Sambhunath Biswas, Brian C. Lovell

Bézier and Splines in Image Processing and Machine Vision

First Edition, SPIN Springer's internal project number, if known

Theory and Applications

August 27, 2007

Springer
Berlin Heidelberg New York
Hong Kong London
Milan Paris Tokyo

Preface

The rapid development of spline theory in the last five decades and its widespread applications in many diverse areas, has not only made the subject rich and diverse but have also made it immensely popular within different research communities. It is well established that splines are a powerful tool and have tremendous problem solving capability. Of the large number of splines discovered so far, a few have established permanent homes in computer graphics, image processing, and machine vision. In computer graphics, their significant role is well documented. Unfortunately this is not really the case in machine vision, even though a great deal of spline-based research has already been done in this area. The situation is somewhat better for image processing. One, therefore, feels the need for something in the form of a report or a book that clearly spells out the importance of spline functions while teaching a course on machine vision. It is unfortunate that despite considerable searching not even a single book in this area was found in the market. This singular fact provides the motivation for writing this book on splines with special attention to applications in image processing and machine vision.

The philosophy behind writing this book lies in the fact that splines are effective, efficient, easy to implement, and have a strong and elegant mathematical background as well. Its problem solving capability is, therefore, unquestionable. The remarkable spline era in computer science, starts when P. E. Bézier first published his work on UNISURF. The subject immediately caught the attention of many researchers. The same situation was repeated with the discovery of Ingrid Daubechi's wavelets. Different wavelet splines are now well known and extensively found in the literature. As splines are rich in properties, they provide advantages in designing new algorithms and hence they have wide scale applications in many important areas. Bézier and wavelet splines, can, therefore, be regarded as two different landmarks in spline theory with wide application in image processing and machine vision, and this justifies the title of the book.

In writing this book, therefore, we introduce the Bernstein polynomial at the very beginning, since its importance and dominance in Bézier spline models for curve and surface design and drawing are difficult to ignore. We have omitted the design problems of curves and surfaces because they are dealt in almost all the books of computer graphics. Some applications in different image processing areas, based on Bézier-Bernstein model, are discussed in depth in chapters 1, 2, 3, and 4, so that researchers and students can get a fairly good idea about them and can apply them independently.

B-splines discussed in chapter 5, are useful to researchers and students of many different streams including Computer Science and Information Technology, Physics, and Mathematics. We have tried to provide a reasonably comprehensive coverage. Attention has been devoted to writing this chapter so that students can independently design algorithms that are sometimes needed for their class work, projects, and research. We have also included applications of B-splines in machine vision because we believe it also has strong potential in research. Beta splines in chapter 6 are relatively new and much work remains to be been done in this area. However, we have tried to discuss them as much as possible and have indicated possible directions of further work.

Discrete splines in chapter 7, are discussed along with the feasibility of their use in machine vision. The application is appropriate and informative. Wavelet splines are relatively new so we have taken special care to write the theory in a clear, straightforward way in chapter 8, so that students and researchers can use it without difficulty. To aid in understanding, we have used examples whenever necessary.

Snakes and active contours are explained in chapter 9 and we discuss their intimate relationship with mathematical splines. Minimizing snake energy using both the original calculus of variations method and the dynamic programming approach are discussed clearly. This also includes problems and pitfalls drawn from several applications to provide a better understanding of the subject. Chapter 10, on the other hand, discusses powerful globally optimal energy minimization techniques, keeping in mind the need of students and researchers in this new and promising area of image processing and machine vision.

Finally, we believe that this book would help readers from many diverse areas, providing a reasonably good coverage of the subject. We believe, it can be used in many different areas of image processing and machine vision. It is our hope that this book differs from many other books as we have made a considerable effort to make these techniques are easy to understand and implement as possible. We do hope the reader will agree with us.

Sambhunath Biswas,
India,
March 2007

Brian C. Lovell
Australia
March 2007

Contents

Part I

Early Background

Part II

Intermediate Steps

Part III

Advanced Methodologies

1

Snakes and Active Contours

1.1 Introduction

1.1.1 Splines and Energy Minimisation Techniques

If we examine the origin and historical usage of splines there is a clear linkage between smooth curves and energy minimisation. Splines were originally thin, wooden strips used in both traditional ship and aircraft building techniques to help create the curved hulls needed to allow the ship or aircraft to travel speedily through the water or air. The splines were used for the process of lofting which is required to expand the small scale plan from a boat designer into the full-size plans required for construction. The rescaling was done by transferring a series of measurements called offsets on to the large lofting floor of the lofthouse and then interpolating these offset points by bending splines into smooth interpolative curves. The wooden splines were bent into shape with the aid of a small number of weights, called ducks, and clamps which kept the wood in position as shown in figure 1.1. The ducks provided positional constraints and the clamps could provide both positional and tangential (i.e. derivative) constraints. Wooden splines have a natural tendency to assume a smooth shape to minimise overall bending energy while satisfying the imposed constraints.

