

Foreword

I would like to thank my supervisor, Roger Midtstraum, for his helpful advice during the course of this work.

Summary

In this thesis the image segmentation system EDISON, was tested against an automatic version snake, which is an algorithm active contour models. The algorithms were tested against each to see if an automatic snake algorithm could be feasible for use in an image database for shape extraction.

The conducted tests showed that EDISON yielded the best results, and that snake should be given further work before being considered.

Contents

1	Introduction	1
1.1	Vision	1
1.1.1	Background	1
1.1.2	The issues	2
1.1.3	The case for QBE	2
1.2	The process of image-retrieval in an image database	2
2	Image segmentation	5
2.1	General background	5
2.2	Different types of segmentation	5
2.3	Threshold based segmentation	6
2.3.1	Automatically assigning threshold	8
2.3.2	Adaptive methods	8
2.3.3	Optimal thresholding	8
2.4	Edge based segmentation	9
2.5	Active contour models	9
2.5.1	Kass snakes	10
2.5.2	Discretising	13
2.5.3	Dynamic contour	13
2.6	Gradient Vector Flow	14
2.6.1	GVF snakes	15
3	Implementation	17
3.1	Test environment	17
3.2	The algorithms	17
3.2.1	EDISON-implemenation	18
3.2.2	Snake-implementation	18
3.3	Other software	19
3.3.1	Image retrieval from database	19
3.3.2	Support software	19

4	Test of segmentation algorithms	21
4.1	Segmentation - one link in the chain	21
4.2	Proposed segmentation goals	21
4.3	Test motives	22
4.4	Testset	22
4.4.1	Test criteria	23
4.4.2	Hand-crafted solution	24
4.4.3	Evaluating the criteria	24
4.4.4	The images in the testset	25
4.5	Solution set	28
4.5.1	Notes on hand made object masks	28
4.6	Test 1 - EDISON	31
4.6.1	About EDISON	31
4.6.2	Setup of EDISON	31
4.7	Result from test 1	32
4.8	Discussion EDISON	35
4.8.1	Troublesome segmentations	36
4.8.2	Summary EDISON	38
4.9	Test 2 - snake	39
4.9.1	About snake	39
4.9.2	Making snake automatic	39
4.9.3	Setup of snake	39
4.10	Results from test 2	40
4.11	Discussion snake	46
4.11.1	General remarks	46
4.11.2	Identified problems	49
4.11.3	Summary snake	53
5	Conclusion	55
5.1	Conclusive remarks	55

Chapter 1

Introduction

1.1 Vision

1.1.1 Background

The number of digital images stored on computers is increasing rapidly. The use of digital cameras and other digital media, such as mobile phones and camcorders contributes to the growing amount of images. A large number of images introduces another problem, that of retrieving one specific image, or several images belonging to a conceptual category.

In the world of image databases, the ability to automatically acquire metadata derived from an image's spatial properties¹, is highly desirable, since the work involved with manually categorizing images can be tremendous and in practice not feasible. This step is considered to be important to facilitate the notion of content-based indexing, as well as content-based image retrieval.

Current approaches to this concept rely among others upon low-level operations on the pictorial data. Methods pertaining to images, e.g. feature extraction, are known to be computationally expensive. As a side note image processing as a discipline itself, is relatively young. The methods involved are still somewhat immature. Therefore, hybrid solutions which leverage both the benefits of textual annotation and spatial information are indeed interesting.

¹With *spatial properties* we mean the intrinsic properties in an image. which have to be extracted with methods from computer vision, to be indexed upon.

1.1.2 The issues

An issue with a large multimedia database is retrieval. Adjeroh et.al. [2] considers that even though textual keywords is a popular way of annotating multimedia data, they are flawed. Manual annotation has problems beyond being labour intensive. More precisely, humans lay their subjective interpretation of the data in the keywords they attach. For instance an image may symbolize one thing to one person, yet have completely different meaning to the next person. Vocabulary may play a role in this process as well. To help overcome these troubles an automated process of content-based indexing is desirable.

With a growing number of images, there comes an urgent need for image retrieval [26]. In later years, content-based indexing has been used on multimedia data, perhaps on images in particular [2]. Images are analyzed to derive inherent spatial metadata. This process tries to extract low-level features as colour, shape, texture and spatial information. These features can be represented by a feature vector and associated with the image. The nature of the algorithms extracting features make the prospect of exact matches between images poor. Therefore when searching multimedia data, a form for similarity measure is normally used [2]. It is difficult to express spatial properties from images using textual queries. Likewise it is difficult to express semantic features with spatial properties. This can be called an semantic gap, a discrepancy between what the user want, and what can be described.

1.1.3 The case for QBE

Query-by-example is one method which is more intuitive [26]. Describing a desired image in terms of feature vectors would be meaningless and absurd to a human user. Using an image, similar to what one wants as output, a MDBMS can use the feature vectors attached to the example image, and search for images with similar feature vectors in the database. Searching for similar vectors can be a problem in its own right. The vectors can be fairly long, and this high dimensionality is often not supported natively in databases. Lu [14] is suggested for further reading in that area.

1.2 The process of image-retrieval

To query for results in an image database, using query-by-example can be outlined as shown in figure 1.2. Note however, this process may vary, depending on method used to query for images, and desired output.

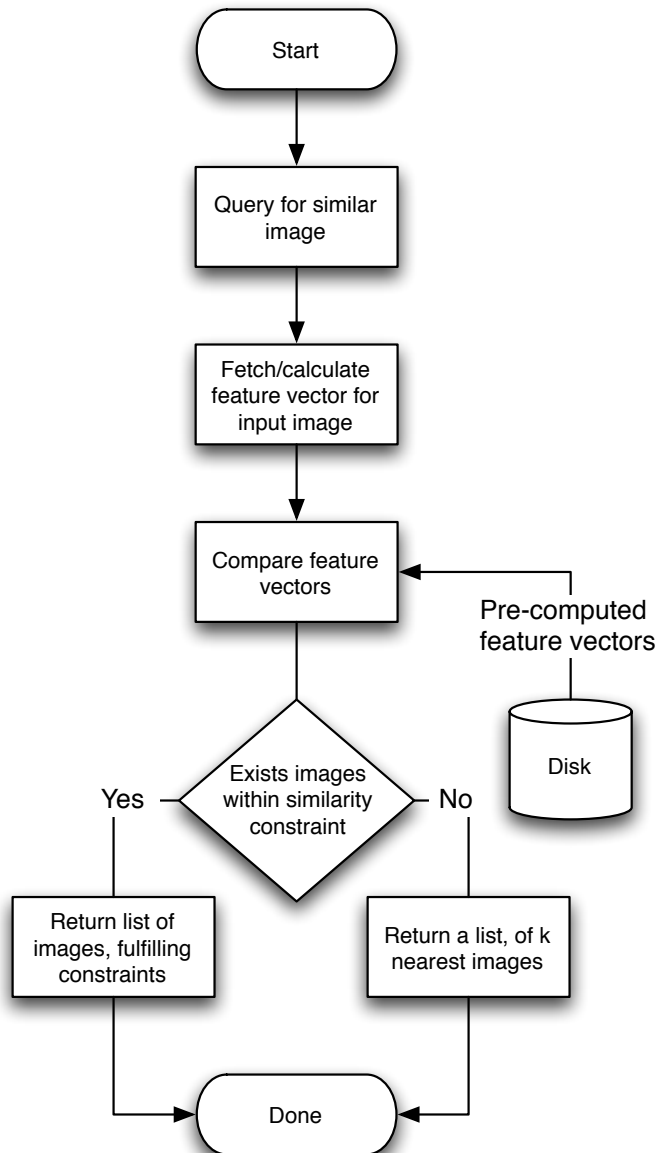


Figure 1.1: Process for an image query

Chapter 2

Image segmentation

This chapter will cover the underlying theory of some segmentation algorithms.

2.1 General background

The objective with segmentation is to apply a label to the pixels. After the segmentation has completed, the pixels will have a label, which effectively divide the pixels into groups that share some spectral property. In the case of image databases, the result is hopefully objects separated from the background of the images.

Segmentation is really one of the first steps on the way to describe objects in images. The process which succeeds image segmentation can be outlined as:

1. Obtaining the shape
2. Describing the shape
3. Devise a way to compare different shapes

Segmentation in general is very hard, and can be a time-consuming operation. One of the struggles is to avoid confusion caused by ambiguity and noise. [22]

2.2 Different types of segmentation

According to [22], segmentation algorithms can roughly be divided into three categories. Threshold based, edge based and region based. In the next sections a short description of the different segmentation approaches will follow.

2.3 Threshold based segmentation

Threshold based segmentation or thresholding can be considered, the technically least complex segmentation approach. First let us consider an image. Images can be viewed as spatial data structures, a matrix containing grey-levels for each pixel in the image. High grey-levels e.g. 255 represents the colour white, whereas value 0 is black. In this discussion only grey images are considered. (Though thresholding can still be applied to colour images after a grayscale conversion.) The underlying idea behind thresholding is that an object comprise grey-levels which are relatively uniform and distinguishable from that of the background. This can be observed in a histogram of the grey-levels, for instance consider figure 2.1. The upper left image shows the original image, the upper right image, shows the histogram of grey-levels. The lower left image, show the threshold version of figure 2.1(a), with a threshold equal to 165, it can be seen that this, divides the image into two groups. And the bowl has been segmented.

The purpose of thresholding, is to separate the object from the background, that is, to extract the object. This is achieved by setting a threshold which divides pixels into object-pixels and background-pixels. Thus the resulting thresholded image is a binary image. In [7] a thresholded image $g(x, y)$ of an image $f(x, y)$ with light objects on a dark background, is defined as:

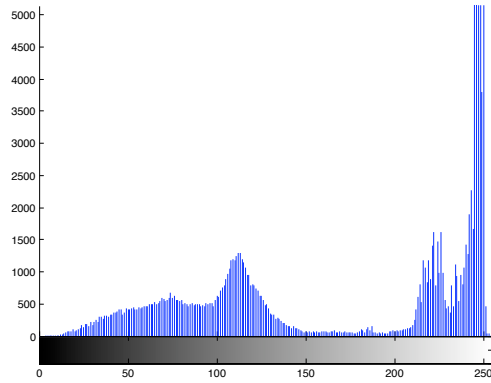
$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T \end{cases} \quad (2.1)$$

By using a single threshold for the entire image, one performs a global thresholding. However often this method is too crude. The same can be said about thresholding in general, still, it remains a popular technique, because of its ease of implementation and speed. The literature [7, 22] list problems such as illumination, which complicates the case of thresholding. Problems arising might be objects not consisting of uniform grey-levels, either as a cause of lighting or some deviations associated with the input-method of the image. This makes the use of a global threshold likely to fail, since different part of the object or image probably need different local thresholds.

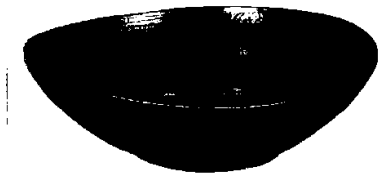
There are more than one technique which can be put to use, to improve the results. For example, the threshold can be adjusted dynamically as the algorithm traverse various parts of the image. There are several ways of optimising the threshold. Many algorithms involve some statistical considerations on the histogram as well as local low-level features in an image.



(a) Original greyscale image



(b) The corresponding histogram of grey-levels



(c) The resulting thresholded image with threshold 165

Figure 2.1: Histogram of grey-levels

2.3.1 Automatically assigning threshold

Gonzalez et.al [7] describe on page 599, of their book the following method for automatically selecting a threshold.

1. Select a starting threshold T , as an estimate.
2. Segment the image in two regions G_1 and G_2 , with this threshold. ($G_1 > T$ and $G_2 \leq T$)
3. Compute average grey-levels, μ_1 and μ_2 for, G_1 and G_2 respectively.
4. Finally, calculate a new threshold $T = \frac{1}{2}(\mu_1 + \mu_2)$. Repeat step 1-4 until the difference between each iteration is below a certain value.

2.3.2 Adaptive methods

Adaptive thresholding is a method which depends not only on threshold, but also spatial location within the image. One approach in this category is to divide an image into a sufficient number of sub-images. This can produce sub-images with bi-modal histograms. A bi-modal histogram is instrumental in obtaining a sound segmentation. Bi-modal can be understood as an image with background of one colour, and one or more objects, of a different colour. This will produce a histogram, with two peaks, one for each class of pixels. The peaks need not be at separate places, depending on the actual colours.

A situation which can occur if a large area of the image or sub-image is dominated by one grey-level or a band of grey-levels, is that the resulting histogram becomes unimodal. This situation may be remedied by further subdivision of the image.

Determination of the actual threshold, can for instance be calculated with the same algorithm as described above.

2.3.3 Optimal thresholding

Another take on threshold estimation, is that of the estimation of the optimal threshold. The method is based on finding the minimal average error in segmentation. The histogram of grey-levels is used as a probability density function. One assumes a known distribution, for which parameters can be calculated and solves the equation which yields the optimal solution. However, as a side-note, this calculation is not trivial per se, and numerical methods must be used.

2.4 Edge based segmentation

Edge based segmentation, involves the use of edge operators [22, 7]. These are operators which give response for sudden changes in grey-values. They give response on change, and consequently, edge operators can be understood as derivatives. The gradient, is often used as an approximation for the derivatives, and therefore, as an edge operator.

