field $V(x,y) = [u(x,y),v(x,y)]$ that minimizes the energy functional

$$\varepsilon = \iint (\mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f|^2 |V - \nabla f|^2) dx dy$$

From this it can be seen that when ∇f is small, the energy is dominated by the sum of squares of the partial derivatives of the vector field, which results in a slowly varying field. When ∇f is large, the second term takes over, and is minimized by setting $V = \nabla f$. The effect from this is that V is kept nearly equal to the gradient of the edge map when it is large, but forced to be slowly-varying in homogenous regions. μ is a regularization parameter that governs the trade-off between the two terms. If there is a lot of noise in the image, μ should be increased to give a smoother vector field.

Numerically, the GVF can be evaluated iteratively using

$$u_{i,j}^{n+1} = (1 - b_{i,j}\Delta t)u_{i,j}^n + r(u_{i+1,j}^n + u_{i,j+1}^n + u_{i-1,j}^n + u_{i,j-1}^n - 4u_{i,j}^n) + c_{i,j}^1 \Delta t$$

$$v_{i,j}^{n+1} = (1 - b_{i,j}\Delta t)v_{i,j}^n + r(v_{i+1,j}^n + v_{i,j+1}^n + v_{i-1,j}^n + v_{i,j-1}^n - 4v_{i,j}^n) + c_{i,j}^2 \Delta t$$

where

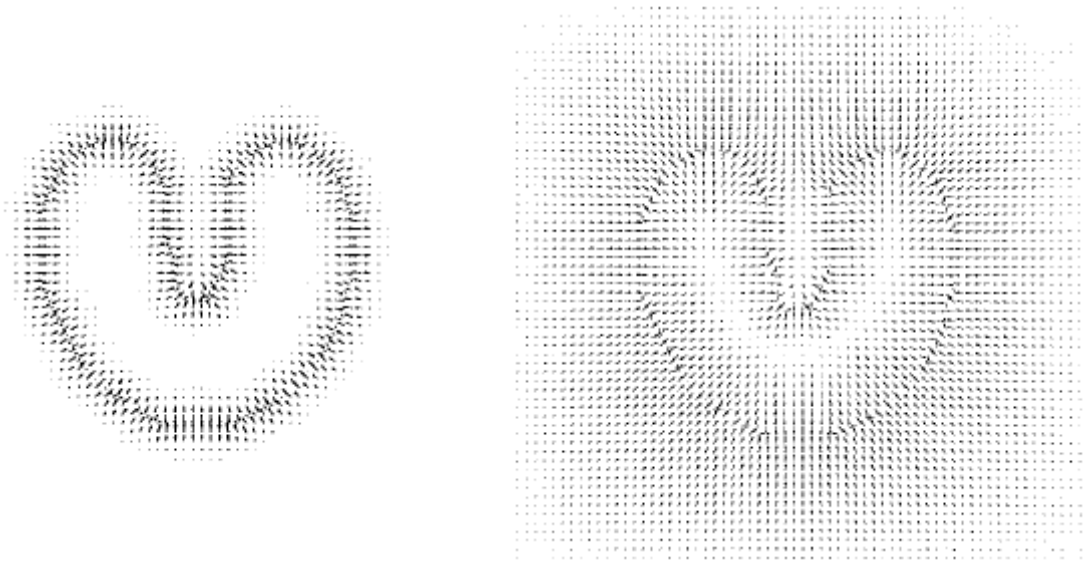
$$b(x,y) = f_x(x,y)^2 + f_y(x,y)^2$$

$$c^1(x,y) = b(x,y) f_x(x,y)$$

$$c^2(x,y) = b(x,y) f_y(x,y)$$

$$r = \frac{\mu \Delta t}{\Delta x \Delta y}$$

Figure 2-8 shows the vectors of a traditional potential field used with active shapes, as well as the gradient vector field of the same shape. It can easily be seen that the GVF has a much longer capture range while the exact position of the edges remain clear. It also helps the contour move into border concavities which is a problem with normal image forces.



a) Standard potential field. This requires very close initialization.

b) GVF potential field. This has very long capture range while the original edge is still clear.

Figure 2-8: Gradient Vector Flow versus standard potential forces [29].

Calculating the Gradient Vector Flow is time consuming, but the results from it are very good [31].

CHAPTER 3: ALGORITHM SELECTION

Selection of the segmentation algorithm to use is the most important part of the project. Using the wrong algorithms it will be impossible to accomplish good results. This chapter will describe factors that must be taken into account when selecting the algorithm, and discuss the different algorithms described in Chapter 2 according to these.

3.1 Relevant Information

Many factors must be considered when choosing the right segmentation algorithm. The algorithm must be suitable for the data sets it will be working on, and the structure that should be segmented. It should also be as fast as possible, at least faster than manual segmentation, and require as little user interaction as possible.

3.1.1 The Data Sets

This thesis will be used as part of a large project at Haukeland University Hospital where they are developing MR-examinations of the kidneys. At this early stage of the project, very few data sets are available, making it difficult to create statistical models for assisting segmentation. Two different MR-machines (one with 1.5 Tesla and one with 3 Tesla) are being used, and different physical methods for getting the best images with these machines are tested. This means that the images currently available are of highly variable quality. The final image acquisition method will not be developed until after the end of this thesis, so the algorithm must be somewhat flexible. Examples of the current images are shown in Figure 3-1.

Figure 3-1: Examples of images

The first image is clearly of poor quality. The signal-to-noise ratio is low, and there are large defects in the image due to poor acquisition parameters. The second image is much better, but it is near impossible to locate the border between the right kidney (to the left in the image) and the liver. The third image is probably best for segmentation because of fairly good contrast around the kidneys.

Several image volumes are taken of each patient. These include anatomical image series and several dynamic image series taken at different time steps after contrast injection. The anatomical images provide the best signal-to-noise ratio and image quality, but the dynamic images provide extra information because different parts of the anatomy are highlighted at different times as the contrast moves through the body. This may help in segmentation of the kidneys, as:

- After 15-20 seconds the contrast highlights the cortex and arteries
- After 1 minute the contrast highlights the cortex and medulla
- After 5 minutes the contrast highlights the renal pelvis and towards the bladder.

The different image series are not registered at this time, so the structures in the images will move and change size between series as the patient moves and breathes. The image slices of each data volume should be fairly well registered by themselves because they are obtained in a very short time frame.

Registration of the different image volumes is a separate part of the project that will be performed at Haukeland University Hospital. They plan on using the segmentation as an aid in registration, so the segmentation algorithm should work on unregistered images. This makes multi-spectral segmentation algorithms, where more than one image series is used, difficult and inaccurate. Some work on multi-spectral segmentation algorithms for kidneys is presented in [32][33].

Each of the data volumes consist of approximately 22 image slices with 512x512 pixels each. The spacial resolution is approximately 0.75mm, and the spacing between slices is approximately 3mm. All of these factors vary as different methods of image acquisition are tested.

The full series of the last image in Figure 3-1 is shown in Figure 3-2. Even though the parameters for image acquisition are changing as the kidney project progresses, there should always be images of this quality.

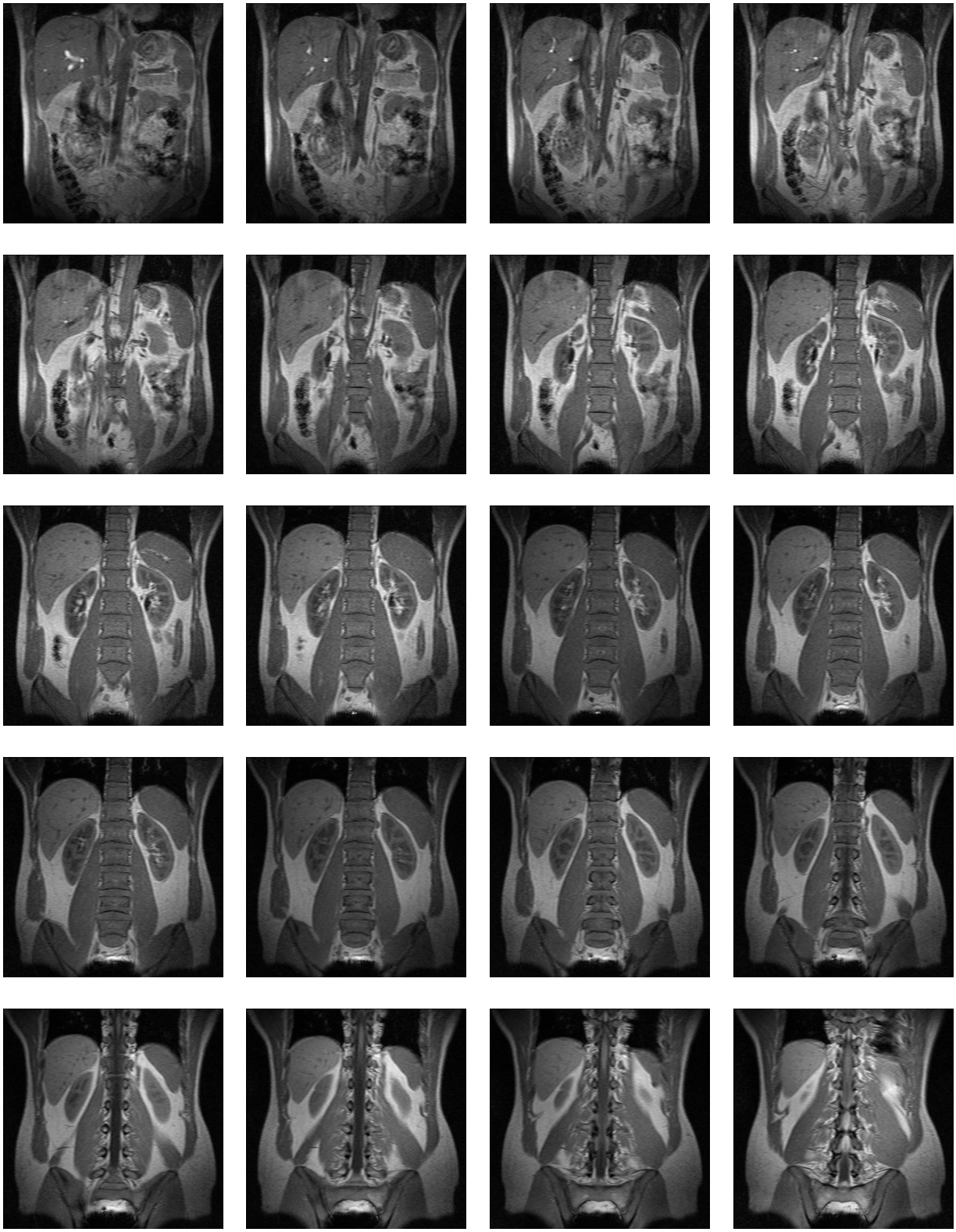


Figure 3-2: An example image series

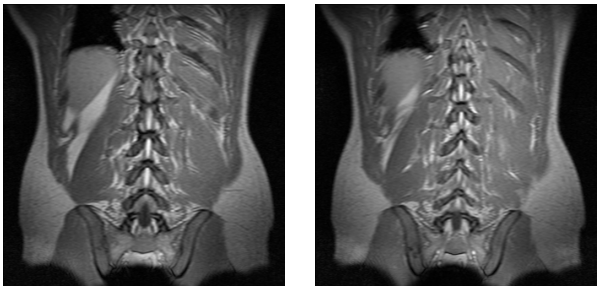


Figure 3-2: An example image series

Some important factors should be noted:

- In the first 5-6 images it is nearly impossible even for the human observer to see where the kidneys are located or if they even stretch to those image slices at all. The dark shape that looks like the kidney in the first three slices is actually the duodenum.
- In images 5-7 and 20, the right kidney (to the left in the image) is divided into two parts in the images.
- In images 7-13 the medial parts of the kidneys (towards the middle) have gaps in them because the renal pelvis in the kidney, which are in fact just a hollow room, gives a similar signal as the surrounding tissue.
- The interior of the kidneys has variable gray levels because the medulla gives a slightly different signal from the cortex.

