Continuous Max-Flow for Image Segmentation with Shape Priors

Master Thesis in Applied and Computational Mathematics

Mari Aurlien Kvile

Department of Mathematics
University of Bergen

June 2014

Acknowledgements

I want to thank everyone who has helped and supported me during my master studies. This would not have been possible without my friends and family who fill my life outside the university. I am grateful for your support and interest in my work even though you sometimes find it difficult to understand what I am working with.

Thanks to my fellow students at the faculty for much needed time outs with lunch breaks and good conversations. I especially want to thank my office buddy, Mary, for cooperation, encouragement and good friendship. It has been a pleasure working with you these years.

I want to thank my supervisor Xue-Cheng Tai for his mathematical ideas and for pushing me to work independently. Finally, I would like to thank Jofrid and Trine for reading through the thesis and commenting on the language.

Mari Aurlien Kvile

Bergen, June 2014

Abstract

In this thesis we propose a stable method for image segmentation with shape priors. The original Chan-Vese intensity based segmentation model with regularisation term is extended to include shape prior information. We study shape priors which are pose invariant under the group of similarity transformations, that is under rotation, scaling and translation. In order to solve this problem robustly and effectively, an algorithm based on the theory of max-flow and min-cut is used in addition to a gradient descent procedure for updating the pose parameters. Comprehensive experiments are provided to demonstrate the behaviour of the proposed method on different images.

Contents

Chapter 1

Introduction

The area of image processing deals with generating and processing digital images in a way suitable for the relevant application, often with the purpose of making it easier to extract information from the images. This is a field that has grown immensely along with the development of the computer, and is still growing. Today, image processing is used in a broad range of applications in defence, meteorology, medicine, industry, archaeology, astronomy, law enforcement, and many other areas. Some examples include mineral and oil exploration, analysis of satellite images for weather prediction, automated license plate reading, restoration of images of unrecoverable objects, contrast enhancement for easier interpretation of x-ray images, and automated inspection for missing parts of products in a factory.

Segmentation is the first task in the process of solving many image analysis problems, and the quality of the segmentation will therefore often determine the result of the rest of the process as well. Consequently, segmentation is an important image processing task, and research on segmentation has increased significantly during the last decade. Segmentation is the process of dividing an image into different regions, for example separating an object of interest from its background, thus making it easier to analyse. This is especially useful in areas where a computer is supposed to automatically identify or recognise objects in an image. For example, in order to automatically read a licence number plate, the computer first has to isolate the licence number plate and then extract the individual characters before it is possible to try and recognise them. Other examples include counting the number of fish in an underwater picture, locating tumours in medical images, object tracking in videos, locating objects in satellite images, and face recognition.

Autonomous segmentation is generally a very difficult task, and if we have any knowledge about the shape of the object to be segmented, this can significantly improve the segmentation result. A segmentation procedure

with inclusion of shape information is called shape prior segmentation. Such procedures allow us to successfully segment images that are too difficult to segment without shape prior information. This can be images with a background containing colours or patterns too similar to the object, images with too weak edges, very noisy images or images where the object is occluded by other objects.

Such an approach is naturally only possible if we actually have some information about the shape of the object to be segmented. This information can be very general, such as a star shaped prior [19] or even more general like the regularisation term in the Chan-Vese segmentation model [4], which incorporates the information that the shape of the object to be segmented is likely to have a relatively short boundary. In the present thesis however, we study more object specific priors, which can be relevant in applications where one has to segment several similar images. It is then possible to use a more time consuming method or a method requiring more manual interaction for successful segmentation of some of the images, and then use the results as shape priors when segmenting the rest of the images. An example is in the segmentation of a video, where the images are quite similar from one frame to another if the frame rate is high enough. There one can use the segmentation result from the previous frame as a shape prior for the segmentation of the next frame. Another possibility is if one has access to a database of example shapes of the same type of objects, and can incorporate statistical information about these shapes into a shape prior (see for example [5] and [13]).

The purpose of this work is to test a new fast and robust method for shape prior image segmentation. We use several of the ideas in the paper by Overgaard et al. [16], where the Chan-Vese model for segmentation is extended to include shape prior information. For the numerical minimisation we include some of the ideas in [21] in order to make the method more robust. Accordingly, a sub-problem is formulated as a continuous min-cut problem, and the theory of max-flow and min-cut is used to solve it with a fast and convex max-flow based algorithm as presented in that paper.

The remainder of this thesis is organised as follows: In chapter 2 some preliminary theory about optimisation and image processing is reviewed. Also, the Chan-Vese segmentation model and the max-flow and min-cut approach are explained. Then, we detail the proposed model in chapter 3. A short discussion on the implementation is the subject of chapter 4, followed by a variety of results in chapter 5. Here, we show the advantages and disadvantages of the proposed method, we discuss the use of different colour models and the number of iterations.

Chapter 2

Preliminaries

In this chapter we recall some theory that lays the foundation for the discussions in the rest of the thesis. First we review some general optimisation theory and methods, then we go on to define a digital image and describe some image processing tools that are used, before defining the segmentation task. In chapters 2.3 and 2.4 two methods that are used in the present work are described.

2.1 Optimisation theory

Optimisation is minimisation or maximisation of a function, often subject to some constraints on its variables. Many practical problems can be formulated as optimisation problems; in business and industry for instance, people want to maximise the efficiency, minimise the costs and maximise the profit. Specifically, many problems in image processing can be modelled as energy minimisation or maximisation problems, which is why optimisation is of interest for this study. Recall that any minimisation problem can be formulated as a maximisation problem and vice versa:

$$\max f(x) = \min -f(x).$$

Therefore, we will in the following only recall definitions and theorems for either minimisation problems or maximisation problems, not both.

2.1.1 Convexity

Convexity is a fundamental concept in optimisation, because problems that possess this property are easier to solve than problems that do not. Convexity is defined as follows:

Definition: *A subset C of a vector space is called convex if for all $x, y \in C$ and for all $\alpha \in [0,1]$ we have $\alpha x + (1-\alpha)y \in C$.*

This means that the straight line connecting two points in a convex set also lies in the set.

Definition: *A functional $f : C \to \mathbb{R}$ is called convex if its domain is a convex set and if for all $x, y \in C$ and for all $\alpha \in [0,1]$ we have $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$.*

If $C \subset \mathbb{R}^n$ this means that the straight line connecting two points on the graph of f lies over or on the graph. Recall that a functional is a special case of a function, $f : C \to V$, where $V \subset \mathbb{R}$. Therefore, a functional will often be referred to as a function, especially if $C \subset \mathbb{R}^n$. The following theorem from [15] explains the advantages of a convex functional:

Theorem: *When f is convex, any local minimiser x^* is a global minimiser of f. If in addition f is differentiable, then any stationary point x^* is a global minimiser of f.*

Therefore, if our optimisation problem consists of minimising a convex functional f, it is enough to find a point x^* where $f'(x^*) = 0$. If the functional is not convex however, we have to search among all local minimisers in order to find the global minimiser. Obviously, this is a much more difficult and time-consuming task.

2.1.2 Gateaux differential

In elementary calculus, we only learn about functionals where $C \subset \mathbb{R}^n$. However, C could also be a set from some general function space, and then the functional is a function of a function rather than a function of real numbers. When we differentiate such a functional, we differentiate it with respect to a function, and the definition of the derivative that we learn in elementary calculus cannot be used. For this purpose, we can use the Gateaux derivative [7]:

Definition: *We say that $f'(x)$ is the Gateaux differential of $f(x)$ if for all $d \in C$ the directional derivative of f at x in the direction d is*

$$f'(x;d) = \lim_{h \to 0} \frac{f(x+hd) - f(x)}{h} = \langle f'(x), d \rangle, \tag{2.1}$$

where $\langle \cdot\,, \cdot \rangle$ is a proper inner product on the function space.

In this thesis we will use the L^2 inner product $\langle u, v \rangle = \int u \cdot v \, dx$.

2.1.3 Gradient descent method

Based on the theorem in section 2.1.1, a natural approach for finding the minimum point of a convex functional is to search for points where $f'(x) = 0$. However, this equation might be impossible to solve analytically. An alternative can be gradient descent method [17], also called steepest descent method, which we explain in the case of a functional $f : \mathbb{R}^n \to \mathbb{R}$. The aim is to solve the optimisation problem

$$x^* = \arg\min_{x \in \mathbb{R}^n} f(x). \tag{2.2}$$

The gradient descent method is an iterative method where a sequence $\{x^{(k)}\}$ which converges to x^* is constructed. We start with an initial guess $x^{(0)}$ and construct the sequence according to

$$x^{(k+1)} = x^{(k)} - \varepsilon^{(k)} \nabla f(x^{(k)}), \tag{2.3}$$

where $\varepsilon^{(k)}$ is the step length. The reason why this iteration leads us towards the minimum point is that $f(x^{(k+1)}) = f\left(x^{(k)} - \varepsilon^{(k)} \nabla f(x^{(k)}\right) < f(x^{(k)})$ as shown below:
The Taylor expansion gives us that (taking away the iteration number for readability)

$$f(x - \varepsilon \nabla f(x)) = f(x) - \varepsilon \nabla f(\xi)^T \nabla f(x)$$

for some $\xi \in (x, x - \varepsilon \nabla f(x))$ provided that f is continuously differentiable. If we write $\xi = x - a\varepsilon \nabla f(x)$ for some $a \in (0, 1)$ we get that

$$f(x - \varepsilon \nabla f(x)) - f(x) = -\varepsilon \nabla f(x)^T \nabla f(x) + O(\varepsilon^2) = -\varepsilon \|\nabla f(x)\| + O(\varepsilon^2)$$

The right hand side becomes negative if ε is small enough and hence $f(x - \varepsilon \nabla f(x)) < f(x)$.

This method can also be used for the more general optimisation problem

$$x^* = \arg\min_{x \in C} f(x) \tag{2.4}$$

where $f : C \to \mathbb{R}$. If C is a subset of a function space, we need to perturb the function x so that the functional f gets a lower value. Then we have to use the Gateaux differential. A function (x) cannot be perturbed with a number $(f'(x; d))$, but it can be perturbed with another function $(f'(x))$. So similarly as above we have that

$$f(x - \varepsilon f'(x)) < f(x) \text{ when } f'(x; d) = \langle f'(x), d \rangle \, \forall \, d \in C,$$

and the minimising sequence is thus constructed according to

$$x^{(k+1)} = x^{(k)} - \varepsilon^{(k)} f'(x^{(k)}). \tag{2.5}$$

2.1.4 Method of Lagrange multipliers

The gradient descent method can be used to solve unconstrained optimisation problems such as (2.4). However, optimisation problems that arise in natural applications often come with a set of constraints that have to be fulfilled. The constraints can be equality constraints, inequality constraints or a mix of these, but in this work we only consider equality constrained optimisation problems:

$$\text{Find } x^* = \arg\max_{x \in C} f(x) \text{ s.t. } g(x^*) = 0. \tag{2.6}$$

A well-known technique for solving such problems is the method of Lagrange multipliers. We will describe this technique for the real case:

$$\text{Find } x^* = \arg\max_{x \in \mathbb{R}^n} f(x) \text{ s.t. } g_i(x^*) = 0 \text{ for } i = 1, \ldots, m. \tag{2.7}$$

The method is based on the following theorem [1][15]:

Theorem: *If x^* is the solution of (2.7), if f and g_i are continuously differentiable and the set $\{\nabla g_i(x^*),\ i = 1, \ldots, m\}$ is linearly independent, then there exists $\lambda^* \in \mathbb{R}^m$ such that $\nabla L(x^*, \lambda^*) = 0$ where*

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x). \tag{2.8}$$

The reasoning behind this is that when $\nabla L(x^*, \lambda^*) = 0$, then $g_i(x^*) = 0\ \forall\, i$ and $\nabla f(x^*)$ is a linear combination of $\nabla g_i(x^*)$. It is clear why $g_i(x^*)$ must be zero when x^* is the solution of (2.7). Moreover, if $\nabla f(x^*)$ is not a linear combination of $\nabla g_i(x^*)$ then there exists a nonzero projection of $\nabla f(x^*)$ along the tangent line of the curve which is the intersection of $g_i(x) = 0$. Hence, f has a positive directional derivative along this tangent line and can be increased without violating any of the constraints $g_i(x) = 0$. Because x^* is the solution of (2.7) this cannot be true, and so $\nabla f(x^*)$ must be a linear combination of $\nabla g_i(x^*)$.

The method of Lagrange multipliers consists thus of looking for critical points of (2.8) in order to find candidates for the solution of (2.7). The method can be extended to the more general maximisation problem (2.6), see for example [6]. The general Lagrange function is of the form

$$L(x, \lambda) = f(x) + \langle \lambda, g(x) \rangle,$$

where $\langle \cdot\,, \cdot \rangle$ is an inner product on the vector space. We see that (2.8) is of this form when $\langle \cdot\,, \cdot \rangle$ is the Euclidean inner product and $g(x) : \mathbb{R}^n \to \mathbb{R}^m$ is given by $g(x) = [g_1(x), g_2(x), \ldots, g_m(x)]$. Thus, this is just a special case of the general Lagrange function.

2.1.5 Augmented Lagrangian method

Another way to solve (2.7) is by combining the objective function and the constraints into a penalty function, and thus solving a sequence of unconstrained optimisation problems instead of the original constrained optimisation problem. One common penalty function is the quadratic one:

$$f(x) - \frac{\gamma}{2} \sum_i g_i^2(x),$$

where $\gamma > 0$ is the penalty parameter. This method generally requires $\gamma \to \infty$ to ensure that g_i is sufficiently close to zero, and there is also a possibility of ill-conditioning. Therefore, the augmented Lagrangian function [15] combines the properties of the quadratic penalty function and the Lagrangian function:

$$L_a(x, \lambda, \gamma) = f(x) + \sum_i \lambda_i g_i(x) - \frac{\gamma}{2} \sum_i g_i^2(x). \tag{2.9}$$

Here, λ is an approximation of the optimal Lagrange multiplier λ^*. In the augmented Lagrangian method we start with an initial λ^0 and γ^0, and in every step we find x^k as the approximate maximiser of $L_a(x, \lambda^k, \gamma^k)$. γ could be increased in every iteration, and λ^k is updated based on the observation that

$$\nabla_x L_a(x^k, \lambda^k, \gamma^k) = \nabla f(x^k) + \sum_i (\lambda_i^k - \gamma^k g_i(x^k)) \nabla g_i(x^k) \approx 0$$

and

$$\nabla_x L(x^*, \lambda^*) = \nabla f(x^*) + \sum_i \lambda_i^* \nabla g_i(x^*) = 0,$$

which suggests that

$$\lambda^{k+1} = \lambda^k - \gamma^k g(x^k). \tag{2.10}$$

This method can also easily be extended to the more general case (2.6). Then, the augmented Lagrangian function has the form

$$L_a(x, \lambda, \gamma) = f(x) + \langle \lambda, g(x) \rangle - \frac{\gamma}{2} \|g(x)\|^2. \tag{2.11}$$

2.2 Image processing basics

2.2.1 Digital images

An image can be defined as a two-dimensional continuous function $I(x)$ defined on $\Omega \subset \mathbb{R}^2$, where $x = [x_1, x_2]$ are spatial coordinates and the function value $I(x)$ has some physical meaning determined by the source of the image [12]. A digital image is the discretised version of an image, where the spatial coordinates and the function values are finite, discrete quantities. Although the images we work with in digital image processing are discrete, it is common and useful to think of them as continuous when constructing methods and developing theory.

Images like the ones you take with your camera are generated in almost the same way as we see the world around us: There is some kind of a sensor sensing the amount of light reflected by the objects being imaged—little light reflected gives a dark colour whereas much light reflected gives a light colour, and so in this case $I(x)$ represents the light intensity reflected by the point x. However, an image can also be generated by reflection or absorption of other sources of energy than normal visible light; for example infrared light, X-rays or ultrasound, and can thus show information that is normally not visible to the naked human eye. In the case of an X-ray image for example, $I(x)$ represents the amount of X-ray energy that has at the point x passed through the object being imaged.

Regardless of how the image has been acquired, it is necessary to digitise the continuous sensed data to create a digital image. The digitising of the coordinate values, *sampling*, is often determined by the number of sensors used in the image acquisition and their placement. Digitising the function values, *quantisation*, is done by picking equally spaced values along the intensity scale and assigning one of these to each point depending on which discrete quantity is closest to the sensed value. Normally, the number of discrete intensity levels is 2^k, where k is an integer. An image with 2^k intensity levels is called a k-bit image. 8-bit images are most common, and such an image has intensity levels that are integers in the interval $[0, 255]$. However, the intensity is often normalised to the interval $[0, 1]$, where 0 represents black and 1 represents white.

Now, the digital image consists of a finite number of picture elements, *pixels*, having a specific location and value. They can be placed in a 2-D array and showed in a coordinate system with the origin being in the top left corner of the image, the x_1-axis pointing downwards and the x_2-axis to the right.

For example,

$$I(x) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0.5 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0.5 & 0.5 & 0.5 & 0 \\ 1 & 1 & 1 & 0.5 & 0.5 & 0.5 & 0.5 & 0 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0 & 0 & 0 \end{bmatrix}$$

is shown in figure 2.1.

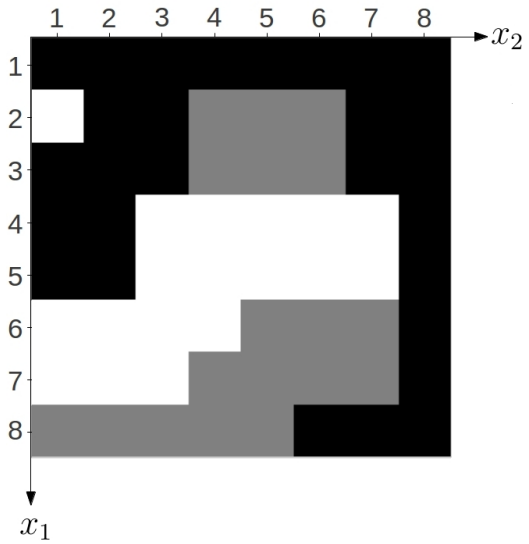


Figure 2.1: The image I shown in a coordinate system.

Neighbours of a pixel

A pixel p with coordinates (x_1, x_2) has four horizontal and vertical neighbours with coordinates (x_1+1, x_2), (x_1-1, x_2), (x_1, x_2+1) and (x_1, x_2-1). The set of these pixels is called the *4-neighbours* of p, denoted $N_4(p)$. The four *diagonal* neighbours with coordinates (x_1+1, x_2+1), (x_1+1, x_2-1), (x_1-1, x_2+1) and (x_1-1, x_2-1) make out the set $N_D(p)$. The union of these two sets is the *8-neighbours* of p, $N_8(p)$.

2.2.2 Different types of images

Binary images

A binary image is a function $I : \Omega \to \{0, 1\}$, that is, the intensity values can only be 0 (black) or 1 (white). Binary images are often used as masks or similarly as characteristic functions for a region. If I is a characteristic function for the region Σ, then

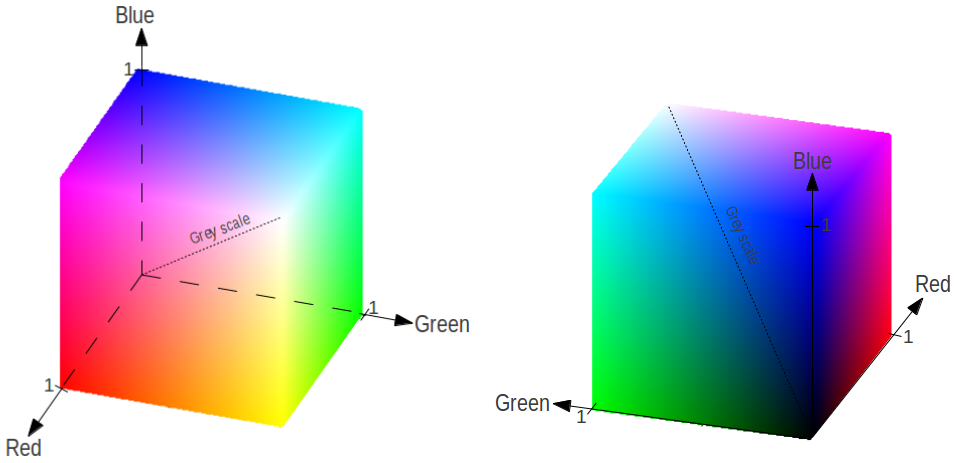
$$I(x) = \begin{cases} 1 & \text{if } x \in \Sigma, \\ 0 & \text{elsewhere.} \end{cases}$$

Grey scale images

A grey scale image is a function $I : \Omega \to S$ where S is a finite subset of $\mathbb{R}$ containing the allowed intensity values.

Colour images

A colour image is a function $I : \Omega \to S \subset \mathbb{R}^n$. There exist several different colour models, for example RGB, CMY, CMYK and HSI [12]. In our experiments we will mostly use the RGB colour model, but the HSI model can also be very useful and we will therefore describe these two models:

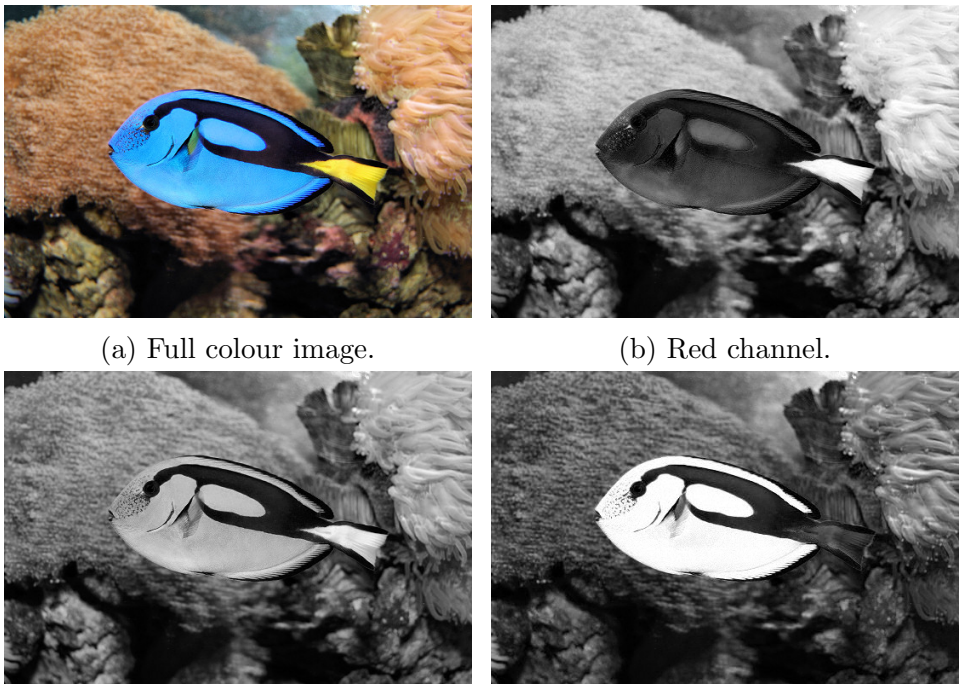


(a) Shown from $(1, 1, 1)$ (b) Shown from origo

Figure 2.2: The RGB colour cube.