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (2.2)$$

The gradient is most often implemented as a spatial mask which is convoluted over the image.

The problem in edge segmentation, can be summarised as generating complete, connected edges. Most edge operators result in partial edges. The problem can sometimes be solved by inspecting the neighbourhood pixels, for matching grey-levels. This can help determine if the edge is a weak one i.e. no supporting edges in the neighbourhood, or a strong one, weak edges are usually discarded.

2.5 Active contour models

Active contour models, popularly called snakes, have gained attention the later years and have been put to use in medical imaging. Then concept of the snake model was introduced by Kass et. al. in [11]. This notion of a snake consist of a parametric contour. The contour is later deformed to encompass the wanted object to be segmented. There are problems associated with the use of snakes however, see [23, 20]. That is, there are some concern regarding the initialisation of the snake contour, and some also with its ability to reach into cavities in image contours. As a parametric structure the snakes have a possibility to be controlled interactively, this has also been a widely used method of snake initialisation. While semiautomatic techniques are the dominating type of snakes, hope is to achieve fully automatic segmentation with snakes.

McInerney and Terzopoulos [16] explain that one reason the snake has been popular in the medical imaging is the use of indigenous constraints in images. Combined with a priori knowledge of the anatomical structures targeted for segmentation, the snake, or shape models could be manipulated with this information.

2.5.1 Kass snakes

The original snake is an energy-minimizing parametric contour, it is deformed by forces which move the contour toward image features as edges [11]. The contour will deform until an energy function 2.4 is minimized. Usually, two forces are included in this function, an internal energy and an external force. The internal energy is related to the snake and controls its stretching and bending. Meanwhile the external force represents the image features, it is responsible for connecting the contour to the image. This results in a snake that is an elastic structure that can bend and form to complex objects in an image. The energy functions must be defined so that the snake reaches a minimum on or near the wanted features in the image.

The contour of the snake in the image plane as represented in [16] is given by

$$\nu(s) = \begin{bmatrix} x(s) \\ y(s) \end{bmatrix} \quad s \in [0, 1] \quad (2.3)$$

The total energy for the snake in the plane is given by¹

$$E_{snake}^* = \int_0^1 E_{snake}(\nu(s)) ds \quad (2.4)$$

$$= \int_0^1 E_{internal}(\nu(s)) + E_{image}(\nu(s)) ds \quad (2.5)$$

Where $E_{internal}$ denotes the internal force controlling the snake, and E_{image} by the force connecting it to the image. The energy of the internal force from the snake, on a point $\nu(s)$ on the contour is stated in equation 2.6. The snake model is inspired by elasticity theory from physics [16].

$$E_{internal}(\nu) = \int_0^1 \frac{1}{2} (\alpha(s) \left| \frac{\partial \nu}{\partial s} \right|^2) + (\beta(s) \left| \frac{\partial^2 \nu}{\partial s^2} \right|^2) ds \quad (2.6)$$

The equation has two parameters $\alpha(s)$ and $\beta(s)$, which control the snake's physical properties, respectively elasticity and bending. Essentially, these parameters are weighting factors, that control how easily the snake should bend, or resist changes. These parameters can be changed to better suit any given situation.

E_{image} should draw the snake to interesting features in the image. This demands that E_{image} has low values for such features in the original image

¹In [11] the total snake energy also contains an external constraint force. This is omitted in the overview here. It can be used to incorporate user guided constraints on the snake energy.

[10]. Edges are especially interesting. Techniques from chapter 2.4 can be employed in the process of creating the image forces. The main intention for doing segmentation is feature extraction and description. Allowing later comparison to other analysed images. In this sense, the main concern will be to separate object from background. So the edges from between the background and object are ideal for image force.

Using edge-operators to enhance existing edges in the image is one of the more common bases for image forces, in the literature of snakes. The image force can be understood as potential function, or like a force field in physics. An area where objects are subject to a force. The force must pull the objects, e.g. the snake to desired positions in the image plane, as mentioned earlier. That translates to defining an expression, which have a minima on these locations, which explains why E_{image} should have small values for interesting image objects.

The range of the force field of E_{image} is important, because it is essentially represents the capture range of this method. If the snake is not initialised within the force field it may take long for the snake to reach an equilibrium, or it may not reach that state at all. Common formulations of E_{image} are:

- $E_{image} = -|\nabla I|^2$
- $E_{image} = -|G_\sigma * \nabla I|^2$

Here I represents the original image, ∇ is the gradient operator. $G_\sigma * I$ represents a convolution on the image I with a Gaussian convolution mask, a blurring operation. The purpose is to attract the snake to high image gradients, in effect, edges in the image. By blurring, the image one can extend the capture range of the force field, and reduce noise in the image. Note also, that it is the magnitude of the gradient that is used in the energy expression. See figure 2.2 for illustrations.

The process of segmenting an image with a snake, is to find a a contour $\nu(s)$ that minimise the energy expression given in equation 2.7. This involves moving and deforming the snake until it stabilises in an energy minima.

$$E(\nu) = \int_0^1 \frac{1}{2}(\alpha(s)|\frac{\partial \nu}{\partial s}|^2) + (\beta(s)|\frac{\partial^2 \nu}{\partial s^2}|^2) + E_{image}(\nu) ds \quad (2.7)$$

The calculus of variation gives the Euler-Lagrange equation shown in 2.8. A minima of $E(\nu)$ in equation 2.7 must adhere to this new equation. Thus, a stable snake must satisfy this equation, since an energy minimum represents the system at rest, a balance between internal and external forces on the snake.

$$-(\alpha(s)\frac{\partial^2 \nu}{\partial s^2}) + (\beta(s)\frac{\partial^4 \nu}{\partial s^4}) + \nabla E_{image}(\nu) = 0 \quad (2.8)$$

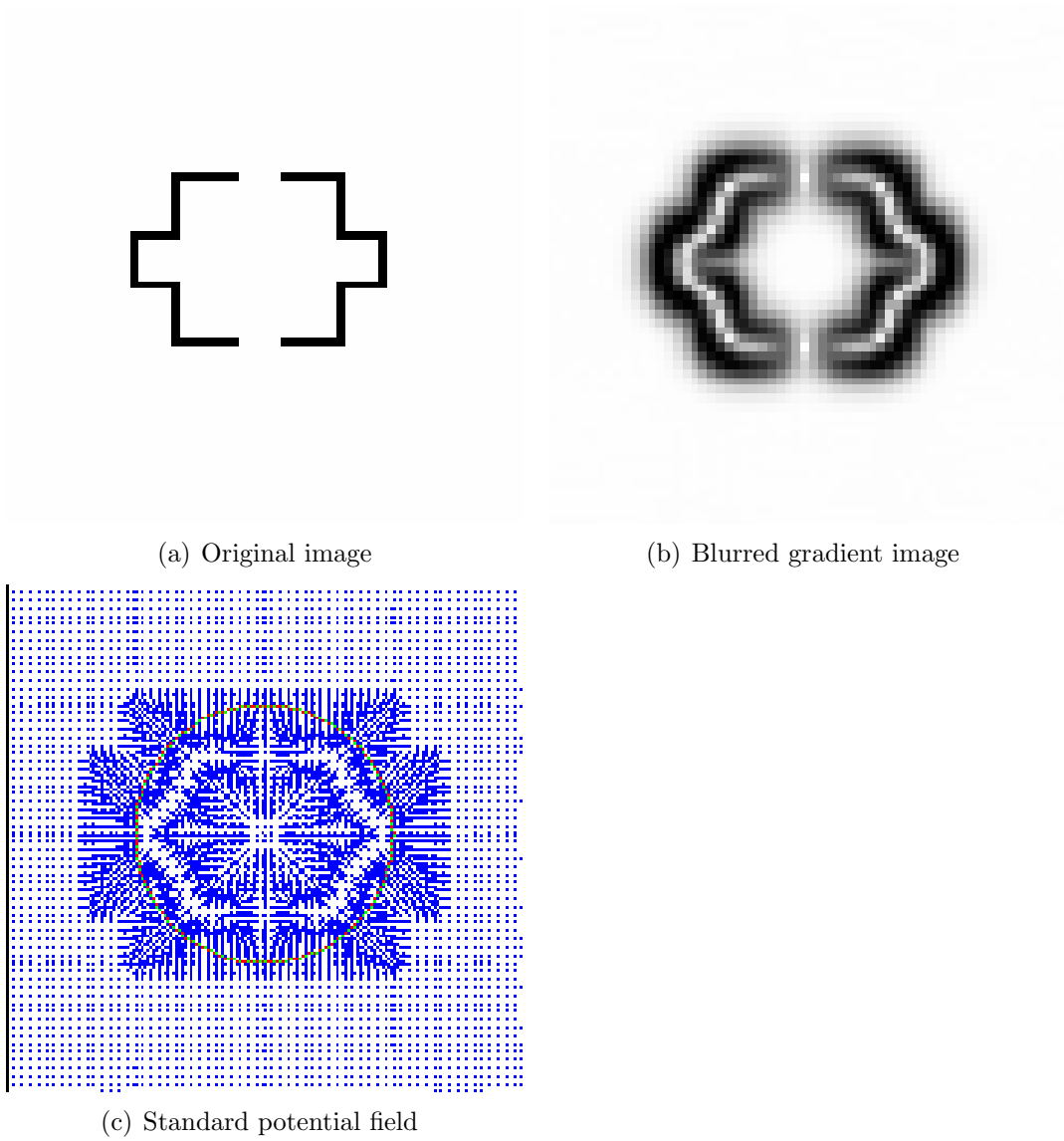


Figure 2.2: Image and standard potential field

2.5.2 Discretising

Digital images are discrete data, and the above expressions must be solved numerically. To approximate the derivatives, one can use finite differences [11, 16]. The snake contour is represented as a vector of parameters. The derivatives become

$$\nu'(s) \approx \nu'(ih) = \nu((i+1)h) - \nu(ih) \quad (2.9)$$

Here h is a step in s , and i is the index, $s = ih$. Further discretisation of the internal energy equation 2.6 with vector notation $\nu_i = (x_i, y_i)^\top = (x(ih), y(ih))^\top$, gives

$$E_{internal} = \alpha_i |\nu_i - \nu_{i-1}|^2 / 2h^2 + \beta_i |\nu_{i-1} - 2\nu_i + \nu_{i+1}|^2 / 2h^4 \quad (2.10)$$

The Euler-Lagrange equation becomes

$$\begin{aligned} & -(\alpha_{i+1}(\nu_{i+1} - \nu_i) - \alpha_i(\nu_i - \nu_{i-1})) \\ & + (\beta_{i+1}(\nu_{i+2} - 2\nu_{i+1} + \nu_i) - 2\beta_i(\nu_{i+1} - 2\nu_i + \nu_{i-1}) + \beta_{i-1}(\nu_i - 2\nu_{i-1} + \nu_{i-2})) \\ & \quad + \nabla E_{image}(\nu) = 0 \end{aligned} \quad (2.11)$$

Representing $\nabla E_{image}(\nu)$ as $\nabla E_{image}(\nu) = (\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}) = (F_x, F_y)$ the Euler-Lagrange equation can be written in matrix form, as two independent equations.

$$Ax + F_x = 0 \quad (2.12)$$

$$Ay + F_y = 0 \quad (2.13)$$

2.5.3 Dynamic contour

A dynamic contour, which is changed iteratively is desirable. By making the contour $\nu(s)$ also dependent on time, this can be achieved. So for a dynamic contour $\nu(s, t)$ the energy can be expressed as

$$\begin{aligned} E(\nu(s, t)) = & \int_0^1 \frac{1}{2} (\mu(s) (\frac{\partial \nu(s, t)}{\partial t})^2 + \gamma(s) |\nu_t|^2 \\ & + \alpha(s) |\frac{\partial \nu(s, t)}{\partial s}|^2 + \beta(s) |\frac{\partial^2 \nu(s, t)}{\partial s^2}|^2) + E_{image}(\nu(s, t)) ds \end{aligned} \quad (2.14)$$

In equation 2.14 $\mu(s)$ and $\gamma(s)$ is the mass density and damping density respectively. The corresponding Euler-Lagrange equation becomes

$$\mu \frac{\partial^2 \nu(s, t)}{\partial t^2} + \gamma \frac{\partial \nu(s, t)}{\partial t} - \alpha \frac{\partial^2 \nu(s, t)}{\partial s^2} + \beta \frac{\partial^4 \nu(s, t)}{\partial s^4} + \nabla E_{image}(\nu(s, t)) = 0 \quad (2.15)$$

We assume the parameters are constant along the contour, we also assume the mass μ is zero. Using the same discretisation approach in section 2.5.2, isolating $x(t)$ and $y(t)$ and solving the equations with matrix-inversion, the solutions are (more details available in [11]):

$$x(t) = (A + \gamma I)^{-1}(\gamma x(t-1) - F_x(x(t-1), y(t-1))) \quad (2.16)$$

$$y(t) = (A + \gamma I)^{-1}(\gamma y(t-1) - F_y(x(t-1), y(t-1))) \quad (2.17)$$

Also a scaling factor is usually included κ , to weigh the external force field.