This energy minimisation approach to producing smooth curves is fundamentally different from the approach taken with Bernstein-Bézier and B-splines. In the former approach the smoothness derives naturally from the minimisation of energy — just as in the case of the wooden spline. By contrast, the latter approach ensures smoothness by representing a shape as a sum of smooth mathematical functions. So it could be persuasively argued that energy minimisation methods are more faithful to the historical concept of splines than the modern concept of mathematical splines developed by Bézier *et al.*

Indeed Kass, Witken, and Terzopoulos [1, 2] referred to snakes as a form of spline in their groundbreaking paper presented at the very first ICCV

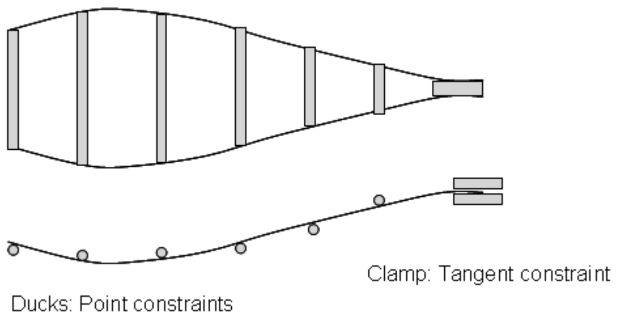

Fig. 1.1. Wooden splines for boat building

held in London in 1987. They introduced the concept of snakes by stating that,"A snake is an energy minimizing spline guided by external contraint forces and influenced by image forces that pull it towards lines and edges." Unlike Bernstein-Bézier splines, the splines of Kass *et al* could determine their own control points directly from the image under analysis by using constraints based on image intensity and gradient. This was an exciting time in computer vision research and represented a major break from image analysis via the sequential linking of low level features such as edges and intensities.

Kass, Witken, and Terzopoulos were working for Schlumberger Palo-Alto Research and were interested in using snakes to speed up the manual labelling of seismic data as required for oil exploration. These seismic images are complicated and their interpretation requires the input of interpretation experts. Indeed, sometimes there would be little agreement between the experts. In the words of Kass *et al* [1]:

> Different seismic interpreters can derive significantly different perceptual organisations from the same seismic sections depending on their knowledge and training. Because a single 'correct' interpretation cannot always be defined, we suggest low-level mechanisms which seek appropriate local minima instead of searching for global minima.

Thus the authors saw snakes as an interactive "power assist" for manual labelling by a human expert rather than as a fully automatic image interpreter algorithm in its own right. The human expert would adjust the snake by hand until it was close to the desired solution, and the snake energy minimisation would do the rest.

In the case of seismic sections, the effort of manually labelling the images is relatively low compared to the huge expense and effort of collection. However in the case of, say, computer analysis of closed circuit television feeds for building security, the process of image analysis must be completely automatic and reliable to be effective. Fortunately, in many image labelling tasks, interpretation is much more straight-forward than for seismic sections. Indeed, in many cases virtually all people would agree on the same interpretation.

For example, when presented with a photograph of an unobstructed person, almost everyone would agree on the same partitioning of such an image into person and background. Yet, this important and seemingly trivial task of image labelling is extraordinarily difficult to achieve automatically. It turns out that snakes and general energy minimisation techniques are some of the most promising methods for automated analysis — though all methods have their weaknesses.

1.2 Classical Snakes

An active contour or snake as proposed by Kass *et al* [1] is a closed or open curve defined within a 2D image domain that is able to evolve or deform to conform to features, such as edges and lines, in the image under analysis. The evolution of the snake is formulated as an iterative energy minimization process in which the snake is deformed to reach a locally minimum energy configuration.

The total energy associated with the snake is defined as the sum of an *internal energy term*, an *external energy term*, and an *external forces term*. The internal energy influences the shape and smoothness of the snake and depends only on the properties of the snake itself, independent of the underlying image (*cf* the bending strain in a wooded spline). The external energy is what causes the snake to align itself with image features and is derived from the underlying image. The force term allows the user to manually force the snake to move in particular directions to aid in finding the best solution.

In general, curves cannot be described by one-dimensional functions as they may double back on themselves, so we parameterise the snakes along their length as follows:

$$\nu(s) = (x(s), y(s)), s \in [0, 1]. \tag{1.1}$$

Thus as s varies from 0 to 1 inclusive, we traverse the entirety of the snake. In practice, we discretise this parameterisation and evaluate the energy of the snake at, say, N sample points, often called control points, along the contour. These points actually define the snake so they must be spaced somewhat closer than would be the case for the control points of a Bernstein-Bézier spline — generally they are spaced just a few pixels apart so that small image features are not missed.

Thus we have initially a set of N points such that

$$\nu_n = \nu(s)|_{s=k/N}, k \in [0..N-1]. \tag{1.2}$$

In other words, we place the N control points successively along the length of the snake at locations $(x_n, y_n) = \nu_n = \nu(s)$ evaluated at monotonically increasing values of s by assumption. Normally, we try to space the points evenly along the snake initially. However, even if we don't, the membrane term (see section 1.3) of the snake internal energy will quickly even out the points during the evolution phase. Figure 1.3 shows the evolution of a closed snake when applied to the cell segmentation problem.