RGB is the abbreviation for the three primary colours red, green and blue, and the RGB colour model adds different intensities of red, green and blue

light together to create almost all visible colours. Equal intensities of two primary colours added together give the secondary colours: red and green yield yellow, red and blue yield magenta, and green and blue yield cyan. This model can be shown as a cube in a coordinate system where the amount of red, green and blue increases along the x, y and z axis (see figure 2.2). Then, black is at the origin, white is at $(1, 1, 1)$ (if the intensities are normalised as they often are) and the primary and secondary colours are at different corners. Points with equal intensities of both red, green and blue make out the grey scale. When an image is represented in the RGB space, it consists of three grey scale images which each show the amount of one of the primary colours in the image, as shown in figure 2.3.



(a) Full colour image. (b) Red channel.

(c) Green channel. (d) Blue channel.

Figure 2.3: RGB decomposition of an image.

HSI is the abbreviation for hue, saturation and intensity. It is a colour model which is closer to the way humans describe colours than the RGB model. We do not describe a colour as a composite of red, green and blue colours, but rather refer to which colour it is, how pure it is and how bright it is. The hue component of an HSI image holds the colour information, the saturation component tells us how pure the colour is, i.e. how much it is mixed with white light, and the intensity component holds information

about the intensity or brightness of the colour. If we rotate the RGB colour cube so that the grey scale is vertical and the cube stands on the black corner with the white corner at the top, the HSI colour model can be obtained. If we look at the cube from above, we see that the rest of the corners form a hexagon where the primary colours are separated by 120° and the secondary colours are 60° from these (this can be seen in figure 2.2a). This hexagon can be simplified to a circle, where the hue is defined by an angle from some reference point which usually is red at 0°. The intensity scale goes along the vertical axis, and since the saturation increases as we move away from the grey scale, it is described by the distance from the vertical axis. This model can be shown as in figure 2.4.

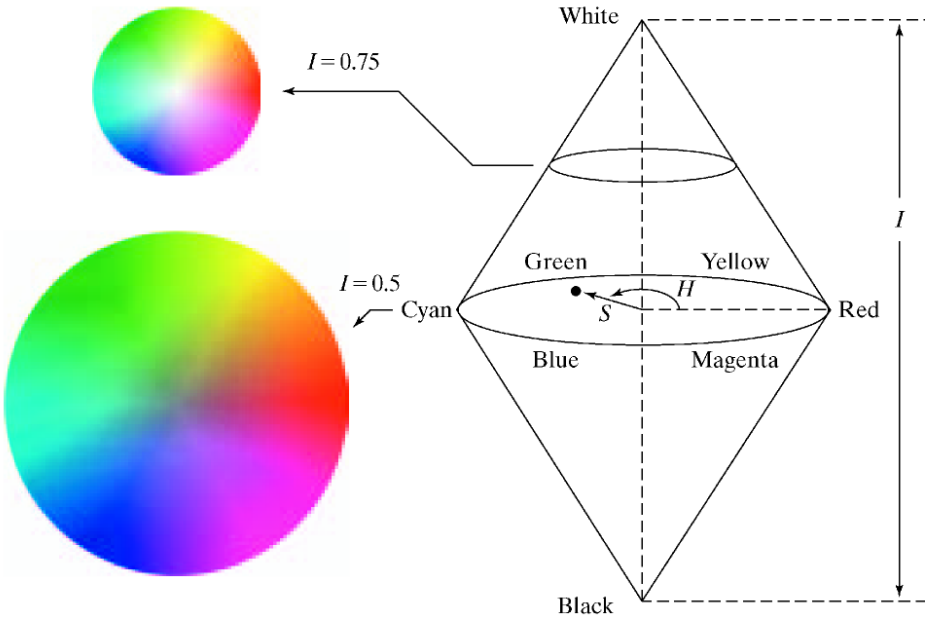
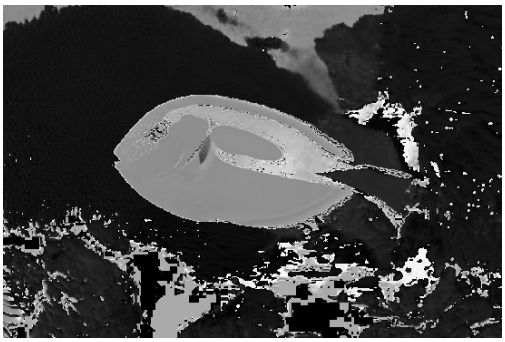
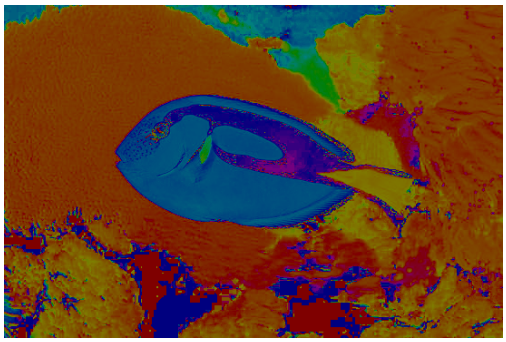


Figure 2.4: The HSI colour model based on circular colour planes.

In figure 2.5 the hue, saturation and intensity components of an image are shown as grey scale images. The image is also shown with hue information only, with saturation information in addition and finally with all information.

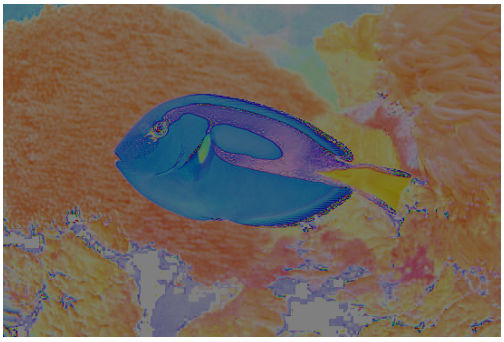


(a) Hue component shown as grey scale image.



(b) Hue component with saturation 1 and intensity 0.5 shown as colour image.

(c) Saturation component shown as grey scale image.



(d) Hue and saturation components with intensity 0.5 shown as colour image.

(e) Intensity component shown as grey scale image.

(f) All components shown as colour image.

Figure 2.5: Image in HSI space.

2.2.3 Mathematical operations used on images

Arithmetic operations

It is important to note that even though the mathematical representation of an image looks like a matrix, most arithmetic operations performed on images are not matrix operations but array operations, that is, they are done element-wise. For example if $I = f \cdot g$, then both I, f and g are images of the same size and $I(x) = f(x) \cdot g(x)\ \forall x \in \Omega$. Arithmetic operations play an important role in digital image processing, for example in noise reduction (averaging images) and shading correction. A common use of image multiplication is in *masking* operations, where a binary mask image is multiplied with the image to extract regions of interest. The mask image consists of ones in the regions we are interested in and zeros in the remaining parts of the image. This is an important tool in image segmentation.

Affine transformations

We can use affine transformations to rotate, scale, translate and shear an image. We transform the coordinates using

$$[x_1, x_2, 1] = [y_1, y_2, 1]\, T, \tag{2.12}$$

where (x_1, x_2) are the pixel coordinates in the transformed image and (y_1, y_2) are the coordinates in the original image. T is a 3 by 3 matrix with elements depending on which operation we want to use:

$$\text{Rotation: } T = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \text{scaling: } T = \begin{bmatrix} \varsigma_1 & 0 & 0 \\ 0 & \varsigma_2 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\text{translation: } T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a_1 & a_2 & 1 \end{bmatrix}, \quad \text{shear: } T = \begin{bmatrix} 1 & 0 & 0 \\ \zeta_1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ or } \begin{bmatrix} 1 & \zeta_2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

It is easy to combine several operations: let T be the product of the matrices corresponding to the operations that are to be performed.

One possibility is to use (2.12) directly to find the new location (x_1, x_2) of each pixel, a technique called *forward mapping*. However, this can cause some trouble since several pixels can be transformed to the same location and some locations might not be given any pixels. Therefore, it is better to use *inverse mapping*, where we for every pixel coordinate in the transformed image find the corresponding coordinate in the original image using $[y_1, y_2, 1] = [x_1, x_2, 1]\, T^{-1}$. This coordinate might not be a pixel location, and

we thus have to use interpolation to assign an intensity value to the pixel in the transformed image. There are several interpolation techniques that can be used, but in this study we only use nearest neighbour interpolation, where the intensity value of (x_1, x_2) will be equal to the intensity value of the nearest neighbouring pixel of (y_1, y_2) in the original image. If (y_1, y_2) is outside the image domain Ω and hence does not have any neighbours, we use the convention that the intensity value will be zero.

Linear spatial filtering and convolution

Spatial filtering can be used for many tasks in image processing. This process consists of moving the centre of a neighbourhood from pixel to pixel in the original image and doing some operation on the pixels enclosed in the neighbourhood to create the output value at the centre location. Specifically, linear spatial filtering creates the output as the sum of products of the filter coefficients and the image pixels. Linear spatial filtering of an image I with a filter w of size $(2a+1) \times (2b+1)$ yields the output O given by:

$$O(x_1, x_2) = \sum_{t_1=-a}^{a} \sum_{t_2=-b}^{b} w(t_1, t_2) I(x_1 + t_1, x_2 + t_2).$$

It is common to use *convolution* to perform spatial filtering. The convolution of $w(x_1, x_2)$ and $I(x_1, x_2)$, denoted by $w(x_1, x_2) \star I(x_1, x_2)$, is defined as

$$w(x_1, x_2) \star I(x_1, x_2) = \sum_{t_1=-a}^{a} \sum_{t_2=-b}^{b} w(t_1, t_2) I(x_1 - t_1, x_2 - t_2).$$

As we can see, linear spatial filtering of an image with a filter is the same as convolution of the image with the filter rotated by 180°.

One common application of linear spatial filtering is to smooth images, which for example can be done in order to reduce noise in an image, or in order to blur an image to reduce irrelevant detail and make the detection of larger objects easier. A smoothing linear spatial filter is also called an averaging filter, since it computes the average of the pixels in its neighbourhood. The filter coefficients of a filter of size $m \times n$ which computes the standard average is $\frac{1}{mn}$. It is also possible to construct filters that compute weighted averages, for example giving more importance to the middle pixel than to the rest in order to reduce blurring.

2.2.4 Segmentation

In this thesis we work with one of the many tasks that the huge area digital image processing consists of; namely segmentation. This is one of the most difficult tasks in digital image processing, and it is an important basis for identifying and recognising objects in an image. To segment an image is to partition it into different parts based on some sort of region descriptors. If Ω represents the whole image domain (as before), segmentation is a process that divides Ω into n subregions Ω_i, such that

(i) $\bigcup_{i=1}^{n} \Omega_i = \Omega$,

(ii) Ω_i is a connected set for $i = 1, 2, \ldots, n$,

(iii) $\Omega_i \cap \Omega_j = \emptyset \ \forall \, i, j : \ i \neq j$,

(iv) $Q(\Omega_i) =$ TRUE for $i = 1, 2, \ldots, n$,

(v) $Q(\Omega_i \cup \Omega_j) =$ FALSE for any adjacent regions Ω_i and Ω_j,

where $Q(\Omega_k)$ is a logical predicate defined over the points in the set Ω_k [12]. There are many different ways to define this logical predicate, but the most common are based on either discontinuity or similarity of intensity values. Methods based on discontinuity of intensity values assume that the intensity values in the different regions are so different from each other that it is possible to detect edges between regions based on local discontinuities in intensity. Methods based on similarity of intensity values use some sort of measure to partition the image into regions where the measure is similar within each region. In this case one can for instance use the average intensity value as a measure to find regions with similar colour, the standard deviation to find regions with similar texture or more generally histograms to find regions with similar intensity distributions. In the next section we will describe one method where segmentation is based on average intensity values.

2.3 Chan-Vese segmentation model

One very popular and well-known model for segmentation is the Chan-Vese model [4]. We will as in [16] formulate this model for segmentation of a grey scale image $I : \Omega \to \mathbb{R}_+$ into two regions; background and foreground, where the foreground contains the object(s) we want to segment from the image. This model can easily be extended to colour images, but for simplicity of

notation we will only formulate it for grey scale images. The Chan-Vese segmentation model consists of minimising the functional

$$E_{CV}(u,c) = J(u) + \frac{\lambda}{2}\left\{\langle 1-u, (I-c_0)^2\rangle + \langle u, (I-c_1)^2\rangle\right\} \tag{2.13}$$

with respect to u and c. The variable u is the characteristic function of a region Σ which represents the foreground. $c = [c_0, c_1]$ where c_0 is the average grey value outside Σ and c_1 the average grey value inside Σ. The constant $\lambda > 0$ is a fixed weight, and $J(u) = \int_\Omega |\nabla u|\, dx$ is the total variation of u.

The idea of this model can easily be explained by using a simple example where the image I consists of two approximately constant regions Ω_0 and Ω_1 with distinct grey values. The two last terms in the energy functional will in this case ensure that the minimiser gives us a region Σ which equals the region Ω_1, which represents the object of interest. These terms (the two terms inside the curly brackets) can also be written as

$$T_1 + T_2 = \int_{\Omega\setminus\Sigma} (I-c_0)^2 dx + \int_\Sigma (I-c_1)^2\, dx, \tag{2.14}$$

since $u(x) = 1$ for $x \in \Sigma$ and $u(x) = 0$ for $x \in \Omega \setminus \Sigma$.

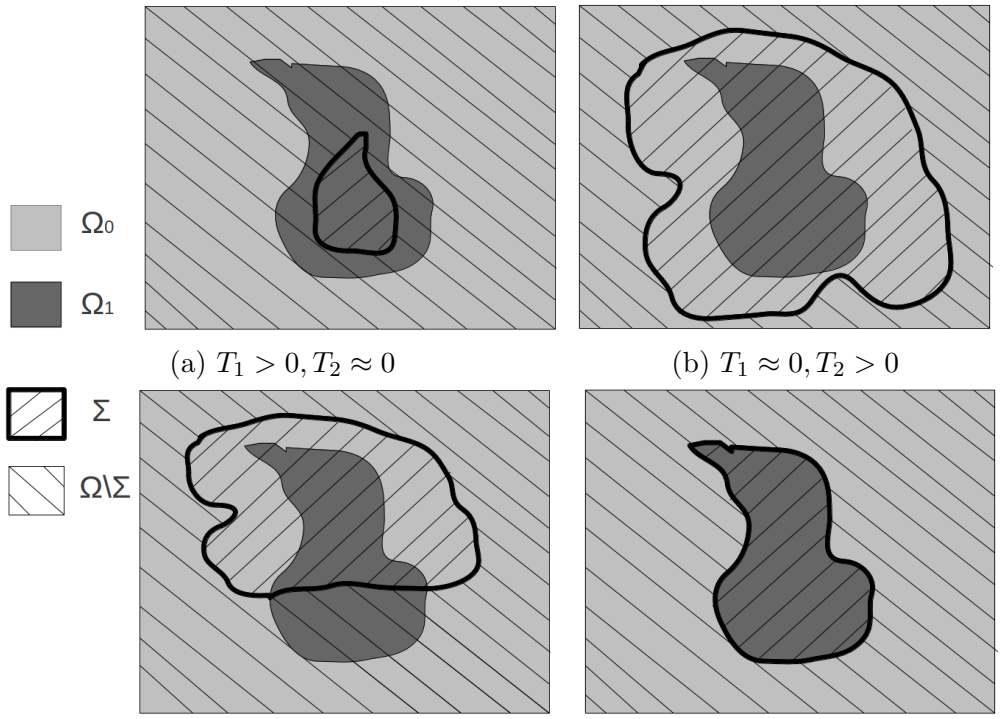
Then, if $\Sigma \in \Omega_1$, c_0 will be somewhere between the grey value inside Ω_0 and the grey value inside Ω_1 (in figure 2.6a you see that $\Omega \setminus \Sigma$ contains both the light and the dark colour), so $T_1 > 0$. c_1 will accordingly be equal to the grey value inside Ω_1 (Σ contains only the dark colour), so $T_2 \approx 0$. If on the other hand $\Omega_1 \in \Sigma$ as in figure 2.6b, then c_1 will have a value in between the grey value in Ω_0 and in Ω_1, so $T_2 > 0$, while c_0 will be approximately equal to the grey value in Ω_0, resulting in $T_1 \approx 0$. Furthermore, if some of Σ is inside Ω_0 and some is inside Ω_1, then both c_1 and c_0 will be in between the grey value inside Ω_0 and the grey value inside Ω_1, so both terms will be > 0 (see figure 2.6c). Finally, if $\Sigma = \Omega_1$ as in figure 2.6d, then c_1 will be equal to the grey value in Ω_1 and c_0 will be equal to the grey value in Ω_0, in which case both terms will be ≈ 0 and the minimum is attained as wanted.

2.3.1 Total variation

The total variation of $u \in L^1(\Omega)$, where $\Omega \subseteq \mathbb{R}^n$ is a bounded open domain, is in [3] and [11] defined as

$$J(u) = \sup\left\{\int_\Omega u \operatorname{div}(\xi)\, dx \mid \xi \in C_c^1(\Omega, \mathbb{R}^n), \|\xi\|_\infty \le 1\right\}. \tag{2.15}$$

$u \in L^1(\Omega)$ means that $u : \Omega \to \mathbb{R}$ is absolutely integrable and $\xi \in C_c^1(\Omega, \mathbb{R}^n)$ means that ξ is one time continuously differentiable and has compact support.



(a) $T_1 > 0, T_2 \approx 0$ (b) $T_1 \approx 0, T_2 > 0$

(c) $T_1 > 0, T_2 > 0$ (d) $T_1 \approx 0, T_2 \approx 0$

Figure 2.6: Chan-Vese fitting terms

A function u is said to be of bounded variation if $J(u) < \infty$, and the space $BV(\Omega)$ consists of all functions in $L^1(\Omega)$ with bounded variation.

If u is continuously differentiable, then $J(u) = \int_\Omega |\nabla u|\, dx$. This is shown below using the divergence theorem [8]:

$$\int_\Omega \operatorname{div}(F)\, dx = \int_{\partial\Omega} F \cdot \nu\, dS,$$

where ν is the outward unit normal vector along $\partial\Omega$. We let $F = \xi u$ and get from the product rule for differentiation that $\operatorname{div}(\xi u) = \operatorname{div}(\xi)u + \xi \cdot \nabla u$. The divergence theorem then gives us

$$\begin{aligned}\int_\Omega \xi \cdot \nabla u\, dx &= \int_\Omega \operatorname{div}(\xi u)\, dx - \int_\Omega \operatorname{div}(\xi) u\, dx \\ &= \int_{\partial\Omega} (\xi u) \cdot \nu\, dS - \int_\Omega \operatorname{div}(\xi) u\, dx.\end{aligned}$$

Since ξ has compact support, the first term on the right hand side vanishes and we have

$$\int_\Omega u \operatorname{div}(\xi)\, dx = -\int_\Omega \xi \cdot \nabla u\, dx \leq \int_\Omega |\xi \cdot \nabla u|\, dx \leq \int_\Omega |\xi||\nabla u|\, dx \leq \int_\Omega |\nabla u|\, dx,$$

where the last step comes from the fact that $\|\xi\|_\infty \leq 1$ and the previous step comes from the Cauchy-Schwarz inequality. When $\xi \to \frac{-\nabla u}{|\nabla u|}$ the supremum is attained and thus $J(u) = \int_\Omega |\nabla u|\, dx$.

If u is the characteristic function of $\Sigma \subset \Omega$ and $\partial\Sigma$ is C^1 then

$$\int_\Omega u \operatorname{div}(\xi)\, dx = \int_\Sigma \operatorname{div}(\xi)\, dx = \int_{\partial\Sigma} \xi \cdot \nu\, dS \leq \int_{\partial\Sigma} |\xi \cdot \nu|\, dS \leq \int_{\partial\Sigma} |\xi||\nu|\, dS \leq |\partial\Sigma|.$$

If we choose $\xi = \nu$ on $\partial\Sigma$ then $\int_{\partial\Sigma} \xi \cdot \nu\, dS = \int_{\partial\Sigma} |\nu|^2\, dS = \int_{\partial\Sigma} dS = |\partial\Sigma|$ and thus $J(u) = |\partial\Sigma|$. Therefore, minimising $J(u)$ is the same as minimising the length of the border of Σ. This is the reason why we need this term in (2.13); while the last terms give us a region Σ which best matches the image I in terms of mean intensity inside and outside the region, the first term controls the length of the border of this region.