2.6 Gradient Vector Flow

C. Xu and J.L. Prince highlighted in [24, 23] certain problems with the traditional snake model. More precisely they showed some drawbacks pertaining to the potential forces guiding the snake contour to edges in the images. The issues were related to the standard potential field used as an external/image force in Kass et.al. [11], and described in section 2.5.1. The main issues are initialisation of the snake contour, its convergence properties, and the ability for the contour to reach into cavities.

The original potential force, makes the snake sensitive to initialisation. If the snake is initialised too far away the wanted features, it may not converge to those features at all, or it can converge to an undesirable feature. Xu and Prince note that some work has been done in this area to partially rectify this specific problem, namely the distance potential field. One issue the distance potential field [5] tried to enhance was the capture range, guiding the snake to edges. While it accomplished that one problem remained, to reach into cavities.

Xu et.al. list several proposals in their articles, which try to solve the mentioned problems. They conclude that when partially solving some the problems mentioned above, they introduce others. For instance they look at multiresolution methods, which have solutions for capture range, but they note, are problematic when specifying how the snake should move between resolutions. In the articles regarding gradient vector flow, Xu. et.al. propose a method, solving both the problem of capture range and cavities. The new method consists of a different type of external force, replacing the image forces mentioned in this report. The new force, or force fields are called gradient vector flow (GVF) fields. The new method of energy minimisation is done through diffusion of gradient vectors of the edge map.

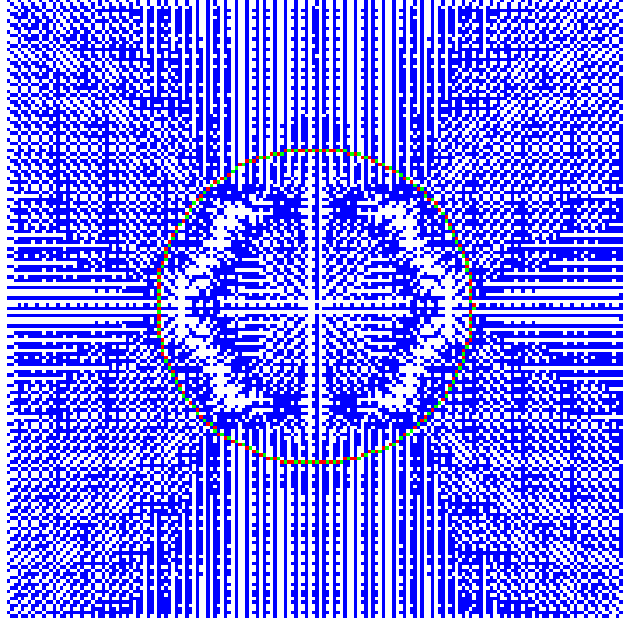


Figure 2.3: GVF field of 2.2(b)

2.6.1 GVF snakes

Xu et.al. call active contours that uses the GVF as an external field, GVF snakes. Since the GVF snake can not be expressed as the traditional Kass snake, the energy-minimisation formulation does not apply. Instead, it is expressed from a force balance condition. In place of the old external force field, a new static potential field is wanted. The theoretical basis for this field comes from the Helmholtz theorem [23].

Equation 2.8 can still be used as the basis for the GVF snake. We can reformulate 2.8 as

$$F_{int} + F_{ext}^{(p)} = 0 \quad (2.18)$$

Replacing the potential external force field with $F_{ext}^{(g)} = \mathbf{g}(x, y)$ gives the equation for the dynamic snake

$$x_t(s, t) = \alpha x''(s, t) - \beta x''''(s, t) + \mathbf{g} \quad (2.19)$$

To obtain a GVF field we need an edge map $f(x, y)$ which contain edge information from the original image $I(x, y)$. If the input image is a grey-level image, suitable definitions are:

$$f(x, y) = |\nabla I(x, y)|^2 \quad (2.20)$$

$$f(x, y) = |\nabla [G_\sigma * I(x, y)]|^2 \quad (2.21)$$

The magnitude of the gradient of the image, or the magnitude of the blurred image, will have the desirable high values for edge features. The gradient is vector consisting of both value and direction. The gradient of an edge map will have vectors in the direction of edges. But like the image force field in connection with the Kass snakes, the vectors only have a large magnitude, in the direct neighbourhood of the edges, resulting in a short capture range. Additionally, areas with solid grey-values, the gradient is zero, and this results in large areas where there is no force, affecting the contour.

To remedy the two shortcomings of capture range, and lack of force in homogenous areas, the new force field is calculated using vector diffusion. This process will spread gradients across the gradient map, effectively extending the reach of the force field. The process will also create vectors pointing into cavities.

In [24] the gradient vector flow field $\mathbf{v}(x)$ is defined as the solution to equation 2.22.

$$\begin{aligned} u_t &= g(|\nabla f|)\nabla^2 u - h(|\nabla f|)(u - \nabla f) \\ u(x, 0) &= \nabla f(x) \end{aligned} \quad (2.22)$$

This yields as a solution to $u(t)$

$$u(t + 1) = u(t) + g(|\nabla f|)\nabla^2 u(t) - h(|\nabla f|)(u(t) - \nabla f) \quad (2.23)$$

Gradient vector flow, can be understood as a diffusion process. In fact it is similar to what goes on in heat conduction, where heat dissipates out in a surface, heating the near surroundings in the surface. Similarly, gradient vector flow, can be seen as a smoothing, where the gradients are smoothed out over the image plane, and as can be seen from the equation above, gradient vector flow, also involves a data term, reenforcing the original edge in the plane, preventing it from being washed out. The equation directly, above, can be seen as consisting of a smoothing term, and a data term.

Chapter 3

Implementation

This chapter will detail aspects of the implementation of the algorithms and other non-essential components used in the testing.

3.1 Test environment

The tests and software used in this thesis, were all run on an Apple Powerbook computer. A brief overview of the technical specifications of the system are as follows.

- Processor: Motorola 1.0 GHz G4
- Memory: 512MB
- OS: Mac OS X
- Java-version: Java J2SE 1.4.2
- Matlab 7.0 R14
- ImageMagick 6 (with PerlMagick)
- Perl 5.8
- Gimp.app 2.0

3.2 The algorithms

Several software packages were used in the tests. Both in the case of EDISON and the snake method, the implementation were based on existing code.

3.2.1 EDISON-implemenation

EDISON can almost be described as a framework for image segmentation. It is a system which performs segmentation with a region based mean shift algorithm. The algorithm was developed at the Robust Image Understanding Laboratory [3] at Rutgers University, by Chris M. Christoudias and Bogdan Georgescu. The theory and methods implemented within EDISON are covered in detail in the following papers [6, 18, 4], which together form the theoretical basis for that system.

The original source code of EDISON can be found on webpage here [3], where it is implemented in C++. In the tests conducted during this thesis a Java-port of the EDISON-system was used. This port was developed by Brian E. Pangburn and Jonathan P. Ayo, it is called jEDISON [19]. The port was compiled, and could be run directly to create region maps of the inputted images.

Supporting software

In addition to the jEDISON bundle, more software was needed to to test EDISON. To carry out the tests of EDISON in the wanted manner, a shell script was created in Perl. This shell script linked into ImageMagick [13], which is a library of image processing functions. The shell script executed EDISON on each image in theset. Furthermore the shell script generate segmentation masks of the outputted region maps. Therefore each region map is examined according to section 4.6.2, and the EDISON is rerun until a satisfying number of objects are identified. Finally a segmentation mask is created.

3.2.2 Snake-implementation

At the core of the snake-implementation, is a framework created by Chenyang Xu and J.L. Prince, in connection to their work on GVF-fields combined with snakes. This framework is available from Xu's and Prince's webpage [25]. This framework covers calculation of snake deformation and GVF force field, as described in section 2.5.

A graphical user interface is used as a wrapper around the snake framework by Xu and Prince. The graphical user interface is based on code from H. Heuch's master thesis. [10]. The code and user interface from Heuch, was originally intended to facilitate semi-automatic segmentation on MR-images of liver slices.

This code has been modified for the purpose of this thesis, to allow for testing of fully automatic snake segmentation. That means that the original

methods used in the program, where one the contour for the current image, was based on the contour from the previous image, is altered. For the purpose of simplicity in testing, the semi-automatic nature of the user interface is retained, where each image is elected manually, however the initialization of the contour is automatic. Further description of the behaviour of the changed program can be found in section 4.9.

3.3 Other software

This section describes other software pieces used in testing.

3.3.1 Image retrieval from database

The tool `SESAM` [1], was used to access the database with images of the artifacts from The Norwegian Folk Museum. This Java-application was developed in connection with J.O. Hauglid's doctoral thesis [9]. `SESAM` was used as-is, to find suitable images for segmentation. The images were saved on the test-computer's harddisc, and used locally from that point on.

3.3.2 Support software

Some pieces of software accompany the actual tested algorithms. These are software pieces developed to make testing more simple, and to help yield comparative results between the different algorithms. Two such extra tools were made.

Image mask tool

This tool (`EdSeg.java`), processes the output from `EDISON`, and converts it to a binary image, with the segmentation mask of the object. The implementation is based on code from Grotan [8].

Conformity percentage tool

The tool (`TestResult.java`), is used to measure to what degree the outputted segmentation mask from the algorithms, match with the segmentation mask from the man made solution. It shares some code with the image mask tool above. Further details can be found in section 4.4.3.

Chapter 4

Test of segmentation algorithms

The chapter contains a test of the snake-algorithm versus the EDISON algorithm

4.1 Segmentation - one link in the chain

Segmentation is an intermediate step in an image-analysis process. As such, it receives data from an earlier step like a preprocessing step, or it just begins to work on raw pictorial data from an image. After the segmentation is finished, the output is forwarded to the next step in the chain of the analysis. It is the overall motive of the analysis, which decides what sort of goals the segmentation should target, therefore the goals will change with the application. The form and nature of the output should also reflect this motive. In the introductory text on segmentation in [7], it is stated that segmentation should be stopped when the applicable objects have been found and separated from the other parts of the image. That is a useful condition in this situation as well.

4.2 Proposed segmentation goals

The current motivation for the image-analysis, is to extract shape. Shape is not an image property, it is an object property. With this in mind, it is clear that the segmentation must yield an object mask, or some measure to successfully separate the object of interest, from the background of the image. In other words, the main object will suffice. The segmentation does not need to discern finer details in the image, such as different regions in object.

It is relevant for the problem at hand, that the shape of the object, with relative ease, can be extracted and calculated from the object, produced by the segmentation. This will help reduce complexity in the representation phase.

Lastly but not least, the quality of the segmentation is important. Several studies have shown that the quality of the segmentation is directly responsible for the quality of the derived search, see [12]. Therefore the desired segmentation algorithm should be as precise as possible, when extracting the object mask.

To sum up, the segmentation goals become:

1. Extract main object of interest
2. Provide easy way of establishing and representing object boundary
3. Accurate segmentation of the object of interest

Often the output from a segmentation operation is a labeled image, where contiguous regions of same texture or spectral properties share the same label. This label is usually a predefined integer. This must be handled when doing shape representation.

4.3 Test motives

The goal of this test is to help determine whether segmentation with automatic snake will be feasible. It can be argued that, if one is to create a better representation of an image, the segmentation and separation of the object, from which the representation is derived, is the most important step. Refer to figure 4.1 for an outline of the process from image to shape query. In [21] a semi-automatic snake-implementation was tested against EDISON, and the results suggested that the snake approach was promising. However for use in an image database, a semi-automatic approach is not practical.

Therefore the main motivation by testing an automatic snake algorithm against a EDISON, is to see whether it can yield better segmentation results, and if it has other properties which might make it more attractive as segmentation algorithm of choice, rather than EDISON.

4.4 Testset

A set of test images from the image database of the Norwegian folk museum is selected to test the applicability of the snake-algorithm versus the segmentation done by EDISON. As the overall goal for this segmentation is to yield a

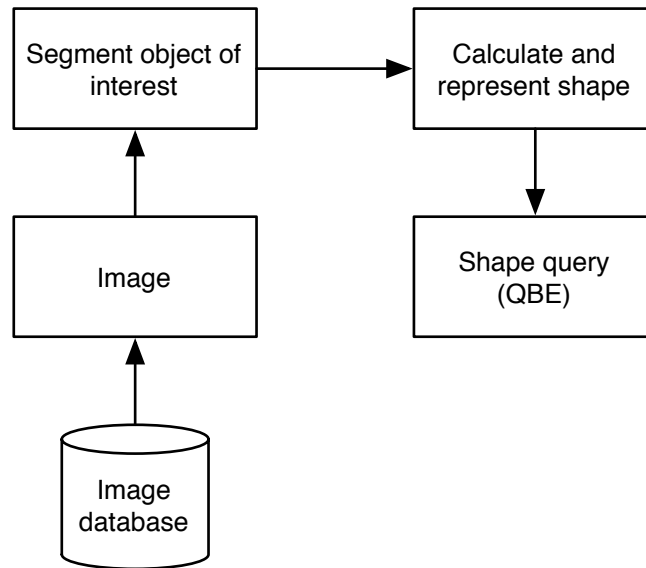


Figure 4.1: Derivation of shape

basis for shape representation, images relevant and suitable for this purpose is selected. The images consists of different types of objects depicted on a neutral background.

A solution will be created for each image by hand, that is a human created, correct segmentation mask. Then afterwards each segmentation algorithm will be measured up against this solution.