3.1.2 Programming Environment

The program and algorithm that is developed will be part of a larger Java workstation that is being developed at Haukeland University Hospital. It is therefore a requirement that the program should be developed in Java using Java Advanced Imaging (JAI). This requires more work for the programming than if for example Matlab could be used, where a lot of basic algorithms are already implemented.

3.2 THE ALGORITHMS

Chapter 2 provides information about several possible algorithms for segmentation of the kidneys. The selection of which algorithm to use is extremely important as the performance of each of the algorithms varies depending on the images they are used on and the problem to be solved.

3.2.1 Discussions About the Algorithms

Each of the methods in Chapter 2 have advantages and disadvantages that that should be considered in the selection of which algorithm to use. This section provides some relevant information for this project about each of the algorithms.

Manual Drawing

Manual drawing is commonly used for segmentation in medical images, also with renography [34]. It is used as the "gold standard" that other segmentation methods are compared against because it gives accurate results where correctness is confirmed by the expert user who performed the segmentation.

There are unfortunately several drawbacks as well. Manual drawing is very slow, and requires the constant attention of an expert user who is then prevented from doing other work. It also does not provide reproducible results because the user may not draw the exact same regions if he or she is to do the same segmentation again.

Thresholding

Thresholding is the fastest segmentation algorithm, but it puts a lot of requirements on the data to be examined. As can be seen in the image series in Figure 3-2, the kidneys share gray levels with several anatomical structures nearby. Different parts of the kidney also have different, and variable, gray levels.

Better results could be obtained by using multi-spectral thresholding where thresholds in different image series are combined. This requires careful registration of the image series, and it is likely that structures that give similar signals in image series also will do wo in others. The results are therefore questionable at best.

Thresholding uses no spatial information, so there will almost always be random pixels that are incorrectly considered to be part of or not part of the desired structures. This also means that information found in one image slice can not be used to help segmentation of neighboring slices, except as a guide.

In [35], a multi-spectral thresholding is used for segmentation of kidneys, but because of problems with respiration-induced motion of the kidneys between image series, it only works with renal transplants with an unchanging position.

Edge-Based Algorithms

The edge-based segmentation algorithms require that good edges can be found that surround the desired structures. Figure 3-3 shows the result of Sobel edge detection on the last image in Figure 3-1.

Figure 3-3: Sobel edge detection of MR image

The edges around the kidneys are fairly good, but where the right kidney meets the liver, there is a large gap in the edge information. Because the renal pelvises of the kidneys are similar to the background, there will also be edge information missing here. This will usually be the case, and makes edge-based segmentation difficult. It can be possible to help edge detection by using information from neighboring image slices, but gaps like the one between the kidney and liver are usually present in several adjacent slices.

There is unfortunately no mathematical formulation of a general “kidney-shape” as individual kidneys are slightly different, so the Hough transform is not available.

Region-Based Algorithms

Region-based segmentation algorithms are based on finding homogenous regions. As can be seen in the images in Figure 3-2, the kidney is not really that homogenous as different internal structures of the kidneys have different gray-levels. This means that the homogeneity criterion can not be too strict. With region merging and region growing, this can cause the region to grow into the liver or other nearby tissue which has a similar gray level.

It is possible to use multi-spectral data from several image series, but again the different internal structures of the kidneys give different signals which makes it difficult to find a good homogeneity criterion. This also requires the image series to be registered.

A multi-spectral region-growing is used in [36] for segmentation of kidneys in rats with good results. But because of the registration problem, the rats have to be anaesthetized so that they are completely still during the MR examination.

Watershed

Watershed segmentation has the benefit that it can use both regional information about gray levels etc. and edge information. If markers are used, it can be determined the number of regions, and how these should be situated according to each other. It can also benefit 3-dimensional data by making regions grow between image slices.

In Figure 3-2 it is seen that the renal pelvises of the kidneys have similar gray levels as the surrounding tissue, even though they should be considered part of the kidneys. It will be very difficult for the watershed algorithm to classify these as kidney without also growing into the surrounding tissue.

In [37], watershed segmentation has been used for segmentation of kidneys with good results.

Model-Based Algorithms

The model-based algorithms have many benefits. They always result in closed contours, and they give a good boundary estimation where edge information is missing because the internal forces maintain a smooth outline in those regions. They are commonly used in medical imaging [38][41], and also specifically for kidney segmentation [42].

The problem with model-based segmentation is that a very close initialization is required [39]. If not, the active contours will not find edges to lock on to, or lock on to incorrect edges. Usually, a statistical model of the desired structure is created by analyzing several manually segmented structures as in [42][43].

3.2.2 Selection of algorithm

From the discussion above it is clear that a model-based segmentation should be used because it provides a smooth and accurate outline of the segmented structures even when edge information is missing or weak as it is around the renal pelvis and between the kidney and liver.

But at this time, there is not enough data available to make a good statistical model of the kidneys that can be used for initialization. Therefore, a watershed segmentation will be performed as an initial step, and this result is used to initialize the active contours. This will also make the algorithm very general so it can easily be applied to other segmentation problems.

A similar approach has been used in [40] with some success.

CHAPTER 4: IMPLEMENTATION

The algorithm that was selected in section 3.2.2 has been implemented for segmentation of kidneys from MR images. The algorithm first performs watershed segmentation to create an initial region which is used for initializing a model-based segmentation method.

This chapter will describe the implementation in detail.

4.1 USER INTERFACE

The user interface for the application is based on the one developed in [44] using Java and Java Advanced Imaging, and looks something like Figure 4-1.

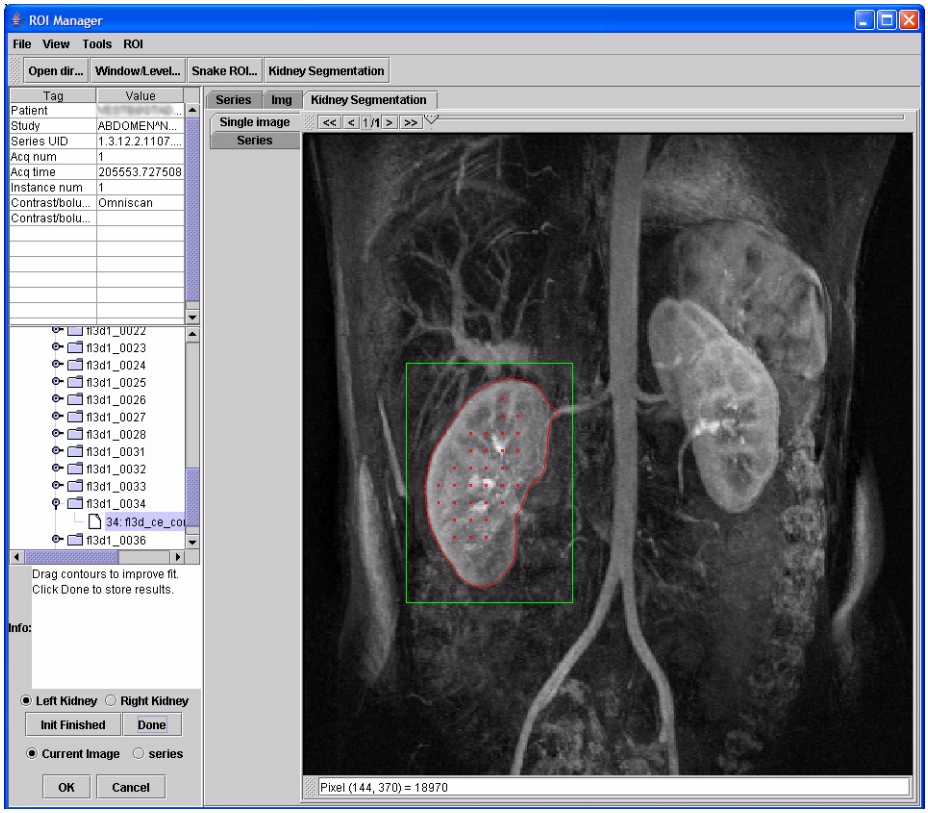


Figure 4-1: The implemented graphical user interface

The user can open and view different image series. He has the option of viewing a series one image at a time, or all images at once. It is also simple to interact with the algorithm by pointing and dragging with the mouse.

When the algorithm is put into clinical use, a different GUI will be used, so it's implementation has not been a priority.

4.2 WATERSHED SEGMENTATION

The implemented watershed algorithm is based on watershed-from-markers using the Image Foresting Transform (IFT) [20][21].

4.2.1 Description of the Algorithm

The algorithm given in the original papers works as follows:

```
1. Initialization
    a) for all voxels p do
        flag(p) = TEMP;
    b) for all non-marker voxels p do
        C(p) = infinity;
    c) for all marker voxels p do
        C(p) = 0;
        EnQueue(p, 0);
2. Propagation
    a) v = DeQueueMin;
    b) flag(v) = DONE;
    c) for each p neighbor of v with flag(p) == TEMP do
        if max{C(v), w(v,p)} < C(p) then
            A) C(p) = max{C(v), w(v,p)};
               L(p) = L(v);
            B) if IsInQueue(p) then
                DeQueue(p);
            C) EnQueue(p, C(p));
```

A priority queue is used for keeping track of the voxels as they are examined. The voxels are sorted according to a given cost, with an internal FIFO restriction for each cost. The FIFO ordering makes sure that the best possible border is created even on plateaus where the gray levels are constant. `EnQueue(p, c)` will place voxel `p` in the queue behind all voxels with a cost `c` or less. `DeQueueMin` will remove the voxel with the lowest cost from the queue and return it, and `DeQueue(p)` will remove voxel `p` from the queue.

Each vertex has:

- a cost `C` that is initialized to 0 for the marker voxels and infinity for all other voxels,
- a label `L` that is initialized to a given value for the marker voxels,
- a flag that is initialized to `TEMP`, and updated to `DONE` when a label has been permanently assigned to it.

The algorithm starts by adding all marker voxels to the queue. Then, as long as the queue is not empty, the first voxel, `v`, is removed and flagged as `DONE`. For each of v's neighbors, `p`, that is not already marked as `DONE`, a weight `w(v,p)` is calculated as the cost of giving `p` the same label as `v`. In the original paper $w(v,p) = |f(p) - f(v)|$ is used which is a simple approximation of the image gradient, but any `w(p,v)` can be used that works for the given segmentation task.

If the maximum of `C(v)` and `w(v,p)` is lower than the cost currently assigned to `p`, voxel `p` will be assigned the new cost of the maximum of `C(v)` and `w(v,p)` and the same label as `v`. To keep the queue from growing more than necessary, `p` will be removed from the queue if it is already in it, before it is added again with it's new cost.

4.2.2 The Implementation

The algorithm starts with the user drawing a rectangular frame around the kidney. This frame only has to be drawn in one image slice, but should have the position and size such as to frame the kidney in all the slices. This frame is used to give an initial position of the kidney, as well as reducing the required memory because only the voxels inside the drawn rectangle are used in the calculations.

Markers for the kidney are created using a binary image that is read from disk. This is a black image with a very rough "kidney shape" drawn in white. The image is scaled so that it is the same size as the initialization rectangle. Every ten pixels in both directions are sampled. If the pixel is 1, a marker is created at that point in the middle image in the MR-series. If the series consists of more than 6 images, markers are also created in the images that are located two slices away from the middle.
This simple initialization also makes the algorithm extremely flexible. To use the same algorithm in another segmentation problem, the only change that needs to be done is to create a new binary image that looks roughly like the desired shape.

For the background, markers are created from all voxels on the border of the rectangle, in all slices. Because the kidney should be fully included in the data volume, the first and last slices should not contain kidney, and are also used as background markers.

The algorithm works very similar to that described in the previous section. The queue is created with an array of Vectors (linked lists). One Vector is available for every possible voxel cost. Ordering the queue in this manner makes queue operations, such as add and remove, faster because there is random access to the voxels of a certain cost. The use of Vectors makes the memory required by the queue dynamic so that it is not necessary to reserve space for all voxels at the same time.

The queue has methods for removing and returning the first voxel, adding new voxels, and moving voxels in the queue when it's cost changes. The latter method is used to simplify steps 2.c.B and 2.c.C in the algorithm.