2.3.2 The truncation lemma

There is, however, a problem with (2.13); namely that u is required to be a binary function. This leads to a non-convex optimisation problem which is hard to solve. Nevertheless, Chan, Esedoglu and Nikolova [2] discovered that for any fixed c this problem can be solved globally by relaxing the constraint on u to allow $u \in K$ where

$$K = \{u \in BV(\Omega) \mid 0 \leq u(x) \leq 1\ \forall x \in \Omega\}. \tag{2.16}$$

We then get a convex problem which is easy to solve. If $u_* \in K$ gives the minimum of (2.13) then the global minimiser u_*^t for the original problem with minimisation over binary u can be found simply by thresholding u_* with a threshold $t \in [0, 1]$:

$$u_*^t(x) = \begin{cases} 1 & \text{if } u_*(x) > t, \\ 0 & \text{otherwise,} \end{cases} \quad \forall x \in \Omega.$$

In order to see that this is indeed the global minimiser, we must first notice that E_{CV} can be rewritten as

$$J(u) + \frac{\lambda}{2}\langle 1, (I - c_0)^2\rangle + \frac{\lambda}{2}\{\langle -u, (I - c_0)^2\rangle + \langle u, (I - c_1)^2\rangle\}.$$

Noticing that the second term does not depend on u, we see that the minimisation over $u \in K$ of E_{CV}, when c is fixed, is equivalent to minimisation of the functional

$$\hat{E}_{CV}(u) = J(u) + \frac{\lambda}{2}\langle (I - c_1)^2 - (I - c_0)^2, u\rangle = J(u) + \langle g, u\rangle,$$

where $g = (I - c_1)^2 - (I - c_0)^2$. Now we can state the lemma and prove it as done in [16]:

The Truncation Lemma: *If* $u_* = \arg\inf_{u \in K} \hat{E}_{CV}(u)$ *then* $u_*^t = \arg\inf_{u \in \{0,1\}} \hat{E}_{CV}(u)$ *for almost all* $t \in [0, 1]$.

Proof. Because of the coarea formula, $J(u_*) = \int_0^1 J(u_*^t)\, dt$, and the layer cake representation, $\langle g, u_* \rangle = \int_0^1 \langle g, u_*^t \rangle\, dt$, we can write $\hat{E}_{CV}(u_*) = \int_0^1 \hat{E}_{CV}(u_*^t)\, dt$. Since u_* is the minimum value $\hat{E}_{CV}(u_*^t) \geq \hat{E}_{CV}(u_*)$. It is logical and well known that whenever $\int f(t)\, dt = 0$ and $f(t) \geq 0$ then $f(t) = 0$ for almost all t. This is exactly what we have here: $\int_0^1 (\hat{E}_{CV}(u_*^t) - \hat{E}_{CV}(u_*))\, dt = 0$ and $\hat{E}_{CV}(u_*^t) - \hat{E}_{CV}(u_*) \geq 0$ so $\hat{E}_{CV}(u_*^t) = \hat{E}_{CV}(u_*)$ for almost all t, i.e. $u_*^t = \arg\inf_{u \in K} \hat{E}_{CV}(u)$ also and since $u_*^t \in \{0, 1\}$ it therefore solves $\arg\inf_{u \in \{0,1\}} \hat{E}_{CV}(u)$. □

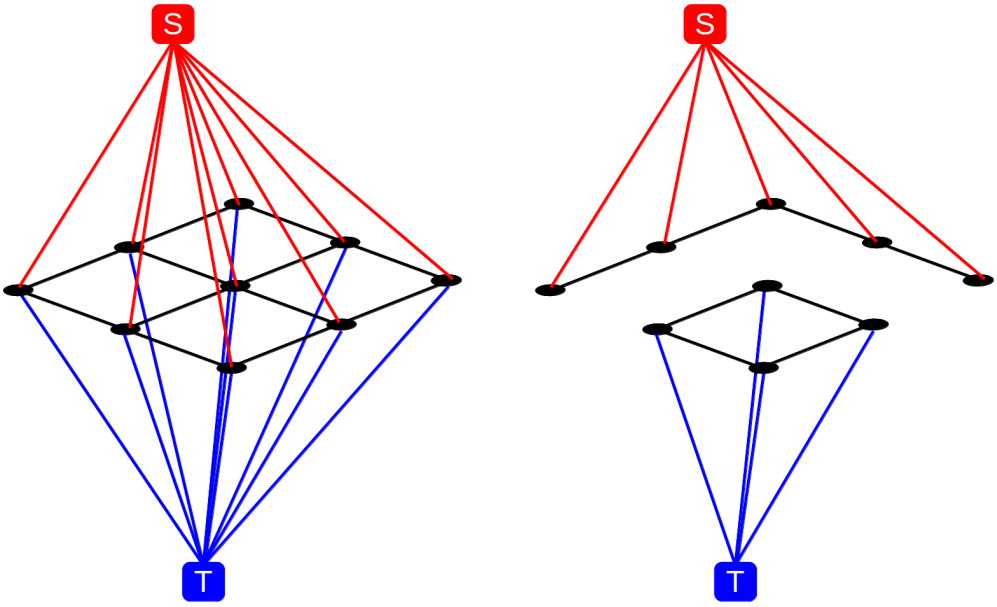
2.4 Max-flow and min-cut

An increasingly popular method for solving energy minimisation problems is to search for the minimal cut over an appropriately constructed graph. The classical theorem of min-cut and max-flow gives us an efficient way to compute this minimal cut: We maximise the corresponding flows instead. In the following sections we explain the min-cut and the max-flow problem and their connection.

2.4.1 Min-cut

A graph is a pair $(\mathcal{V}, \mathcal{E})$ composed of a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$. In connection with image processing the vertex set consists of all the image pixels together with two terminal vertices: the source s and the sink t. Every pixel is connected to its 4-neighbours by spatial edges e_n, as well as to both terminal vertices by the two terminal edges e_s and e_t (see figure 2.7a). Now the segmentation task is to partition all the pixels in Ω into two disjoint groups; background and foreground. If we associate the terminal node s with foreground and t with background then this can be thought of as cutting off one of the terminal edges from each pixel so that it either has an edge to s or an edge to t, and also cutting off all the spatial edges between pixels belonging to different groups (see figure 2.7b).

We have now divided the set $\mathcal{V}$ into one subset $\mathcal{V}_s$ containing the terminal vertex s and all the pixels connected to it, and another subset $\mathcal{V}_t$ containing the terminal vertex t and all the pixels connected to it. This two-partition



(a) (b)

Figure 2.7: Figure showing a graph consisting of 9 pixels, two terminal vertices and spatial and terminal edges. In 2.7b some of the edges are cut off; the pixels are partitioned into two groups. The cut consists of the edges that are shown in 2.7a but not in 2.7b.

cut we now have made is called an *s-t cut*. To find out where this cut should be made, we assign a non-negative cost $C(e) \geq 0$ to each edge $e \in \mathcal{E}$. The cut energy is defined as the sum of the costs of the edges that are cut, and we want to find the cut that minimises this energy:

$$\min_{\mathcal{E}_{st} \subset \mathcal{E}} \sum_{e \in \mathcal{E}_{st}} C(e) \text{ , where } \mathcal{E}_{st} = \{e \in \mathcal{E} \mid e = (v_1, v_2), v_1 \in \mathcal{V}_s, v_2 \in \mathcal{V}_t\}. \quad (2.17)$$

2.4.2 Max-flow

Another closely related way to look at this graph, which also can be called a network, is to think of each edge as a pipe. We want to find the maximum amount of water that can flow through these pipes from the source s to the sink t (and now these terms make much more sense than for the min-cut problem). The cost of an edge is now the capacity of this pipe, and for this pipe network we have some constraints that need to be satisfied: Naturally the flow p through a pipe cannot exceed the pipe's capacity, and also the amount

of water flowing out of a vertex should be the same as the amount of water flowing into the vertex. This can be formulated mathematically as follows:

- Capacity of source flows:

$$0 \leq p_s(v) \leq C_s(v), \tag{2.18}$$

  where $p_s(v)$ is shorthand for $p(e_s(v))$ and $C_s(v)$ is shorthand for $C(e_s(v))$.

- Capacity of sink flows:

$$0 \leq p_t(v) \leq C_t(v), \tag{2.19}$$

  where $p_t(v)$ is shorthand for $p(e_t(v))$ and $C_t(v)$ is shorthand for $C(e_t(v))$.

- Capacity of spatial flows:

$$|p(e_n)| \leq C(e_n). \tag{2.20}$$

  For the spatial edges e_n linking two vertices from $\mathcal{V} \setminus \{s, t\}$, the flow can have two directions. The flow through an edge $e_n = (v_1, v_2)$ is positive if it goes from v_1 to v_2 and negative if it goes from v_2 to v_1—this is the reason for the absolute value in the inequality above.

- Conservation of flows:

$$\sum_{w \in N_4(v)} p((v, w)) - p_s(v) + p_t(v) = 0, \tag{2.21}$$

  where $N_4(v)$ is a 4-connectivity neighbourhood of pixel v.

Thus the maximum flow problem amounts to maximising the amount of water flowing out from s subject to the above constraints:

$$\max_{p_s, p_t, p} \sum_{v \in \mathcal{V} \setminus \{s,t\}} p_s(v), \text{ subject to (2.18), (2.19), (2.20) and (2.21).} \tag{2.22}$$

2.4.3 Max-flow min-cut theorem

It can be shown that

Theorem 5.1 in [9]: *For any network the maximal flow value from s to t is equal to the minimal cut capacity of all cuts separating s and t.*

Here, the capacity of a cut is the sum of the capacities/costs of the arcs in the cut. That is, the theorem says that (2.17)=(2.22). This theorem connects max-flow problems with min-cut problems, so that when we want to find the minimum of an energy function by searching for the minimal cut over some graph, we can instead find it by maximising the corresponding flow, which gives us a much easier and faster algorithm.

2.4.4 Continuous max-flow and min-cut

The max-flow and min-cut problems can also be formulated in the continuous setting [21]. We now have a continuous set of points in a domain Ω in addition to the source s and the sink t. The constraints on flows are then:

$$p_s(x) \leq C_s(x), \quad \forall x \in \Omega; \tag{2.23}$$
$$p_t(x) \leq C_t(x), \quad \forall x \in \Omega; \tag{2.24}$$
$$|p(x)| \leq C(x), \quad \forall x \in \Omega; \tag{2.25}$$
$$\mathrm{div} p(x) - p_s(x) + p_t(x) = 0, \quad \text{a.e. } x \in \Omega; \tag{2.26}$$
$$p(x) \cdot \nu = 0 \quad \text{on } \partial\Omega, \tag{2.27}$$

where a.e. is short for "for almost every" and ν is the outward normal to the boundary $\partial\Omega$. The continuous max-flow model can be formulated as:

$$\sup_{p_s, p_t, p} \int_\Omega p_s(x)\, dx, \text{ subject to constraints (2.23) through (2.27).} \tag{2.28}$$

The constraint (2.26) is an equality constraint and we can thus use the method of Lagrangian multipliers to reduce the number of constraints. The problem is to find $r^* = (p_s^*, p_t^*, p^*)$ which satisfies (2.23), (2.24) and (2.25) such that

$$r^* = \arg\max_r f(r) \text{ s.t. } g^x(r^*) = 0 \quad \forall\, x \in \Omega,$$

where $f(r) = \int_\Omega p_s(x)\, dx$ and $g^x(r) = \mathrm{div}\, p(x) - p_s(x) + p_t(x)$. Since Ω is a continuous domain, we have infinitely many constraints. Thus, the Lagrange multiplier u is now a function, and we use the L^2 inner product in the Lagrangian function:

$$\begin{aligned} L(r, u) &= f(r) + \int_\Omega u(x) g^x(r)\, dx \\ &= \int_\Omega p_s(x)\, dx + \int_\Omega u(x)(\mathrm{div}\, p(x) - p_s(x) + p_t(x))\, dx \\ &= \int_\Omega [(1 - u(x)) p_s(x) + u(x) p_t(x) + u(x) \mathrm{div}\, p(x)]\, dx. \end{aligned}$$

We must find a saddle point of L subject to (2.23), (2.24) and (2.25) in order to solve (2.28). L is linear in both r and u and hence concave u.s.c. for fixed u and convex l.s.c. for fixed p, p_s and p_t. Since in addition the constraints on flows are convex, the conditions of the minimax theorem [7]

are satisfied. We know therefore that at least one saddle point exists, which can be found by solving

$$\min_u \sup_{p_s,p_t,p} \left\{ \int_\Omega [(1-u(x))p_s(x) + u(x)p_t(x) + u(x)\mathrm{div}\, p(x)]\, dx \right\} \qquad (2.29)$$
$$\text{s.t.} \quad p_s(x) \le C_s(x), \quad p_t(x) \le C_t(x), \quad |p(x)| \le C(x) \quad \forall x \in \Omega.$$

This is called the primal-dual model [21] and is equivalent to the continuous maximal flow model (2.28) which also is called the primal model.

In order to optimise (2.29) over the flow variables p_s, p_t and p we first look at the generalised maximisation problem

$$f(q) = \sup_{p \le C} p\, q. \qquad (2.30)$$

If $q < 0$ then $p\,q$ is maximised if $p = -\infty$ which gives $f(q) = \infty$. If $q = 0$, it does not matter what p is chosen to be; $f(q) = 0$ anyhow. If $q > 0$, then $p\,q$ is maximised if $p = C$ which gives $f(q) = C\,q$. That is,

$$f(q) = \begin{cases} Cq & \text{if } q \ge 0, \\ \infty & \text{if } q < 0. \end{cases} \qquad (2.31)$$

Now, we can rearrange the primal-dual model (2.29) as

$$\min_u \left\{ \int_\Omega [\sup_{\substack{p_s(x) \\ \le C_s(x)}} p_s(x)(1-u(x)) + \sup_{\substack{p_t(x) \\ \le C_t(x)}} p_t(x)u(x)]\, dx + \sup_{\substack{|p(x)| \\ \le C(x)}} \int_\Omega u(x)\mathrm{div}\, p(x)\, dx \right\}. \qquad (2.32)$$

Similar to the discussion in section 2.3.1 and due to (2.27), notice that

$$\sup_{|p(x)| \le C(x)} \int_\Omega u(x)\mathrm{div}\, p(x)\, dx = \int_\Omega C(x)|\nabla u(x)|\, dx. \qquad (2.33)$$

If we now restrict $u \in K$, then both $1 - u(x)$ and $u(x)$ is non-negative (remember the definition of K in (2.16)). By comparing the two first terms in (2.32) to (2.30) we thus see that by (2.31) and (2.33), the primal-dual model (2.32) can be written as the equivalent dual model

$$\min_{u \in K} \left\{ \int_\Omega [(1-u(x))C_s(x) + u(x)C_t(x) + C(x)|\nabla u(x)|]\, dx \right\}. \qquad (2.34)$$

If $u \notin K$ then the energy would be infinite, but since we know that a saddle point exists, this cannot be the case. So the primal model (2.28), the primal-dual model (2.29) and the dual model (2.34) are all equivalent to each other

[21]. The dual model can also be called the continuous min-cut model, and hence we have just showed that the max-flow min-cut theorem is also valid for the continuous case. We see that if u is the characteristic function for $\mathcal{V}_s \setminus \{s\}$, then (2.34) is

$$\min_{\mathcal{V}_s} \left\{ \int_{\mathcal{V}_t \setminus \{t\}} C_s(x)\, dx + \int_{\mathcal{V}_s \setminus \{s\}} C_t(x)\, dx + \int_{\partial(\mathcal{V}_s \setminus \{s\})} C(x)\, dx \right\},$$

and so it is easy to see the connection with (2.17) if $C_s(x)$ are the costs on terminal edges from the terminal s, $C_t(x)$ are the costs on terminal edges to the terminal t and $C(x)$ are the costs on spatial edges. The pixels that belong to $\mathcal{V}_t \setminus \{t\}$ have the edges from s cut off so they contribute to the cut energy with $C_s(x)$, whereas the pixels that belong to $\mathcal{V}_s \setminus \{s\}$ contribute with $C_t(x)$. Along the edge of $\mathcal{V}_s \setminus \{s\}$ the spatial edges are cut off, and here $C(x)$ contributes to the energy.

Chapter 3

The proposed method

In this chapter we continue by describing the method we propose for shape prior segmentation. We largely follow the discussion in [16], and combine the results here with the max-flow based algorithm proposed in [21]. For the sake of simplicity we describe the method only for grey scale images, but it is easily extended to colour images and this is in fact how we have implemented the method.

3.1 Reformulating the Chan-Vese functional

The starting point for our method is the Chan-Vese segmentation model described in chapter 2.3, where we want to include a priori shape information into the segmentation process. The natural choice for representation of the prior shape Σ' is a characteristic function f, since this is how the segmented region Σ is represented in the Chan-Vese model. The shape prior can be obtained from an already segmented image of an object similar to the object we are going to segment, or from a database of shapes from the relevant object class. In the latter case, the mean of these shapes can be used as the shape prior. There are also several other possible ways to utilise such a database, for example by incorporating more of the statistical information available. Nevertheless, this will not be done in this work, where we will use a very simple shape prior term; namely the squared norm of the difference between the prior shape and the segmented shape. Thus the functional to be minimised is

$$E(u, c, f) = E_{CV}(u, c) + \frac{\eta}{2}\|u - f\|^2 \tag{3.1}$$

for some $\eta > 0$. Because of the quadratic term, this problem cannot be solved simply by relaxing the constraint on u from $u(x) \in \{0, 1\}$ to $u \in K$

and using the truncation lemma as we can do with (2.13). Therefore it is necessary to reformulate the problem.

For binary u the CV-functional (2.13) may be rewritten as

$$E_{CV}(u,c) = J(u) + \frac{\lambda}{2}\|I - I_{\text{model}}\|^2, \tag{3.2}$$

where $I_{\text{model}} = c_0(1-u) + c_1 u$ (see figure 3.1). This is easy to see because $u(x) = 1$ for $x \in \Sigma$ and $u(x) = 0$ for $x \in \Omega \setminus \Sigma$, so

$$\int_\Omega [(1-u)(I-c_0)^2 + u(I-c_1)^2]\,dx = \int_\Sigma (I-c_1)^2\,dx + \int_{\Omega\setminus\Sigma} (I-c_0)^2\,dx$$

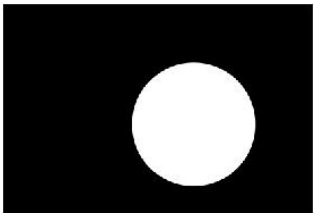
and

$$\int_\Omega (I-I_{\text{model}})^2\,dx = \int_\Omega (I-c_0(1-u)-c_1u)^2\,dx = \int_\Sigma (I-c_1)^2\,dx + \int_{\Omega\setminus\Sigma} (I-c_0)^2\,dx.$$

(a) u.

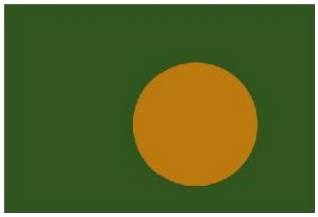
(b) The contour of Σ shown on the image I.

(c) I_{model}.



(d) Another u.

(e) The contour of the corresponding Σ shown on the image I.



(f) I_{model}.

Figure 3.1: Illustration of I_{model} corresponding to two different u. It is easy to see that the last I_{model} (figure 3.1f) is much more similar to the image I than the first I_{model} (figure 3.1c), which means that the last u (figure 3.1d) gives the minimum energy (3.2) of these two and therefore gives the best segmentation.

If we also define a shape prior image model $I_{\text{prior}} = b_0(1-f) + b_1 f$ for grey values b_0 and b_1, then shape prior segmentation can be formulated as the minimisation over all characteristic functions u of the functional

$$E(u,c,f,b) = E_{CV} + E_{\text{prior}} = J(u) + \frac{\lambda}{2}\|I - I_{\text{model}}\|^2 + \frac{\mu}{2}\|I_{\text{model}} - I_{\text{prior}}\|^2. \quad (3.3)$$

The connection with (3.1) is easily seen when we notice that close to convergence $b_0 \approx c_0$ and $b_1 \approx c_1$. If these equalities are exact, we have

$$\begin{aligned}\frac{\mu}{2}\|I_{\text{model}} - I_{\text{prior}}\|^2 &= \frac{\mu}{2}\|c_0(1-u) + c_1 u - c_0(1-f) - c_1 f\|^2 \\ &= \frac{\mu}{2}\| - c_0 u + c_1 u + c_0 f - c_1 f\|^2 = \frac{\mu}{2}(c_1 - c_0)^2\|u - f\|^2.\end{aligned} \quad (3.4)$$

With this functional we do not only consider the difference between the two shapes as in (3.1), but also the differences in average intensity values inside and outside the regions represented by u and f.

We want to minimise (3.3), and we minimise with respect to one variable at a time while keeping the others fixed.

3.2 Minimising w.r.t. u and c

Now, prior data b and f are kept fixed at iteration k, and u and c are to be updated. $E(u,c,f,b)$ can be reformulated by completing the squares:

$$\begin{aligned}&\frac{\lambda}{2}(I - I_{\text{model}})^2 + \frac{\mu}{2}(I_{\text{model}} - I_{\text{prior}})^2 \\ &= \frac{\lambda}{2}I^2 - \lambda I I_{\text{model}} + \frac{\lambda}{2}I_{\text{model}}^2 + \frac{\mu}{2}I_{\text{model}}^2 - \mu I_{\text{model}} I_{\text{prior}} + \frac{\mu}{2}I_{\text{prior}}^2 \\ &= \frac{\lambda+\mu}{2}I_{\text{model}}^2 - (\lambda I + \mu I_{\text{prior}})I_{\text{model}} + (\frac{\lambda}{2}I^2 + \frac{\mu}{2}I_{\text{prior}}^2) \\ &= \frac{\lambda+\mu}{2}\left[I_{\text{model}}^2 - 2\frac{\lambda I + \mu I_{\text{prior}}}{\lambda+\mu}I_{\text{model}} + \frac{\lambda I^2 + \mu I_{\text{prior}}^2}{\lambda+\mu}\right] \\ &= \frac{\lambda+\mu}{2}\left[I_{\text{model}}^2 - 2\frac{\lambda I + \mu I_{\text{prior}}}{\lambda+\mu}I_{\text{model}} + \frac{(\lambda I + \mu I_{\text{prior}})^2}{(\lambda+\mu)^2}\right. \\ &\qquad\qquad \left. + \frac{\lambda I^2 + \mu I_{\text{prior}}^2}{\lambda+\mu} - \frac{(\lambda I + \mu I_{\text{prior}})^2}{(\lambda+\mu)^2}\right]\end{aligned}$$

$$= \frac{\lambda+\mu}{2}\Bigg[\left(I_{\text{model}} - \frac{\lambda I + \mu I_{\text{prior}}}{\lambda+\mu}\right)^2 + \frac{\lambda^2 I^2 + \mu\lambda I_{\text{prior}}^2 + \mu\lambda I^2 + \mu^2 I_{\text{prior}}^2 - \lambda^2 I^2 - 2\lambda\mu I I_{\text{prior}} - \mu^2 I_{\text{prior}}^2}{(\lambda+\mu)^2}\Bigg]$$

$$= \frac{\lambda+\mu}{2}\left[\left(I_{\text{model}} - \frac{\lambda I + \mu I_{\text{prior}}}{\lambda+\mu}\right)^2 + \frac{\lambda\mu}{(\lambda+\mu)^2}(I - I_{\text{prior}})^2\right],$$

and so we get

$$E(u, c, f, b) = J(u) + \frac{\lambda+\mu}{2}\|I_{\text{model}} - I_{\text{eff}}\|^2 + \frac{\lambda\mu}{2(\lambda+\mu)}\|I - I_{\text{prior}}\|^2,$$

where $I_{\text{eff}} = \frac{\lambda}{\lambda+\mu}I + \frac{\mu}{\lambda+\mu}I_{\text{prior}}$ (see figure 3.2).