4.4.1 Test criteria

In order to determine which algorithm is most suitable for our purpose, segmentation goals 1 and 3 will be tested. Goal 2 needs more discussion, as there are several possibilities this can be achieved. Pros and cons by both methods will be inspected.

Even segmentation goals 1 and 3 can in fact be a matter of controversy. As humans we each have our own view, of what a correct segmentation might be. In this test this is remedied in this test, by using images suitable for shape extraction, in which the main object of interest is clear, and can be segmented by hand.

A numerical measure for the quality of this segmentation, is calculated by the degree the segmentation mask given by the algorithms, conform to

the hand-crafted solution to the segmentation. This action serves a dual-purpose. Firstly we get a numerical measure on the accuracy of the algorithm. Secondly, we reduce the subjectivity, when deciding which algorithm has the best segmentation mask. Nevertheless a qualitative consideration of the segmentation may still be in place.

4.4.2 Hand-crafted solution

This section outlines procedure for attaining correct solution.

1. Each image is loaded into an image-editor
2. An outline is drawn around the object of interest
3. This outline is a closed curve, so that the area it surrounds can be filled with a colour label
4. The outline is filled with a determined colour, which is shared by all these images.
5. The filled outline should be saved as a flattened, binary image, representing a correct segmentation of the image.

This sequence details the step in creating a logical mask, which will become the solution to the given image.

4.4.3 Evaluating the criteria

Checking the test criteria will be done by comparing logical masks from the output of the segmentation algorithm, with the solution, given from the file created above. The logical mask, given by the segmentation algorithm in question, is analyzed pixel by pixel, checking whether it represents a correct solution, as given by the solution mask.

Calculating percentage of conformity

First, the number of object pixels in the solution mask is calculated. Next, the number of pixels in the test mask representing correct object pixels in the solution mask is counted. Then the number of erroneously marked as object pixels in the test mask is counted. This step tests if the object mask in question has object pixels outside the boundaries of the solution mask. That situation is a detriment to the quality of the mask, and should there

be taken into account. Finally a percentage of conformity is calculated by the following formula:

$$\frac{(\text{\#correct pixels}) - (\text{\#erroneous pixels})}{(\text{\#object pixels})}$$

This will provide a measure, as to how well an object mask conforms to the solution mask. However it should be noted that this measure, is just a measure, it serves as a tool to bring objectiveness into the the testing process. It is not a perfect measure for match between the masks, this is not desirable, as there are some uncertainty in the calculations, as the the solution is hand-crafted.

4.4.4 The images in the testset

Table 4.1: The testset of images

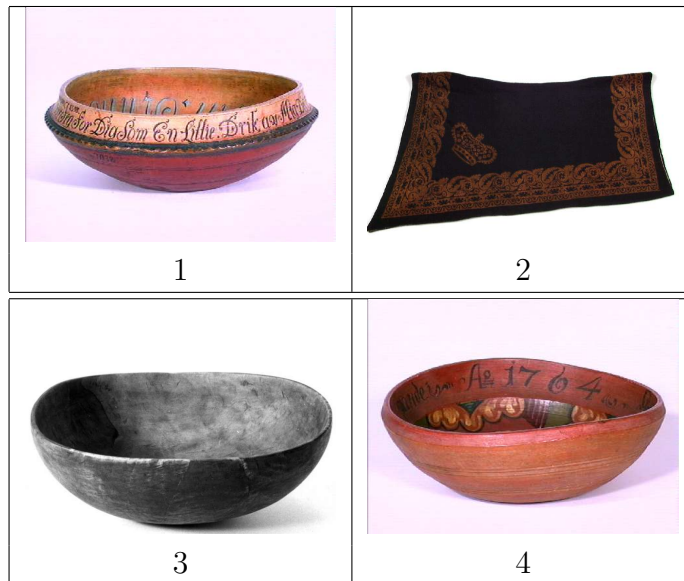


Table 4.1: (continued)



Table 4.1: (continued)

 <p>11</p>	 <p>12</p>
 <p>13</p>	 <p>14</p>
 <p>15</p>	 <p>16</p>
 <p>17</p>	 <p>18</p>
 <p>19</p>	 <p>20</p>

Table 4.1: (continued)

19	20
----	----

4.5 Solution set

The hand-crafted objects masks, which will serve as our solution material can be found in table 4.2

4.5.1 Notes on hand made object masks

These masks were created by the author, according to section 4.4.2. As these masks are hand made, the object masks will not be completely perfect. They are however, to the best of the authors ability, as accurate as possible. As such they should still be usable as solution material. The point where error is likely introduced is around the borders, however the discrepancy is likely so small, that it will not affect the calculations notably.

Table 4.2: The hand-made object masks

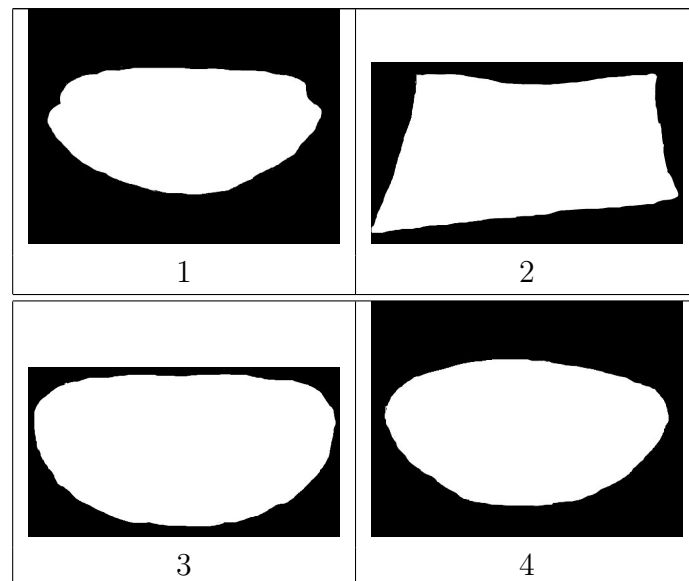


Table 4.2: (continued)

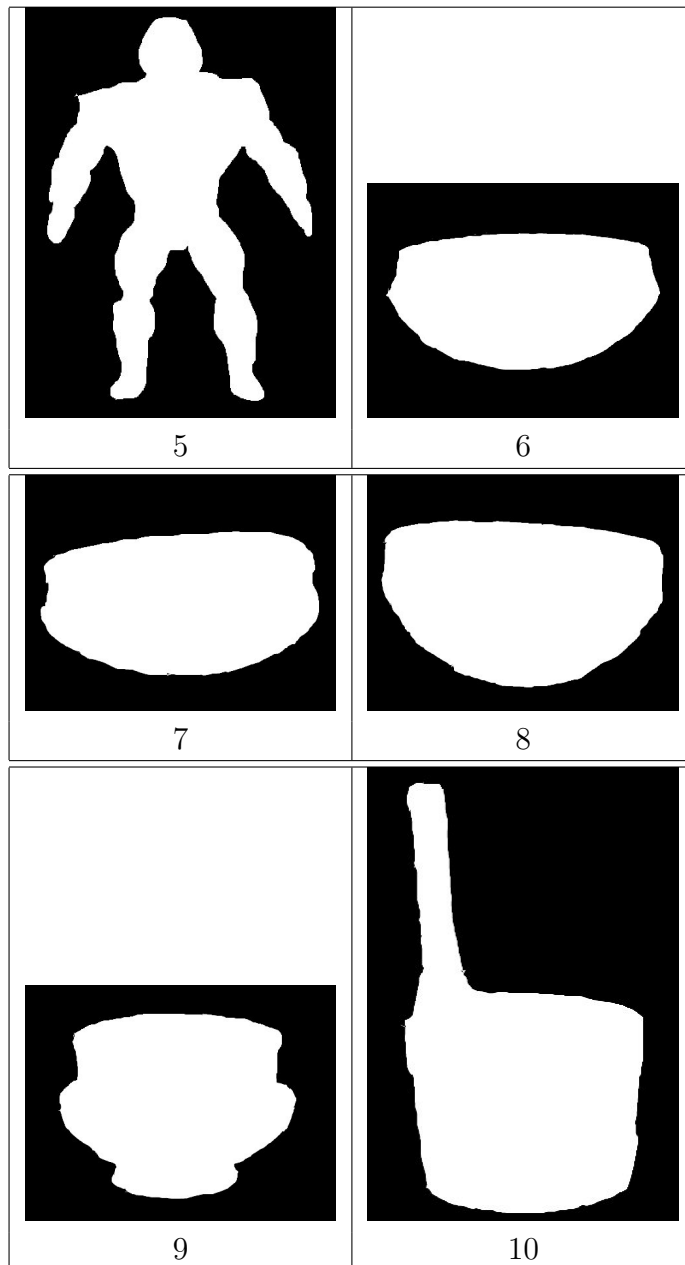


Table 4.2: (continued)

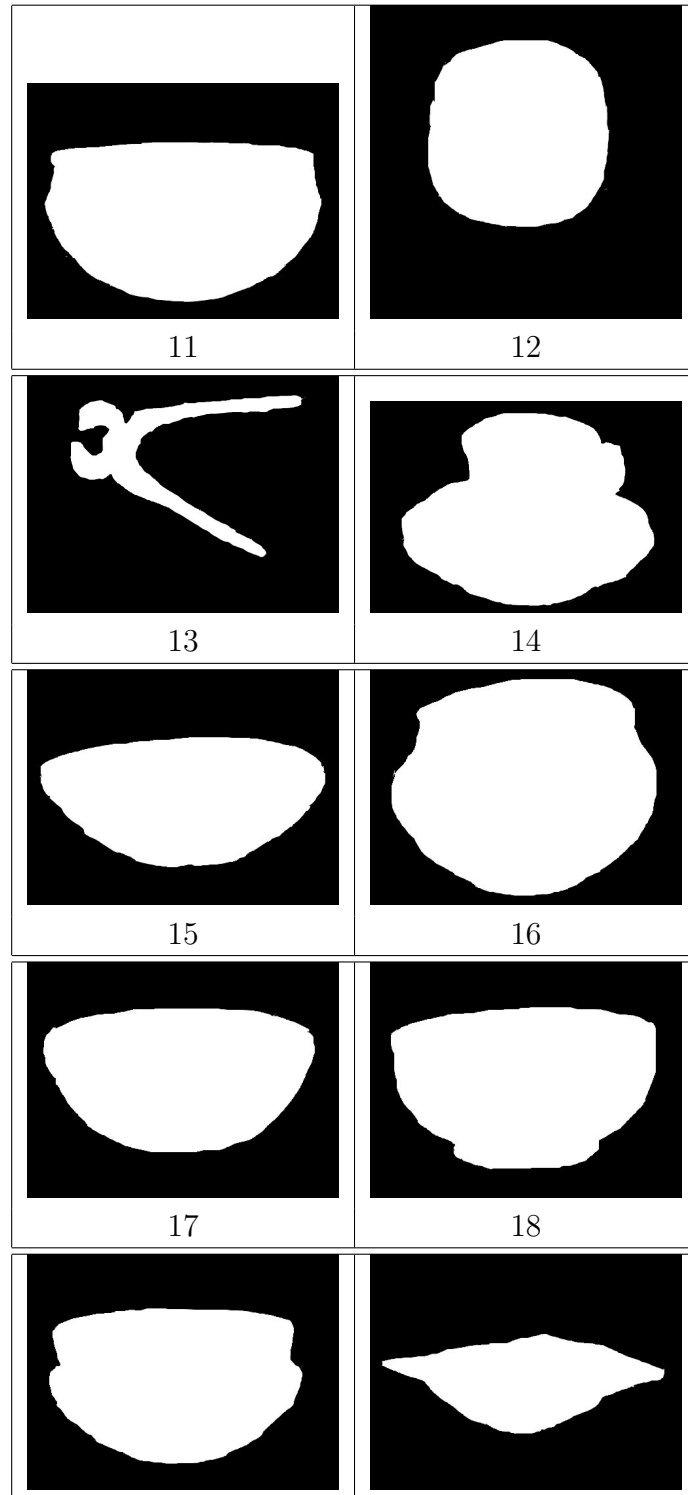


Table 4.2: (continued)

19	20
----	----

4.6 Test 1 - EDISON

The first step in testing the applicability of the snake algorithm for shape extraction, is to obtain a basis of comparison. In previous work in the field of content-based image retrieval, see Hauglid [9] and Grøtan [8], the algorithm EDISON was tested and found to yield the best segmentation results. Therefore to establish a comparative basis, the snake-algorithm will be tested along with EDISON.

4.6.1 About EDISON

EDISON is an image segmentation system consisting of several segmentation techniques. These techniques are covered in the following papers by Comaniciu [6] and Meer et.al. [18]. At the heart of the EDISON is the mean shift-algorithm. For in-depth details about this approach, please consult the above mentioned papers.

4.6.2 Setup of EDISON

Since there exist previous work, where EDISON have been used on images from the PRIMUS database¹, these interim results will be exploited in this test. Hauglid [9] tuned EDISON on a subset of images from the PRIMUS database. In this work, a scheme were used, where particularly one parameter of EDISON was tweaked. The parameter is `minRegion` or `MiniumRegionArea`, from `jEDISON` and `EDISON` respectively. This parameter controls how large a region must be, in order to be consider a region by the algorithm. The goal is to separate at least one object from the background.