Each voxel keeps track of it's label, cost and flag, as in the original algorithm, but it also has a list of all it's neighbors that are flagged as DONE. This is used to weight the cost that is assigned to it in step 2.c.A. If many of it's neighbors are permanently assigned to the same label as this voxel, it's weighted cost is low, but if many of it's neighbors are permanently assigned to other labels, it's weighted cost is higher. The queue is sorted according to this weighted cost. This is done to keep the regions as compact as possible while they are growing, and reduce the risk of having a region grow out of it's desired borders through a small hole in the edge information.

The cost that is given to a voxel is calculated in the same way as in the original algorithm using $w(v,p) = |f(p) - f(v)|$. But because the distance between image slices is much greater than the distance between pixels in each image, arc weights across image slices are penalized. C(p) is

$$C(p) = \begin{cases} w(v,p), \text{ if v and p are in the same image slice} \\ 1,\ 1 \times w(v,p), \text{ if v and p are in different slices} \end{cases}$$

The reason for this is that the size of the kidney can change substantially between two image slices. This increases the risk that voxels on the edge of the kidney in one slice can grow into nearby structures with similar gray levels in a neighboring slice where the kidney is smaller.

4.3 ACTIVE CONTOUR

The implemented algorithm is based on the discrete dynamic contour model [25][31][45] which was described in section 2.6.2.

4.3.1 Implementation

The implementation is similar to the implementation in [44]. A separate contour is created for each of the image slices, with links between them. Each contour is a linked list of vertices. Each vertex keeps track of it's position, velocity, acceleration, next and previous neighboring vertices in the same contour, it's nearest vertices in the neighboring image slices, and it's last five positions. It is influenced by internal and external forces similar to the ones described in section 2.6.2.

Internal Forces

For each vertex V_i in one image slice, the local curvature c_i is calculated as the vector describing the edge from V_{i-1} to V_i, subtracted from the vector from V_i to V_{i+1}.

When the algorithm works in 3D data, a vector l_i is calculated as the sum of the edges from V_i to it's nearest vertices in the two neighboring image slices. This vector is then scaled and added to c_i. This helps to maintain a similar shape and size between image slices.

The curvature vector is then evaluated to a positive or negative length $c_{i,r}$ along the radial direction $\hat{r}_i$. When the local curvatures are evaluated for all the vertices, the strength $s_{in,i}$ of the internal force $f_{in,i}$ for each vertex is calculated as

$$s_{in,i} = c_{i,r} - \frac{1}{2}c_{i-1,r} - \frac{1}{2}c_{i+1,r}$$

This is equivalent to the convolution described in section 2.6.2.

The internal force is then easily created along the radial direction using the dot product

$$f_{in,i} = s_{in,i}\hat{r}_i$$

External Forces

The image forces are based on the gradient of each image. The gradient magnitude of the image is created using the horizontal and vertical Sobel operators. The resulting edge image is then blurred heavily to increase the capture range of the image forces. Because this also softens the edges and makes results less accurate, the original gradient image is added to the blurred image to make the precise edges stand out.

From this enhanced gradient image, eight gradient images are created for the directions north, north-east, east, south-east, south, south-west, west and north-west. The magnitude of these images give the strength of the image force in each of these directions. These are created as vectors and stored in a 3-dimensional array of vectors.

When the image force for a given point is requested, an interpolation scheme is used to find the best force field as a combination of the vectors in the four closest voxels in the same slice.

Gradient Vector Flow (GVF) was considered as the image force to get longer capture range, but was rejected. One of the benefits from GVF is that it better pulls the contour into boundary concavities, but because the image of the kidneys may have incorrect concavities, i.e. in the renal pelvis, it is better that the contour prefers a straight path in these cases.

A deflating balloon force (see section 2.6.3) is also used in the initialization phase of the algorithm. This is created as a scaled version of the radial vectors at each vertex.

Resampling

Resampling is mainly done as a two-step process:

1. All vertices of the snake are examined. If the distance between two vertices is more than three pixels, a new vertex is added between them.
2. All vertices of the snake are examined again. If the distance between two vertices is less than one pixel, these two vertices are replaced by one new vertex positioned directly between them.

At the end of the resampling, several vertices have been added and removed, so many links to vertices in neighboring slices are lost. These are therefore created again using a simple process. For the first vertex in each contour, the entire neighboring contours are searched for the closest vertices which are used. When the neighboring vertices for the first vertex have been found, a local search in the 20 vertices closest to the neighbor is used to find the nearest neighbor to the next vertex. This process continues around the contour.

Because of the computational demands in the resampling step, the contours are only resampled once every ten iterations.

Self-intersection Avoidance

A problem with active contours is that the contours may intersect with themselves forming loops. This is clearly undesirable. To guarantee that there can be no self-intersections adds very high computational demands [46][47] to the contours and is usually not done.

A simple method has been implemented that greatly reduces this problem. Every 300 iteration steps, a binary image is created with 1 for every pixel inside the contour, and 0 for all the others. Determining if a pixel is inside or outside is a simple matter of counting intersections with the contour. By doing this, the pixels inside of a loop are considered to be inside the contour even if though they are on the "outside" side of the active contour.

When this binary image has been created, the contour is recreated by tracing around the connected region of value 1 and adding a vertex at every step. This removes all loops without putting restraints on the movement of the vertices.

User Interaction

In some cases the contours may lock on to incorrect edges in the images, which will give an incorrect result. To prevent this, the user can interact with the contour by dragging directly on it's vertices. If a single vertex V_i is moved, it will have little effect, because the internal forces working on the vertex will be very large and pull it back to it's original position. By moving only one vertex at a time, a large number of user interactions are required for moving just a small part of the contour. To prevent this, the 10 closest vertices to V_i in both directions are also moved a distance

$$d_r = d_i \cdot \left(1 - \left(\frac{|r-i|}{10}\right)^{1,5}\right)$$

where d_i is the distance V_i was dragged by the user, and $|r-i|$ is the number of vertices between V_r and V_i. This gives a wider "grasp" on the contour, which also feels more natural to the user.

4.3.2 Initialization

The results from the watershed step are usually fairly good, but in some instances the regions are split into two or more parts, sometimes with with very small regions where the watershed region has grown from one slice to another incorrectly.

To help the second problem, and make the borders of the regions smoother, a morphological opening is performed on the result of the watershed. This is an erosion by a structuring element, followed by a dilation of the same structuring element. The causes all parts of the region that can not contain the structuring element to be removed, while the rest of the region remains unchanged.

The results from this opening may still have two or more regions in some of the slices. This is a problem for initialization of the active contour, because there should be only one contour for each image slice.

The contour is therefore initialized as a rectangle in each image slice given by the bounding rectangle of the opening result of the watershed segmentation. This contour is then placed under the influence of internal forces and the deflating balloon force described in the last section. No image forces are used, but the vertices are locked in position when they enter the watershed regions. This makes the contours shrink until they make a natural border that contains all large regions that were produced from the watershed segmentation.

To determine when the contours have converged to this result may be difficult. It is obvious that they have converged when they stop moving, but when the vertices are close to their local minima, they will often oscillate around it. To separate this small movement from a movement towards the edge, the distance d_i is calculated as the difference between the current position of vertex V_i and it's position 5 iteration steps ago:

$$d_i = |p_{i,t} - p_{i,t-5\Delta t}|$$

If this distance is large, it is likely that the vertex is moving towards it's best position, but if it is small, it has probably found it's local minima and is at rest or oscillating around it.

When the maximum distance $d = max(d_i)$ is below a given threshold, it is assumed that the contour has found a good initial shape, and the initialization is finished.

4.3.3 Progression

After the initialization phase is finished, the algorithm continues in a similar fashion. But now image forces are used instead of the deflating balloon forces. This makes the contours move towards edges in the images.

When the user is satisfied with the fit of the contours, he or she may stop the algorithm. If he does not, a similar convergence criterion as that described in the previous section is used based on the maximum distance d that any vertex has moved in the last five iterations. But instead of stopping the contour when d is below a threshold, the weights for the internal and external forces are reduced slowly when the movement is below the threshold. This will gradually reduce any oscillating movement and help the contour come to a complete rest with each vertex at it's best possible position.

CHAPTER 5: RESULTS

This chapter will show the results of different segmentation tasks using the implemented model described in Chapter 4. All the images used are real MR-images, and equivalent to the images the application will face in daily use.

The processing times reported here are obtained using a 1.5MHz Pentium-M laptop with 1024MB memory.

5.1 Case 1 - Right Kidney from One Slice With Contrast

The first test is to segment the right kidney (to the left in the image) on the single image shown in Figure 5-1. This image is in fact a 7.8mm thick volume that is compressed into a single image. The use of a contrast fluid and a long acquisition time makes the kidneys stand out, together with the aorta and other major blood vessels.

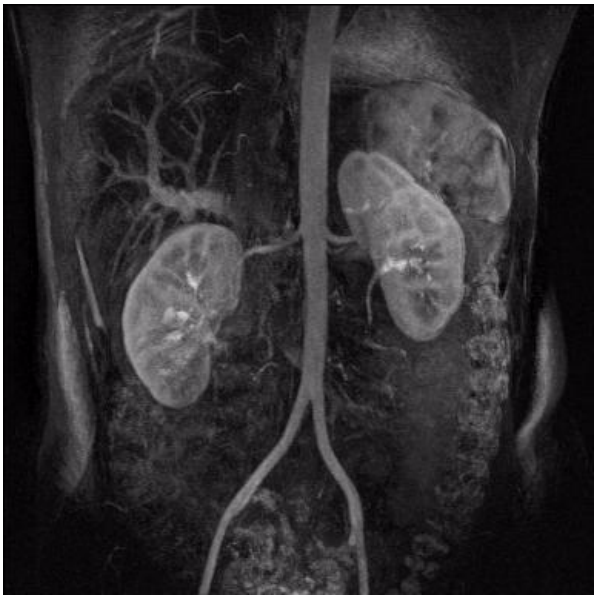


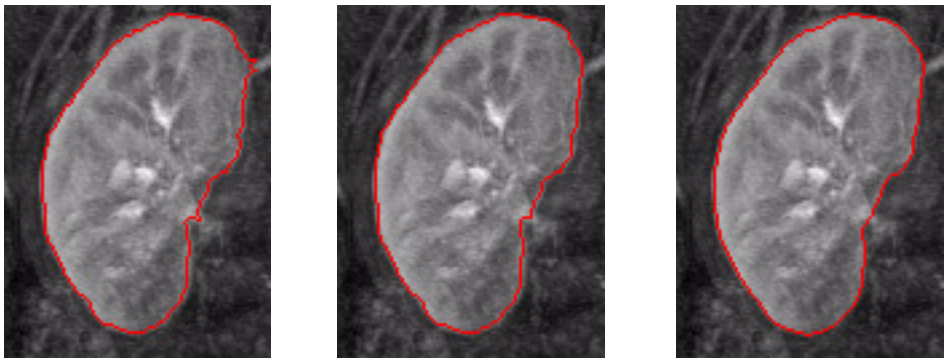
Figure 5-1: Case 1 source image

Because the contrast around the kidney is good, segmentation should not be too difficult. Possible problem areas would be the blood vessel entering the kidney, and some poor contrast along the renal pelvis.

Segmentation result

The watershed algorithm finished in 0,89 seconds and gave the result in Figure 5-2a). This is a close segmentation with only a few small flaws. At several points results seem to lie slightly inside the desired contour. Where the major blood vessel enters the kidney in the upper right-hand corner, the segmentation has moved slightly into the vessel, which has a similar gray-level to the kidney.

After the morphological opening is performed, the results are slightly better, as shown in Figure 5-2b). The region is still slightly inside the desired contour, but the dent in the blood vessel is removed, and the outline is a bit smoother. The opening took 1,12 seconds.



a) Watershed result b) Opening result c) Active contour result

Figure 5-2: Close up on the results for Case 1

The active contour also works relatively quickly. The initialization phase finished in 1025 iterations and 2,11 seconds, and the contour came to a complete rest as shown in Figure 5-2c) after only 139 more iterations. The fact that this was so fast proves that the watershed segmentation gave a very good initialization for the contour.

The resulting contour falls very closely along the borders of the kidneys and has a smoother outline than the previous results because of the internal forces. The complete time from drawing the rectangle around the kidney until the snake had finished was 6,069 seconds. Manual segmentation of this kidney takes 2-3 minutes.