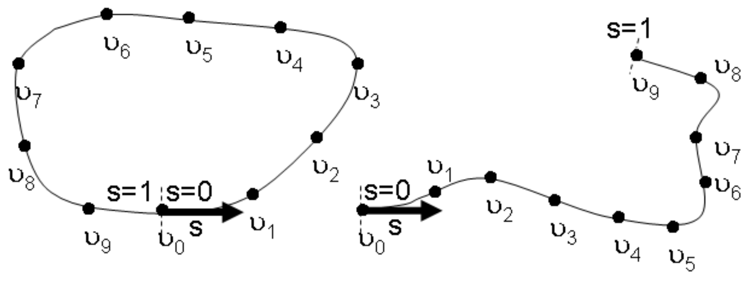

Fig. 1.2. Parameterised and discretised closed and open snakes

1.3 Energy Functional

The initial position of the snake is usually specified by the user based on *a priori* knowledge of the image under analysis. Often the initial snake may be drawn with a mouse or drawing tablet for convenience. Once initialised, the evolution of the snake can be considered as the process of minimising the following energy functional[1]:

$$E_{snake} = \int_0^1 E_{int}(\nu(s)) + E_{image}(\nu(s)) + E_{forces}(\nu(s)) \tag{1.3}$$

where E_{int} is the internal energy term, E_{image} is the image energy term, and E_{forces} is the external forces constraints term. In Kass *et al* [1], the internal energy of the snake is defined as follows:

$$E_{int}(\nu(s)) = \alpha(s) \underbrace{\left|\frac{\partial}{\partial s}\nu(s)\right|^2}_{\text{membrane term}} + \beta(s) \underbrace{\left|\frac{\partial^2}{\partial s^2}\nu(s)\right|^2}_{\text{thin-plate term}} /2\,. \tag{1.4}$$

The spline energy is defined by a first-order term controlled by $\alpha(s)$ and a second-order term controlled by $\beta(s)$. The first-order term provides behaviour similar to the elasticity exhibited by a membrane[2] and the second-order term provides behaviour similar to the stiffness exhibited by a thin metal plate.

[1] A functional is a function of a function.

[2] The equation (1.4) is a membrane equation known from mechanics combined with a stiffness-term.

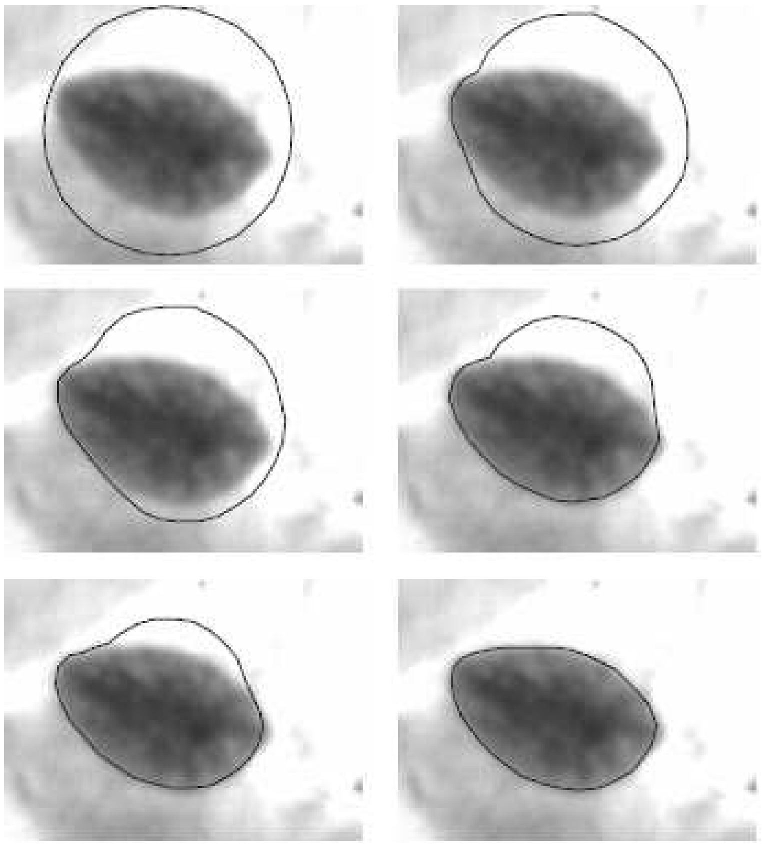

Fig. 1.3. Application of a Kass *et al* closed snake for screening for cervical cancer using Pap smear images. Here we wish to segment the cell nucleus from the cytoplasm (from [3]).

The behaviour of snakes is easier to understand if we examine the discrete form of the internal energy as follows [4]:

$$E_{int}(\nu_i) = \alpha_i \underbrace{|\nu_i - \nu_{i-1}|^2}_{\text{membrane term}} + \beta_i \underbrace{|\nu_{i+1} - 2\nu_i + \nu_{i-1}|^2}_{\text{thin-plate term}}. \tag{1.5}$$

Now the membrane term

$$|\nu_i - \nu_{i-1}|^2 = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$$

just represents the square of the distance between successive control points. Since E_{int} is the sum of the squared distances between the control points, this energy is minimised when the distances are all equal and the control points are collinear. In the case of an open snake this low energy configuration will be satisfied by a straight line with uniformly spaced control points. Note that if there are no external forces imposed, the sum of the membrane terms in the snake energy is minimised by contracting all control points into a single point; just like a soap bubble which becomes a tiny droplet when the air escapes. The membrane term is often considered to be providing *elasticity* — it makes the snake shrink during evolution somewhat like a stretched elastic band.