The last square does not depend on u and c and can hence be disregarded. Then, we update u and c by minimising the CV-functional

$$E_{CVe}(u, c) = J(u) + \frac{\lambda+\mu}{2}\left\{\langle 1-u, (I_{\text{eff}}^k - c_0)^2\rangle + \langle u, (I_{\text{eff}}^k - c_1)^2\rangle\right\}, \quad (3.5)$$

following the same reasoning as in the reformulation from (2.13) to (3.2) above, only backwards.

Now, the quadratic term which gave us problems in (3.1) is gone. Comparing (3.5) with (2.13), we see that the only difference is $\lambda+\mu$ instead of just λ and I_{eff}^k instead of I, and so following the discussion in section 2.3.2, this problem can be solved by relaxing the constraint on u from $u(x) \in \{0, 1\}$ to $u \in K$ and truncating the solution.

It can be shown that this functional is bi-convex, i.e. convex in u when c is kept fixed and convex in c when u is kept fixed. Therefore, we first minimise with respect to u and then with respect to c:

$$u^{k+1} = \arg\min_{u \in K} E_{CVe}(u, c^k), \quad (3.6)$$

$$c^{k+1} = \arg\min_{c \in \mathbb{R}^2} E_{CVe}(u^{k+1}, c) \quad (3.7)$$

3.2.1 Minimising w.r.t. u

For the sub-problem (3.6) we depart from the discussion in [16], and use instead a continuous max-flow and min-cut approach to find the minimum. We see that if we put $C_s(x) = (I_{\text{eff}}^k(x) - c_0^k)^2$, $C_t(x) = (I_{\text{eff}}^k(x) - c_1^k)^2$ and $C(x) = \frac{2}{\lambda+\mu}$, then minimising (3.5) over $u \in K$ is equivalent to solving (2.34), so our problem is in fact a min-cut problem. The reasoning behind

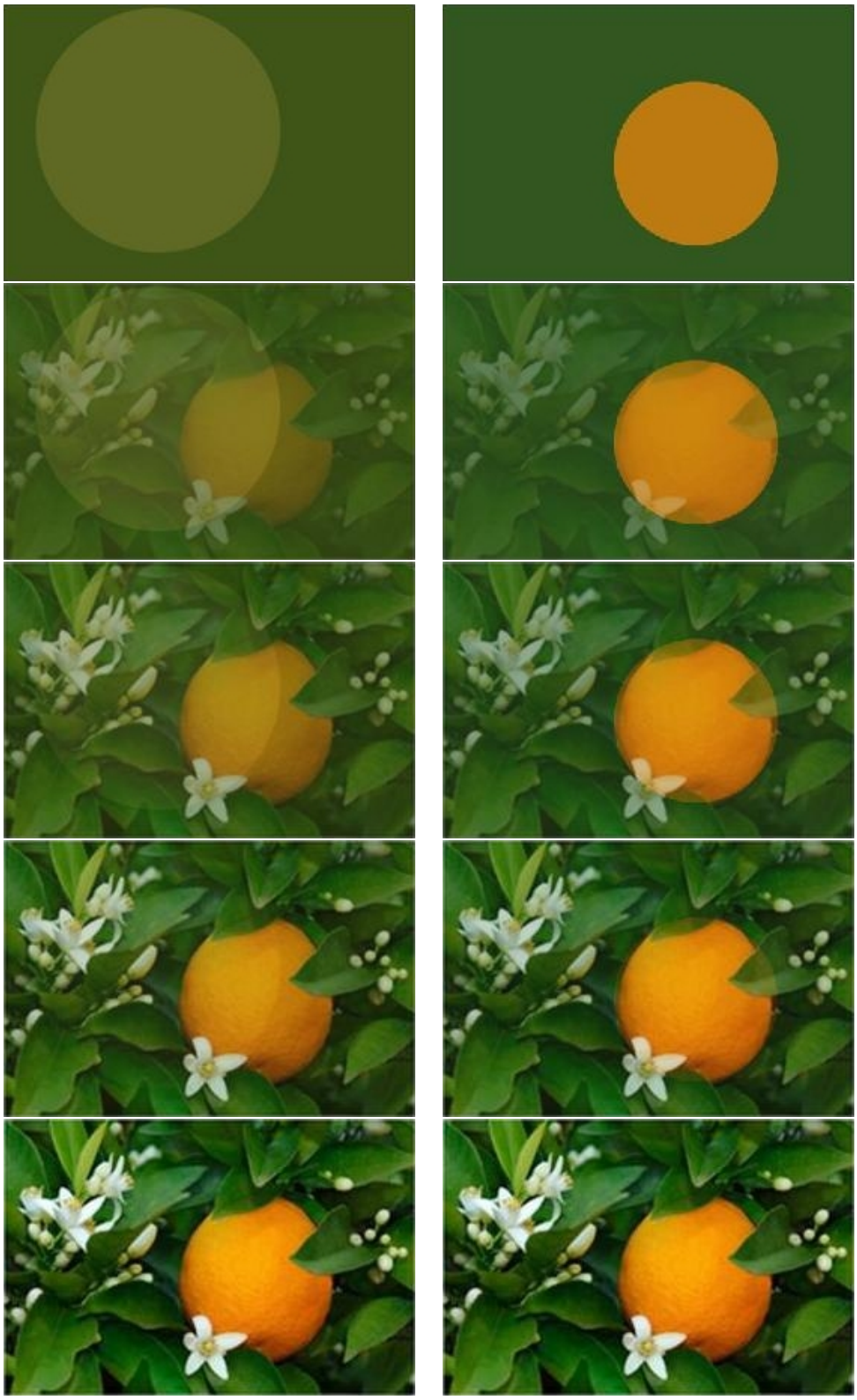


Figure 3.2: Illustration of I_{eff} where f is equal to the u in figure 3.1a in the first column and to the u in figure 3.1d in the second column. The first row is with $\lambda = 0$ and $\mu = 4$, the second with $\lambda = 1$ and $\mu = 3$, the third with $\lambda = 2$ and $\mu = 2$, the fourth with $\lambda = 3$ and $\mu = 1$ and the fifth with $\lambda = 4$ and $\mu = 0$.

these cost expressions is as follows: C_s is the cost of cutting off e_s, that is the cost of labelling a pixel as t; as background. Since c_0 is the average intensity value in the background it makes sense that C_s is the difference between the intensity value and c_0: If they are close, it is natural to label the pixel as background so the cost is small, but if they are very different the cost is high because there should be a good reason to label the pixel as background if the intensity value is closer to the foreground intensity value. Here, where we look at the intensity value of I_{eff} instead of I in the cost expressions, we also take the shape prior information into account. The constant $C(x)$, which we from now on call α, is a measure of how important it is that the segmented region has a short boundary.

Now, since (2.34) is equivalent to (2.28), we use an algorithm for (2.28) to minimise w.r.t. u. The algorithm for continuous max-flow is based upon the augmented Lagrangian method. In addition to adding the Lagrangian term for the equality constraint $\text{div}p(x) - p_s(x) + p_t(x) = 0$ as in the primal-dual model (2.29), we also add a penalty term and arrive at the augmented Lagrangian function

$$L_a(p_s, p_t, p, u) = \int_\Omega p_s \, dx + \int_\Omega u(\text{div}p - p_s + p_t) \, dx - \frac{\gamma}{2}\|\text{div}p - p_s + p_t\|^2, \quad (3.8)$$

where $\gamma > 0$. In order to update p_s, p_t and p, this function should be maximised subject to the constraints (2.23), (2.24) and (2.25). If now $\text{div}p(x) - p_s(x) + p_t(x)$ is far from 0, the negative penalty term will be large in absolute value, which is something we need to avoid when maximising L_a. We optimise (3.8) with respect to one variable at a time while fixing the others (so-called alternating directions method of multipliers [21]), and repeat the steps until convergence:

1. Optimising p by fixing other variables:

$$p^{k+1} = \arg\max_{\|p\|_\infty \leq \alpha} L_a(p_s^k, p_t^k, p, u^k).$$

The first term is independent of p and can hence be disregarded. We can complete the square in the two last terms:

$$\begin{aligned} &u^k(\text{div}p - p_s^k + p_t^k) - \frac{\gamma}{2}(\text{div}p - p_s^k + p_t^k)^2 \\ &= -\frac{\gamma}{2}\left((\text{div}p - p_s^k + p_t^k)^2 - \frac{2u^k}{\gamma}(\text{div}p - p_s^k + p_t^k)\right). \end{aligned}$$

We want this to be equal to $-\frac{\gamma}{2}(a^2 - 2ab)$ so we choose $a = \text{div}\, p - p_s^k + p_t^k$ and $b = \frac{u^k}{\gamma}$. Since $-\frac{\gamma}{2}b^2$ is independent of p it can be added to the

expression, so we get $-\frac{\gamma}{2}(a^2 - 2ab + b^2) = -\frac{\gamma}{2}(a-b)^2$ and

$$p^{k+1} = \underset{\|p\|_\infty \leq \alpha}{\arg\max} \; -\frac{\gamma}{2}\|\mathrm{div}p - M^k\|^2,$$

where $M^k = p_s^k - p_t^k + \frac{u^k}{\gamma}$. Because of the divergence term, we cannot find an exact solution to this maximisation problem, and we therefore use the gradient descent method. Then we have to find the Gateaux derivative of $G(p) = \int_\Omega \frac{\gamma}{2}(\mathrm{div}p - M^k)^2\,dx$:

$$\begin{aligned}
G'(p;d) &= \lim_{h\to 0} \frac{G(p+hd) - G(p)}{h} \\
&= \lim_{h\to 0} \frac{\gamma}{2h} \int_\Omega [(\mathrm{div}p + h\mathrm{div}d - M^k)^2 - (\mathrm{div}p - M^k)^2]\,dx \\
&= \lim_{h\to 0} \frac{\gamma}{2h} \int_\Omega h\mathrm{div}d(2\mathrm{div}p + h\mathrm{div}d - 2M^k)\,dx \\
&= \lim_{h\to 0} \frac{\gamma}{2} \int_\Omega \mathrm{div}d(2\mathrm{div}p + h\mathrm{div}d - 2M^k)\,dx \\
&= \gamma \int_\Omega \mathrm{div}d(\mathrm{div}p - M^k)\,dx,
\end{aligned}$$

where the third equality follows from the fact that $a^2 - b^2 = (a-b)(a+b)$. To get rid of the divergence of d we use the divergence theorem: $\int_\Omega \mathrm{div}F\,dx = \int_{\partial\Omega} F\cdot\nu\,dS$ with $F = d(\mathrm{div}p - M^k)$:

$$\mathrm{div}(d(\mathrm{div}p - M^k)) = \mathrm{div}(d)(\mathrm{div}p - M^k) + d\cdot\nabla(\mathrm{div}p - M^k), \text{ hence}$$

$$\begin{aligned}
&\gamma \int_\Omega \mathrm{div}d(\mathrm{div}p - M^k)\,dx \\
&\qquad = \gamma \int_\Omega \mathrm{div}(d(\mathrm{div}p - M^k))\,dx - \gamma \int_\Omega d\cdot\nabla(\mathrm{div}p - M^k)\,dx \\
&\qquad = \gamma \int_{\partial\Omega} d(\mathrm{div}p - M^k)\cdot\nu\,dS - \gamma \int_\Omega d\cdot\nabla(\mathrm{div}p - M^k)\,dx.
\end{aligned}$$

We impose $(\mathrm{div}p - M^k)\cdot\nu = 0$ on $\partial\Omega$ and get $G'(p;d) = \langle G'(p), d\rangle$ where $G'(p) = -\gamma\nabla(\mathrm{div}p - M^k)$. One gradient descent step with step size ε and a projection to enforce $\|p\|_\infty \leq \alpha$ then gives us an approximate solution to the maximisation problem:

$$p^{k+1} = \mathrm{proj}_\alpha\left(p^k - \varepsilon G'(p^k)\right) = \mathrm{proj}_\alpha\left(p^k + \tilde{\gamma}\nabla(\mathrm{div}p^k - M^k)\right), \quad (3.9)$$

where $\tilde{\gamma} = \varepsilon\gamma$ and proj_α is the projection onto the convex set $S_\alpha = \{q \mid \|q\|_\infty \leq \alpha\}$.

2. Optimising p_s by fixing other variables:

$$p_s^{k+1} = \underset{p_s(x)\leq C_s(x)}{\arg\max}\ L_a(p_s, p_t^k, p^{k+1}, u^k).$$

We complete the square as above and get

$$p_s^{k+1} = \underset{p_s(x)\leq C_s(x)}{\arg\max} \int_\Omega p_s\, dx - \frac{\gamma}{2}\|p_s - N^k\|^2,$$

where $N^k = \mathrm{div}p^{k+1} + p_t^k - \frac{u^k}{\gamma}$. Here, we can maximise at each point to maximise the integral since the points are independent of each other (we do not have any divergence or gradient term or suchlike). The integrand is concave with respect to $p_s(x)$, so we can find the maximum point simply by setting the derivative of the integrand w.r.t $p_s(x)$ equal to zero:

$$p_s^*(x) = \underset{p_s(x)}{\arg\max}\, p_s(x) - \frac{\gamma}{2}(p_s(x) - N^k(x))^2$$

$$\Leftrightarrow 1 - \gamma(p_s^*(x) - N^k(x)) = 0 \Leftrightarrow p_s^*(x) = \frac{1}{\gamma} + N^k(x).$$

Since we also need $p_s(x) \leq C_s(x)$ we take the minimum of these values:

$$p_s^{k+1} = \min\{\frac{1}{\gamma} + N^k, C_s\}. \tag{3.10}$$

3. Optimising p_t by fixing other variables:

$$p_t^{k+1} = \underset{p_t(x)\leq C_t(x)}{\arg\max}\ L_a(p_s^{k+1}, p_t, p^{k+1}, u^k) \ = \underset{p_t(x)\leq C_t(x)}{\arg\max}\ -\frac{\gamma}{2}\|p_t - O^k\|^2,$$

where $O^k = -\mathrm{div}p^{k+1} + p_s^{k+1} + \frac{u^k}{\gamma}$. As above, the integrand is concave and we maximise at each point:

$$p_t^*(x) = \underset{p_t(x)}{\arg\max}\ -\frac{\gamma}{2}(p_t(x) - O^k(x))^2$$

$$\Leftrightarrow -\gamma(p_t^*(x) - O^k(x)) = 0 \Leftrightarrow p_t^*(x) = O^k(x).$$

Since we also need $p_t(x) \leq C_t(x)$ we again take the minimum of these values:

$$p_t^{k+1} = \min\{O^k, C_t\}. \tag{3.11}$$

4. The Lagrange multiplier u is updated by one step of gradient descent, with γ as time step according to the augmented Lagrangian method:

$$u^{k+1} = u^k - \gamma(\mathrm{div}p^{k+1} - p_s^{k+1} + p_t^{k+1}). \tag{3.12}$$

Even though γ could also be updated (increased), we have chosen to keep it fixed.

3.2.2 Minimising w.r.t. c

The subproblem (3.7) is a simple quadratic optimisation problem, and we can easily find the solution by setting the gradient equal to zero and solving the equation:

$$c_0^{k+1} = \arg\min_{c_0} E_{CVe}(u^{k+1}, c_0, c_1^k) \Longleftrightarrow \frac{\partial E_{CVe}}{\partial c_0}(u^{k+1}, c_0^{k+1}, c_1^k) = 0$$

$$\begin{aligned}\frac{\partial E_{CVe}}{\partial c_0}(u^{k+1}, c_0^{k+1}, c_1^k) &= \frac{\lambda+\mu}{2}\langle 1 - u^{k+1}, -2(I_{\text{eff}}^k - c_0^{k+1})\rangle \\ &= -(\lambda+\mu)\langle 1 - u^{k+1}, I_{\text{eff}}^k\rangle + c_0^{k+1}(\lambda+\mu)\langle 1 - u^{k+1}, 1\rangle = 0 \\ \Longleftrightarrow c_0^{k+1}\langle 1 - u^{k+1}, 1\rangle &= \langle 1 - u^{k+1}, I_{\text{eff}}^k\rangle \Longleftrightarrow c_0^{k+1} = \frac{\langle 1 - u^{k+1}, I_{\text{eff}}^k\rangle}{\langle 1 - u^{k+1}, 1\rangle}.\end{aligned} \tag{3.13}$$

Similarly we get

$$\begin{aligned}\frac{\partial E_{CVe}}{\partial c_1}(u^{k+1}, c_0^{k+1}, c_1^{k+1}) &= \frac{\lambda+\mu}{2}\langle u^{k+1}, -2(I_{\text{eff}}^k - c_1^{k+1})\rangle \\ &= (\lambda+\mu)[-\langle u^{k+1}, I_{\text{eff}}^k\rangle + c_1^{k+1}\langle u^{k+1}, 1\rangle] = 0,\end{aligned}$$

hence

$$c_1^{k+1} = \frac{\langle u^{k+1}, I_{\text{eff}}^k\rangle}{\langle u^{k+1}, 1\rangle}. \tag{3.14}$$

3.3 Minimising w.r.t. b

Suppose that c and u have been updated and are now held fixed. f is also fixed, and we minimise (3.3) w.r.t. b. The calculation is similar to the calculation for c and gives

$$b_0^{k+1} = \frac{\langle 1 - f^k, I_{\text{model}}^{k+1}\rangle}{\|1 - f^k\|^2} \qquad \text{and} \qquad b_1^{k+1} = \frac{\langle f^k, I_{\text{model}}^{k+1}\rangle}{\|f^k\|^2}. \tag{3.15}$$

3.4 Updating f

In this thesis as in [16], we consider pose invariant priors, i.e. $f = f_0 \circ T$ where f_0 is a shape template and T is a similarity transformation. However, unlike Overgaard et al., we use a transformation with rotation about a specified centre, which is the centre of the region represented by the characteristic

function f_0. The reason for this is that with rotation about the origin (i.e. the upper left corner of the image) as in [16], it is easier to get stuck in local minima. Indeed, if f represents the same object as u with the same size and at the same place only a bit rotated, we would with this method be at a local minimum. Rotation of f about the origin would give a displacement of f from u and result in a larger difference between f and u, although rotation about the centre of the object would result in $f = u$ (see figure 3.3).

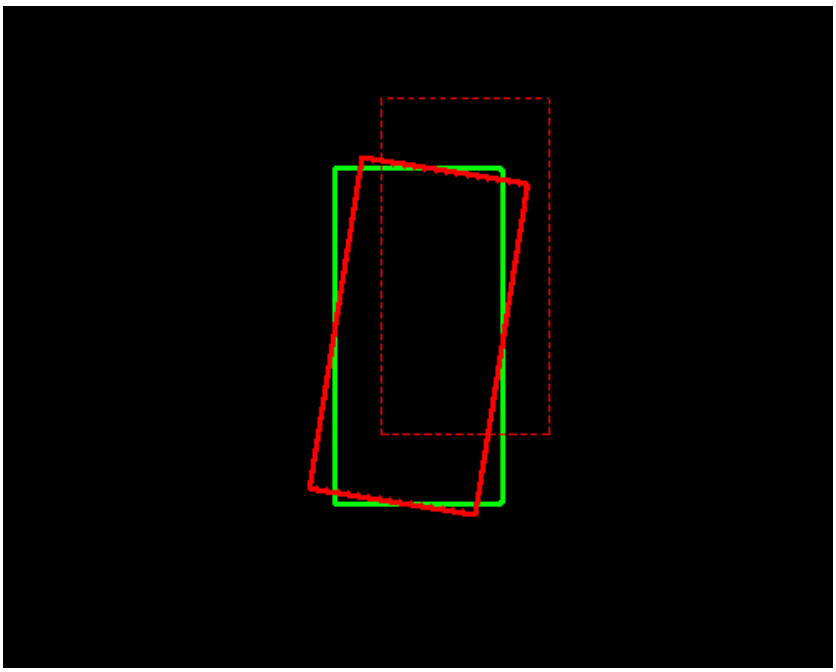


Figure 3.3: The red line is the contour of f and the green line is the contour of u. f is rotated $\frac{\pi}{20}$ radians clockwise about the centre of the region represented by u. The dashed red line shows f after rotation $\frac{\pi}{20}$ radians counterclockwise about the origin.