Discussion

Table 5-1 summarizes the results at the different steps of Case 1.

Segmentation method	Time from start	Size in voxels	Over-segmented	Under-segmented
Manual	~2 min	6482	0	0
Watershed	~1 sec	6423	0,61 %	1,52 %
Opening	~2 sec	6402	0,35 %	1,58 %
Active Contour	~6 sec	6482	0,16 %	0,16 %

Table 5-1: Result summary for Case 1

In this fairly simple case, the watershed algorithm alone provided almost sufficient results, but each step of the algorithm improved the segmentation further. In the final result after the active contour, there is virtually no difference between the automatic segmentation and the manually drawn region. When this takes only 6 seconds and requires no user interaction, it is clearly a very good result.

5.2 CASE 2 - RIGHT KIDNEY FROM 22 SLICES WITHOUT CONTRAST

The second case study was performed on the image series shown in Figure 3-2. This is an anatomical series of 22 image slices. The spatial resolution is 0.74mm, and the slices are spaced 3mm apart. The right kidney (on the left in the image) was to be segmented.

Some factors that may cause difficulties when segmenting the right kidney from this image series are:

- In the first 5 images, it is difficult to determine the location of this kidney, or if it is even present at all.
- In images 6-10, the renal pelvis makes it very difficult to separate parts the medial side of the kidney from the surrounding tissue.
- In some image slices, especially 11-16, the edges separating the kidney from the liver on the upper left-hand side, and from the muscle tissue on the lower right-hand side, are fairly weak.

This section will describe the results of the segmentation for all the different steps of the algorithm.

Watershed Segmentation

Figure 5-3 shows close-ups of the results from the watershed segmentation for images 2-21. The first and last images are not included because the initialization does not allow them to contain kidney. The size of the images was determined by the frame the user drew to initialize the algorithm.

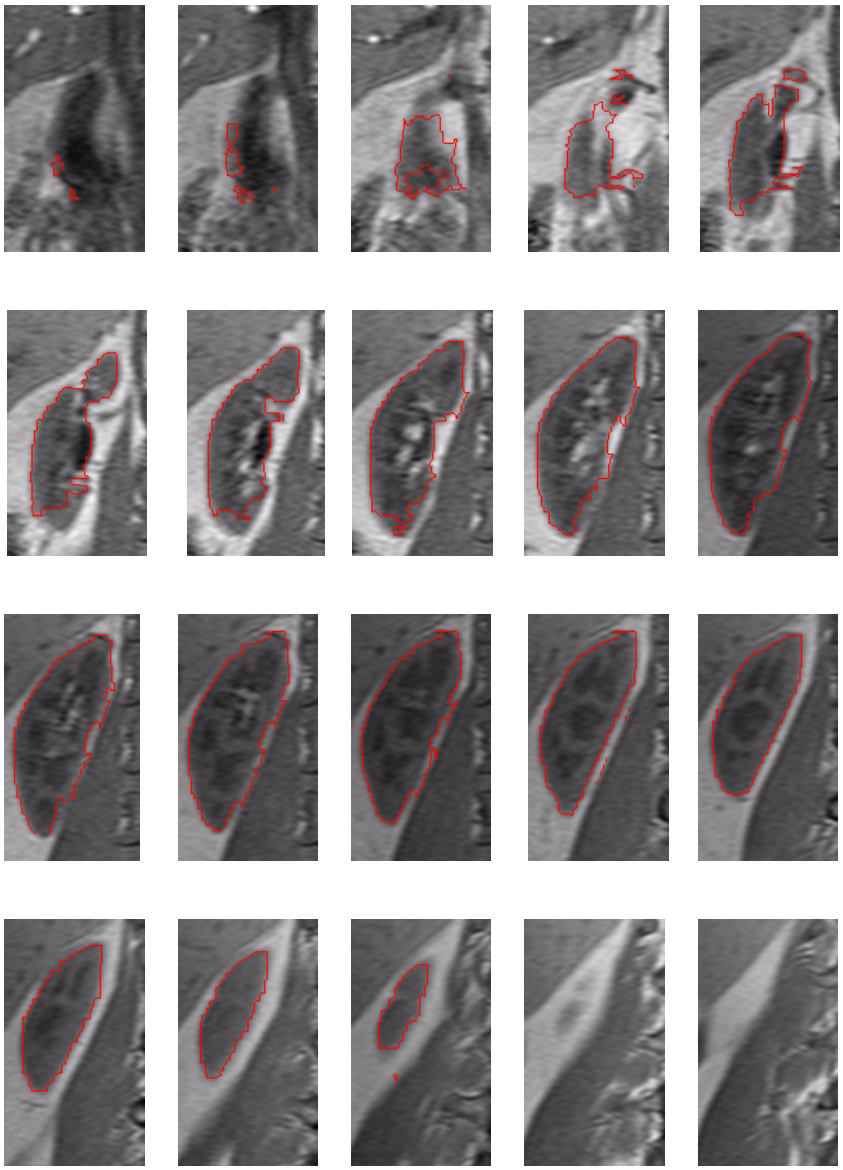


Figure 5-3: Watershed segmentation for Case 2

These results were obtained in 43 seconds, which is less than 2 seconds per image slice.

- In images 1-2, the watershed segmentation has only spread to a few small and unconnected regions. This is actually not a bad result because the kidney is not present in these slices. The dark region that looks like it is the duodenum.
- In image 3, the watershed has made a rough outline of the lower part of the kidney, but there is a large hole in the middle that has not been included. A single pixel has spread from the neighboring slice to the upper part of the kidney.
- In images 4-5, most of the kidney that is clearly visible has been marked by the algorithm, except from the bottom part. These regions are in several unconnected parts. On the right side, two narrow rays have grown towards the muscle tissue.
- Images 6-8 are segmented fairly well except for the bottom part which has not been found, and some dents in the renal pelvis.
- In images 9-14, all of the kidney seems to be included, but because of weak edges on the right side, the watershed has grown towards the muscle tissue here.
- In images 15-18, the segmentation seems good, but it has not stretched all the way to the border at all points. In image 18, there is also a small region below the kidney that has incorrectly been marked as kidney. This could cause problems for initialization of the active contour as it will stop when it hits this point.
- In image 19, the watershed is not present, even though the kidney is present in two small separate regions.
- In image 20, there watershed is correctly not present.

Morphological Opening

The morphological opening of the watershed regions took 33 seconds using a 7x7 structuring element and gave the result shown in Figure 5-4.

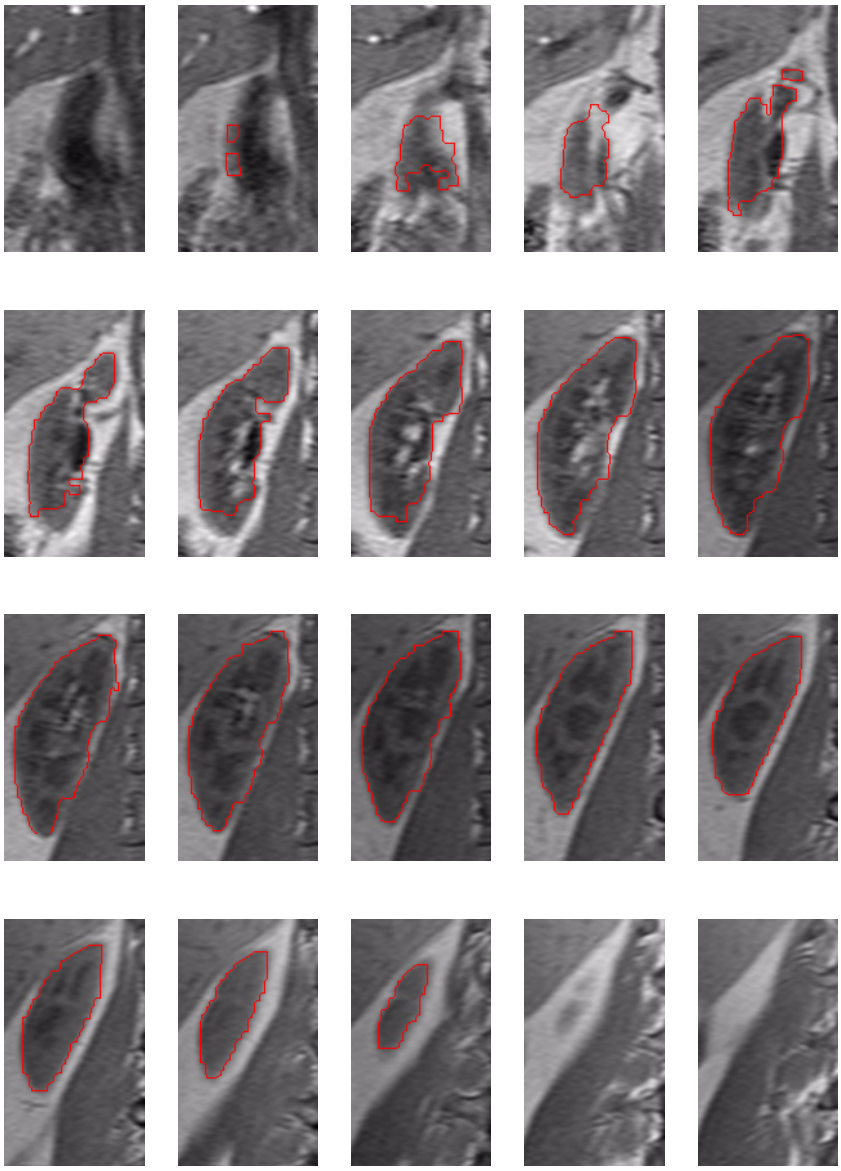


Figure 5-4: Opening of watershed result for Case 2

This result seems slightly better than after the watershed segmentation.

- In image 1, the small regions that were incorrectly present after the watershed have been completely removed.
- In image 2, some of the incorrect regions have been removed, but not all.
- In images 3-4, the small regions that were present in the upper parts of the kidney have been incorrectly removed by the opening. This may cause problems for the active contour as it will be initialized far from it's desired border in these images. The hole in image 3 has been opened so it now forms a substantial dent in the region instead.
- In images 4-5, the two rays going towards the muscle tissue have been completely removed.
- In image 18, the small incorrect region that could cause problems for the active contour has been completely removed.
- In the rest of the images, there is not a big difference from the watershed result, but the outlines are somewhat smoother, and the parts that have grown towards the muscle tissue have been reduced.

Active Contours Without User Interaction

The active contour did not come to a rest by itself, but after two minutes it had stabilized as shown in Figure 5-5 without any user interaction.