The membrane term also penalises curvature indirectly because curvature increases the snake energy by increasing the distance between the control points — a straight line is always the shortest distance between two points in a Euclidean space. Note that in the case of a closed snake there must always be some curvature to allow the snake to connect back on to itself.

The second-order or thin-plate term in (1.5) penalises changes in curvature and makes the snake behave like a thin metal plate.[3] The thin-plate term only penalises *changes* in the distance between the control points. This becomes obvious if we rewrite the argument of the modulus in the thin-plate term of (1.5),

$$\nu_{i+1} - 2\nu_i + \nu_{i-1},$$

in the form

$$(\nu_{i+1} - \nu_i) - (\nu_i - \nu_{i-1}).$$

So unlike the membrane term, minimisation of the thin-plate term does not provide the elastic behaviour that collapses the snake to a single point under evolution. Rather it provides *stiffness* as exhibited by, say, a thin metal plate that ensures that both the control-points and the curvature are uniformly distributed. This term makes the snake form smooth curves during evolution just like the traditional wooden spline for lofting in shipbuilding. Thus during evolution a closed snake with no external constraints will tend to become circular due to the stiffness provided by the thin-plate term before it finally collapses to a single point due to the elasticity provided by the membrane term.

The image energy is formulated so that its value is minimal at the location of the desired image features. Kass *et al*[1] considered a weighted set of features based on lines, edges, and terminations (*i.e.,* the end points of lines) as follows:

$$E_{image} = w_{line} E_{line}(\nu_s) + w_{edge} E_{edge}(\nu_s) + w_{term} E_{term}(\nu_s). \tag{1.6}$$

In this chapter and henceforth we will only consider edge energy, so a suitable image energy term is:

$$E_{image} = E_{edge}(\nu(s)) = -\left|\nabla I(x,y)\right|^2 \tag{1.7}$$

where $\nabla I(x,y)$ is the gradient of the image intensity.

[3] *cf* a thin-plate spline is the surface with minimum mean square second derivative energy that interpolates a given collection of points.

Finally the force term can be expressed by the following term [2]:

$$E_{force} = -k(x_1 - x_2)^2 \tag{1.8}$$

This force energy E_{force} represents the energy of a spring connected between a point x_1 on the contour and some point x_2 in the image plane. In practice, there could be multiple force terms — one for each spring added. These forces may be used by a human expert to direct and guide the evolution of the snake.

Special cases and variations on a theme

By adjusting the values of the α and β terms in the internal energy of (1.5), snakes of varying elasticity and stiffness can be produced. If β_i is set to zero at control point ν_i, we allow the snake to become second-order discontinuous (flexible) at ν_i and develop a corner. This is analogous to folding a piece of corrugated cardboard to make a cardboard box — the fold then behaves like a flexible hinge between the stiff cardboard sides. This property allows snakes to better conform to corners of objects such as car numberplates and allows for the possibility of embedding shape grammars into snakes.

In some applications, the contractive behavior of the membrane term is inconvenient as it may pull the snake away from the best solution. In such cases, setting α to a low value or zero yields the so-called *thin-plate splines* which behave much more like wooden splines and are best compared to Bernstein-Bézier and B-splines.

Due to the contractive nature of the membrane term, snakes must always be initialised outside the region of interest, so they can contract down onto the image features like a contracting elastic band. In some situations this behaviour may be inconvenient. For this reason, Cohen [5] proposed both inflationary and deflationary forces normal to the surface of closed snakes to force them to either grow or shrink as illustrated in figure 1.4; he used the term *balloons* to refer to these modified closed snakes. Balloons can be initialised either within or outside image objects of interest. Figure 1.5 shows the application of a balloon to the cell image segmentation problem.

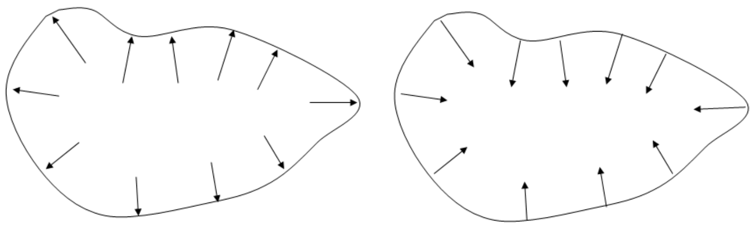

Fig. 1.4. Balloons with inflationary and deflationary forces.