Therefore, $f(x) = T^* f_0(x) = (f_0 \circ T)(x) = f_0(T(x))$ where the transformation is a mapping of the form

$$y = T(x) = \varsigma^{-1} R^{-1}(x - a - s) + s,$$

where $s \in \mathbb{R}^2$ is the centre of rotation, $\varsigma > 0$ denotes scaling, R rotation and a translation. The natural parameterisation used is

$$a \in \mathbb{R}^2, \quad \varsigma = e^{\sigma} (\sigma \in \mathbb{R}), \quad R(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} (\phi \in \mathbb{R}),$$

so $y = e^{-\sigma}R(-\phi)(x - a - s) + s$ and

$$\begin{aligned} f(x_1, x_2) &= f_0(y_1, y_2) \\ &= f_0(e^{-\sigma}((x_1 - a_1 - s_1)\cos\phi + (x_2 - a_2 - s_2)\sin\phi) + s_1, \\ &\qquad e^{-\sigma}(-(x_1 - a_1 - s_1)\sin\phi + (x_2 - a_2 - s_2)\cos\phi) + s_2). \end{aligned} \tag{3.16}$$

Now, we simplify our model (3.3) a little. E_{CV} is independent of f and so we only need to consider E_{prior}. If we use the simplification in (3.4), we have

$$E_{\text{prior}} = \inf_T \left\{ \frac{\mu}{2}(c_1 - c_0)^2 \int_\Omega (u(x) - f_0(T(x)))^2 \, dx \right\}. \tag{3.17}$$

In order to update f we need to find the mapping T which gives the infimum above. That is, we need to update the pose parameters $\rho = (\rho_1, \rho_2, \rho_3, \rho_4)$ $= (a_1, a_2, \sigma, \phi)$. We can denote the mapping by $T(\rho)$, so the shape prior $f(x) = T(\rho)^* f_0(x)$. We ignore the constants $\mu(c_1 - c_0)^2$ and use gradient descent on the function $E(\rho) = \frac{1}{2}\int_\Omega (u(x) - T(\rho)^* f_0(x))^2 \, dx$. The pose parameters are updated as

$$a^{k+1} = a^k - \varepsilon_a \frac{\partial E}{\partial a}, \qquad \sigma^{k+1} = \sigma^k - \varepsilon_\sigma \frac{\partial E}{\partial \sigma}, \qquad \phi^{k+1} = \phi^k - \varepsilon_\phi \frac{\partial E}{\partial \phi},$$

where $\frac{\partial E}{\partial a} = (\frac{\partial E}{\partial a_1}, \frac{\partial E}{\partial a_2})$. The partial derivatives of E are given by

$$\frac{\partial E}{\partial \rho_i} = \int_\Omega (T(\rho)^* f_0(x) - u(x)) \frac{\partial}{\partial \rho_i}(T(\rho)^* f_0(x)) \, dx = \langle f - u, \frac{\partial (T^* f_0)}{\partial \rho_i} \rangle. \tag{3.18}$$

Before we calculate the partial derivatives $\frac{\partial}{\partial \rho_i}(T^* f_0)$ we notice that

$$\nabla_x f = \nabla_x (T^* f_0) = e^{-\sigma} \begin{bmatrix} f_{0y_1}\cos\phi - f_{0y_2}\sin\phi \\ f_{0y_1}\sin\phi + f_{0y_2}\cos\phi \end{bmatrix},$$

where $\nabla_x f$ is the gradient of f with respect to the spatial coordinates x_1 and x_2. Now we calculate the partial derivatives:

$$\begin{aligned} \frac{\partial (T^* f_0(x))}{\partial a_1} &= f_{0y_1}(x)(-e^{-\sigma}\cos\phi) + f_{0y_2}(x) e^{-\sigma}\sin\phi \\ \frac{\partial (T^* f_0(x))}{\partial a_2} &= f_{0y_1}(x)(-e^{-\sigma}\sin\phi) + f_{0y_2}(x)(-e^{-\sigma}\cos\phi) \\ \Rightarrow \frac{\partial (T^* f_0)(x)}{\partial a} &= e^{-\sigma} \begin{bmatrix} -f_{0y_1}(x)\cos\phi + f_{0y_2}(x)\sin\phi \\ -f_{0y_1}(x)\sin\phi - f_{0y_2}(x)\cos\phi \end{bmatrix} = -\nabla_x f(x). \end{aligned} \tag{3.19}$$

$$\begin{aligned}\frac{\partial(T^* f_0(x))}{\partial\sigma} &= -e^{-\sigma}[f_{0y_1}(x)((x_1 - a_1 - s_1)\cos\phi + (x_2 - a_2 - s_2)\sin\phi) \\ &\quad + f_{0y_2}(x)(-(x_1 - a_1 - s_1)\sin\phi + (x_2 - a_2 - s_2)\cos\phi)] \\ &= -e^{-\sigma}\begin{bmatrix} f_{0y_1}(x)\cos\phi - f_{0y_2}(x)\sin\phi \\ f_{0y_1}(x)\sin\phi + f_{0y_2}(x)\cos\phi \end{bmatrix} \cdot \begin{bmatrix} x_1 - a_1 - s_1 \\ x_2 - a_2 - s_2 \end{bmatrix} \\ &= -\nabla_x f(x)^T (x - a - s). \end{aligned} \tag{3.20}$$

$$\begin{aligned}\frac{\partial(T^* f_0(x))}{\partial\phi} &= e^{-\sigma}[f_{0y_1}(x)(-(x_1 - a_1 - s_1)\sin\phi + (x_2 - a_2 - s_2)\cos\phi) \\ &\quad + f_{0y_2}(x)(-(x_1 - a_1 - s_1)\cos\phi - (x_2 - a_2 - s_2)\sin\phi)] \\ &= e^{-\sigma}\begin{bmatrix} f_{0y_1}(x)\cos\phi - f_{0y_2}(x)\sin\phi \\ f_{0y_1}(x)\sin\phi + f_{0y_2}(x)\cos\phi \end{bmatrix} \cdot \begin{bmatrix} x_2 - a_2 - s_2 \\ -(x_1 - a_1 - s_1) \end{bmatrix} \\ &= \nabla_x f(x)^T \Lambda (x - a - s), \end{aligned} \tag{3.21}$$

where $\Lambda = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ is clockwise rotation by 90 degrees.

By putting (3.18) together with (3.19), (3.20) and (3.21), we see that the partial derivatives to be used in the gradient descent step are

$$\frac{\partial E}{\partial a} = -\langle f - u, \nabla_x f\rangle, \qquad \frac{\partial E}{\partial \sigma} = -\langle f - u, \nabla_x f^T (x - a - s)\rangle,$$
$$\frac{\partial E}{\partial \phi} = \langle f - u, \nabla_x f^T \Lambda (x - a - s)\rangle.$$

However, there is a problem with using these partial derivatives since the time steps have to be chosen differently and with great care. Therefore, Overgaard et al. derived a new expression for the gradient of E in [16], which involves a scaling of the partials and makes it possible to use the same time step for all parameters. This gradient has the following components:

$$\frac{\partial E}{\partial a} = \frac{-\langle f - u, \nabla_x f\rangle}{\|\nabla_x f\|^2}, \qquad \frac{\partial E}{\partial \sigma} = \frac{-\langle f - u, \nabla_x f^T (x - a)\rangle}{\||x - a - s|\nabla_x f\|^2},$$
$$\frac{\partial E}{\partial \phi} = \frac{\langle f - u, \nabla_x f^T \Lambda (x - a)\rangle}{\||x - a - s|\nabla_x f\|^2}. \tag{3.22}$$

Now, the pose parameters can be updated according to

$$\rho^{k+1} = \rho^k - \varepsilon \nabla E. \tag{3.23}$$

The proposed method is summarised in algorithm 1. See the end of section 5 for an explanation of the number of iterations in the for-loop.

Algorithm 1 The proposed method

Set the starting values c_0^1, c_1^1, b_0^1 and b_1^1.
Let $a^1 = 0$, $\phi^1 = 0$, $\sigma^1 = 0$.
Let $\alpha = \frac{2}{\lambda+\mu}$.
Let $f^1 = f_0$.
$I_{\text{eff}}^1 = \frac{\lambda}{\lambda+\mu} I + \frac{\mu}{\lambda+\mu}(b_0^1(1-f^1) + b_1^1 f^1)$
$C_s^1 = (I_{\text{eff}}^1 - c_0^1)^2$ and $C_t^1 = (I_{\text{eff}}^1 - c_1^1)^2$
$u^1(x) = \begin{cases} 1 \text{ if } C_s^1(x) \geq C_t^1(x), \\ 0 \text{ otherwise.} \end{cases}$
$p^1(x) = 0$, $p_s^1(x) = \min\{C_s^1(x), C_t^1(x)\}$ and $p_t^1(x) = p_s^1(x)$.
Let $k = 1$.
while u not converged **do**
  $p^{k+1} = p^k$, $p_s^{k+1} = p_s^k$, $p_t^{k+1} = p_t^k$ and $u^{k+1} = u^k$.
  for $j = 1$ to 3 **do**
    $p^{k+1} = \text{proj}_\alpha \left(p^{k+1} + \tilde{\gamma} \nabla (\text{div} p^{k+1} - p_s^{k+1} + p_t^{k+1} - \frac{u^{k+1}}{\gamma}) \right)$
    $p_s^{k+1} = \min\{\frac{1}{\gamma} + \text{div} p^{k+1} + p_t^{k+1} - \frac{u^{k+1}}{\gamma}, C_s\}$
    $p_t^{k+1} = \min\{-\text{div} p^{k+1} + p_s^{k+1} + \frac{u^{k+1}}{\gamma}, C_t\}$
    $u^{k+1} = u^{k+1} - \gamma(\text{div} p^{k+1} - p_s^{k+1} + p_t^{k+1})$
  end for
  $c_0^{k+1} = \dfrac{\langle 1 - u^{k+1}, I_{\text{eff}}^k \rangle}{\langle 1 - u^{k+1}, 1 \rangle}$
  $c_1^{k+1} = \dfrac{\langle u^{k+1}, I_{\text{eff}}^k \rangle}{\langle u^{k+1}, 1 \rangle}$
  $I_{\text{model}}^{k+1} = c_0^{k+1}(1 - u^{k+1}) + c_1^{k+1} u^{k+1}$
  $b_0^{k+1} = \dfrac{\langle 1 - f^k, I_{\text{model}}^{k+1} \rangle}{\|1 - f^k\|^2}$
  $b_1^{k+1} = \dfrac{\langle f^k, I_{\text{model}}^{k+1} \rangle}{\|f^k\|^2}$
  $a^{k+1} = a^k + \varepsilon \dfrac{\langle f^k - u^{k+1}, \nabla_x f^k \rangle}{\|\nabla_x f^k\|^2}$
  $\phi^{k+1} = \phi^k - \varepsilon \dfrac{\langle f^k - u^{k+1}, \nabla_x f^{kT} \Lambda (x - a^k - s) \rangle}{\| |x - a^k - s| \nabla_x f^k \|^2}$
  $\sigma^{k+1} = \sigma^k + \varepsilon \dfrac{\langle f^k - u^{k+1}, \nabla_x f^{kT} (x - a^k - s) \rangle}{\| |x - a^k - s| \nabla_x f^k \|^2}$
  $f^{k+1} = f_0(e^{-\sigma^{k+1}} R(-\phi^{k+1})(x - a^{k+1} - s) + s)$
  $I_{\text{eff}}^{k+1} = \frac{\lambda}{\lambda+\mu} I + \frac{\mu}{\lambda+\mu}(b_0^{k+1}(1 - f^{k+1}) + b_1^{k+1} f^{k+1})$
  $C_s^{k+1} = (I_{\text{eff}}^{k+1} - c_0^{k+1})^2$ and $C_t^{k+1} = (I_{\text{eff}}^{k+1} - c_1^{k+1})^2$
  Let $k = k + 1$.
end while

Chapter 4

Implementation issues

The code for the method was written from scratch in MATLAB. We implement the method for colour images, so I is a three-dimensional array. This has some minor practical consequences; c_0, c_1, b_0 and b_1 are now vectors with three components rather than scalars and must together with u and f be of the same size as I when doing some of the calculations. Also, $C_{s(/t)}(x)=\|I_{\text{eff}}(x) - c_{0(/1)}\|_2^2$. Otherwise, everything is the same as for grey scale images.

When implementing the method in MATLAB, it has to be discretised. The digital images are already discrete, so $I(x_1, x_2)$ is $I(i, j)$ where $i = 1, \ldots, m$ and $j = 1, \ldots, n$. The number of rows in the image is m, and n is the number of columns. The discretisation techniques used to update p, p_s and p_t have benefited from the ideas described in [21]. We must also discretise the operators used in the rest of the algorithm. Addition, subtraction, multiplication and division do not constitute any problems, since these operations are done element-wise. The derivative and the integral however, have to be carefully discretised so that the discrete case will be consistent with the continuous:

4.1 Approximating the derivative

When discretising the derivative it is natural to look at the formal definition of the derivative in order to find a good approximation:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

This definition leads us to the forward-difference approximation [10]

$$f'(x_k) \approx f'_+(x_k) = \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k}.$$

Another possibility is to use negative h to get the backward-difference approximation

$$f'(x_k) \approx f'_-(x_k) = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}.$$

There are of course several other possibilities, e.g. central differences, but in the present work only these two were used. In our case, the nodes x_k are the image pixels so $x_{k+1} - x_k = 1$ for all k. Then, denoting the direction of the derivatives as upper indices x_1 and x_2, the partial derivatives will be approximated by

$$I_+^{x_1}(i,j) = I(i+1,j) - I(i,j) \quad \text{or} \quad I_-^{x_1}(i,j) = I(i,j) - I(i-1,j)$$

and

$$I_+^{x_2}(i,j) = I(i,j+1) - I(i,j) \quad \text{or} \quad I_-^{x_2}(i,j) = I(i,j) - I(i,j-1).$$

Observe that, when calculating the derivative for all image pixels, the definition above uses pixels outside the image domain for the derivative on the first or last row and column depending on which approximation is used. For forward differences some of the choices for defining those pixels are $I(m+1,j) = I(m,j)$, $I(m+1,j) = 0$ or $I(m+1,j) = I(1,j)$ and similar for $I(i,n+1)$. For backward differences we can define $I(0,j) = I(1,j)$, 0 or $I(m,j)$ and similar for $I(i,0)$.

4.2 Approximating the integral

Our approximation of the integral is based on the trapezoidal rule [10] where the interval $[a,b]$ is divided into n subintervals of equal length h:

$$\int_a^b f(x)\,dx \approx \sum_{k=0}^{n-1} \frac{h}{2}(f_k + f_{k+1}) = h(f_0/2 + f_1 + f_2 + \cdots + f_{n-1} + f_n/2),$$

where f_k is short for $f(x_k)$. As above, since the nodes x_k are the image pixels, $h = 1$. Also, seeing that images often are quite large matrices with a large number of pixels, the approximation of the integral will not be much more inaccurate if f_0 and f_n are not divided by 2. This simplifies matters, and the two-dimensional integral will hence be approximated as

$$\int_\Omega I(x_1,x_2)\,dx \approx \sum_{i=1}^{m}\sum_{j=1}^{n} I(i,j).$$

4.3 Smoothing prior to differentiation

When implementing the formulas for updating the pose parameters for f, we had some trouble making the discrete case consistent with the continuous. This trouble arose from the approximation of the derivative. When calculating the gradient descent direction for the updating of a, $\langle f - u, \nabla_x f \rangle$ must be calculated. When the supports of u and f do not overlap, the result is the same as if $u = 0$ because $\nabla_x f = 0$ in all points where $u \neq 0$. Consequently, $\langle f - u, \nabla_x f \rangle = \int_\Omega f \nabla_x f \, dx$. Looking at this in one dimension, we know that $(\frac{1}{2} f^2)' = f f'$ so $\int_a^b f f' \, dx = [\frac{1}{2} f^2]_a^b = 0$ as f has compact support. That is, $\langle f, f' \rangle = 0$. The same should hold in the discrete case, but as this one-dimensional example shows, it is not straightforward to get this:

$$\begin{array}{rccccccc} f: & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ f'_+: & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ f f'_+: & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ f'_-: & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ f f'_-: & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}$$

It appears that when using forward differences $\langle f, f' \rangle \approx \sum f f'_+ < 0$ and when using backward differences $\langle f, f' \rangle \approx \sum f f'_- > 0$, so this is not consistent with the continuous case. Furthermore, if the support of u overlaps on the right side of the support of f, then $\sum (f - u) f'_+ = 0$ and $\sum (f - u) f'_- > 0$. If it covers the left side $\sum (f - u) f'_+ < 0$ and $\sum (f - u) f'_- = 0$. Hence, both the forward and the backward difference approximation "favour" one side, which is an unwanted behaviour.

The solution to this problem is to smooth the function prior to the difference approximation. Observe that, for example looking at backward differences, the problem in the example above is the -1 to the right of the support of f; the backward difference approximation "stretches" the support to the right. Therefore we want to smooth the function in such a way that the support is "stretched" to the left in order to counteract this behaviour of the difference approximation. If we let $\tilde{f}(i) = \frac{1}{2}(f(i+1) + f(i))$ where $\tilde{f}$ denotes the smoothed f, we have

$$\begin{array}{rccccccc} f: & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ \tilde{f}: & 0 & 0.5 & 1 & 1 & 0.5 & 0 & 0 \\ \tilde{f}'_-: & 0 & 0.5 & 0.5 & 0 & -0.5 & -0.5 & 0 \end{array}$$

Notice that the support of the derivative $\tilde{f}'_-$ is symmetric around the support of f and $\sum f \tilde{f}'_- = 0$ as wanted. Moreover, if the support of u overlaps the support of f on the right side $\sum (f - u) \tilde{f}'_- > 0$ which is what we want since

f should approach u which means that a must increase. The opposite is true if the support of u overlaps the support of f on the left side.

Naturally, the same symmetric derivative can be obtained if we instead smooth f in such a way that the support is stretched to the right and use forward differences. However, using an averaging filter of even size $(2d \times 2d)$ and convolving f with this filter in MATLAB yields an $\tilde{f}$ with support stretched to the left just as described above. Therefore, backward differences is used when calculating the gradient of f.

Chapter 5

Experimental results

We have tested our method on several natural colour images and will show some of the results in this section. The shape priors were obtained by painting with white over the desired shape and black over the background, or in some cases downloaded from [14]. The same shape prior was often used for several different images of similar objects. The parameters used in all experiments are $\gamma = 0.3$, $\tilde{\gamma} = 0.16$ and $\varepsilon = 1$, while λ and μ must be chosen differently for each image. As can be seen from (3.3) and (3.5), the value of λ controls the influence of the intensity values in the segmentation process, whereas the value of μ controls the influence of the shape prior. Moreover, the sum of λ and μ regulates the importance of the length of the boundary of Σ; if $\lambda + \mu$ is large, $J(u)$ will not contribute much to the energy and can thus be allowed to be relatively large.

This is illustrated in figures 5.1 and 5.2. The shape prior used in figure 5.1 is obtained from an image of a different butterfly. It does not have the exact same shape as the butterfly which is to be segmented from this image, and so the segmentation result will not be very good when μ is large relative to λ as in the second row, where the result is too similar to the shape prior. As λ is increased and μ decreased the result improves; observe that in the third row the result differs from the prior in several places, and is closer to the actual outline of the butterfly. In the fourth row the segmentation result is quite close to the butterfly; notice in particular that the whole head of the butterfly is contained in the output, in contrast to in the two previous results. However, since λ is so large relative to μ, two unwanted dark spots with a colour more similar to the colours of the butterfly than to the colours of the background are also included.

Figure 5.2 illustrates how the allowed length of the boundary can affect the segmentation result. Notice in the second row that the contour is allowed to be too long, resulting in some small white flowers also being enclosed by

u. In the third row λ and μ are decreased, and the result is consequently almost perfect, whereas the contour in the fourth row is a bit too short and so some of the background between the petals is included in the result.

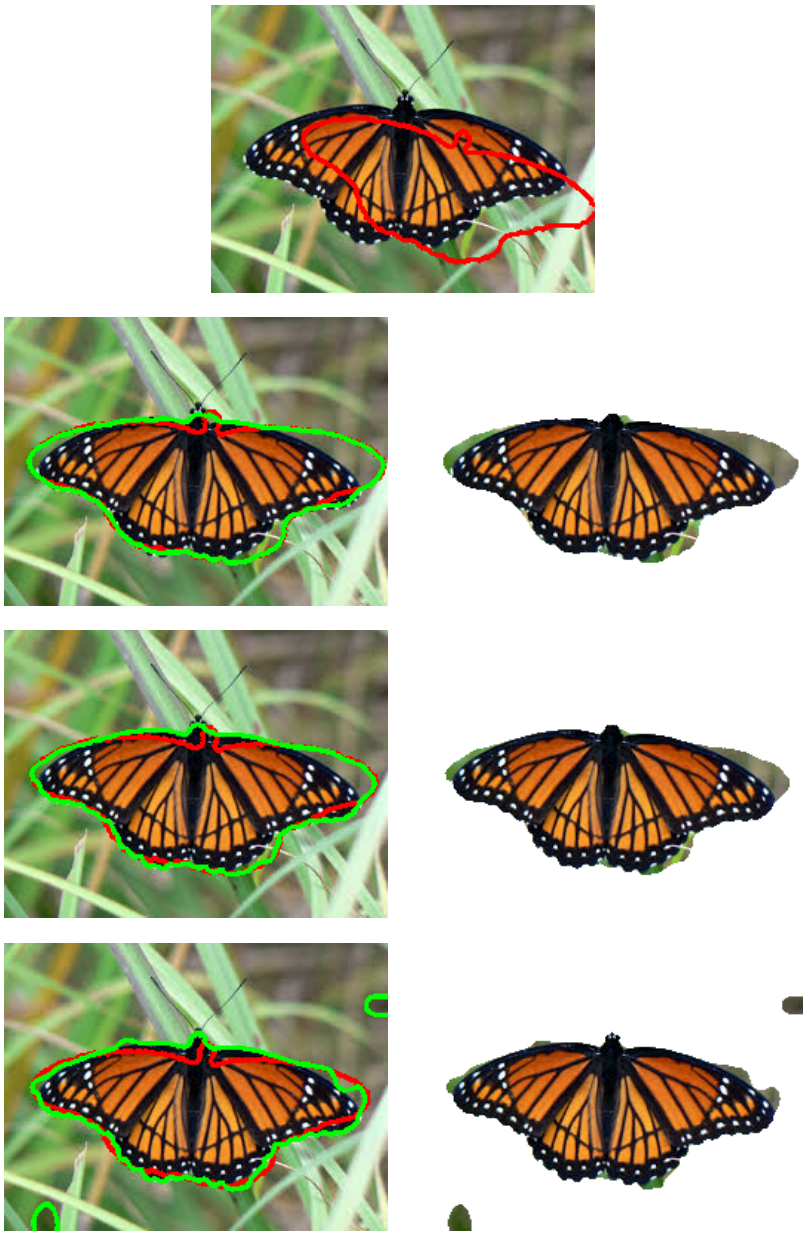


Figure 5.1: First row: Initial image with the initial position of the shape prior shown as a red contour. Second row: Segmentation result with $\lambda = 1.5$, $\mu = 3.5$. Third row: Segmentation result with $\lambda = 2$, $\mu = 2$. Fourth row: Segmentation result with $\lambda = 3$, $\mu = 1.5$.