In the test `minRegion` was changed by the following regime

1. Initial value of `minRegion` = 15%
2. If the current percentage did not at least yield one one object, it is reduced by 50%
3. Repeat percentage decrease, until one object is separated from background during segmentation

¹PRIMUS is the name of the database containing the images from the Norwegian folk museum

The other parameters are:

- `colorRadius = 6.5`
- `spatialRadius = 7`

These two parameters are from the Java edition, `jEDISON`, they were kept constant while `minRegion` was changed.

4.7 Result from test 1

The object masks produced by `EDISON` are included in table 4.3. Also the percentage of each masks conformity against the solution is given in table 4.4. The percentages are rounded up to nearest whole percentage.

Table 4.3: The object masks produces by `EDISON`

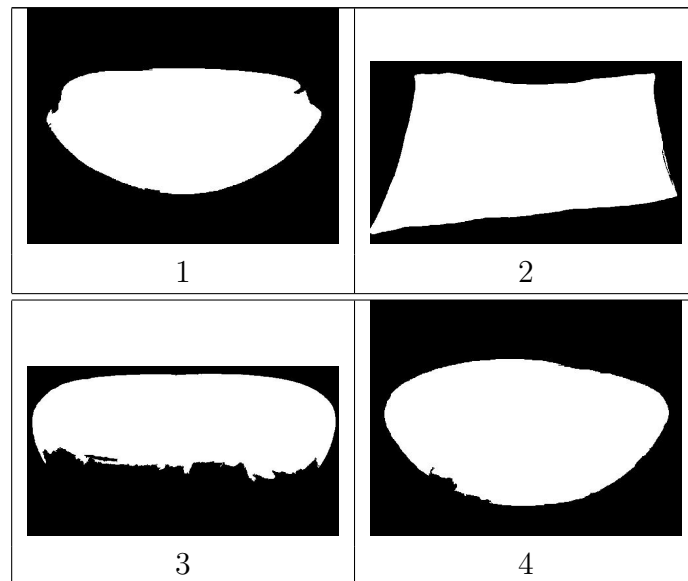


Table 4.3: (continued)

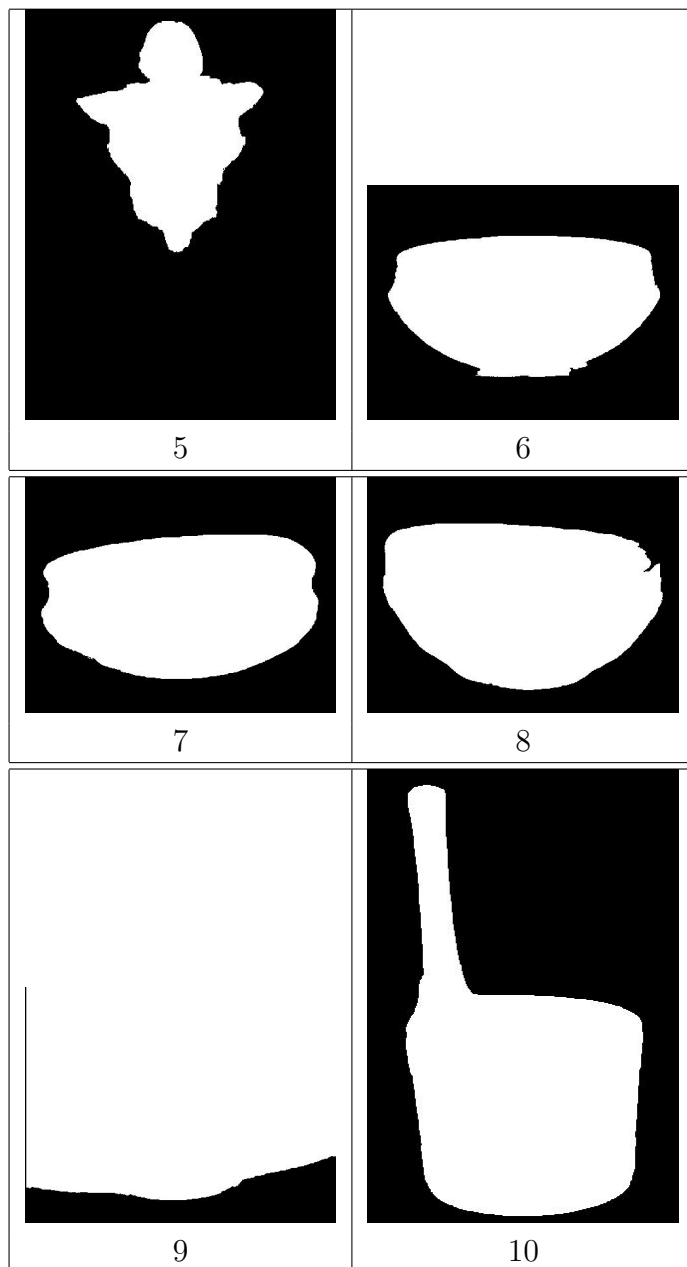


Table 4.3: (continued)

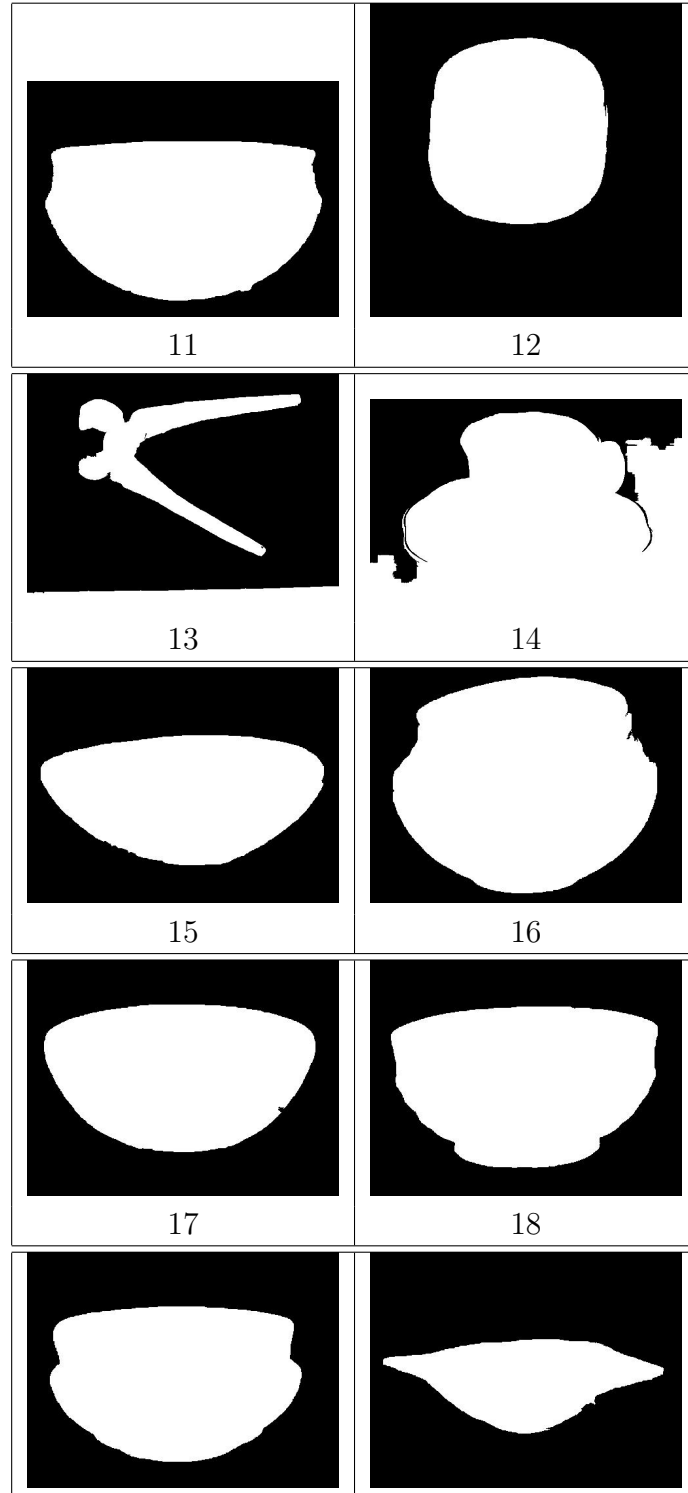


Table 4.3: (continued)

19	20
----	----

Name	%	Correct pixels	Error pixels	total pixels
TS1	97%	58303	79	60108
TS2	98%	145951	126	148983
TS3	65%	100697	265	154952
TS4	97%	69722	136	71842
TS5	48%	26639	82	54839
TS6	97%	66340	965	67410
TS7	98%	73531	170	75074
TS8	97%	82211	123	84732
TS9	16%	75482	63105	75708
TS10	97%	50739	106	52131
TS11	98%	80457	150	81734
TS12	98%	63843	48	65091
TS13	0%	27760	29349	31042
TS14	57%	136631	56597	139600
TS15	97%	61426	191	62915
TS16	98%	100125	76	102533
TS17	98%	69131	533	69962
TS18	98%	76482	189	78134
TS19	98%	70728	88	71914
TS20	94%	32731	117	34742

Table 4.4: Percentage object mask conformity

4.8 Discussion EDISON

The result from running EDISON on the test, show that the majority of the produced object masks conform to the hand-made solution rather well. The percentage of conformity between the solution and the test result is shown in table 4.4, for each image. It is obvious that the algorithm works well on the type of images selected for testing, images which are inherently suitable for shape extraction. Most of the of the object masks match so closely, that they are hard to distinguish from the hand crafted solution masks. A fact which is reflected in the percentages shown in table 4.4.

Some of the results stand out however, with a lacking segmentation. According to the percentages, image 3, 5, 9, 13 and 14 all have lesser object separation than the other images. The rest of the set of images end up with masks conforming close to 100%.

Upon visual inspection of these results, we see that the all the mentioned images have a degraded object mask where parts of the object has been merged with background. Inspecting image number 13, it seems as the object is nearly perfectly separated, yet it has the lowest percentage. This is the result of a weakness in the method for calculating object mask conformity. More on this in the next section.

4.8.1 Troublesome segmentations

To better understand the problems discovered with segmentation with EDISON, the process will be examined closer. Here all the troublesome segmentations will be discussed, as the reason for them are different in each case.

TS13

The reason why image 13 gives such a poor percentage, despite it seems to be a good segmentation. In figure 4.2 we can compare the different version of test image 13. The original image depicts the tool and a photographic ruler

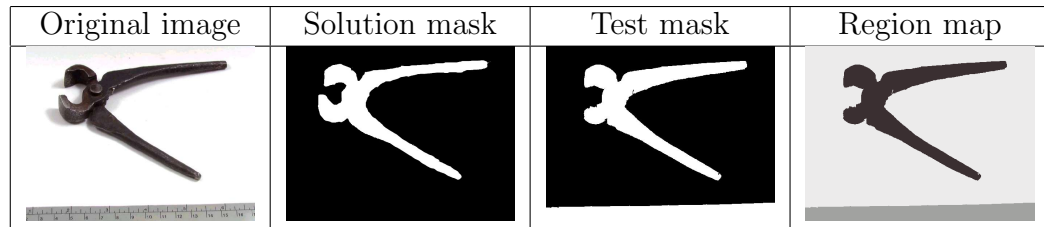


Figure 4.2: TS13

at the bottom. A comparison between the solution mask and the test mask, shows that a part of the tool is not marked as an object. But at the bottom of the test mask, the image is white, indicating it is an object. The result is that the ruler is also labeled as an object. This will, when the percentage calculation is run, affect the count of erroneous pixels. Thus leaving a low percentage, the reason the percentage becomes so small, is that the number of correct object pixels is small compared to the number of error pixels. Further inspection of the direct output from EDISON, the region map, it becomes clear that the method of generating the object mask is unsatisfactory.

TS3 and TS5

The first images in the test set which experienced problems with the segmentation were TS3 (fig. 4.3) and TS5 (fig. 4.4). In these two cases there is not a reason to question the percentage awarded to the test masks, when visually comparing the test results with the solution masks. For reasons of comparison we inspect the region maps. For TS3 it seems that the post-processing

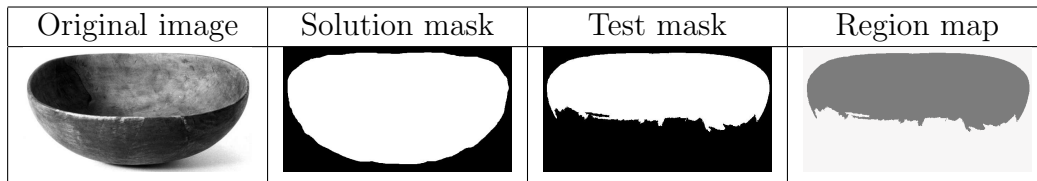


Figure 4.3: TS3

of the region map is not at fault this time. And again for TS5. It is clear that EDISON has only managed to successfully extract the torso of the toy figure.