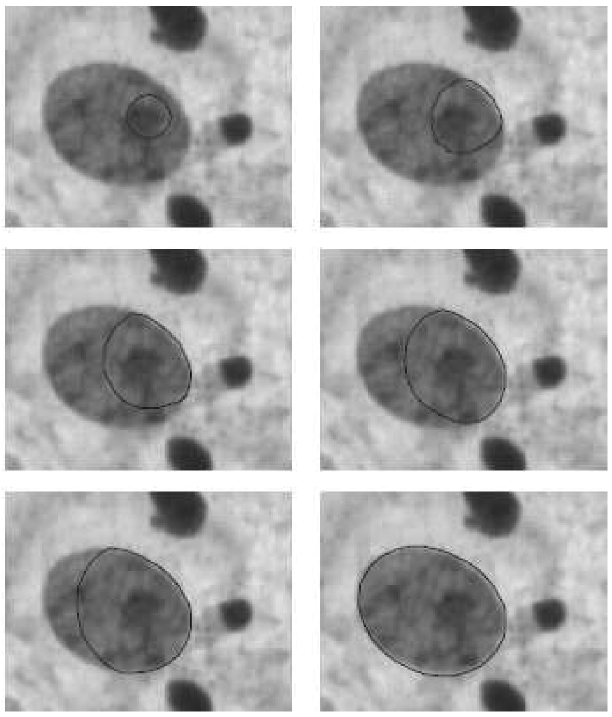

Fig. 1.5. Balloon applied to the cell image segmentation problem (from [3]).

1.4 Minimizing the Snake Energy Using the Calculus of Variations

As minimising the snake energy is an optimisation problem we can use techniques from calculus of variations. In particular, we will use Lagrangian multipliers.

Following the development of Amini, Weymouth, and Jain [4], we let $E_{ext} = E_{image} + E_{forces}$ where E_{ext} is the external energy. Substituting (1.4) into (1.3) we have

$$E_{snake} = \int_0^1 \alpha(s) \left| \frac{\partial}{\partial s} \nu(s) \right|^2 + \beta(s) \left| \frac{\partial^2}{\partial s^2} \nu(s) \right|^2 + E_{ext}(\nu(s))\, ds. \tag{1.9}$$

For simplicity, we represent the integrand by $F(s, \nu_s, \nu_{ss})$, then the Euler-Lagrange necessary condition for minimisation is derived by

$$F_\nu = \frac{\partial}{\partial s} F_{\nu_s} + \frac{\partial^2}{\partial s^2} F_{\nu_{ss}} = 0. \tag{1.10}$$

Substituting the terms in the above equation, we obtain a pair of independent Euler-Lagrange equations,

$$-\alpha x_{ss} + \beta x_{ssss} + \frac{\partial E_{ext}}{\partial x} = 0$$

and

$$-\alpha y_{ss} + \beta y_{ssss} + \frac{\partial E_{ext}}{\partial y} = 0.$$

This is best to solve numerically. The Euler-Lagrange equations with

$$f_x(i) = \partial E_{ext} / \partial x_i$$

and

$$f_y(i) = \partial E_{ext} / \partial y_i$$

are discretised, yielding

$$\begin{aligned}\alpha_i(\nu_i - \nu_{i-1}) - \alpha_{i+1}(\nu_{i+1} - \nu_i) &+ \beta_{i-1}(\nu_{i-2} - 2\nu_{i-1} + \nu_i) \\ &- 2\beta_i(\nu_{i-1} - 2\nu_i + \nu_{i+1}) + \beta_{i-1}(\nu_i - 2\nu_{i+1} + \nu_{i+2}) \\ &+ (f_x(i), f_y(i)) = 0.\end{aligned}$$

Writing the equation in matrix forms, one for x and another for y yields

$$Ax + f_x(x, y) = 0$$

and

$$Ay + f_y(x, y) = 0.$$

We can now solve for position vectors iteratively by,

$$x_t = (A + \gamma I)^{-1}(\gamma x_{t-1} - f_x(x_{t-1}, y_{t-1}))$$

and

$$y_t = (A + \gamma I)^{-1}(\gamma y_{t-1} - f_u(x_{t-1}, y_{t-1})).$$

Amini, Weymouth, and Jain identified several problems with the above calculus of variations approach as originally proposed by Kass, Witten, and Terzopoulos in 1987. In particular, they raised the following objections:

1. There is a significant risk that the above procedure does not converge
2. Optimality cannot be guaranteed as the Euler-Lagrange equations are a necessary but not a sufficient condition for optimality in a local sense
3. Constraints are required to be differentiable, which cannot be guaranteed in general
4. The requirement for differentiability of the images will lead to instability unless the image is smoothed leading to poor localisation of features
5. If a snake is not subject to appropriate external forces it will contract to a line or a point

6. If a snake is not placed close to image features, it will not get attracted.

For these reasons, they proposed the dynamic programming approach to minimizing the energy [4]. This method has now been adopted as the standard algorithm by most researchers and will be used henceforth in this chapter.