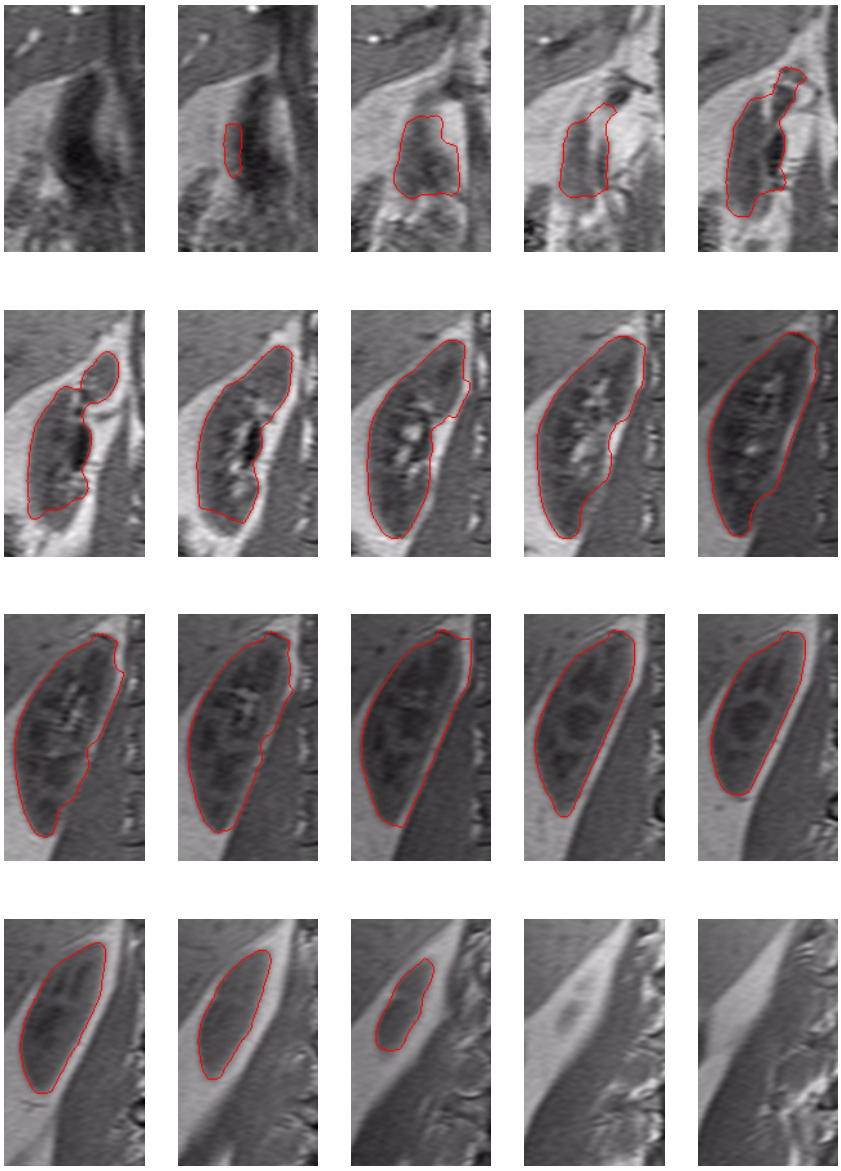


Figure 5-5: Active contours for Case 2 without user interaction

All of these contours are much smoother than the previous results, which is more consistent with the kidneys. They are also single contours and not unconnected regions.

- In image 1, there was no initialization for the active contour after the morphological opening, so there was correctly no kidney found.
- In image 2, there has not been much change from the opening, but the two regions that were present have been merged into one.
- In image 3, the dent that was present in the lower part has been completely, and correctly, removed. This indicates that the internal forces do a good job in the initialization so that the contour keeps a smooth outline, which is consistent with the kidney.
- In images 4-7, the active contour has not been able to move to the bottom edge of the kidney, but the borders are much smoother, and narrow dents that were present after the opening have been removed.
- In image 8, the contour has moved to the bottom edge of the kidney.
- In images 9-13, there is little difference from the opening, but larger parts of the border have moved to the nearby edge of the muscle tissue. This is because that edge has a higher gradient than the edge of the kidney, causing the contour to be pulled towards it..
- In images 14-18, the resulting contours are smoother and moved closer to the border of the kidney, giving a very good segmentation.

Active Contours With User Interaction

With some user interaction, the segmentation can be further improved. The results shown in Figure 5-6 were obtained after 34 interactions, in the form of pulling on the contour, which took about two minutes to complete.

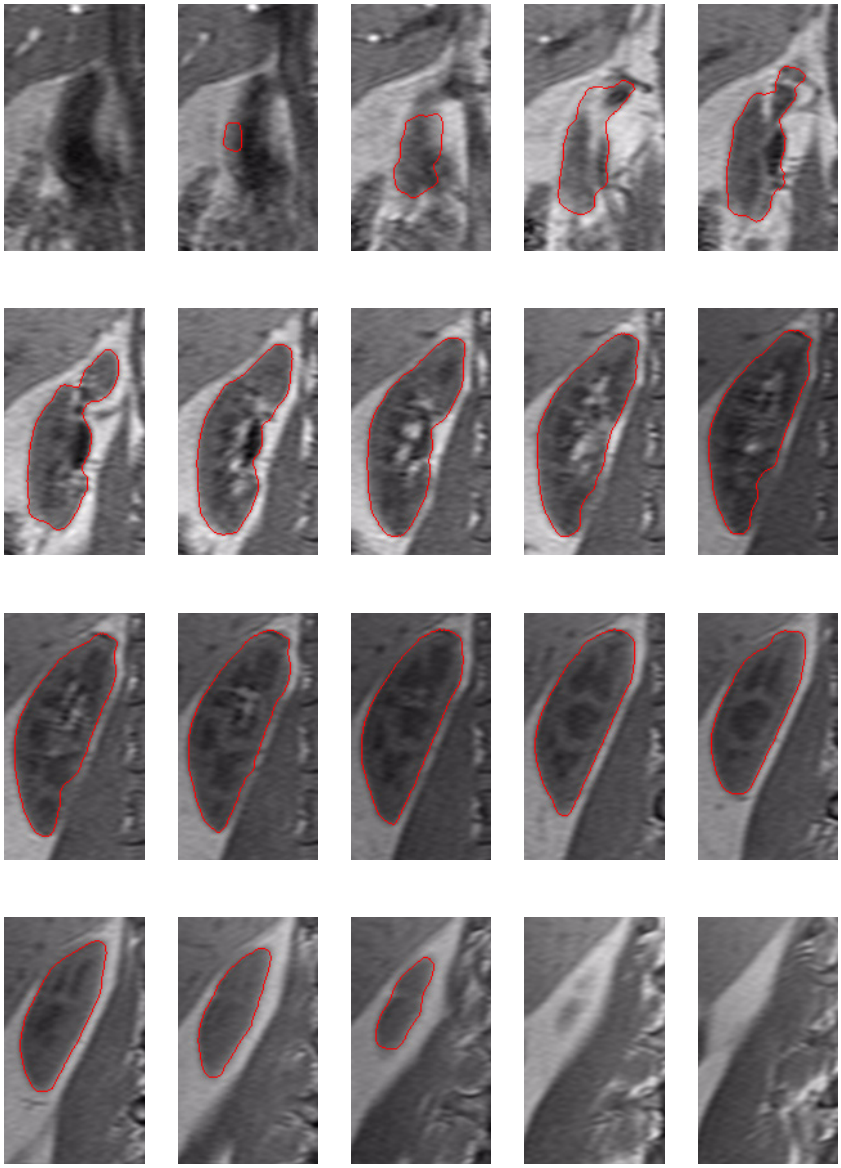


Figure 5-6: Active contours of Case 2 with user interaction

The user performing the interaction is not a trained radiologist, and thus had some problems locating the kidney in the first slices. The results are still very good.

- In image 2, the size of the incorrectly segmented region has been reduced greatly, but it is not completely removed.
- In image 3, the contour follows the kidney much better than the previous results, but the contour has not been pulled to the top part of the kidney, because the user did not recognize this as kidney.
- In images 4-6, the contour follows the edge of the kidney closely. A few places where there are dents in the visible kidney, this has resulted in dents in the segmented region as well.
- In images 7-18, the results are very close to the correct borders of the kidney. Where the contour had moved to the incorrect border of the muscle tissue, it has been pulled back to the correct edge and remained there.

Discussion

Table 5-2 shows the percentage of over-segmentation (voxels that have incorrectly been included in the region) and under-segmentation (voxels that have incorrectly not been included) in the segmentation for each of the images and for the volume as a whole. This provides information on how close the segmentation was to the manual segmentation which is assumed to be correct.

Image	Manual size	Watershed	Opening	Contour automatic	Contour with interaction
0	0	0 / 0	0 / 0	0 / 0	0 / 0
1	0	134 / 0	0 / 0	0 / 0	0 / 0
2	0	471/ 0	310 / 0	521 / 0	298 / 0
3	2171	34,2%/38,8%	28,2%/41,7%	41,7%/22,1%	9,67%/22,7%
4	3158	7,47%/32,9%	0,41%/36,0%	0,50%/30,4%	7,47%/4,14%
5	4020	5,47%/19,9%	3,00%/24,4%	5,02%/9,84%	7,38%/4,72%
6	5265	1,55%/19,4%	1,46%/20,3%	1,91%/13,6%	2,48%/4,93%
7	6223	0,65%/15,3%	0,48%/16,6%	0,51%/7,19%	0,61%/0,25%
8	6742	2,59%/9,27%	2,07%/10,6%	3,11%/0,25%	0,47%/0,17%
9	6888	6,27%/3,15%	5,40%/3,36%	9,10%/0,21%	1,66%/0,31%
10	7107	7,11%/3,09%	6,65%/3,29%	11,5%/1,47%	0,57%/2,04%
11	7050	7,57%/1,63%	6,62%/1,64%	9,63%/0,34%	0,34%/0,69%
12	7161	6,36%/3,03%	5,69%/3,14%	8,67%/1,38%	0,64%/0,48%
13	6893	5,15%/4,09%	4,20%/4,25%	13,5%/1,47%	0,39%/0,43%
14	6383	1,34%/5,84%	0,76%/5,93%	0,10%/1,89%	0,32%/0,20%
15	6390	0,52%/6,67%	0,70%/6,97%	0,25%/1,83%	0,37%/1,16%
16	4502	0,64%/7,50%	0,64%/7,52%	0,84%/0,33%	0,71%/0,33%
17	3100	1,16%/7,25%	1,16%/7,29%	1,51%/0,93%	1,32%/0,90%
18	1781	1,51%/18,24%	0,50%/18,24%	0,72%/5,55%	0,67%/5,67%
19	315	0 / 315	0 / 315	0 / 315	0 / 315
20	0	0 / 0	0 / 0	0 / 0	0 / 0
21	0	0 / 0	0 / 0	0 / 0	0 / 0
Volume	85153	5,45%/9,82%	4,12%/10,4%	6,86%/4,71%	1,92%/2,28%

Table 5-2: Summary table of results

From this it can be seen that the watershed algorithm quite heavily under-segments the kidney, especially in images 3-8. This happens because certain parts, especially the renal pelvis, have a gray level that is closer to the surrounding tissue than the rest of the kidney. Over-segmentation is also substantial because of incorrectly marked voxels, especially in the first slices where the kidney is not present.

The morphological opening reduces the over-segmentation because random voxels are removed, but the under-segmentation is even higher because some of these random voxels were actually correct, especially in images 4-7.

The active contours drastically reduce the under-segmentation. This happens because the internal forces make a more or less straight line even where edge information is missing, and this more closely resembles the actual shape of the kidney. This illustrates that the watershed segmentation can provide a good initialization for the active contours even though the watershed segmentation itself is heavily under-segmented. Over-segmentation increases slightly because the contours a few places are pulled towards the edges of nearby tissue instead of the edge of the kidney.

With user interaction, the results were naturally much better. To get these results, only a minimum of interaction was required, and the total time from beginning to end was less than 7 minutes. Manual segmentation took approximately one hour for the same data set.

From this, it is obvious that the algorithm works well also in 3-dimensional data with several image slices. Very little user interaction is required to obtain a very good segmentation much faster than with manual drawing.

5.3 CASE 3 - LEFT KIDNEY FROM 22 SLICES WITHOUT CONTRAST

The third case study was performed on the same image series as Case 2, but this time the left kidney (to the right in the image) was the target of the segmentation. This seems like a simpler problem than Case 2 because it is easier to see where the kidney disappears in the first images. The left kidney is also further away from other tissue of similar gray level. But the same problems exist with the renal pelvis.

Watershed Segmentation

For this case, the watershed segmentation took 47 seconds to complete. With 22 image slices, this is still just over 2 seconds per image. The results were as shown in Figure 5-7.