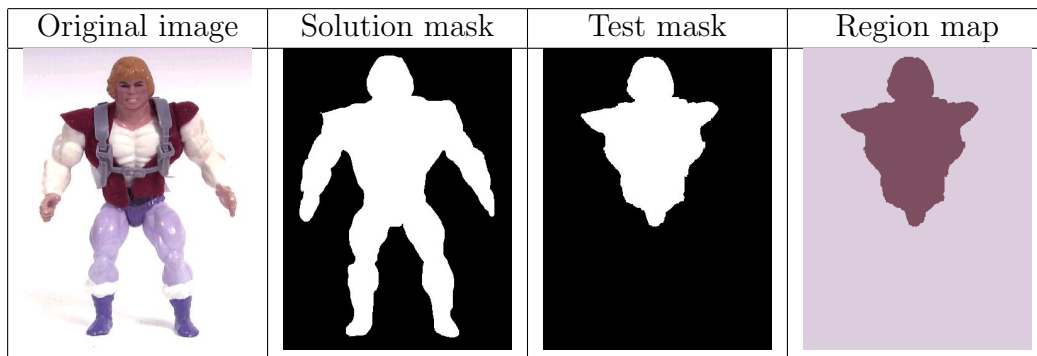


Figure 4.4: TS5

TS9

The test mask of image TS9 (fig. 4.5) leaves little of the original object intact. The only thing which still is detectable is the bottom arc. The region map, show clearly that EDISON has successfully segmented the bowl from the background. Yet again it seems like the post-processing step is at fault, losing the detected object mask.

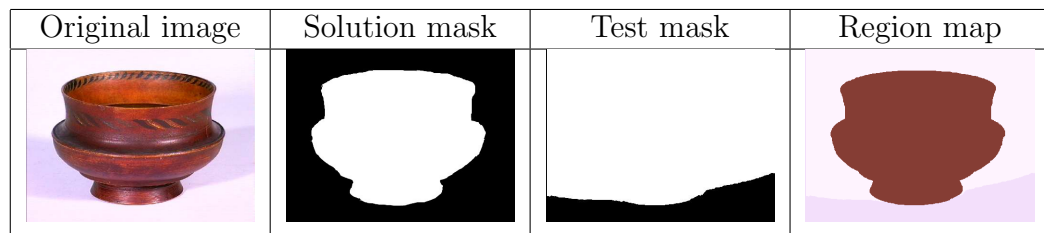


Figure 4.5: TS9

TS14

The final image, which got a low percentage, was TS14 (fig. 4.6). When looking at the resulting test mask, that is easily understandable. It is possible to see the contours of the cup, but it has merged with the background. In

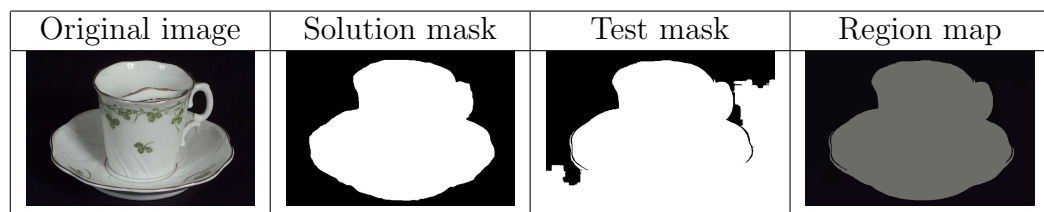


Figure 4.6: TS14

this case it becomes hard to understand how the the resulting mask from EDISON end up looking as it does. Loading the region map in an image editor, reveals that the black colour is not uniform, but there are several hues, indistinguishable to the human eye, thus resulting in a mask where the cup is merged with the background.

4.8.2 Summary edison

Overall the performance of EDISON was good. In the majority of cases the object was successfully separated from the background, and could be used for further shape analysis. In the cases where the segmentation did not produce a satisfactory result, the main fault was the post-processing of the segmentation, not the segmentation itself. Therefore a better post-processing method should be made. An improved method will also boost the quality of the overall EDISON results.

4.9 Test 2 - snake

The topic of this section will be the test of the snake-algorithm, and its results.

4.9.1 About snake

Snakes, or active contour models, have as shown in the papers [16, 15, 17], gained results in medical imaging, in recent years. In [21] the author tested a semi-automatic variant of the snake algorithm, which yielded promising results. Therefore the outset here is to test the applicability of an automatic version of the snake algorithm for shape extraction in image databases.

The main problem in creating an automatic snake, is to create a good initialization. In previous work, and most other implementations of snakes, the process is semi-automatic. The step which is under human control, is to create a starting contour, which the algorithm subsequently deforms, to match the underlying object.

In this test we wish to test the feasibility of an automatic snake segmentation. This is accomplished by automating the initialization process of the contour. Second we let the algorithm deform this initial contour and see how it fits the underlying object. These results are compared to those of EDISON and with the results from [21] in mind.

4.9.2 Making snake automatic

In this test a relatively simple method is used to initialize the snake or contour. Before deformation and resampling the contour, the user can choose to initialize the contour from the interface. This will begin a process where a coarse thresholding segmentation is run in the background, to pinpoint where in the image one might find an object. This temporary output is then analyzed, and the biggest region is identified. The region's center of gravity and bounding box is calculated. In the the test set, with images depicted on neutral background, it is very likely that this bound box confine the object of interest. Therefore the initial contour is set to be equal to the bounding box.

4.9.3 Setup of snake

Parameters

The initial value of the parameters used in the snake algorithm are as shown below. Except for the value of μ , the values are the standard values used in

[10]. A further explanation of these parameters can be found in 2.5.

- $\mu = 0.03$
- $\alpha = 0.1$
- $\beta = 10$
- $\gamma = 1$
- $\kappa = 0.6$

Force field

The performance of the snake algorithm is not only governed by its parameters. The force field plays an important part, in this test, the GVF force field will be used. The force field, is created upon an edge map of the given image. The initial edge map is created with Matlab's canny edge detector. More information on this edge detection method can be found in [22]. The second test with snake, will use another edge detection mechanism as a basis for the GVF field, where the image convoluted with the sobel masks.

Other notes

For every image in the test the snake algorithm, ran for 150 iterations. A selection of images will be further analyzed, and will run for more iterations. To speed up the the process the force field will be calculated at half resolution, this was found to have no negative implications in [10]. An overview of the segmentation process is visualized in figure 4.7

4.10 Results from test 2

The segmentation masks from snake, with GVF field calculated with edgemap based on the Canny edge detector can be seen in table 4.6. The accompanying table which show how well the segmentation masks fit the solution set are found in table 4.5.

The results from the second snake test, where we created an edgemap with the sobel operator are found in table 4.8. In table ?? the degree of conformity with the solution set is listed.

Table 4.6: The object masks produced by snake (canny)

Table 4.6: (continued)

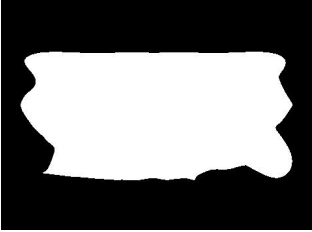

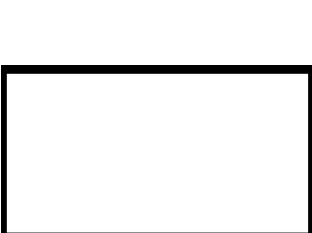
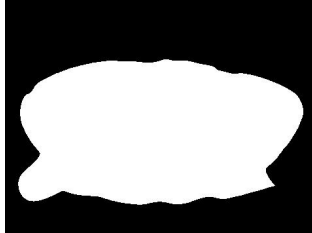
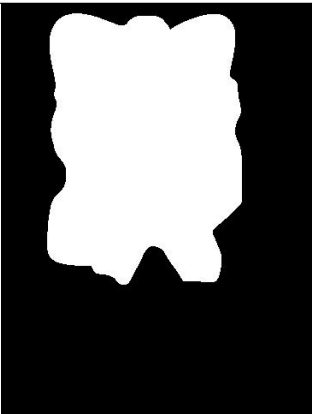
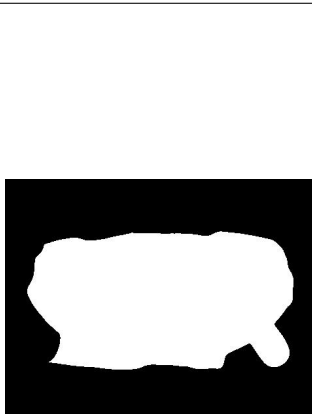
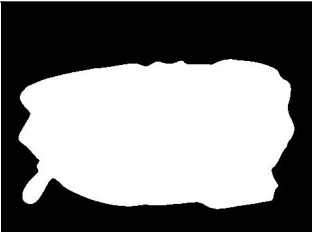
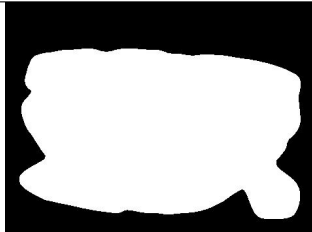
 1	 2
 3	 4
 5	 6
 7	 8

Table 4.6: (continued)

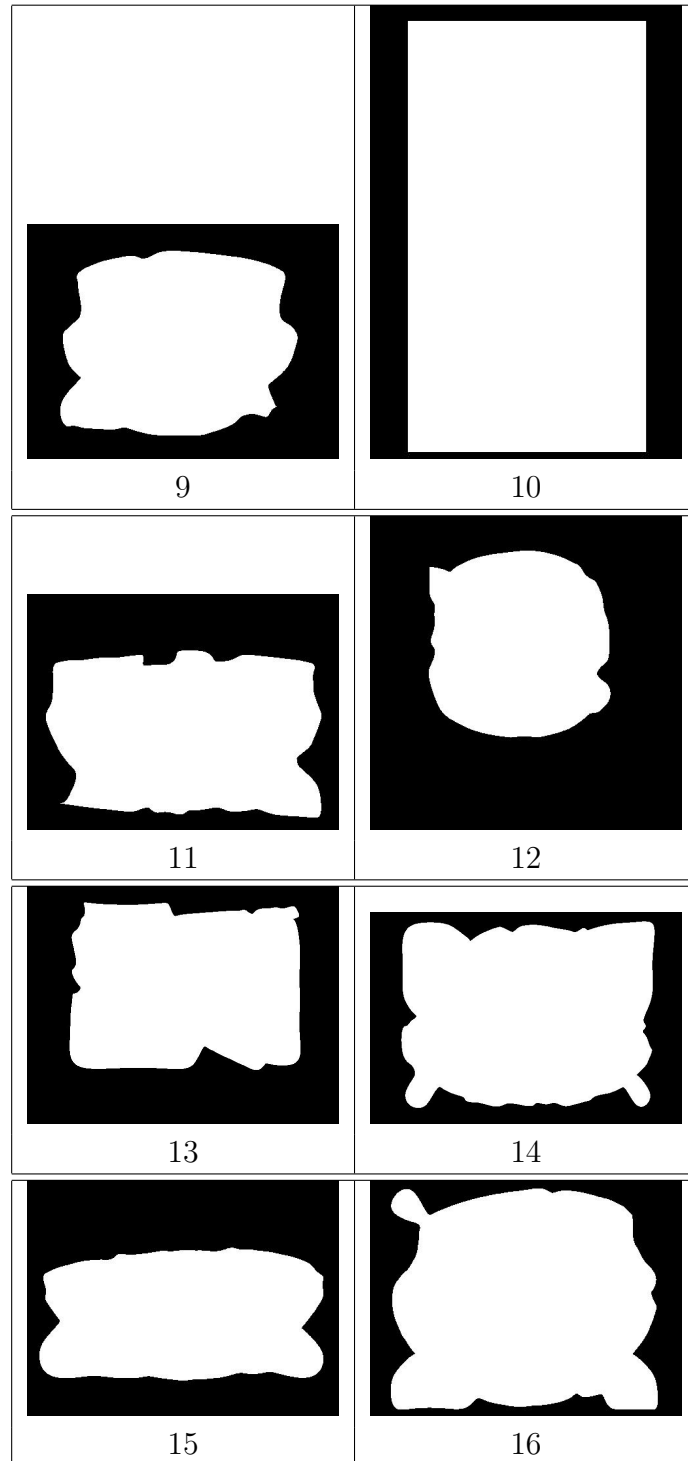


Table 4.6: (continued)

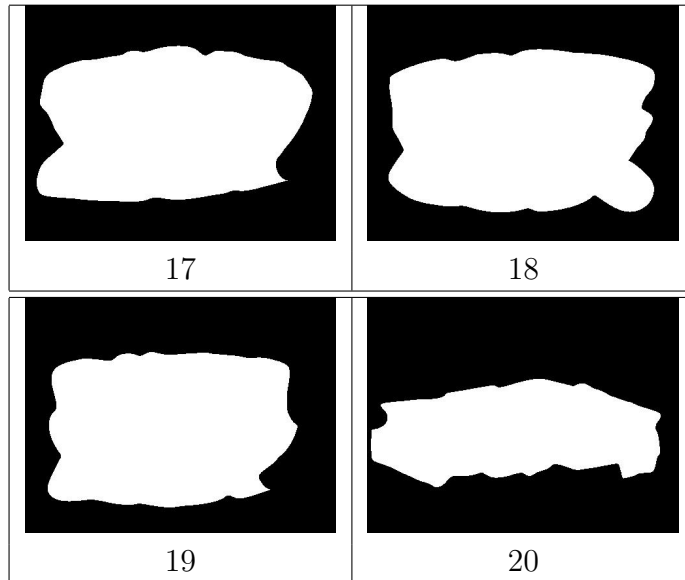


Table 4.8: The object masks produced by snake (sobel)