1.5 Minimising the Snake Energy Using Dynamic Programming

One of the most popular methods today is the dynamic programming approach as implemented by the Viterbi algorithm [6] as proposed by Amini, Weymouth, and Jain [4] and extended by Geiger *et al* [7]. The approach of dynamic programming is to solve the optimization problem by studying a collection, or family, of problems where the particular problem in question is a member. This concept is known as *embedding.*

The Viterbi method is closely related to Dijkstra's algortithm [8] which solves for the shortest path in a network between two points by finding the shortest paths to all points. The major difference is that the Viterbi algorithm calculates the shortest path on a *trellis* whereas Dijkstra's algorithm find the shortest path in a network. Returning to the snake minimisation problem at hand, instead of attempting to find the local minimum directly, the Viterbi algorithm efficiently evaluates a very large set of alternative solutions in the neighbourhood of the current best solution and then picks the minimum. The process is repeated until convergence is attained.

The dynamic programming formulation of snakes requires the snake to be discretized to a finite set of points in the image pixel domain as before. To limit the number of possible solutions examined, the position of each control point on the snake on the next iteration is constrained to a finite set of positions, $x_i \in X_i$, where each set X_i contains m positions. With the snake discretised and the domain of possible solutions constrained in this manner, the set of all possibilities for the next configuration of the snake can be visualised as a trellis as illustrated in figure 1.6.

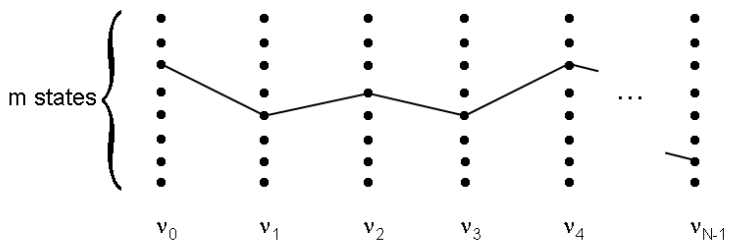

Fig. 1.6. Snake configuration space visualised as a trellis.

It is possible, but not at all practical, to exhaustively enumerate all possible configurations to determine the snake with minimum energy. This would require $\mathrm{O}(m^N)$ evaluations of the energy function, where m is the number of candidate positions and N is the number of control points forming the snake. This is prohibitively expensive for even small values of m and N. For example, if $m = N = 30$, this task would require $30^{30} = 2 \times 10^{44}$ evaluations. Assuming each energy evaluation takes just 1 microsecond, the exhaustive minimisation would require almost 7×10^{30} years — much, much longer than the age of the universe. Yet with the Viterbi algorithm we can calculate the exact same minimal value in just $O(Nm)$ time which is equivalent to just 900 evaluations, or barely one millisecond!

The Viterbi algorithm is therefore deservedly referred to as a *fast* algorithm. Along with the more famous family of Fast Fourier Transform algorithms, it is one of the classic fast algorithms of digital signal processing [9]. The inventor of the algorithm, Andrew Viterbi, was a co-founder of Qualcomm, a wireless telecommunications research and development company based in San Diego, California. The algorithm still finds wide application in communications including the widely-used V32 and V90 telephone modem standards. It is instrumental in decoding highly efficient trellis and convolutional codes for high-speed data communication. The algorithm also finds application in pattern recognition where it forms the basis of the forward and backward algorithms for learning and recognition via hidden Markov models for speech and gesture recognition [10].

Returning to the problem at hand, the Viterbi algorithm is used to efficiently calculate the optimal configuration of the snake which minimises total energy. This is possible because of the decoupled form of the discrete internal energy function of (1.5). We observe that the internal energy of each control point is only dependent on the points immediately preceding and following it. So the total energy of the snake can be written in the form:

$$E_{snake} = E_1(\nu_0, \nu_1, \nu_2) + E_2(\nu_1, \nu_2, \nu_3) + \ldots + E_{N-2}(\nu_{N-3}, \nu_{N-2}, \nu_{N-1}). \tag{1.11}$$

Dynamic Programming and the Principle of Optimality

Dynamic programming in general can be applied to any problem which observes the *Principle of Optimality.* Bellman [11], the inventor of dynamic programming, states:

> An optimal policy has the property that, whatever the initial state and optimal first decision may be, the remaining decisions constitute an optimal policy with regard to the state resulting from the first decision.

If a problem observes the Principle of Optimality it means that optimal solutions of subproblems can be used to find the optimal solutions to the overall problem.

In the case of the shortest (or equivalently minimum energy) path problem in a network as addressed by Dijkstra's algorithm [8], this means that all subpaths A to B, say, of the shortest path from A to Z must themselves be shortest paths. In order to explain this seemingly obvious but very powerful principle better, let's consider the the shortest path between the cities of Brisbane and Sydney. We assume that the shortest path between these cities passes through the city of, say, Armidale. Immediately we can say that the shortest path between Brisbane and Armidale is just that section of the Brisbane-Sydney shortest path that lies between Brisbane and Armidale. Why? Well, if there existed a path between Brisbane and Armidale that was shorter than the one already found, then that original path from Brisbane to Sydney via Armidale could not have been the shortest path between those cities — so we have proof by contradiction. Hence all subpaths of a shortest path must themselves be shortest paths between their respective endpoints. Reversing the argument, we see that new shortest paths can be found by recursively extending known shortest paths.