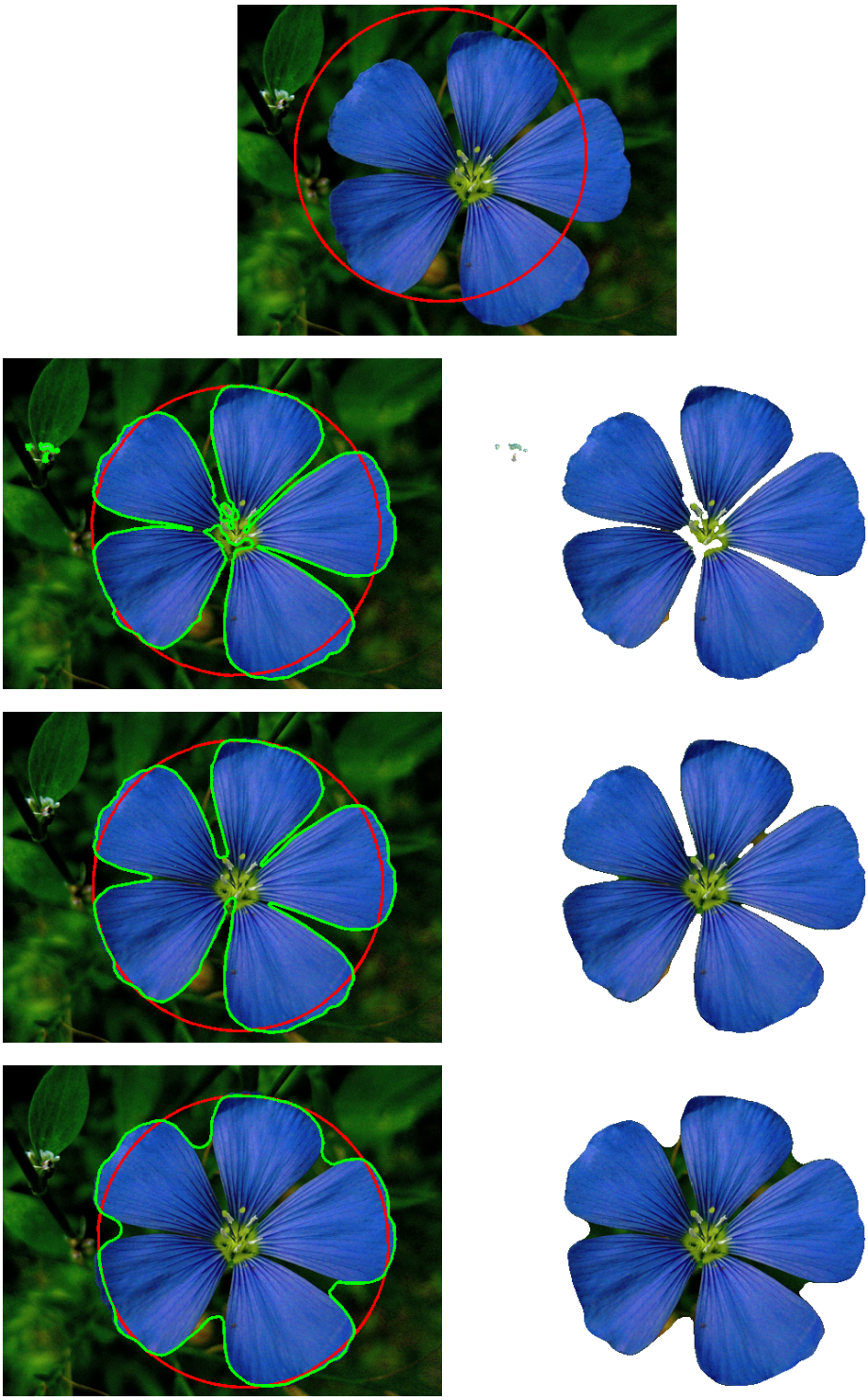


Figure 5.2: First row: Initial image with the initial position of the shape prior shown as a red contour. Row 2-4: Segmentation result with $\lambda = 10$ and $\mu = 7.5$, $\lambda = 3$ and $\mu = 2.25$, and $\lambda = 0.5$ and $\mu = 0.375$.

Figure 5.3 shows an image of two apples and one banana shaped as a face lying in the grass, and the segmentation result when using the method with $\lambda = 4$ and $\mu = 0$, i.e. with no shape prior information. We want to segment out the banana, and observe that since the apples and the banana have approximately the same colour, this is impossible to do without a shape prior.

Figure 5.3: An image and the segmentation result without shape prior information

The shape prior used in this experiment is a black-and-white image of a banana which is not exactly of the same shape as the banana in the image, yet very close. Figure 5.4 shows the segmentation results for different initial poses of the prior. In these experiments $\lambda = 1$ and $\mu = 3$ are used. The reason for this choice is that if λ is increased or μ decreased, the segmentation result will in addition to the banana include the unwanted apples. Observe that the method works very well for several different initial poses, but if the prior covers more of the apples than of the banana, the pose updating will obviously lead the prior towards the apples, as we see in the fourth row. Also, if the prior neither covers the apples nor the banana as in the fifth row, the shape prior is already at a local minimum. The reason for this is that the initial Σ with the characteristic function u will consist of the apples and the banana, while Σ' with the characteristic function f is the banana prior. If we make a small change in the pose parameters, Σ and Σ' will still not overlap and $\int_\Omega (u(x) - f(x))^2 \, dx$ will accordingly not change. Thus, the segmentation result will be the same as the result without a shape prior.

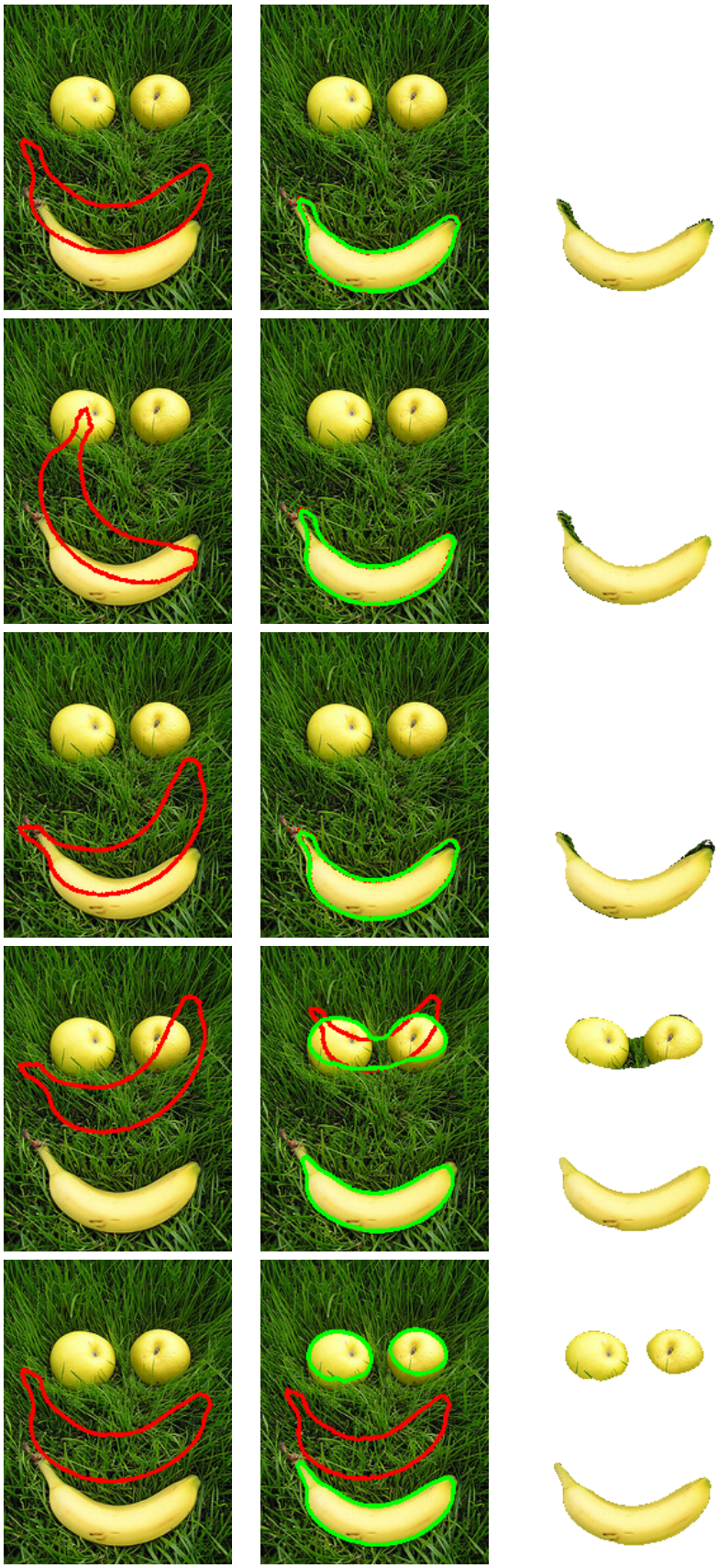


Figure 5.4: First column: Initial image with the initial position of the shape prior shown as a red contour. Second column: Segmentation result shown as a green contour, with the final position of the shape prior. Third column: Segmentation result.

In our next experiment, the object of interest is a paracanthurus hepatus, also called blue surgeonfish or blue tang, shown in figure 5.5a. These fish are difficult to segment, since they both consist of light colours like the blue body and the yellow tail, and dark colours like the stripe along the body. Segmentation without shape prior information often yields an unsatisfactory result without the stripe, as shown in figure 5.5b.

(a) Paracanthurus hepatus

(b) Segmentation result without shape prior information

Figure 5.5

In figure 5.6, the segmentation results using a shape prior downloaded from [14] are presented. Notice that the tail of the prior is shorter than the tail of this fish, hence it is not segmented perfectly. The parameters for this experiment were $\lambda = 1.5$ and $\mu = 2.5$. λ is larger and μ smaller than in the previous experiment, as there are not so many background objects that could easily be labelled as foreground if λ was chosen too large.

The first and second row show successful segmentation results, while the third and fourth row show incidents where the prior gets stuck in a local minimum. The local minimum in the fourth row is due to the fact that when the initial pose of the prior only covers the blue part of the fish, the initial segmentation result u also only contains the blue parts (as in figure 5.5b without the tail), and then the final prior position shown is clearly a minimum for $\int_\Omega (u - f)^2 \, dx$. The local minimum in the third row is not as easy to understand because the initial prior position is not far from the initial prior position in the first row, which yielded a successful result. What happens here is that since the initial prior position does not cover the yellow tail, initial u consists of the blue parts and some of the dark parts of the fish, and this is what the prior tries to cover. However, in the first experiment, the initial prior position is so close to the tail that during the process of enlarging the prior, the prior encloses a part of the tail resulting in u including some

of it and the prior changing its course. This does not happen in the third experiment. Still, when μ is increased to 3 (not shown), the same happens here resulting in an improved segmentation result, although the tail becomes a bit broader than in row one and two in an effort to resemble the prior shape more closely.

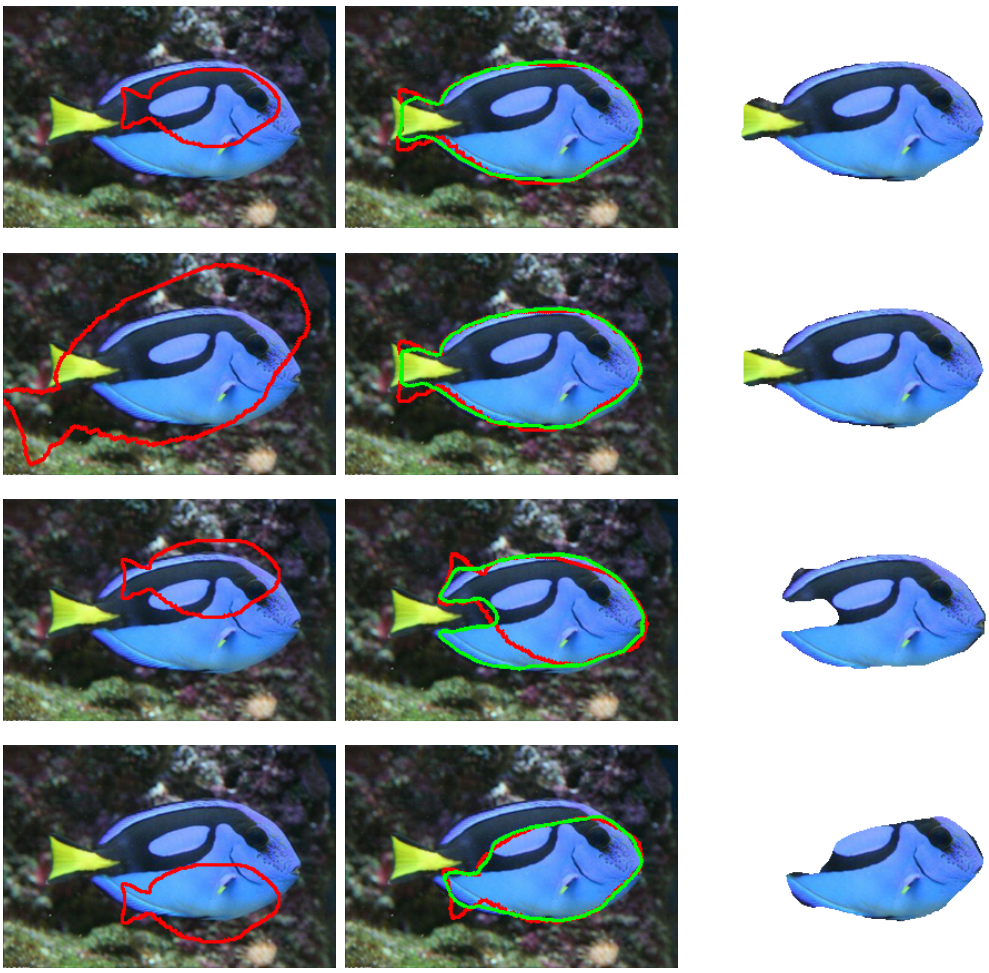


Figure 5.6: First column: Initial image with the initial position of the shape prior shown as a red contour. Second column: Segmentation result shown as a green contour, with the final position of the shape prior. Third column: Segmentation result.

Figure 5.7 displays segmentation results for the image of a blue flower shown in figure 5.2, with a "perfect" shape prior. The results for two different initial poses of the shape prior are shown, in order to illustrate the fact that it sometimes can be a problem if the prior is rotated too much relative to the

object we want to segment. In the first column, the result becomes perfect even though the prior is both scaled down, translated and rotated $\frac{\pi}{8}$ radians counterclockwise. In the second column, the prior is neither scaled down nor translated, only rotated by $\frac{\pi}{4}$. Apparently, this was too much; the prior continues to move counterclockwise and gets stuck in a local minimum. In the first column $\lambda = \mu = 2$ is used, and in the second column $\lambda = 1.5$ and $\mu = 2.5$.

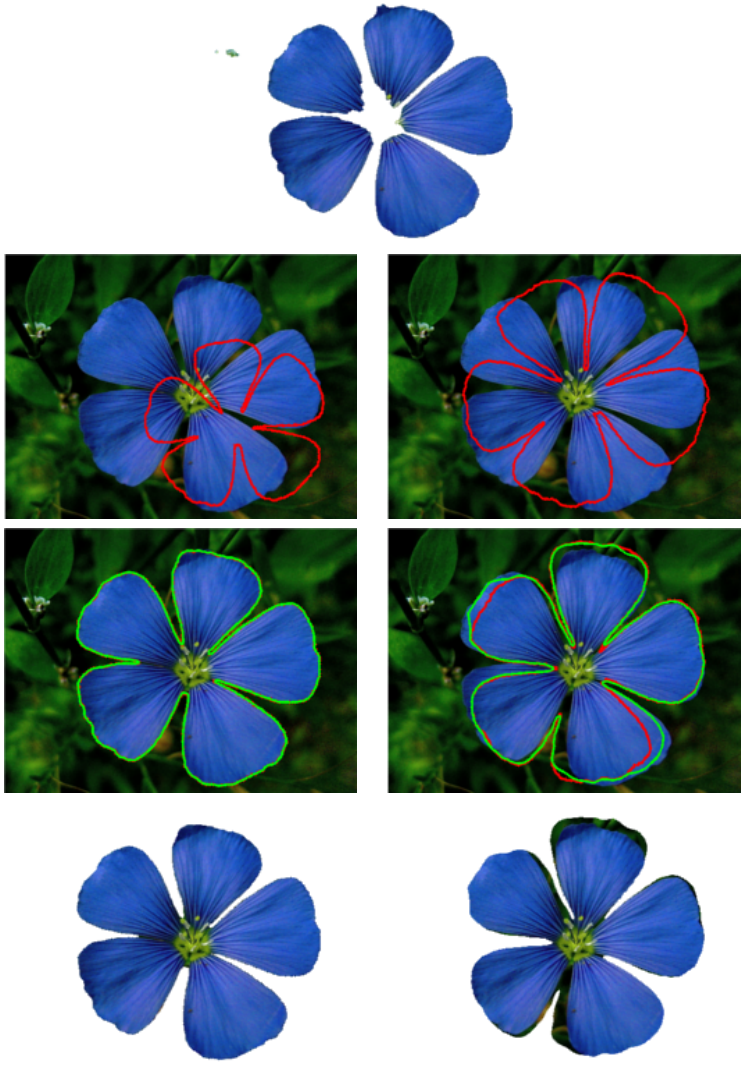


Figure 5.7: First row: Segmentation result with no shape prior information. Second row: Initial image with the initial position of the shape prior shown as a red contour. Third row: Segmentation result shown as a green contour, with the final position of the shape prior. Fourth row: Segmentation result.

Figure 5.8: An image and the segmentation result without shape prior information.

Figure 5.8 shows a leaf lying on a messy table, and the just as messy segmentation result obtained with no prior shape information. In figure 5.9 the much better results that are obtained when including the shape prior which was used in [16] are presented. Two initial positions that lead to successful segmentation results are shown, but in the third row we have the same problem as in figure 5.7: The prior gets stuck in a local minimum because it is too much rotated in relation to the leaf in the image.

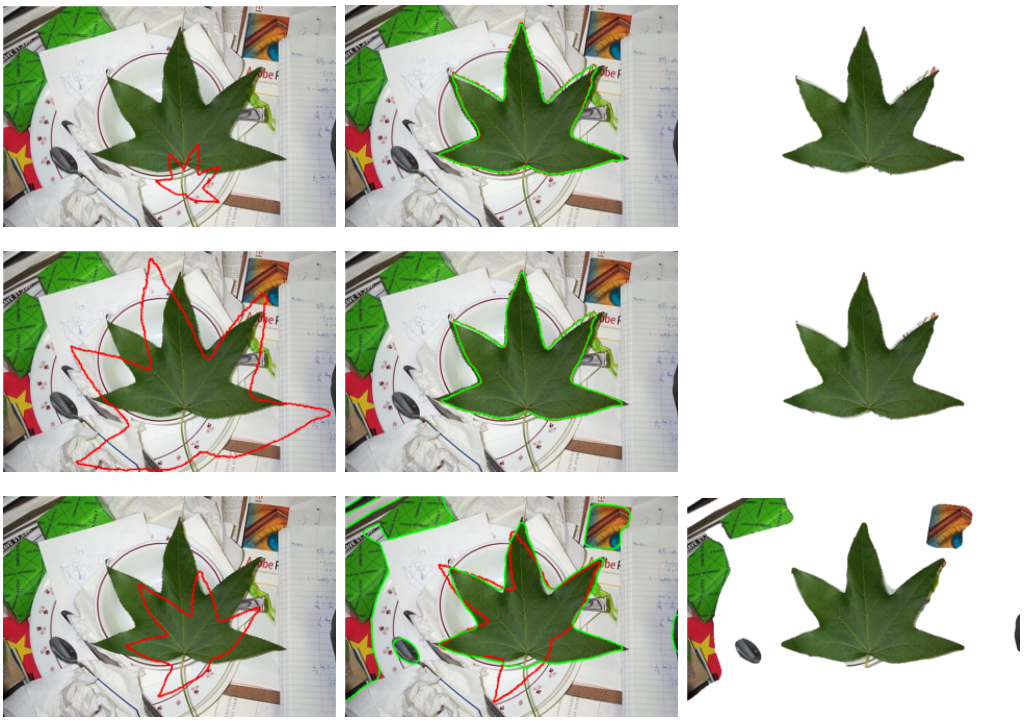


Figure 5.9: First column: Initial image with the initial position of the shape prior shown as a red contour. Second column: Segmentation result shown as a green contour, with the final position of the shape prior. Third column: Segmentation result. Parameters: $\lambda = 0.5$, $\mu = 2$.

If the prior is very similar to the object of interest, it is often easy to choose the right parameters and get a perfect segmentation result. On the other hand, if the prior is rather different, it can be difficult to choose the right parameters. As we saw in figure 5.1, if μ is too large relative to λ, the result will be too close to the prior. If μ is too small however, the result may besides the object of interest contain areas from the background with a similar colour. We want to improve the result in the last row of figure 5.1 by trying to remove the two spots. We decrease λ and μ while keeping the same ratio (i.e. decrease the allowed length of the boundary), and get the perfect result in figure 5.10 with $\lambda = 1$ and $\mu = 0.5$. Here, another initial pose than in figure 5.1 is shown, but the result is the same with both initial poses.