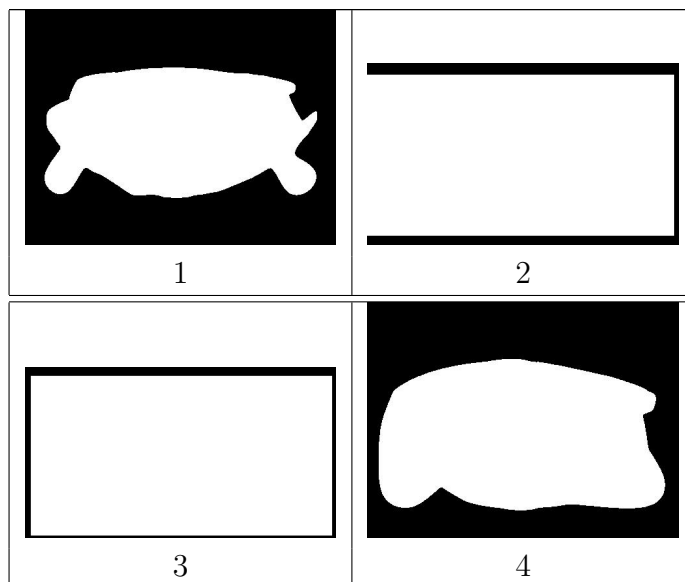


Table 4.8: (continued)

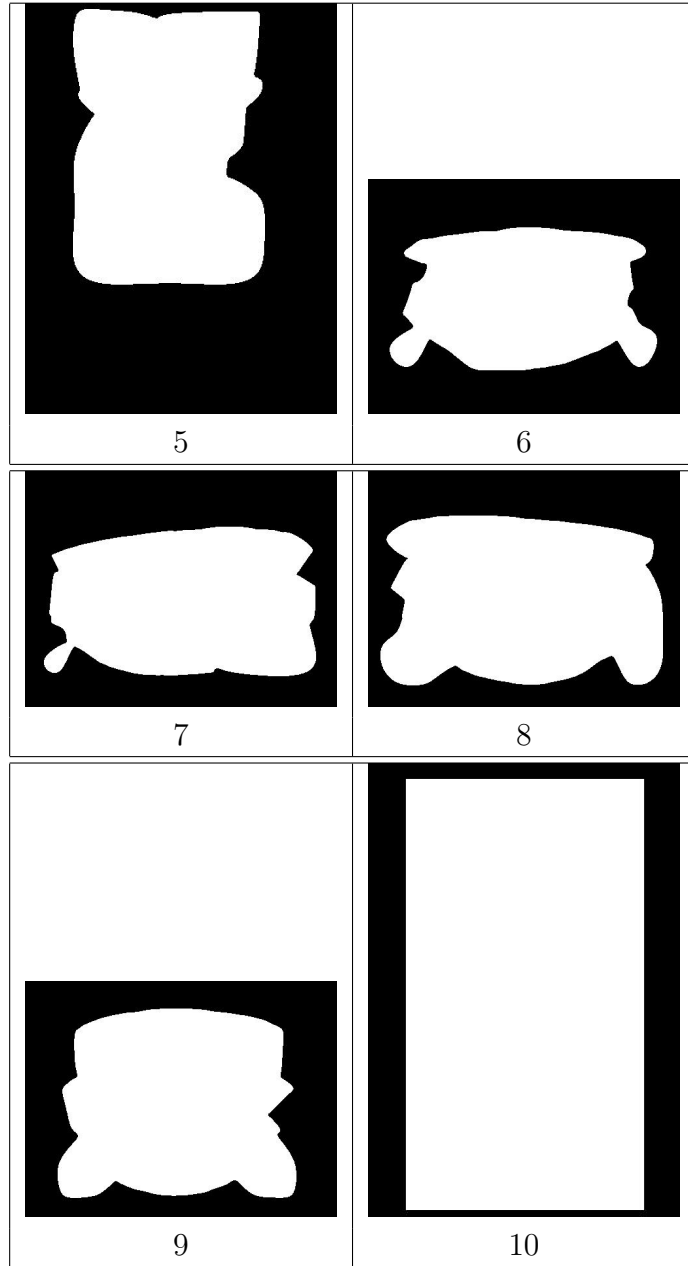


Table 4.8: (continued)

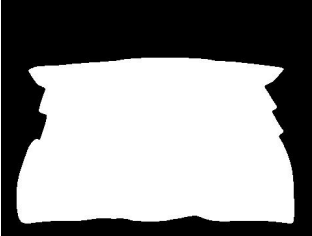
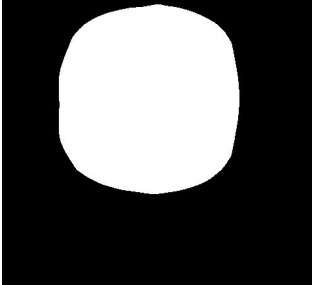
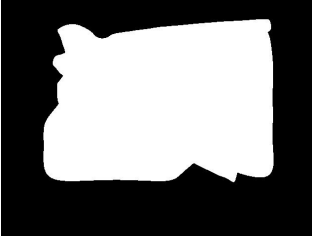
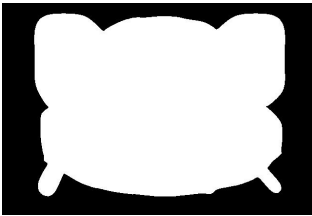
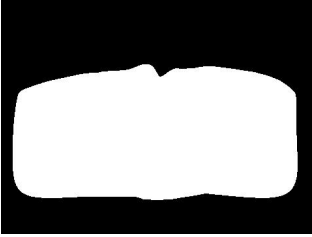
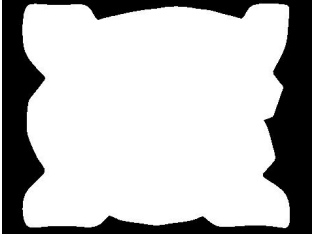
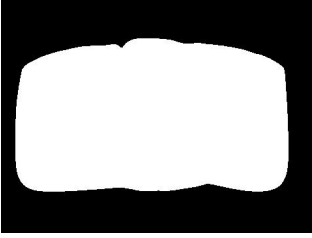
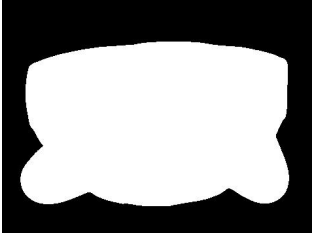
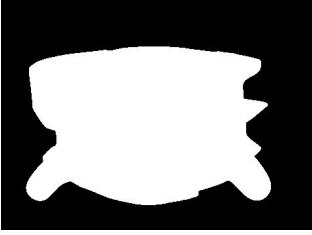
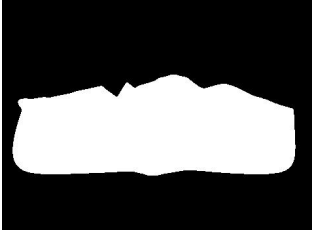
 11	 12
 13	 14
 15	 16
 17	 18
	

Table 4.8: (continued)

19	20
----	----

4.11 Discussion snake

In this section the results from the automatic snake segmentation will be discussed, along with the identified problems pertaining to automatic snake segmentation.

4.11.1 General remarks

It is clear when comparing the segmentation masks with the solution set, that there are discrepancies. One feature which all of the the masks coming from snake seem to have in common, is the presence of "ears" situated in the corner of the object. This feature is clearly not desirable. When inspecting the masks the edges of the object mask seem to be less smooth than what was the case with EDISON, particularly the masks in table 4.6. These deficiencies will be revisited shortly.

Another observation is that the algorithm failed segmentation for three images, image 2, 3 and 10. This issue will be discussed later, but again it is a fault which did not exist in EDISON. The second batch of segmentation masks made with snake, produced slightly different results. The masks in in this batch have overall a smoother edge along the mask. However, as is the case the case with the first batch, the segmentation tend to so exceed the boundaries of the object. In the case of snake, the contour of the snake has failed to snap onto the innermost object boundaries. These problems will now be looked into, as will be shown there is not necessarily one single problem, but a combination which combined, degrade the quality of the snake segmentation in the test set.

It might be justifiable to consider the segmentation masks qualitatively. The values in the table of conformity, does indeed suggest that the quality of segmentation is less than that of EDISON, however the numbers are not always largely different from that in table from EDISON. This is indicative of the difficulty, in assigning visual quality, a numerical value. To sum up some of the identified problems with the snake segmentation:

- "Ears" and nooks along the edge of the object mask
- Edge of segmentation mask, seem to fall to ease along edges, not part of object

Name	%	Correct pixels	Error pixels	total pixels
mask01	87 %	59501	7401	60108
mask02	0 %	148968	51209	148983
mask03	0 %	154153	39713	154952
mask04	91 %	70000	4468	71842
mask05	34 %	37898	19110	54839
mask06	89 %	64795	4684	67410
mask07	91 %	72496	4234	75074
mask08	87 %	82630	8682	84732
mask09	91 %	74385	5828	75708
mask10	0 %	52105	46135	52131
mask11	88 %	79217	7538	81734
mask12	96 %	63348	877	65091
mask13	0 %	29800	104543	31042
mask14	69 %	133394	37314	139600
mask15	80 %	61262	10884	62915
mask16	86 %	100646	12629	102533
mask17	87 %	68926	8054	69962
mask18	87 %	76122	8073	78134
mask19	92 %	69990	4003	71914
mask20	53 %	33576	15054	34742

Table 4.5: Object mask conformity in snake with canny based GVF field

Name	%	Correct pixels	Error pixels	total pixels
mask01	87 %	57337	5262	60108
mask02	0 %	148968	51209	148983
mask03	0 %	154153	39713	154952
mask04	82 %	69614	10617	71842
mask05	27 %	37410	22586	54839
mask06	82 %	60508	5212	67410
mask07	88 %	71908	5765	75074
mask08	83 %	80797	10677	84732
mask09	82 %	73467	11523	75708
mask10	11 %	52105	46135	52131
mask11	76 %	75927	13698	81734
mask12	98 %	64229	388	65091
mask13	0 %	28044	99900	31042
mask14	70 %	132238	35044	139600
mask15	75 %	61878	14628	62915
mask16	77 %	99540	20428	102533
mask17	78 %	69442	14688	69962
mask18	86 %	77096	10005	78134
mask19	85 %	65764	4457	71914
mask20	49 %	33259	16310	34742

Table 4.7: Object mask conformity in snake with sobel based GVF field

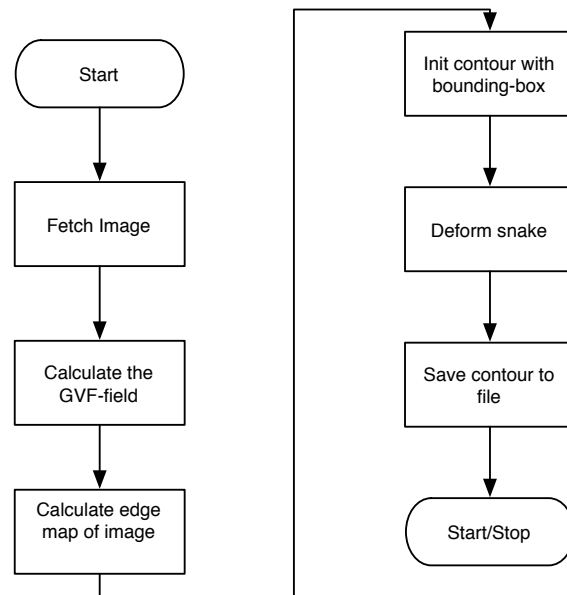


Figure 4.7: Snake segmentation process

- Segmentation mask covers area outside of object

4.11.2 Identified problems

When the segmentation was inspected in the previous section, it was clear that the segmentation was not optimal. The number of erroneous pixels in the segmentation mask are consistently higher for the snake masks. This is the opposite result when snake was tested in semi-automatic fashion in [21]. This section will discuss and identify the source of the experienced problems.