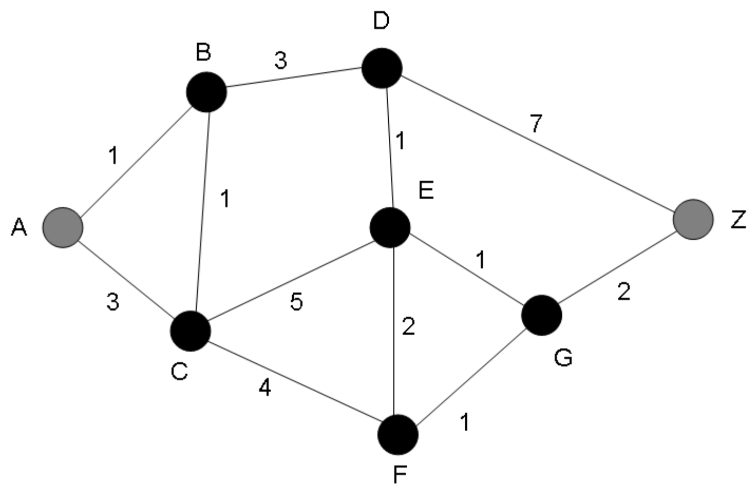

Fig. 1.7. Shortest path in a graph or network problem.

Dijkstra's Algorithm for the shortest path on a network

The above insight leads directly to Dijkstra's algorithm for the single-source shortest path problem for a directed graph with nonnegative edge weights. Let us determine the shortest path from node A to Z, say, in the network of figure 1.7. We know initially that the shortest path to A is the null path of cost 0. From the Principle of Optimality we know our solution can be obtained by extending known shortest paths, in this case the null path. So we follow all paths leading out of A to reach the following nodes with their associated path costs:

$$\text{(known)}\ \ A0 \mid B1, C3.\ \ \text{(trial)}$$

Here we have partitioned our nodes into *known*, where we now know the shortest path to the node, and *trial* where we are yet to determine the shortest path. Now AB of cost 1 must be the shortest path from A to B as any alternative path must go via AC which has cost 3 already. It will always be the case that the trial node with minimum path cost found so far will be the end node of a new shortest path. Now we know that the shortest path AZ can be constructed by extending shortest paths, so now we follow all paths leading out of B, except those leading back to known nodes, and then add B to the known nodes list, which yields the following nodes and costs:

$$\text{(known)}\ \ A0, B1 \mid C2, D4.\ \ \text{(trial)}$$

In this case, we have found an alternate route to C via B with cost 2 which is lower than the cost found so far. So we have updated the cost to C with the new value. We now know that there is no shorter path to C other than the one we have found since any alternate path would have to go via D which has cost 4. This process is repeated until we reach reach node Z. The complete sequence is given in Table 1.1. All that remains to complete the algorithm is to maintain a list of backward pointers for each node which will allow us to backtrack along the shortest paths to the starting node. The steps of the algorithm including the backward pointers are shown in Table 1.2 and the pseudo-code is given in figure 1.8.

A	B	C	D	E	F	G	Z
0	∞	∞	∞	∞	∞	∞	∞
0	1	3	∞	∞	∞	∞	∞
0	**1**	2	4	∞	∞	∞	∞
0	**1**	**2**	4	7	6	∞	∞
0	**1**	**2**	**4**	**5**	6	6	11
0	**1**	**2**	**4**	**5**	**6**	6	11
0	**1**	**2**	**4**	**5**	**6**	**6**	8
0	**1**	**2**	**4**	**5**	**6**	**6**	**8**

Table 1.1. Evolution of Dijkstra's algorithm on the network of Figure 1.7. Shortest path costs to known nodes are shown in bold. The cost of the shortest path from A to Z is 8.

A	B	C	D	E	F	G	Z
0	∞	∞	∞	∞	∞	∞	∞
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

A	B	C	D	E	F	G	Z
0	1	3	∞	∞	∞	∞	∞
$\emptyset$	A	A	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

A	B	C	D	E	F	G	Z
0	**1**	2	4	∞	∞	∞	∞
$\emptyset$	A	B	B	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

A	B	C	D	E	F	G	Z
0	**1**	**2**	4	7	6	∞	∞
$\emptyset$	A	B	B	C	C	$\emptyset$	$\emptyset$

A	B	C	D	E	F	G	Z
0	**1**	**2**	**4**	5	6	∞	11
$\emptyset$	A	B	B	D	C	$\emptyset$	D

A	B	C	D	E	F	G	Z
0	**1**	**2**	**4**	**5**	6	6	11
$\emptyset$	A	B	B	D	C	E	D

A	B	C	D	E	F	G	Z
0	**1**	**2**	**4**	**5**	**6**	6	11
$\emptyset$	A	B	B	D	C	E	D

A	B	C	D	E	F	G	Z
0	**1**	**2**	**4**	**5**	**6**	**6**	8
$\emptyset$	A	B	B	D	C	E	G

A	B	C	D	E	F	G	Z
0	**1**	**2**	**4**	**5**	**6**	**6**	**8**
$\emptyset$	A	B	B	D	C	E	G

Table 1.2. Stages of Dijkstra's algorithm on the network of figure 1.7 including the backward pointers. Shortest path costs to known nodes are shown in bold. The shortest path to Z is the path ABDEGZ of cost 8.