Figures 5.11 and 5.12 show segmentation results both with priors from other images of a similar object and with priors obtained from the image, and illustrate the obvious fact that a better prior gives a better segmentation result.

(a) Segmentation result with no shape prior information.

(b) Initial image with the initial position of the shape prior shown as a red contour.

(c) Segmentation result shown as a green contour, with the final position of the shape prior.

(d) Segmentation result.

Figure 5.10

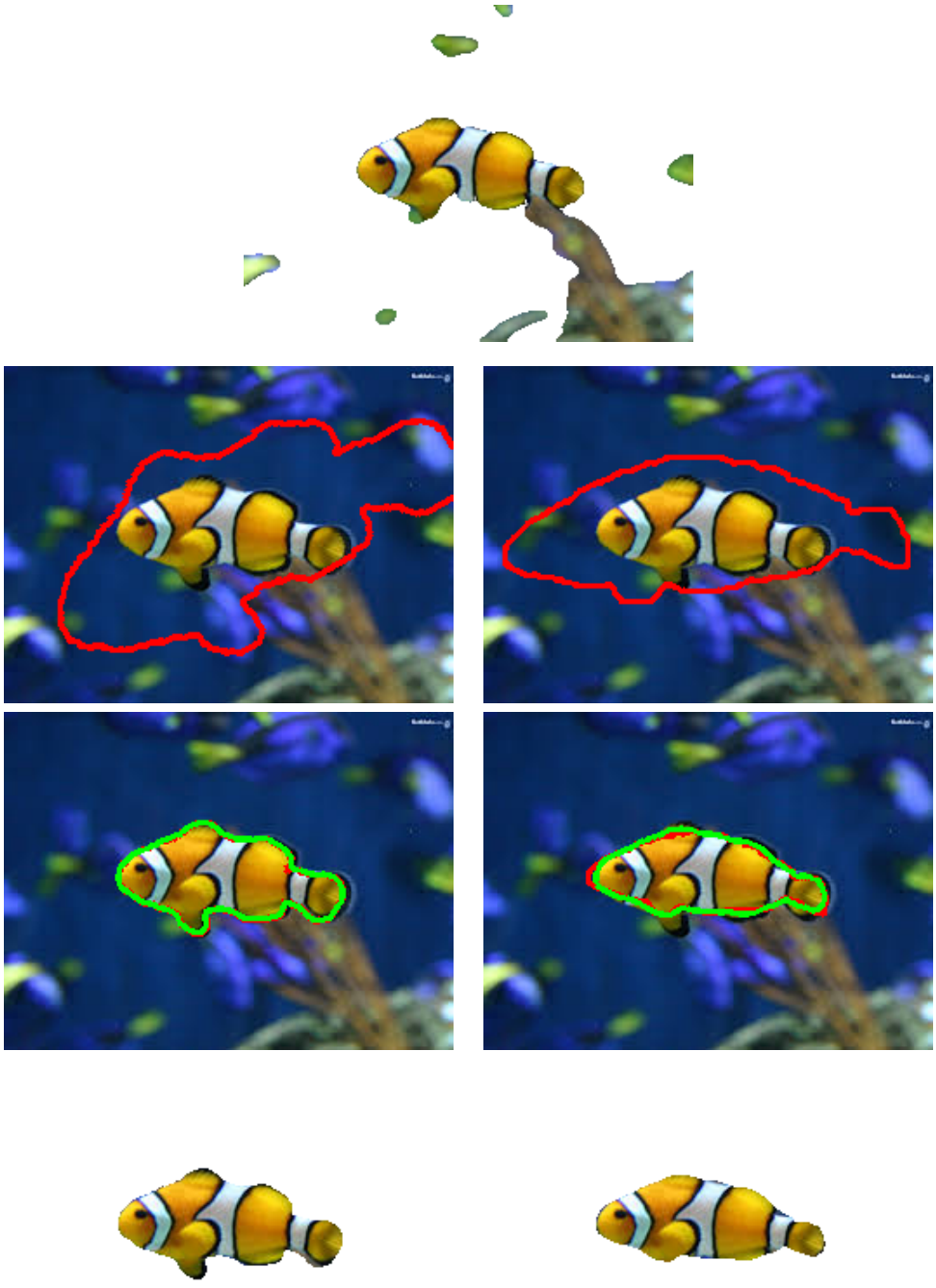


Figure 5.11: First row: Segmentation result with no shape prior information. Second row: Initial image with the initial position of the shape prior shown as a red contour. Third row: Segmentation result shown as a green contour, with the final position of the shape prior. Fourth row: Segmentation result. Left column: Prior obtained from the image, $\lambda = 1$, $\mu = 2.5$. Right column: Prior from [14], $\lambda = 1.2$, $\mu = 1.5$.

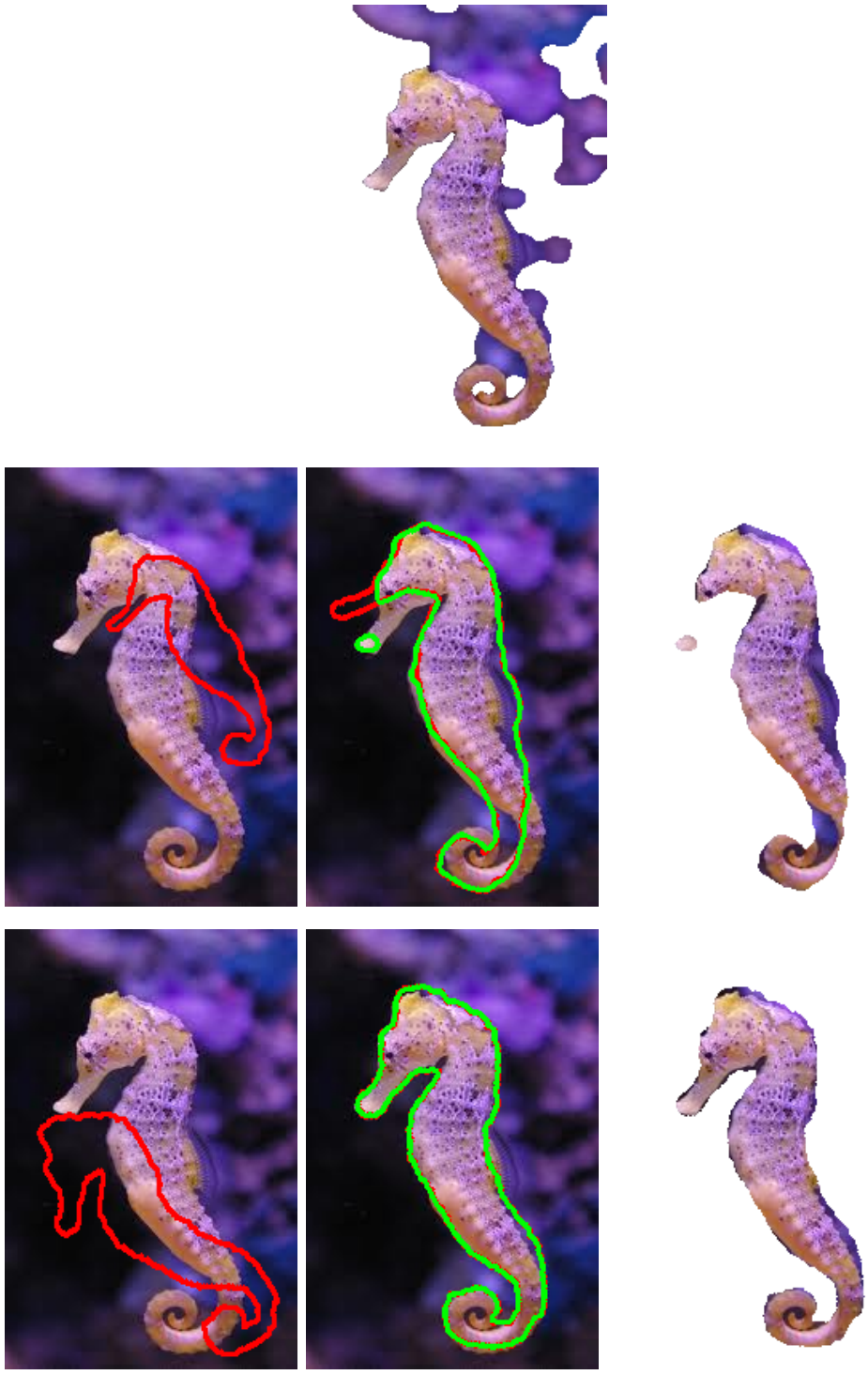


Figure 5.12: Top: Segmentation result with no shape prior information. First column: Initial image with the initial position of the shape prior shown as a red contour. Second column: Segmentation result shown as a green contour, with the final position of the shape prior. Third column: Segmentation result. Second row: Prior obtained from another image of a seahorse, $\lambda = 1.9$, $\mu = 3$. Third row: Prior from this image, $\lambda = 1$, $\mu = 3$.

Figures 5.13 and 5.14 show six more successful experiments. Only one initial pose is shown for each experiment, although several other initial poses that produced satisfying results have been tried as well.

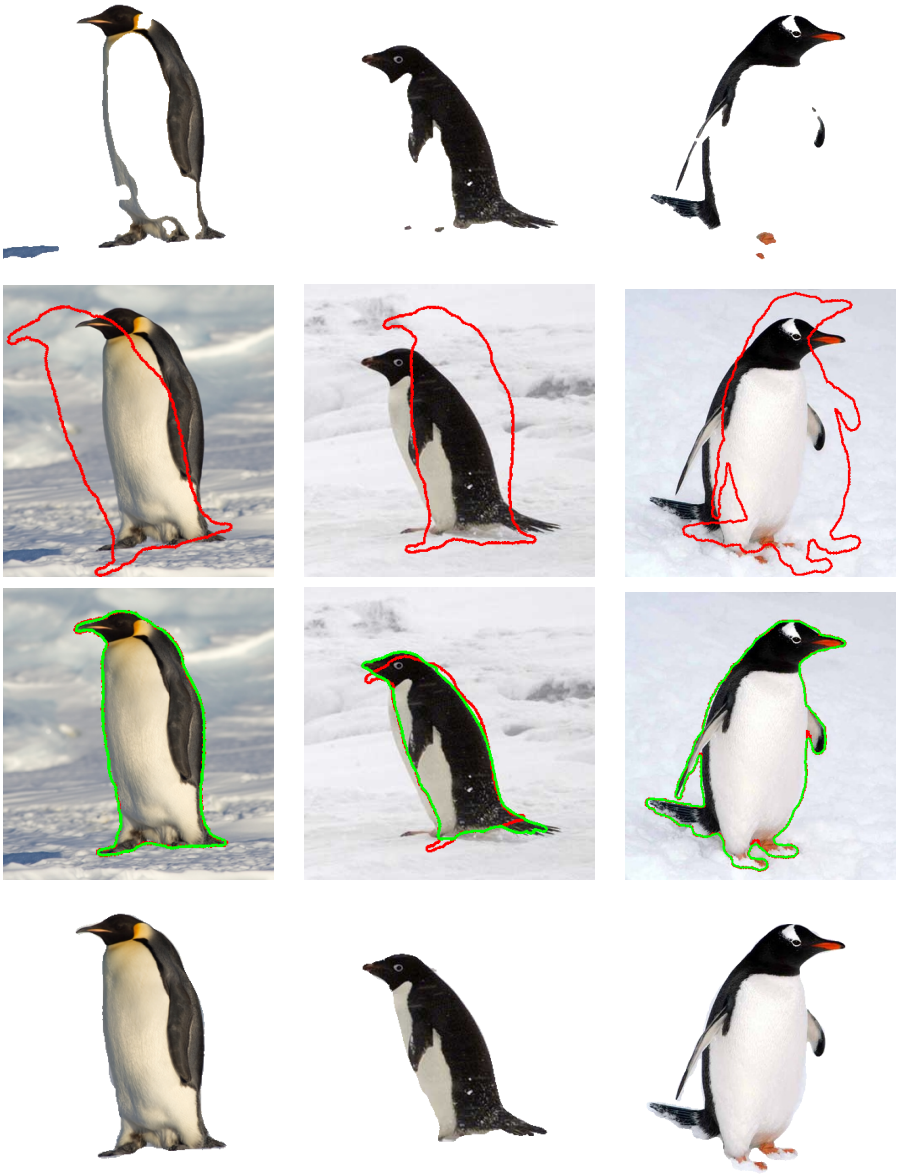


Figure 5.13: First row: Segmentation result with no shape prior information. Second row: Initial image with the initial position of the shape prior shown as a red contour. Third row: Segmentation result shown as a green contour, with the final position of the shape prior. Fourth row: Segmentation result. Parameters left to right column: $\lambda=1$ and $\mu=5$, $\lambda=\mu=2$, $\lambda=1$ and $\mu=3$.

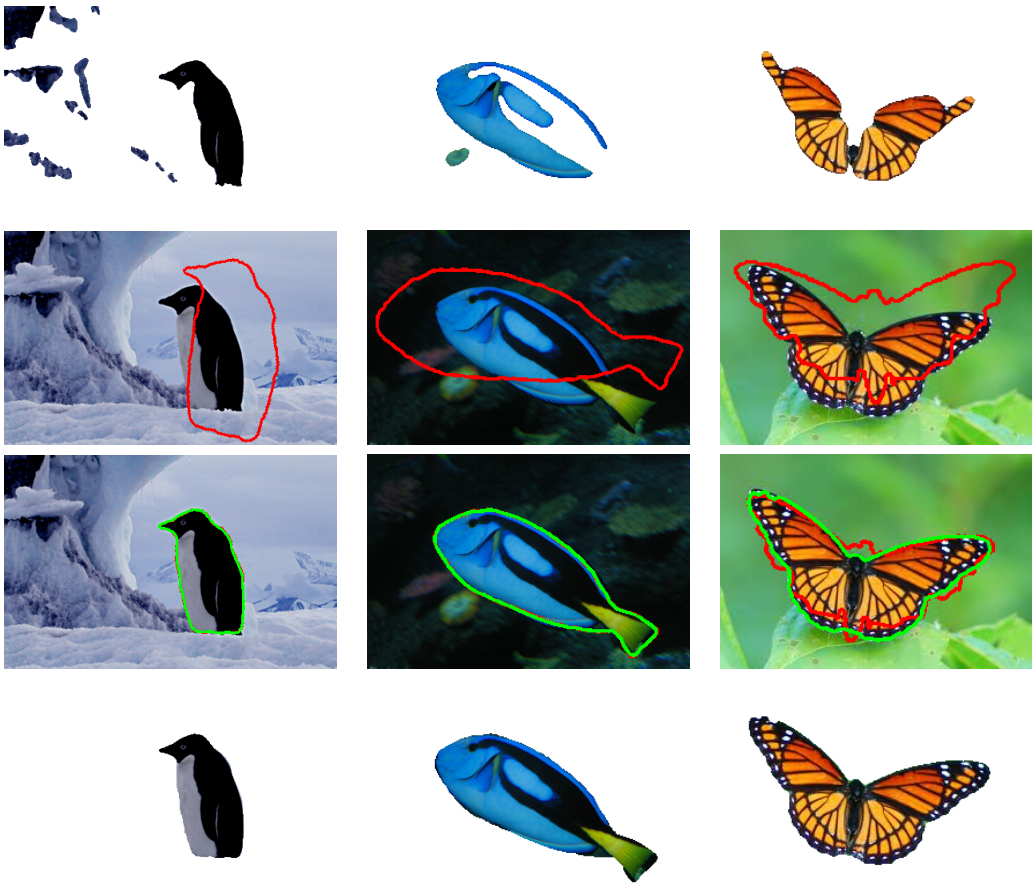


Figure 5.14: First row: Segmentation result with no shape prior information. Second row: Initial image with the initial position of the shape prior shown as a red contour. Third row: Segmentation result shown as a green contour, with the final position of the shape prior. Fourth row: Segmentation result. Parameters left and middle column: $\lambda = 1$ and $\mu = 3$. Right column: $\lambda = 3$ and $\mu = 1$.

Images of penguins on snow are generally very difficult to segment, because a large part of their body has the same colour as the snow. Therefore, the priors have to be quite perfect and μ large relative to λ in order to get a good result in these experiments. A shape prior obtained from the image is used in three of the penguin images; only in the second column of figure 5.13 is another prior used—the same prior as in the first column—and the result is still reasonably good. This is possible because in this image even the white parts of the penguin are a little bit darker than the snow, and so μ does not have to be that large in order to cause the output to include the white parts of the body. In the other images

however, the background has an intensity value in between the intensity of the dark and the light parts of the penguin, which complicates matters.

For the image of the blue surgeonfish in the middle column of figure 5.14, a prior obtained from another image of the same type of fish is used. A butterfly prior from [14] is used in the experiment in the right column, and even though this shape prior is quite different from the butterfly in the image, this gives really good results.

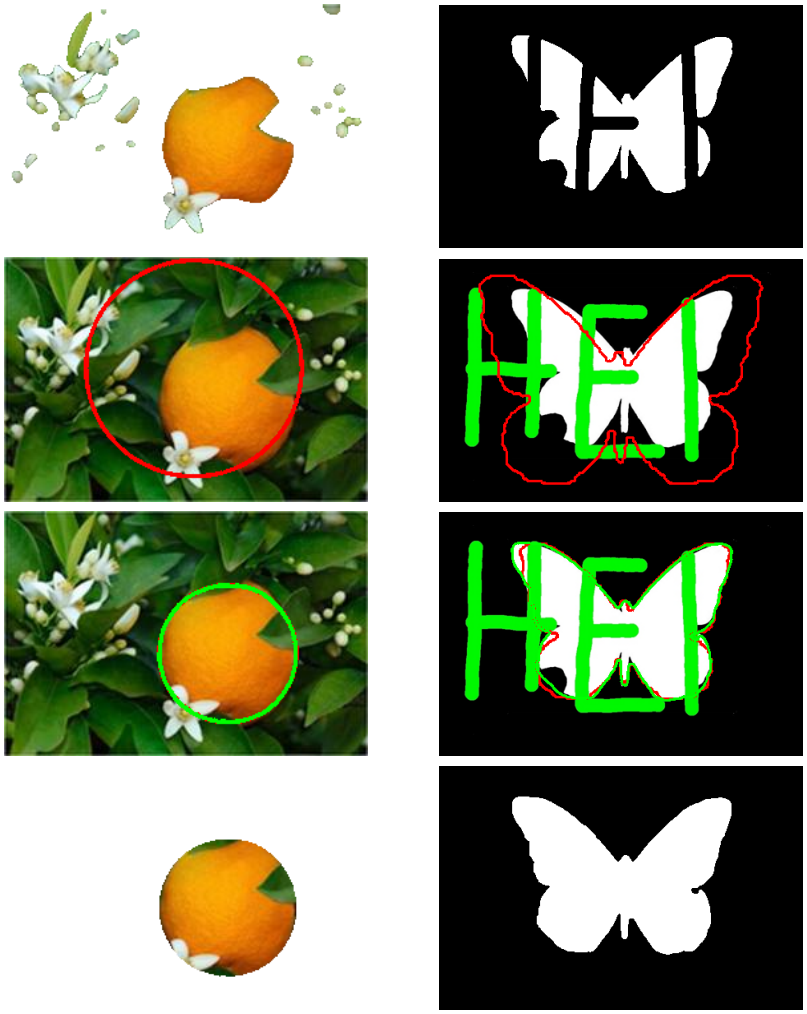


Figure 5.15: First row: Segmentation result (first column) or u (second column) with no shape prior information. Second row: Initial image with the initial position of the shape prior shown as a red contour. Third row: Segmentation result shown as a green contour, with the final position of the shape prior. Fourth row: Segmentation result / u. Parameters left column: $\lambda = 1$ and $\mu = 3$, right column: $\lambda = 2$ and $\mu = 2.5$.

Figure 5.15 illustrates an application of shape prior image segmentation where the aim is to recover objects that are occluded by other objects or have missing parts. In the left column the segmentation of an image of an orange on its bush is shown. The orange is occluded by leaves and a flower, and the segmentation with a circle as shape prior recovers the shape of the orange. The right column shows the segmentation of an image constructed especially for this example; a piece of the butterfly is erased and a word is scribbled across it. Segmentation with a shape prior downloaded from [14] gives a fairly good representation of the original butterfly.

The next example is provided to show that the proposed method is stable with respect to initialisations of c_0, c_1, b_0 and b_1. In all experiments the initialisations $b_0 = c_0$ and $b_1 = c_1$ are used. We performed segmentation on an image of a seahorse with the shape prior showed in the top row of figure 5.16, which is the same as in the first row of figure 5.12. Initial c_0 and c_1 were varied, and we observed that for most initialisations the result was as in the second row of the figure. This is not a perfect result; because of the seaweed in the background μ must be relatively large, and so the segmentation result is too close to the shape prior, which is quite different from the seahorse in the image. However, the result is much better than without a shape prior, and we get this result with a lot of different initialisations—12 are shown in the figure, and many more were tried. Naturally, the initialisation where c_0 is approximately the mean colour of the background and c_1 the mean colour of the seahorse (shown as the upper left image) works fine. The second initial colours image (upper right) also shows a natural initialisation, but surprisingly, the third one also functions. Furthermore, c_0 as black and c_1 as white gives a good result, since the foreground is light and the background is dark. Even though the opposite, i.e. c_0 as white and c_1 as black, does not work, the very similar initialisation with c_0 as light yellow and c_1 as dark blue does! Also, c_0 as green and c_1 as black worked, and c_0 as white and c_1 as any primary or secondary colour. In addition, we found that all combinations of primary and secondary colours also give the same result.

Even though c_1 as black performs well with c_0 as green, the result with any other primary or secondary colour as c_0 is as shown in the third row, where the prior has grown too much. If we use an initialisation with c_0 as the approximate mean colour of the foreground and c_1 as the approximate mean colour of the background, we get a result opposite of what we want, that is, the seahorse is labelled as background (shown in the fourth row). This also happens with c_0 as white and c_1 as black and for the initial colours shown in the bottom initial colours image.