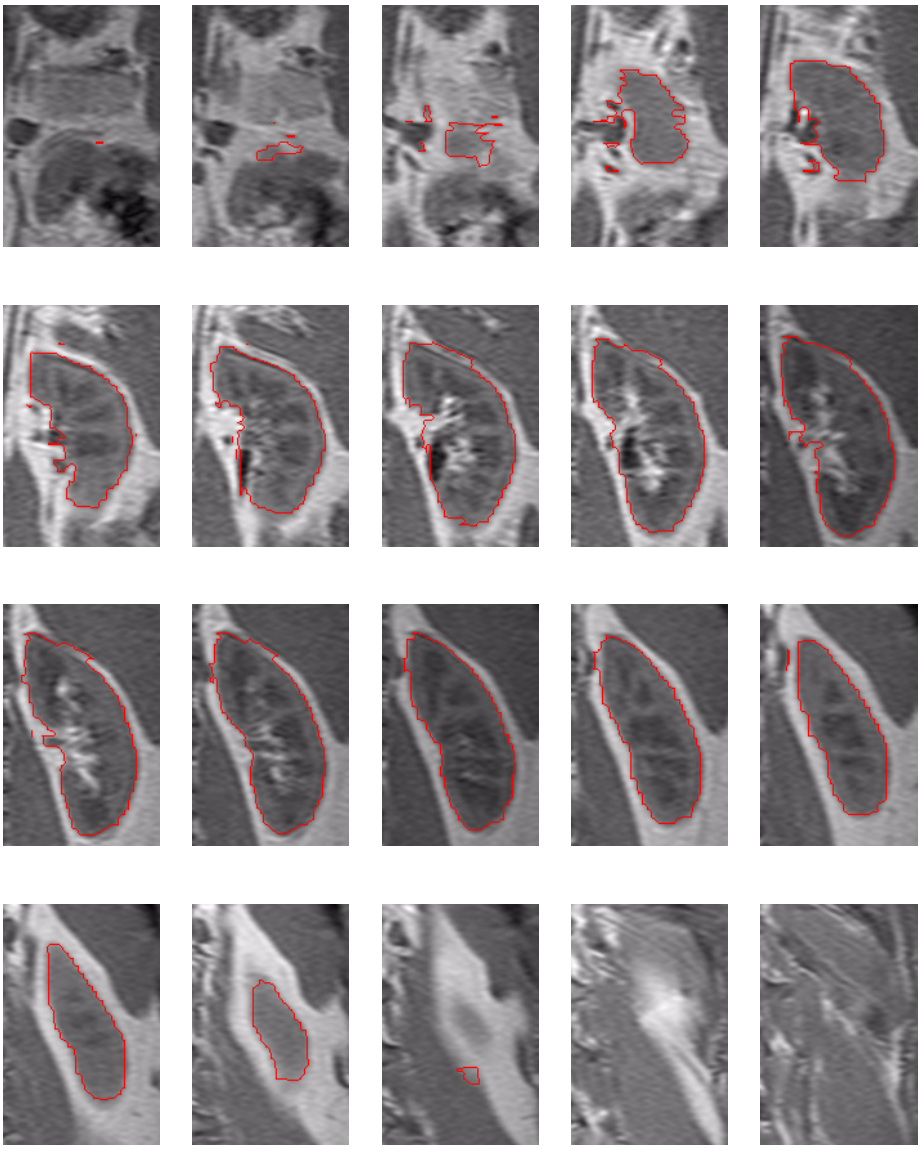


Figure 5-7: Watershed segmentation for Case 3

This seems to be a very good initial segmentation segmentation, but there are a few problems.

- In images 1-2, there is no kidney present, so the watershed should not have entered these images.
- In images 3-5, a fairly good outline of the kidney has been found, but there are also some smaller regions that are incorrectly segmented.
- Images 6-17 are segmented very well with the watershed algorithm, except for small irregularities on the left side of the kidney, and a few places where the watershed has grown to the border of the spleen in the upper right-hand corner.
- In image 18, there is a small segmented region, but this has completely missed it's correct position, and is placed well below the kidney.
- In images 19-20, there is correctly no kidney found.

Morphological Opening

The morphological opening of the result from the watershed segmentation finished in 40 seconds. The results are shown in Figure 5-8.

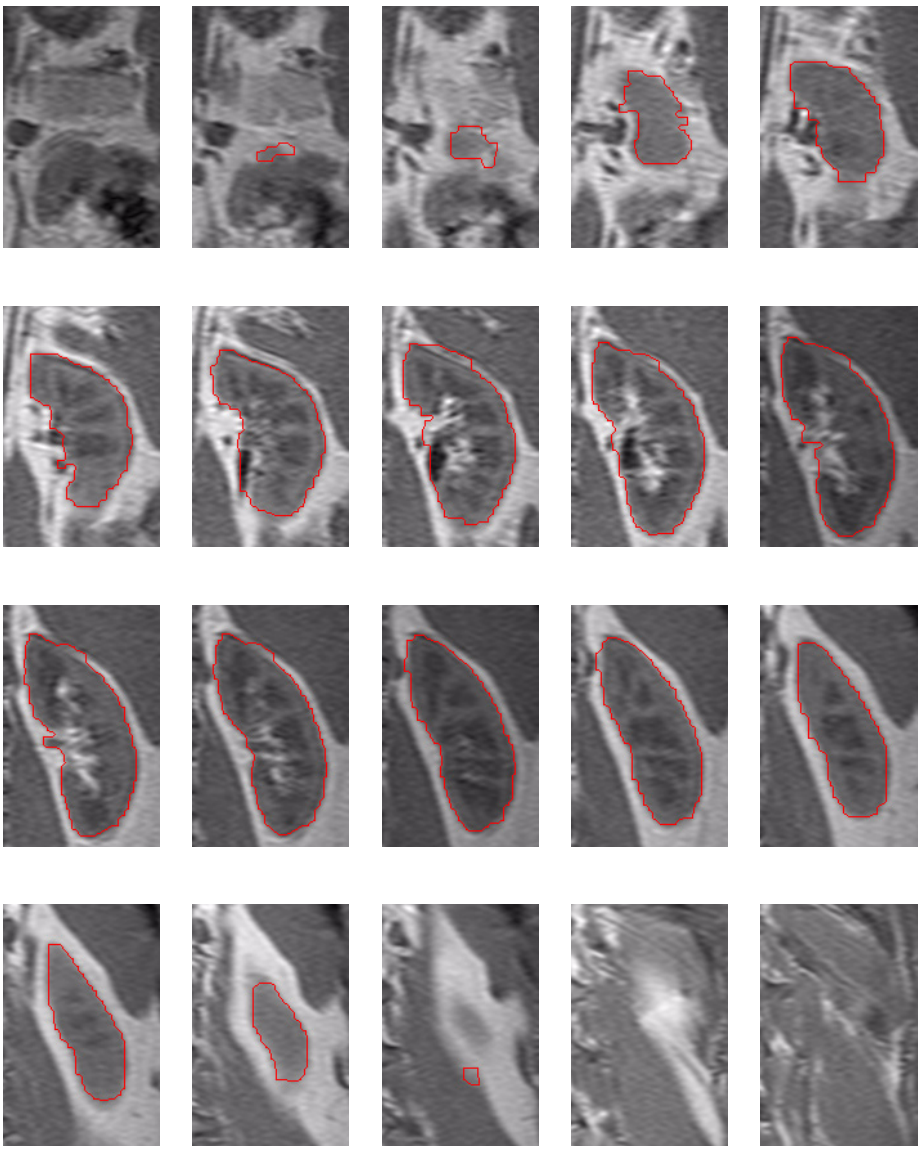


Figure 5-8: Opening of watershed result for Case 3

Here the results have been improved a great deal from the watershed segmentation. The outlines are much smoother, and most of the small irregularities have been removed.

- In image 1, the incorrectly segmented regions from the watershed segmentation have been completely removed.
- In image2, most of the incorrect regions have been removed, but there is still one small part left.
- In images 3-17, the results are very good. The noise on the left part of the kidney has been removed, but the segmentation still follows the border of the spleen instead of the kidney in a few places.
- In images 18-20, there is no significant change from the watershed result.

Active Contours Without User Interaction

As in Case 2, the active contours did not converge and stop by themselves, but after only one minute they were almost at rest as shown in Figure 5-9 without any user interaction.

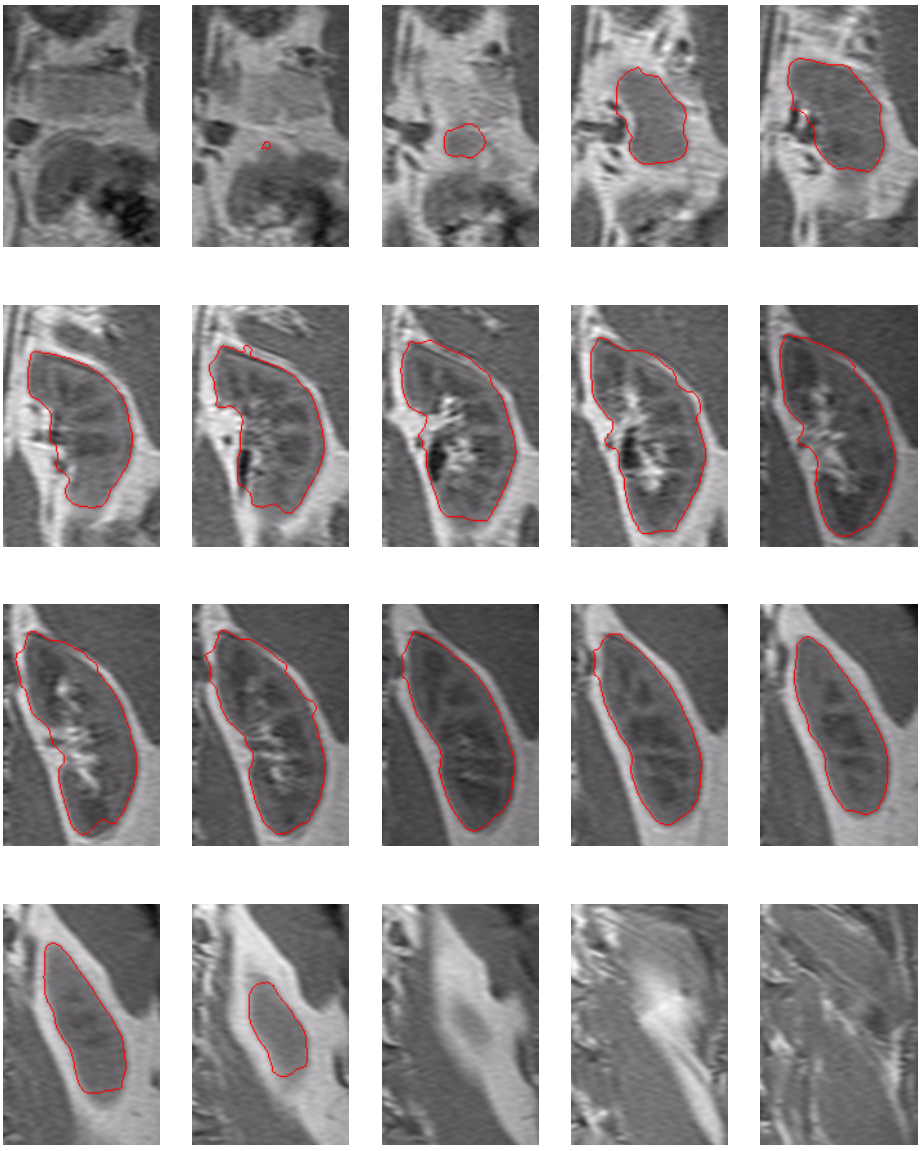


Figure 5-9: Active contours for Case 3 without user interaction

In this case, the opened results from the watershed segmentation were already very good, but the active contours give smoother regions.

- In image 2, the contour has shrunk to an almost insignificant size. This is good since there is actually no kidney present in this image.
- In image 3, the contour has found the boundary of the kidney and placed itself around this. After the opening, this region was a little too big.
- Images 4-17 provide very similar results to the opening of the watershed, but a bit more of the contour has been pulled to the edge of the spleen, as well as a strong edge in the upper left in images 11-14.
- In image 18, the segmented region has disappeared completely. This was in a way correct because there was no kidney where the contour was located, but there was a part of the kidney higher in the image that had not been found, and it could have been desirable to pull this contour to that area.

Active Contours With User Interaction

With some user interaction, the segmentation can be made as good as perfect. After only 15 user interactions in the form of pulling on the contours, the segmentation result was as shown in Figure 5-10.