Automatic snake initialization

One apparent candidate, which can explain some of the difficulties seen, is the way the the contour now is automatically initialized. When the bounding box is used as initialization, there are some points on the contour that will be located further away from the object than desirable. These points are the corner points of the bounding box, specifically. The problem which may arise, is that the contour points will be too far way from the force field and thus will not be guided by the force field to the desired features in the image, namely the edges of the object of interest. In figure 4.8(a) we can see the effect where parts of the contour has failed to snap onto the object. Image 4.8(b) shows

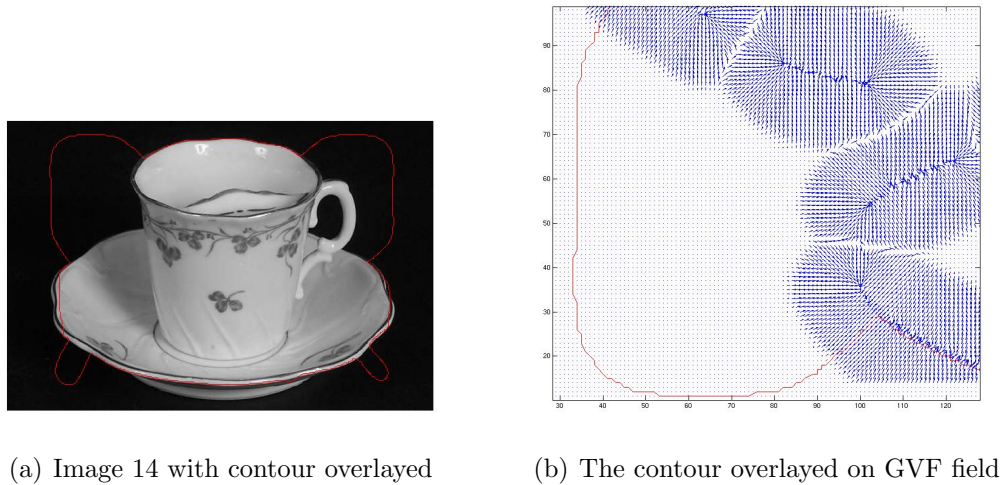


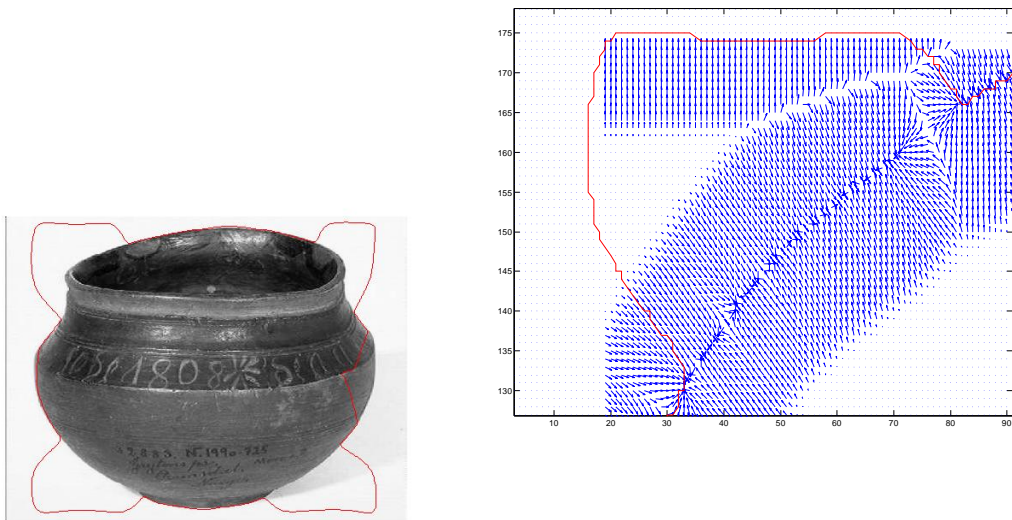
Figure 4.8: Image 14 with contour and part of the underlying GVF field

the GVF field, around the area where the contour has not converged against the object. It is clear from this picture that there are no, or very little forces, pulling the contour into the right direction. To overcome this problem there are a few possibilities to try out. One can in principle increase the value of μ when calculating the GVF field, thus resulting in spreading out the vectors in the GVF field, resulting in larger capture range. However, increasing the parameter too much, and you might end up confusing the contour, since the vectors located elsewhere in the force field will also be spread out. Another possibility is to make a better initialization of the contour, but as there will be a vast amount of shapes, other difficulties may arise. It might be worthwhile to attach a force to the contour points located in the corners of the bounding box, pointing towards the center of gravity. This would give some initial movement to the contour, and might lead it into the force field where it is affected by the other vectors.

The GVF field

Some problems were identified with the calculation of the GVF field. On some images the GVF code seemed to produce a border around the object with vectors pointing directly at the image edge. It is not certain what causes this effect, there might be some problems as to how the snake framework calculates the second order derivatives near the border. It warrants for further testing to check whether this calculation can be improved. To remedy this problem, the force field was nullified outside of the object's bounding box. This made the problem of non-converging contours in some images, while

small traces of this vector border persisted in other images. Figure ?? show the horizontal band of outwards-pointing vectors, which in practice traps the contour from converging inwards to the object. The part shown is the lower left part of the contour. Bands as these, in combination with being placed



(a) Image 16 with contour overlaid

(b) Erroneous band of vectors trapping the contour

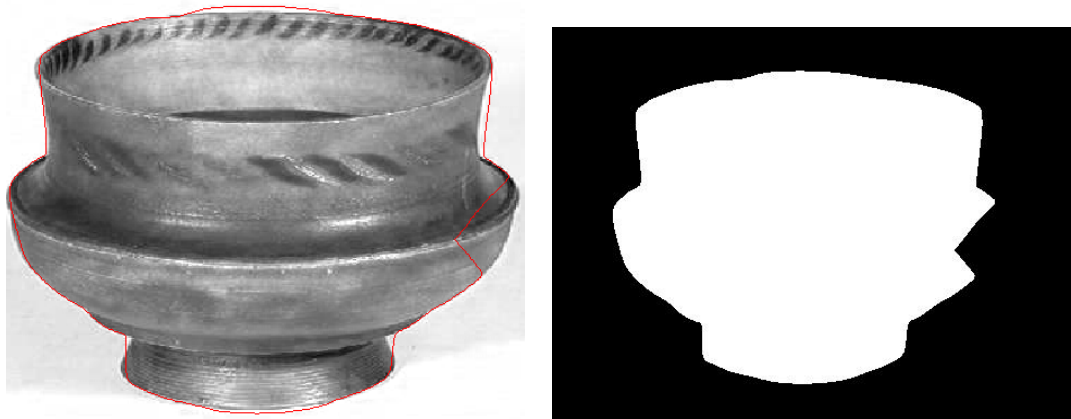
Figure 4.9: Image 16 with contour and part of the underlying GVF field

on the extreme border of the image, resulted in the failed segmentations of image 2,3 and 10.

Convergence and parameters

In this test the iteration count were capped at 150 for all images, however different images may need a different amount of iterations to converge to the correct solution. This is a major problem in automatic application of snake, an issue which easily overcome in semi-automatic situations. More iterations would have yielded better results for some images. Which is the case for image 9, see figure 4.11(a). To find a satisfactory solution for the convergence problem might not be possible. In situations as these, where the algorithms are based on derivatives, and second order derivatives, you can end with a very tiny change from the iteration before, so in fact, the contour

will change just slightly and unnoticeable. Some images will converge to a wrong solution, or there might be an error in the force field, which traps the contour, and you experience no change between iterations. If one looks aside from the cases where segmentations will fail, it might be possible to define a lower threshold of change, and iterations which yield less change than that threshold, should be terminated. More testing is required on this subject.



(a) Image 9 with contour overlaid, after 250 iterations and new GVF field (b) New segmentation mask after more iterations

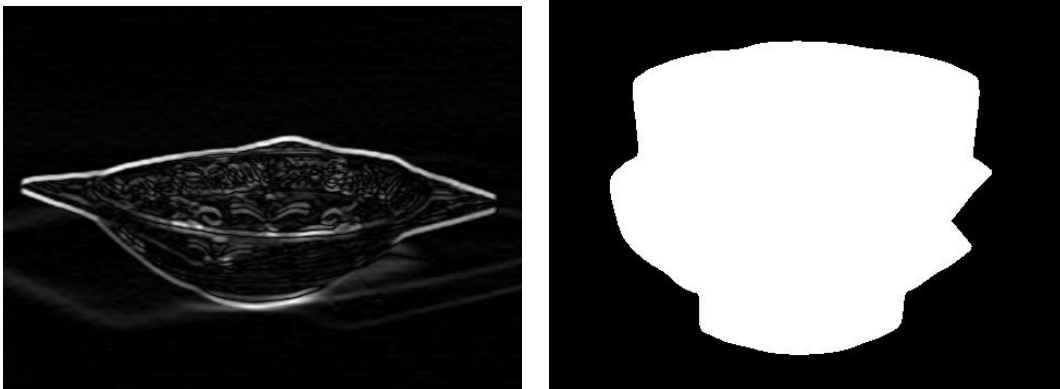
Figure 4.10: Image 9 segmented again with more iterations and a more powerful GVF field

Parameters are another problem with snakes, whereas EDISON is designed to be used in a non parametric fashion, snakes are not. Therefore it is a large body of work, to identify ideal parameters for different images. In an image database with a diverse collection of images, this is likely not even possible to find some common ground, for all parameters. This is perhaps why, it is particularly in medical imaging, that snake has gained ground first, where series of similar images are analyzed serially (organs). Changing parameters can make a difference, in figure 4.10 the value μ was changed to 0.15, in addition to up the iteration count. The higher value of μ means that the GVF field will spread out more giving a higher chance of affecting the contour, a higher iteration count, also gives the contour more time to converge to the correct solution. This yielded an almost perfect solution for image 9.

Edge map

An issue where the contour seemed to attach to an edge outside of the actual object, are usually caused by shadows in the original images. Even though

the background were reasonably homogenous, the shadows are still captured by some edge detectors, and when they are captured, they will contribute just as much to the foreground as the desired object edges. This is demonstrated in figure 4.11, we can see that the sobel edge map has more emphasis on object edges, that is the reason why the second batch of snake images came out with more smooth edges also.



(a) Sobel edge map

(b) Canny edge map

Figure 4.11: Two edge maps compared

4.11.3 Summary snake

All in all, the combination of the issues laid out above, and the relatively straightforwardness of EDISON it seems snake will need more work, before it is feasible for use in image databases.

Chapter 5

Conclusion

5.1 Conclusive remarks

An automated version of the snake algorithm has been tested. It has also been tested side by side another system, EDISON. The results of the EDISON test was uplifting the system yielded very good results. In addition the cases where the system did not deliver, was the fault of the post-processing algorithm was not robust enough.

The snake tests revealed certain difficulties in using an automatic version of snake, which introduce some uncertainty and problems in certain situations, detailed in the previous chapter. Therefor at this time, in the short run EDISON will be the most suitable choice of the two, for use as shape extraction in image databases. At this time, there are enough challenges with the snake algorithm, to not see it as a feasible choice for image databases in the short term.

Bibliography

- [1] *SESAM - Searching Supported by Analysis of Metadata*, Madrid, March 2002.
- [2] Donald A. Adjeroh and Kingsley C. Nwosu. Multimedia database management - requirements and issues. *IEEE Multimedia*, 4(3), 1997.
- [3] Chris M. Christoudias and Bogdan Georgescu. The robust image understanding laboratory. <http://www.caip.rutgers.edu/riul/research/robust.html>.
- [4] Christopher M. Christoudias, Bogdan Georgescu, and Peter Meer. Synergism in low level vision. *16th International Conference on Pattern Recognition*, IV:150–155, 2002.
- [5] L.D. Cohen and I. Cohen. Finite-element methods for active contour models and ballons for 2-d and 3-d images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:1131–1147, November 1993.
- [6] D. Comanicu and P. Meer. Mean shift: A robust aproach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5), May 2002.
- [7] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 2001.
- [8] Magnus Grøtan. Søk i bildedatabaser med fokus på innholdsbasert bildegjenfinning. Master's thesis, Norwegian University of Science and Technology, 2003.
- [9] Jon Olav Hauglid. *User Interfaces for Accessing Information in Digital Repositories*. PhD thesis, Norwegian University of Science and Technology, 2005.
- [10] Håkon Heuch. Segmentation of the liver from mr and ct images. Master's thesis, Norwegian University of Science and Technology, 2003.

- [11] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1988.
- [12] A. Langmyr and M. Grøtan. Gjenfinning av bilder i multimedia-databaser. Technical report, Norwegian University of Science and Technology, 2002.
- [13] ImageMagick Studio LLC. Imagemagick webpage. <http://www.imagemagick.org/>.
- [14] Guojun Lu. Techniques and data structures for efficient multimedia retrieval based on similarity. *IEEE Transactions on Multimedia*, 4(3), September 2002.
- [15] Tim McInerney and Demetri Terzopoulos. Topologically adaptable snakes. In *Proceedings of the Fifth International Conference on Computer Vision*, pages 840–850, June 1995.
- [16] Tim McInerney and Demetri Terzopoulos. Deformable models in medical image analysis: a survey. *Medical Image Analysis*, 1(2):91–108, 1996.
- [17] Tim McInerney and Demetri Terzopoulos. Topology adaptive deformable surfaces for medical image volume segmentation. *IEEE Transactions on Medical Imaging*, 18(10), October 1999.
- [18] Peter Meer and Bogdan Georgescu. Edge detection with embedded confidence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:1351–1365, 2001.
- [19] Brian E. Pangburn and Jonathan P. Ayo. jedison. From www.greatmindsworking.com <http://sourceforge.net/projects/eb1a/>.
- [20] Stan Sclaroff and Lifeng Liu. Deformable shape detection and description via model-based region grouping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(5), May 2001.
- [21] Ole Marius Smestad. Image databases -content based image retrieval, focus on segmentation. Technical report, Norwegian University of Science and Technology, 2004.
- [22] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image processing, Analysis, and Machine Vision*. PWS Publishing, second edition, 1999.

- [23] Chenyang Xu and Jerry L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing*, 7(3), March 1998.
- [24] Chenyang Xu and Jerry L. Prince. Gradient vector flow deformable models. *Handbook of Medical Imaging*, pages 159–169, 2000.
- [25] Chenyang Xu and J.L. Prince. Active contours and gradient vector flow. <http://iac1.ece.jhu.edu/projects/gvf/>.
- [26] Atson Yoshitaka and Tadao Ichikawa. A survey on content-based retrieval for multimedia data. *IEEE Transactions on Knowledge and Data engineering*, 4(3), 1999.