It is reasonable to conclude that any sensible initialisation will give a

good result (if the initial position of the shape prior and the parameters are good); the result does not depend to a large extent on the initialisations of c_0, c_1, b_0 and b_1.

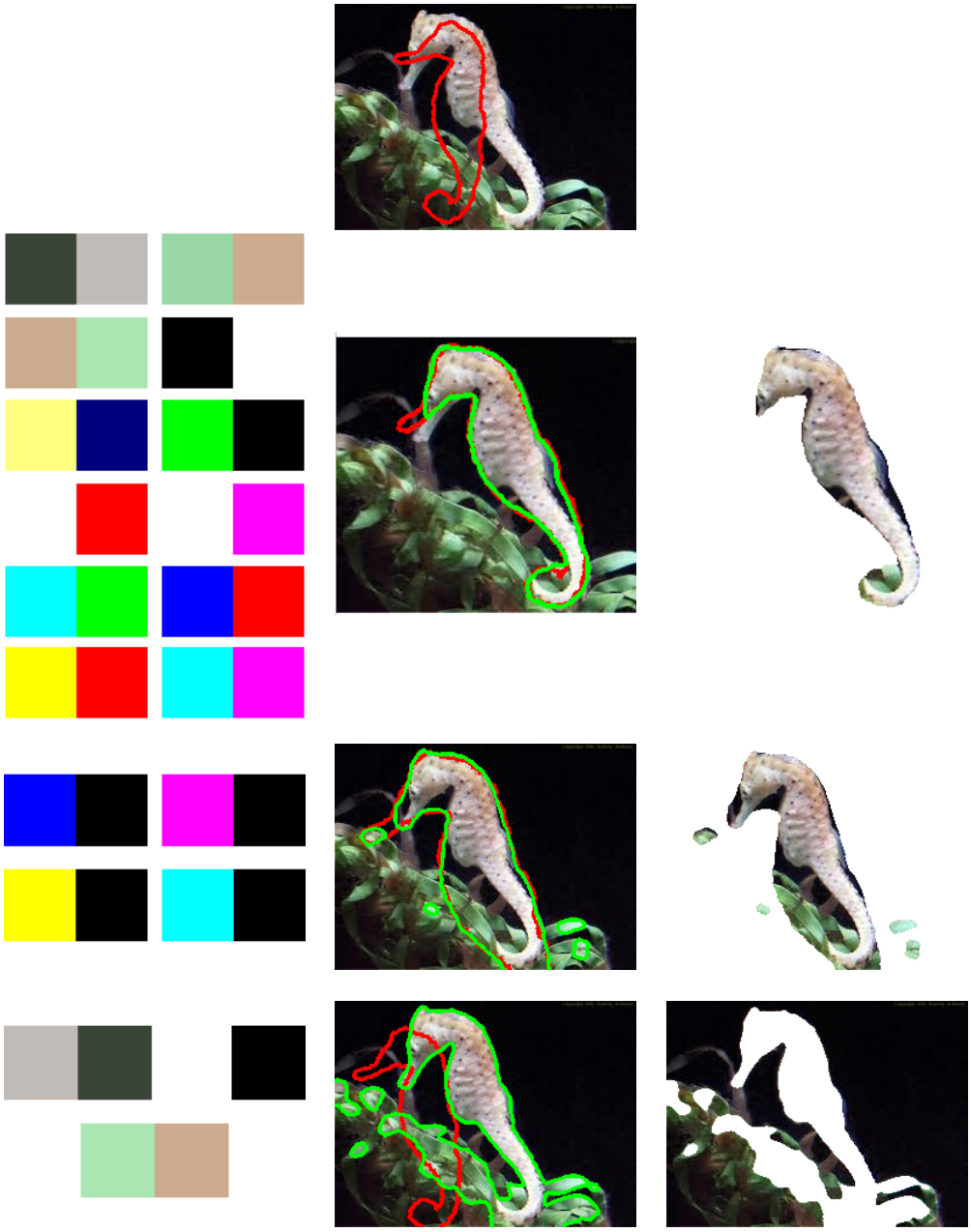


Figure 5.16: First column: Some initial colours shown as images with c_0 to the left of the image and c_1 to the right. Second and third column: Corresponding segmentation result. Parameters: $\lambda = 1.5$ and $\mu = 2.5$.

Colour models

In all the examples above, we have used the RGB colour model, because most images are stored in this format. However, in some cases, the HSI colour model can give an advantage over the RGB model. As an example, we look at the image shown in RGB and HSI space in figures 2.3 and 2.5, respectively. The segmentation result obtained with no shape prior information is shown in figure 5.17, and we can see that the segmentation in HSI space is better than in RGB space, even without a shape prior.

(a) RGB space (b) HSI space

Figure 5.17: Segmentation result without shape prior.

Figure 5.18 shows the segmentation results with the same shape prior as in the middle column of figure 5.14. In RGB space, μ has to be large relative to λ in order to force the result to include the tail and the dark stripe along the body. Then the segmentation result is too similar to the prior shape, but if λ is increased in order to make the contour adapt more to the object, the result will be very similar to the result without a shape prior, as we see in the second row of the figure. In HSI space (row 3), however, we get a satisfactory result even with an initial pose of the prior which is further away from the object. Looking at the decompositions in figures 2.3 and 2.5, this is not surprising. Observe that the dark stripe along the body has a very different value from the rest of the body in both the green and the blue channel, and the tail is different in both the red and the blue channel. On the other hand, in HSI space, the stripe is only significantly different in intensity and the tail only in hue. Therefore, the segmentation result can, when using HSI, be allowed to depend more on the values of the image and less on the shape prior, and the final contour can thus adapt more to the object.

Here, we use both hue, saturation and intensity information in the segmentation process. However, it is possible to utilise the advantages of HSI space even more. Depending on the nature of the image, using only one or

two of the channels can sometimes be a good idea. For colour based segmentation, the hue and saturation images are most frequently used [12]. Another idea is to extract the most relevant information from each channel, for example representing a colour by its hue value if the saturation is so high that this is meaningful, and else representing it by its intensity value, as done in [18]. We have not experimented with these possibilities in this work, we merely note that it may further improve the results.

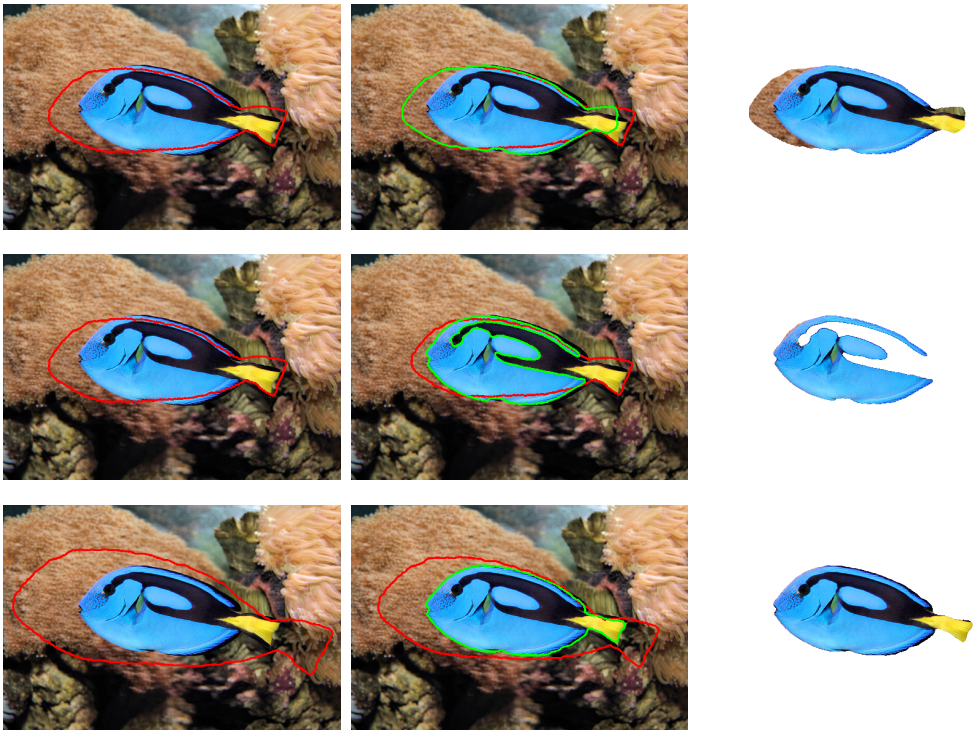


Figure 5.18: Segmentation with RGB colour model in the first and second row and HSI colour model in the third. First column: Initial position of the shape prior. Second column: Segmentation result shown as a green contour, with the final position of the shape prior in red. Third column: Segmentation result. Parameters rows 1-3: $\lambda = 1.7$ and $\mu = 3$, $\lambda = 2$ and $\mu = 3$, and $\lambda = \mu = 2$.

Number of max-flow iterations

Experiments on several different images showed that there is no point in actually minimising w.r.t. u, as [16] suggests. In fact, just a few iterations of the steps for minimising w.r.t. u is enough—the result will in most cases be almost the same regardless of the maximum number of iterations used to update u, and in some cases the result will be worse for a larger number of iterations. We found through experiments that between 2 and 4 iterations yield the best results and shortest computation time, and in a majority of the experiments 3 iterations gave the shortest computation time. Therefore, a maximum number of 3 iterations is used in the algorithm.

One of the experiments was done on the butterfly in figure 5.10. The parameters were as in the experiment discussed in relation to this figure, but the maximum number of iterations in the for-loop on j in algorithm 1 was changed and a stopping criteria in the case of convergence of u was added. The results are shown in table 5.1. We see that a maximum number of 3 iterations gave the shortest computation time, and the segmentation result is actually a little bit better too. This can be seen in figure 5.19. It is not a great difference, but we see that with 100 iterations there are more parts outside the butterfly in the final result than with 3 iterations.

Max it on u	Time	Outer it
100	8.20	41
50	6.18	41
25	4.64	36
10	3.91	37
5	3.09	40
4	2.80	40
3	2.61	43
2	2.98	60
1	4.59	120

Table 5.1: Computation time in seconds and number of iterations on k needed for different number of maximum iterations on continuous max-flow.

The actual number of iterations needed to minimise w.r.t. u can also show us why choosing 3 as maximum number of iterations is a good choice. The bar chart in figure 5.20 shows the number of iterations in the for-loop until convergence of u in every step when we used 100 as the maximum number of iterations. We see that it is only in the beginning that a large number of iterations are needed to minimise w.r.t. u, but at this point it is not possible

Figure 5.19: The difference in segmentation result with 100 and 3 as maximum number of iterations. The red dots show parts that are included in the result with 100 but not with 3, and the green dots show parts that are included in the result with 3 but not with 100.

to get a good segmentation anyhow, since the position of the shape prior in most cases is entirely wrong. Therefore, using a lot of computation time to make u converge is pointless in the beginning. Further on, only a few iterations are needed to minimise w.r.t. u, and so we do not loose accuracy by choosing 3 for the maximum number of iterations. However, if we only do one iteration, an unnecessary number of outer iterations (on k) will be needed, and it is unnecessary much work to also update the position of the prior when it is really only u that needs to be updated. This can also be seen in the table as only one iteration leads to a tripled number of outer iterations and an increased computation time.

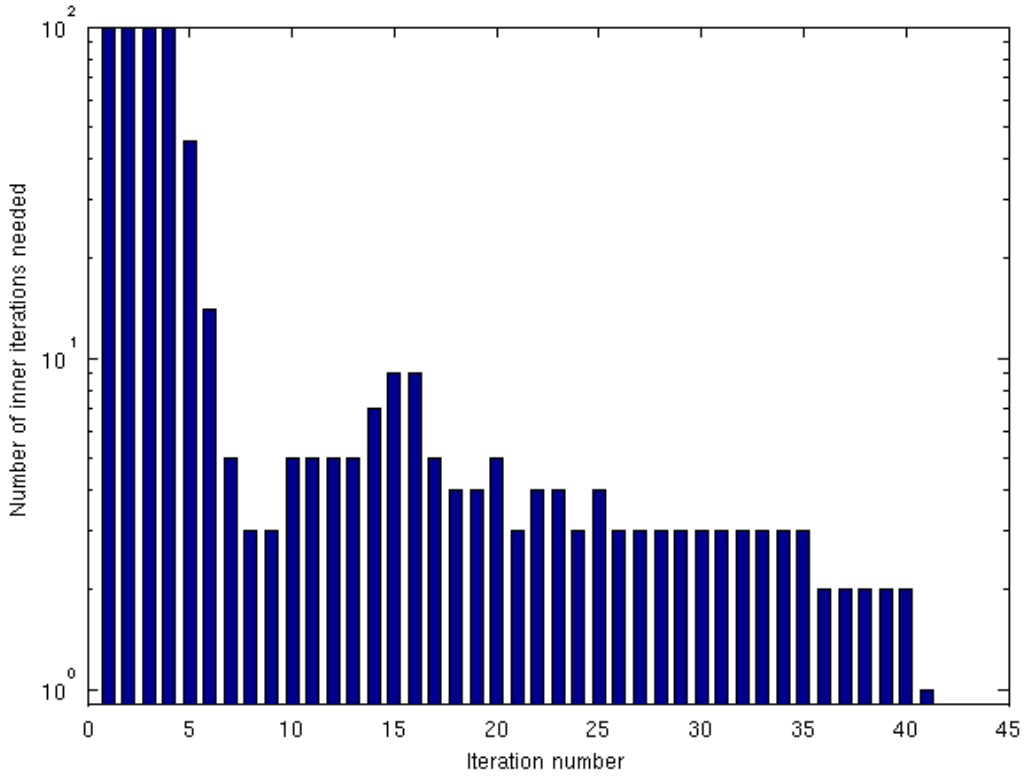


Figure 5.20: Number of iterations until convergence of u for every outer iteration, plotted on a logarithmic scale.

Chapter 6

Summary and conclusion

In this work we proposed a method for image segmentation with shape priors. We used a shape prior in the form of a characteristic function for the shape, and included the possibility of rotation, scaling and translation of this shape to make it match more closely with the object which was to be segmented. The starting point for the model was the Chan-Vese functional, which was extended with a term measuring the area of the difference between the two sets representing the foreground (segmented object) and the prior shape.

This energy functional was reformulated as a Chan-Vese model in order to be able to obtain a convex problem. Then, the functional was minimised with respect to one variable at a time. Minimisation with respect to the shape prior was achieved by updating the pose parameters with an efficient gradient descent procedure which eliminated problems with step size selection. For minimisation with respect to the characteristic function of the foreground region, u, the problem was reformulated and a fast and efficient max-flow based algorithm was explained and used. This approach is so efficient that on average only three iterations were needed for convergence of u in every step.

We tested our algorithm on several images and showed that shape prior segmentation gives better results than segmentation without shape information. The method proved to be stable and works well for several different initialisations of the pose of the shape prior and of the mean intensity values for foreground and background.

However, some initial poses will still result in the prior getting stuck in a local minimum, but this problem cannot be avoided since the energy functional is not convex with respect to the pose parameters. Specifically, the shape prior must, at its initial position, cover more of the object of interest than of other regions with similar colours, in order for the segmentation to be successful. Also, the result depends on the similarity between the shape prior

and the object of interest. For images with very similar colours in foreground and background one must use a very good shape prior and a large μ to get satisfactory segmentation results. For images with more distinct colours however, μ does not have to be as large, and so the method can give very good results even with a prior that is very different from the object. All in all, the proposed method works very well on a large variety of natural colour images.

Bibliography

Books and articles

[1] R. Adams: *Calculus: A Complete Course*. Pearson Addison Wesley, 2006.

[2] T. Chan, S. Esedoḡlu and M. Nikolova: *Algorithms for Finding Global Minimizers of Image Segmentation and Denoising Models*. In SIAM J. Appl. Math., vol. 66(5):pp. 1632–1648, 2006.

[3] T. Chan and J. Shen: *Image Processing and Analysis: Variational, PDE, Wavelet, and Stochastic Methods*. Society for Industrial and Applied Mathematics, 2005.

[4] T. Chan and L. Vese: *Active Contours Without Edges*. In Image processing, IEEE transactions on, vol. 10(2):pp. 266–277, 2001.

[5] G. Charpiat, O. Faugeras and R. Keriven: *Shape Statistics for Image Segmentation with Prior*. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pp. 1–6, IEEE, 2007.

[6] F. Clarke: *Functional Analysis, Calculus of Variations and Optimal Control*, vol. 264 of *Graduate Texts in Mathematics*. Springer, London, 2013.

[7] I. Ekeland and R. Témam: *Convex Analysis and Variational Problems*. Classics in Applied Mathematics, Society for Industrial and Applied Mathematics, 1999.

[8] L. Evans: *Partial Differential Equations*. Graduate studies in mathematics, American Mathematical Society, 2010.

[9] L. Ford and D. Fulkerson: *Flows in Networks*. Rand Corporation research study, University Press, 1962.

[10] C. Gerald and P. Wheatley: *Applied Numerical Analysis*. Always learning Pearson, Pearson Education, 2007.

[11] Giusti: *Minimal Surfaces and Functions of Bounded Variation*. Monographs in Mathematics, Birkhäuser Boston, 1984.

[12] R. Gonzalez and R. Woods: *Digital Image Processing*. Pearson/Prentice Hall, 2008.

[13] J. Kim, M. Çetin and A. Willsky: *Nonparametric Shape Priors for Active Contour-Based Image Segmentation*. In Signal Processing, vol. 87(12):pp. 3021 – 3044, 2007.

[14] B. Kimia: *A Large Binary Image Database*. LEMS Vision Group at Brown University, `http://www.lems.brown.edu/~dmc/main.html`, 2002.

[15] J. Nocedal and S. Wright: *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering, Springer, 2006.

[16] N. C. Overgaard, K. Fundana and A. Heyden: *Pose Invariant Shape Prior Segmentation Using Continuous Cuts and Gradient Descent on Lie Groups*. In *Scale Space and Variational Methods in Computer Vision*, edited by X.-C. Tai, K. Mørken, M. Lysaker and K.-A. Lie, vol. 5567 of *Lecture Notes in Computer Science*, pp. 684–695, Springer Berlin Heidelberg, 2009.

[17] A. Quarteroni, R. Sacco and F. Saleri: *Numerical Mathematics*. Texts in Applied Mathematics, Springer, 2007.

[18] S. Sural, G. Qian and S. Pramanik: *Segmentation and Histogram Generation Using the HSV Color Space for Image Retrieval*. In *ICIP. VIIth Digital Image Computing: Techniques and Applications, Sun C., Talbot H., Ourselin*, pp. 589–592, 2002.

[19] O. Veksler: *Star Shape Prior for Graph-Cut Image Segmentation*. In *Computer Vision–ECCV 2008*, pp. 454–467, Springer Berlin Heidelberg, 2008.

[20] J. Yuan, E. Bae and X.-C. Tai: *A Study on Continuous Max-Flow and Min-Cut Approaches*. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2217–2224, IEEE, 2010.

[21] J. Yuan, E. Bae, X.-C. Tai and Y. Boykov: *A Study on continuous Max-Flow and Min-Cut Approaches*. Technical report CAM-10-61, UCLA, CAM, UCLA, 2010.

[22] J. Yuan, E. Bae, X.-C. Tai and Y. Boykov: *A Spatially Continuous Max-Flow and Min-Cut Framework for Binary Labeling Problems*. In Numer. Math., vol. 126(3):pp. 559–587, 2014.

[23] J. Yuan, C. Schörr and G. Steidl: *Simultaneous Higher-Order Optical Flow Estimation and Decomposition*. In SIAM J. Sci. Comput., vol. 29(6):pp. 2283–2304 (electronic), 2007.

Images

[24] C.-A. Fooks: *Pacific Blue Tang*. Queensland, Australia, `http://www.livingtravel.com/australia/queensland/reef_fish.htm`, 2014.

[25] H. Richter: *Citrus Sinensis*. richterfoto, iStockphoto, `http://www.istockphoto.com/stock-photo-8965418-citrus-sinensis.php?st=b9b02df`, 2009.

[26] B. Maxur: *A Viceroy Butterfly*. flickr, iStockphoto, `https://www.flickr.com/photos/44545509@N00/3836891925/`, 2009.

[27] Imigion: *Awsome Dark Blue Flower*. imigion, `http://www.imgion.com/images/01/Awesome-Dark-Blue-Flower-.jpg`, 2013.

[28] M. Porter: *Happy Fruit Grass*. flickr, `https://www.flickr.com/photos/libraryman/95516793/`, 2003.

[29] R.A. Butler: *Blue Tang*. Mongabay.com, San Fransisco, `http://travel.mongabay.com/panama/images/pan01-0490.html`, 2006.

[30] Wallpapers2014: *Cool Fish Wallpaper*. `http://wallpapers2014.com/cool-fish-wallpaper/`, 2014.

[31] Fanpop: *Seahorse*. `http://www.fanpop.com/clubs/seahorses/images/30701586/title/seahorse-photo`, 2006-2014.

[32] S. Blanc: *Photos de Manchot Empereur*. Horizons Partages, `http://www.sblanc.com/phototheque-polaire/photos-dantarctique/photos-de-manchot-empereur/`, 2005.

[33] M. Berger: *Pingvin*. Dagbladet, http://www.dagbladet.no/nyheter/2005/10/09/445797.html, 2005.

[34] Ztona: *Gentoo Penguin*. http://ztona.org/gentoo-penguin/, 2014.

[35] A. Mandemaker: *Adelie Penguin*. Wikimedia, http://commons.wikimedia.org/wiki/File:Adelie_Penguin_%28Pygoscelis_adeliae%29_on_iceberg.jpg, 2006.

[36] Ozwildlife: *Blue Tang*. Underwater World, Mooloolaba, Sunshine Coast, Queensland, http://www.ozanimals.com/Fish/Blue-Tang/Paracanthurus/hepatus.html, 2014.

[37] J. Cross: *The Viceroy Butterfly*. Northwest Ohio Nature, http://www.ohio-nature.com/viceroy-butterfly.html, 2006-2013.

[38] Foto Hewan: *Kuda Laut Mas*. http://fotohewan.info/ciri-khusus-kuda-laut-beserta-klasifikasinya/kuda-laut-mas/, 2013.