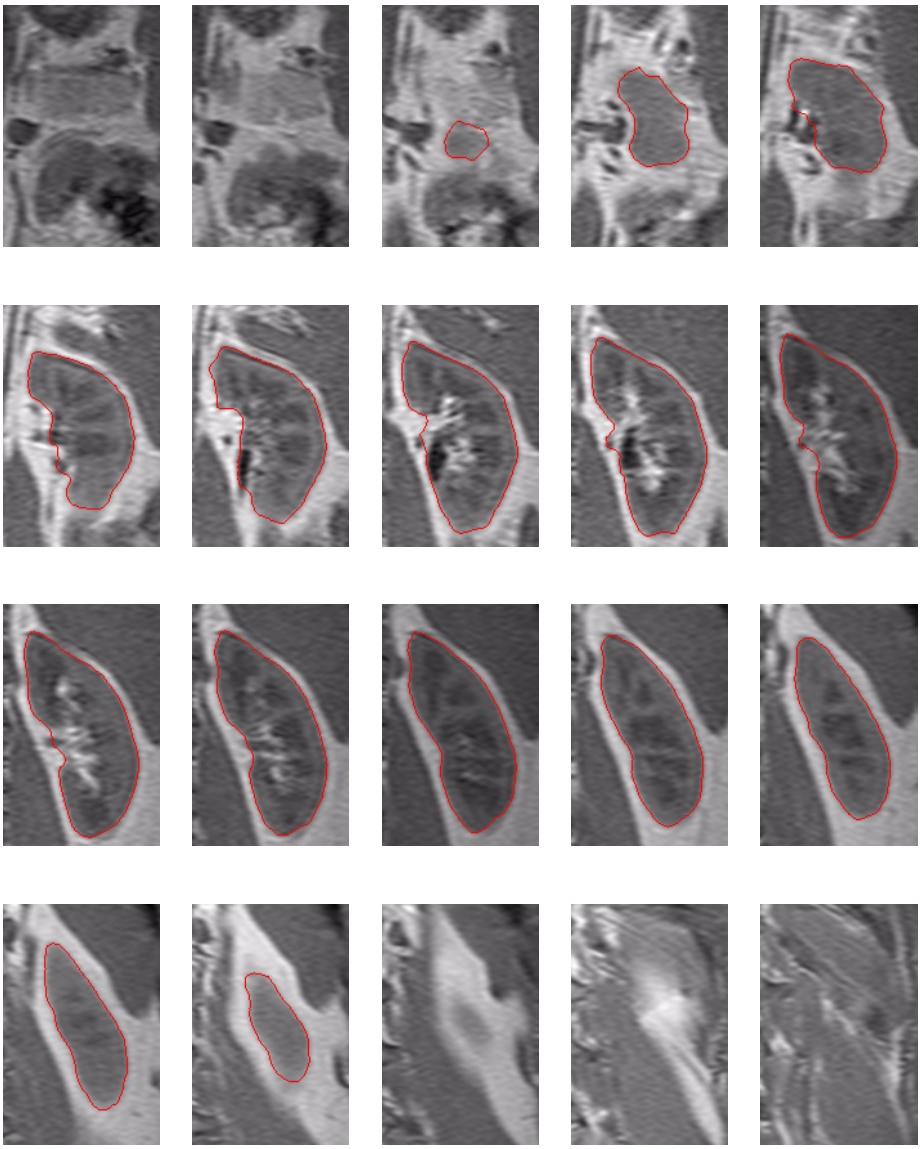


Figure 5-10: Active contours for Case 3 with user interaction

This contour follows the borders of the kidney almost perfectly. In image 2, the small region that was left after the previous step has now been completely removed. Since there was no active contour in image 18 when the user interaction began, it was impossible in the current implementation to include the small part of the kidney which is present there.

Discussion

No manual segmentation was available for this kidney to compare the results against, but from visual observations it is clear that the watershed segmentation did a good job of segmenting this kidney. Each of the steps improved the results further, and the end result, which was obtained in less than 6 minutes with a minimum of user interaction, seems near perfect.

5.4 SUMMARY

The implemented algorithm has been tested with segmentation of three different kidneys, one from a 2-dimensional image, and two from 3-dimensional data volumes. In all tests, the final results were close to perfect with a minimum of user interaction required. In the 2-dimensional case, the final result was acquired after 6 seconds. In the 3-dimensional cases, the complete segmentation took 15-20 seconds per image slice, which is about 10 times faster than manual drawing.

The segmentation algorithm begins with a watershed segmentation. This does a fairly good job, but there tend to be some holes in the kidney, as well as some incorrectly labelled voxels. These incorrect voxels are usually gathered in small clusters that have grown from a neighboring image slice. To reduce this problem, a morphological opening of the watershed result is performed. This removes many of the incorrectly labelled regions, and smoothens the outline of the segmented region somewhat.

The watershed and opening tends to under-segment the kidneys, especially in the renal pelvis area which is difficult to separate from the background. The active contours use the segmented regions after opening of the watershed result for initialization, but the internal forces reduce the effect of the under-segmentation in this step because straight lines are created where there are holes in the watershed result.

When the active contours are free to move on their own, they occasionally lock on to strong, but incorrect, edges that are near the kidney. But in many cases they also move to the correct edges, which improves the result further.

By interacting with the snake, the user can shape the contours as he or she wants, and drag them from incorrect edges to the right ones. But because the initialization from the previous steps is so good, very little user interaction is required to get near perfect results.

CHAPTER 6: CONCLUSIONS AND FUTURE WORK

The task of this project has been to implement a semi-automatic segmentation of kidneys from magnetic resonance images. The implementation will be used in a larger workstation developed at Haukeland University Hospital for kidney examinations. In this setting, the segmentation will be used for volume estimation, image registration and visualization of the kidneys.

The segmentation was implemented as a two step-process. Active contours are used to get the final result, but these require a very close initialization if they are to find the right borders. To accomplish this, a watershed segmentation is performed using markers that separate the kidney from the background. The initial contours are created from the watershed result.

Testing has been performed on real data sets containing slice images of the abdomen and kidneys in particular.

In the following section conclusions will be made for different stages of the algorithm. The chapter also includes suggestions for future work.

6.1 CONCLUSIONS

6.1.1 Watershed Segmentation

A watershed-from-markers algorithm has been implemented to provide an initialization for the active contours. All the user has to do is to draw a frame around the kidney in one of the image slices. A binary image that is read from disk is used to create markers. This image is currently a very rough drawing that resembles a kidney shape. This is one of the strengths of the algorithm. If it is desirable to use the algorithm for segmentation of another body part, or any other structure for that matter, all that is necessary is to make a binary image that roughly resembles the desired structure. The watershed segmentation is very flexible, and the rest of the algorithm follows it’s result.

The watershed segmentation works fast, using approximately two seconds per image slice in 3-dimensional data, and gives fairly good results.

6.1.2 Translation from Watershed to Active Contour

A problem with the watershed segmentation is that the result often has more than one region in each image slice. This makes it impossible to simply trace around the watershed result to get the active contour since there should be only one contour per image.

To get rid of the smallest regions, which are usually the result of noise or other imperfections in the images, a morphological opening is performed on the segmentation results. This improves the results significantly because the small regions may sometimes be placed far from the kidney in some slices, which would cause problems for initialization of the active contours.

But even after the opening, there may still be more regions left in each image. A solution to this problem has been developed where the contours are initialized as rectangles that bound all the remaining regions. The contours are then put under the influence of internal forces and a deflating balloon force that makes them shrink. When vertices reach the segmented regions from the opened watershed segmentation, they are locked in position. The result is that the contours follow the regions that are used for initialization closely, but where the regions are missing, the internal forces make sure the contours get a natural border that binds the regions together.

6.1.3 Active Contour

The active contour is based on the discrete dynamic contour that was presented by Lobregt and Viergever in [25]. It works with one contour for each image slice, but edges to vertices in neighboring slices are used when creating the internal forces. This makes the contours less likely to move away from their correct position.

Because the watershed segmentation provides a good initialization for the active contour, it locates the edges of the kidneys quickly, but in some cases, stronger edges along nearby tissue such as the liver, may attract the contours and pull them away from the desired edge. The user can then interact with the model by dragging the contour back to the correct border.

Some user interaction is usually required to get perfect results, but on average, about two interactions per image slice are enough.

6.1.4 Overall

The implemented model is of great help for the segmentation of kidneys. Manual segmentation requires hours of intense concentration from the user, and can cause health problems because it is very static work. With this algorithm, the user only has to supervise to process while the computer does the work. Instead of painstakingly drawing the exact contours, he or she just has to perform a few operations with the mouse. In addition to being easier, the automatic segmentation is also about 10 times faster than manual drawing.

A similar approach of using watershed segmentation to initialize active contours has been presented by Lapeer, Tan and Aldridge in [40]. In their model, the user had to place a number of markers both inside and outside of the desired structure to initialize the watershed. With this implementation, it is enough to draw a rectangle in one image that frames the kidney. Their method for converting the watershed result into active contours was also different from the one implemented here.

6.2 FUTURE WORK

Some suggestions that may improve the system further are:

- Using multi-spectral data. Both the watershed segmentation and the active contour can probably work better if data from more image series are used. This requires that the different series are registered, which is a difficult problem in itself.
- 3-dimensional implementation of active contours. If the active contours work directly in 3D, it would be easier to find the globally best solutions which could give better results.
- Use of statistical information. When more MR-examinations have been performed, and kidneys have been segmented from these, information about expected shape, gray level, texture, etc. should be extracted and used to guide the segmentation.
- Automatic placement of kidney. If the algorithm can locate the area of the kidney automatically, more user interaction can be saved because the user will no longer have to draw the initial framing rectangle. This automatic placement could be done using statistical information as described above.

APPENDIX A: KIDNEY ANATOMY AND FUNCTION

The kidneys are two bean-shaped organs that are responsible for cleansing the blood and keeping the conditions in the body stable. The following is based on [48][49][50].

A.1 PLACEMENT

The two kidneys are located near the middle of the back, just below the rib cage. The right kidney is usually slightly lower than the left because of pressure from the liver.

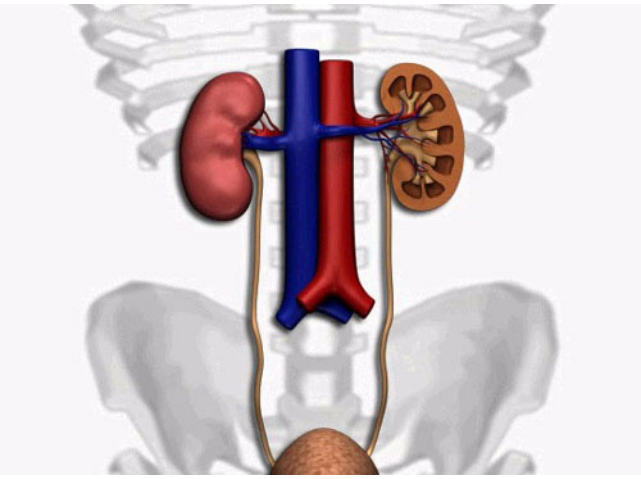


Figure A-1: Kidney placement [51]

Figure A-1 shows the placement of the kidneys on either side of the descending artery and inferior vein. The ureters go from the kidneys to the urinary bladder in the bottom of the figure.

A.2 ANATOMY

Each kidney is approximately 11cm long, 5.5cm wide and 3.5cm thick, weighing 150grams. The internal structure is shown in Figure A-2.

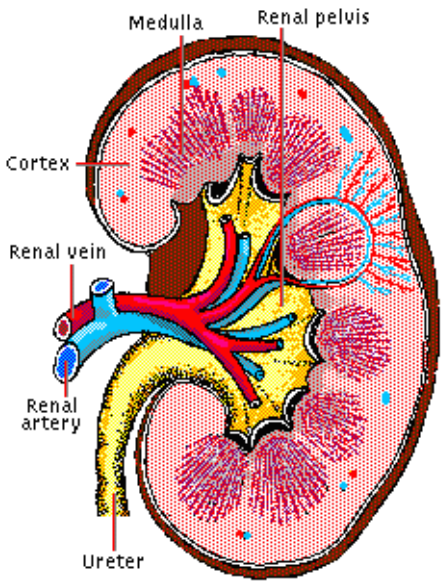


Figure A-2: Kidney anatomy [52]

The cortex is the outermost part of the kidney. More towards the center is the medulla, which contains 10-15 pyramids that consist of 100 000 nephrons each that are used for filtering the blood. Each nephron consists of a glomerulus which is located in the top of the pyramids, and a complex tube system that goes from the glomerulus to the loop of Henley in the bottom of the pyramids. It then goes back to the top, before ending up in the renal pelvis through collecting ducts.

A.3 FUNCTION

Every minute, 20% of the blood passes through the kidneys. In the glomeruli, all of the blood plasma except for the plasma proteins is filtered into the tube system. This means that every day, 180 liters of fluids are filtered out of the blood together with waste materials as well as useful substances. In the tubes of the nephrons, 178.5 liters of water and most of the useful substances are reabsorbed by the blood. The rest gathers in the renal pelvis and follows the ureter to the bladder. To get rid of all the waste materials that build up, the body needs to extract at least 0.5 liters of urine every day, even if no water is introduced.

APPENDIX B: REFERENCES

1. *O. Haraldsen.* ***MR.*** *Lecture in Medicine for Non-Medical Students. Norwegian University of Science and Technology (2004).*

2. ***Prosjektplan: MR nyrer med funksjonell avbildning - en tverrfaglig satsning.*** *Haukeland University Hospital (2004).*

3. *H.J. Smith and K.I. Gjesdal.* ***Magnetisk resonans - historikk og teoretisk grunnlag.*** *Tidsskrift for Den Norske Lægeforening. Vol. 120 (2000) pp. 931-935.*

4. *J.P. Hornak.* ***The Basics of MRI.*** *http://www.cis.rit.edu/htbooks/mri/ (june 2005)*

5. *V.S. Lee, H. Rusinek, M.E. Noz, P. Lee, M. Raghavan and E.L. Kramer.* ***Dynamic Three-dimensional MR Renography for the Measurement of Single Kidney Function: Initial Experience.*** *Radiology. Vol. 227 (2003) pp. 289-294*

6. *M. Sonka, V. Hlavak and R. Boyle.* ***Image Processing, Analysis, and Machine Vision.*** *2nd edition. Brooks/Cole Publishing Company (1999). ISBN 0-534-95393-X.*

7. *J.D. Yang, Y.S. Chan and W.H. Hsu.* ***Adaptive Thresholding Algorithm and its Hardware Implementation.*** *Pattern Recognition Letters. Vol. 15 (1994) pp. 141-150.*

8. *P. Soille.* ***Morphological Image Analysis.*** *2nd edition. Springer (2002). ISBN 3-540-42988-3.*

9. *J.F. Canny.* ***A Computational Approach to Edge Detection.*** *IEEE transactions on pattern analysis and machine intelligence. Vol. 8 (1986) pp. 679-698.*

10. *P.V.C. Hough.* ***Methods and Means for Recognizing Complex Patterns.*** *US Patent 3,069,654 (1962).*

11. *A.X. Falcao, J.K. Udupa, S. Samarasekera, S. Sharma, B.E. Hirsh and R.D.A. Lotufo.* ***User-Steered Image Segmentation Paradigms: Live Wire and Live Lane.*** *Graphical Models and Image Processing. Vol. 60 (1998) pp. 233-260.*

12. *R.J. Frank.* ***Optimal Surface Detection Using Multi-Dimensional Graph Search: Applications to Intravascular Ultrasound.*** *Master thesis, University of Iowa (1996).*

13. *S.L. Horowitz and T. Pavlidis.* ***A Graph-Theoretic Approach to Picture Processing.*** *Computer Graphics and Image Processing. Vol. 7 (1978) pp. 282-291.*

14. J.B.M. Roerdink and A. Meijster. **The Watershed Transform: Definitions, Algorithms and Parallelization Strategies**. Fundamenta Informaticae. Vol. 41 (2001) pp. 187-228.

15. P. Soille and L. Vincent. **Watersheds in Digital Spaces: An Effective Algorithm Based on Immersion Simulations**. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 13 (1991) pp. 583-598.

16. G. Lin, U. Adiga, K. Olson, J.F. Guzowski, C.A. Barnes and B. Roysam. **A Hybrid 3D Watershed Algorithm Incorporating Gradient Cues and Object Models for Automatic Segmentation of Nuclei in Confocal Image Stacks**. Cytometry Part A. 56A(2003) pp. 23-36.

17. F. Meyer and S. Beucher. **Morphological Segmentation**. Journal of Visual Communication and Image Representation. Vol. 1 (1990) pp. 21-46.

18. L. Vincent. **Morphological Grayscale Reconstruction in Image Analysis: Applications and Efficient Algorithms**. IEEE Transactions on Image Processing. Vol. 2 (1993) pp. 176-201.

19. F. Meyer. **Un algorithme optimal de ligne de partage des eaux**. Reconnaissance des Formes et Intelligence Artificelle, 8e congrés. Vol. 2 (1991) AFCET pp. 847-857.

20. R. Lotufo and A. Falcao. **The Ordered Queue and the Optimality of the Watershed Approaches**. Mathematical Morphology and Its Applications to Image and Signal Processing (J. Goutsias, L.Vincent and D.S. Bloomberg, eds.), Kluwer Academic Publishers (2000).

21. P. Felkel, M. Bruckschwaiger and R. Wegenkittl. **Implementation and Complexity of the Watershed-from-Markers Algorithm Computed as a Minimal Cost Forest**. Computer Graphics Forum, Vol. 20 (2001) pp. 26-35.

22. M. Kass, A. Witkin and D. Terzopoulos. **Snakes: Active Contour Models**. Internal Journal of Computer Vision. Vol. 1 (1987) pp. 321-331.

23. T. McInerney and D. Terzopoulos. **Deformable Models**. Chapter in Handbook of Medical Imaging: Processing and Analysis (I. Bankman ed.), Academic Press (2000).

24. J.V. Miller, D.E. Breen and M.J. Wozny. **Extracting Geometric Models Through Constraint Minimization**. Proceedings to the First IEEE Conference on Visualization (Visualization '90), (1990) pp. 74-82.

25. S. Lobregt and M. Viergever. **A Discrete Dynamic Contour Model**. IEEE Transactions on Medical Imaging. Vol. 14 (1995) pp. 12-24.

26. L.D. Cohen. ***On Active Contour Models and Balloons****. Computer Vision, Graphics, and Image Processing: Image Understanding. Vol. 53 (1991) pp. 211-218.*

27. L.D. Cohen and I. Cohen. ***Finite-Element Methods for Active Contour Models and Balloons for 2-D and 3-D Images****. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 15 (1993) pp. 1131-1147.*

28. L.D. Cohen. ***Deformable Surfaces and Parametric Models to Fit and Track 3D Data****. IEEE International Conference on Systems, Man, and Cybernetics. Vol. 4 (1996) pp. 2451-2456.*

29. C. Xu and J.L. Price. ***Snakes, Shapes, and Gradient Vector Flow****. IEEE Transactions on Image Processing. Vol. 7 (1998) pp. 359-369.*

30. C. Xu and J.L. Price. ***Gradient Vector Flow Deformable Models****. Chapter in Handbook of Medical Imaging (I. Bankman ed.), Academic Press (2000) pp. 159-169.*

31. H. Heuch. ***Segmentation of the Liver from MR and CT images****. Master Thesis, Norwegian University of Science and Technology (2003).*

32. Y. Boykov, V.S. Lee, H. Rusinek and R. Bansal. ***Segmentation of Dynamic N-D Data Sets via Graph Cuts using Markov Models****. Medical Image Computing and Computer-Assisted Intervention (MICCAI), LNCS 2208 (2001) pp. 1058-1066.*

33. Y. Sun, M.P. Jolly and J.M.F. Moura. ***Integrated Registration of Dynamic Renal Perfusion MR Images****. IEEE International Symposium on Image Processing, Singapore. (2004).*

34. A.J. Huang, V.S. Lee and H. Rusinek. ***Functional Renal MR Imaging****. Magnetic Resonance Imaging Clinics of North America. Vol. 12 (2004) pp. 469-486.*

35. J.A. de Priester, A.G.H. Kessels, E.L.W. Giele, J.A. den Boer, M.H.L. Christiaans, A. Hasman and J.M.A. van Engelshoven. ***MR Renography by Semiautomated Image Analysis: Performance in Renal Transplant Recipients****. Journal of Magnetic Resonance Imaging. Vol. 14 (2001) pp. 134-140.*

36. Y. Sun, J.M.F. Moura, D. Yang, Q. Ye and C. Ho. ***Kidney Segmentation In MRI Sequences Using Temporal Dynamics****. Proceedings to 2002 IEEE International Symposium on Biomedical Imaging. (2002) pp. 98-101.*

37. *R.M. Summers, C.M.L. Agcaoili, M.J. McAuliffe, S.S. Dalai, P.J. Yim, P.L. Choyke, M.M. Walther and W.M. Linehan.* ***Helical CT of von Hippel-Lindau: Semi-Automated Segmentation of Renal Lesions****. Proceedings to 2001 International Conference on Image Processing. Vol. 2 (2001) pp. 293-296.*

38. D. Freedman, R.J. Radke, T. Zhang, Y. Jeong, D.M. Lovelock and G.T.Y. Chen. ***Model-Based Segmentation of Medical Imagery by Matching Distributions****. IEEE Transactions on Medical Imaging. Vol. 24 (2005) pp. 281-292.*

39. D.C. Barber and D.R. Hose. ***Automatic Segmentation of Medical Images Using Image Registration: Diagnostic and Simulation Applications****. Journal of Medical Engineering & Technology. Vol. 29 (2005) pp. 53-63.*

40. R.J. Lapeer, A.C. Tan and R. Aldridge. ***Active Watersheds: Combining 3D Watershed Segmentation and Active Contours to Extract Abdominal Organs from MR Images****. Proceedings to 5th International Conference on Medical Image Computing and Computer-Assisted Intervention. Vol. 2499 (2002) pp. 596-603.*

41. R. Boscolo, M.S. Brown, M.F. McNitt-Gray. ***Medical Image Segmentation with Knowledge-Guided Robust Active Contours****. RadioGraphics. Vol. 22 (2002) pp. 437-448.*

42. B. Tsagaan, A. Shimizu, H. Kobatake and K. Miyakawa. ***An Automated Segmentation Method of Kidney Using Statistical Information****. Proceedings of Medical Image Computing and Computer Assisted Intervention. Part I (2002) pp. 556-563.*

43. F. Gibou, D. Levy, C. Càrdenas, P. Liu and A. Boyer. ***Partial Differential Equations Based Segmentation for Radiotherapy Treatment Planning****. Mathematical Biosciences and Engineering. Vol. 2 (2005) pp. 209-226.*

44. E.R. Ree. ***ROI Manager****. Thesis, research project at Norwegian University of Science and Technology, Department of Computer and Information Science. (2004).*

45. S. Johannessen and P.M. Joyce. ***Model Based Segmentation, Applications to CT and MR Images of the Liver****. Master Thesis, Norwegian University of Science and Technology. (2004).*

46. D. MacDonald, N. Kabani, D. Avis and A.C. Evans. ***Automated 3-D Extraction of Inner and Outer Surfaces of Cerebral Cortex from MRI****. NeuroImage. Vol. 12 (2000) pp. 340-356.*

47. X. Han, C. Xu and J.L. Prince. ***A Topology Preserving Level Set Method for Geometric Deformable Models****. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 25 (2003) pp. 755-768.*

48. J.G. Bjålie, E. Haug, O. Sand, Ø.V. Sjaastad and K.C. Toverud. ***Menneskekroppen, Fysiologi og Anatomi****. 5.opplag. Gyldendal Norsk Forlag (2003). ISBN 82-00-41831-6*

49. *L. Bjørnsen and P. Hermansen.* ***Nyrenes sirkulasjon, morfologi og funksjon visualisert med MR****. Særoppgave ved Radiologisk seksjon, Institutt for kirurgiske fag, Haukeland Universitetssykehus (2004).*

50. *National Kidney and Urologic Diseases Information Clearinghouse.* ***Your Kidneys and How They Work****. http://kidney.niddk.nih.gov/kudiseases/pubs/yourkidneys/ (february 2005).*

51. *Fox River Watch.* ***PCBs and Kidney, Bladder and Urothelial Cancers****. http://www.foxriverwatch.com/kidney_cancer_PCBs_menu.html (february 2005).*

52. *J.H. Southard.* ***Experimental Organ Preservation****. http://research.surgery.wisc.edu/southard/ (february 2005).*