It is useful to visualise the evolution of Dijkstra's algorithm as an expanding wavefront. At each stage of the algorithm, the minimum distance node is found and paths leading out of this node are followed yielding another shortest path. Dijkstra's algorithm is closely related to the Fast Marching Algorithm introduced by Sethian [12, 13] which is used in both Level Sets and Geodesic Active Contours.

```
function Dijkstra(G, w, s)           // G = graph
                                     // w = costs
                                     // s = start node
for each vertex v in V[G]            // Initializations
      d[v] := infinity
      previous[v] := undefined
d[s] := 0
S := empty set                       // S = known nodes
Q := V[G]                            // Q = trial nodes
while Q is not an empty set          // The algorithm
      u := Extract_Min(Q)
      S := S union {u}
      for each edge (u,v) outgoing from u
             if d[u] + w(u,v) < d[v]
                  d[v] := d[u] + w(u,v)
                  previous[v] := u
```

Fig. 1.8. Pseudocode for Dijkstra's Algorithm for finding the shortest path on a graph or network.

Viterbi Algorithm for the Shortest Path on a Trellis

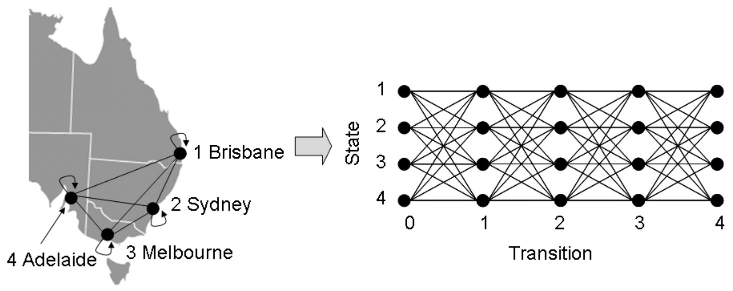

Fig. 1.9. Air transportation network converted to a trellis.

The Viterbi algorithm finds the shortest path on a trellis rather than on a network. A trellis can be derived from a network by associating a state with each node of the network and then representing the set of states vertically as a column. This column of states is then replicated horizontally to indicate increasing increments of time or equivalently transitions or "hops" between states. Connecting lines are used to show the allowable state transitions and possible paths. Note that self-transitions are often permissible on a trellis and may incur non-zero cost.

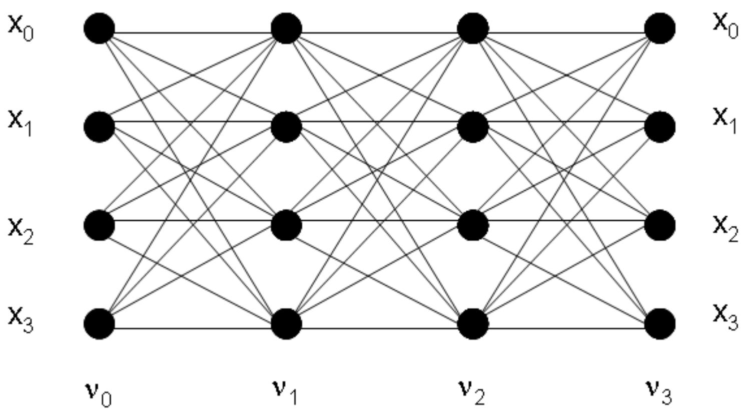

Fig. 1.10. Shortest path problem in a trellis.

Figure 1.9 shows the paths available through an air transportation network between a set of cities represented by a trellis. Each flight from city to city would be just one leg of an overall itinerary (*cf* path) and a self-transition could be a joyflight returning to the city of departure. In the above example we show direct flights between all cities, but that is not always the case in a general air transportation network. If there were no direct flights between two particular cities, a common problem is finding the cheapest itinerary requiring just N stopovers. This is a problem that can be rapidly solved using the Viterbi algorithm as follows.

Consider the trellis illustrated in Figure 1.10 with $M = 4$ states and length $N = 4$ hops. We examine paths from the starting node[4] ν_0 in state $x_i \in [x_0..x_{M-1}]$ reaching a terminating node ν_{N-1} in state $x_j \in [x_0..x_{M-1}]$ on the right. Let the distance measure $d(i, j, k)$ denote the cost of transitioning from state x_i in node ν_k to state x_j in node ν_{k+1}.

Now the cost of a path on the trellis is simply the sum of the costs along the path. For example the cost of travelling from ν_0 to ν_3 along the illustrated path in Figure 1.11 is given by

$$\text{Total Cost} = d(2, 1, 0) + d(1, 2, 1) + d(2, 3, 2).$$

Now we wish to determine the shortest path between two nodes on the trellis. The possible number of paths in a fully-connected[5] trellis equals M^N which can be an enormous number — exhaustive evaluation of all possible paths is out of the question. The trick that makes the Viterbi algorithm so incredibly fast is again due to the Principle of Optimality. This tells us that if we maintain the shortest path to each of the M states as we progress through the trellis, we can find the overall shortest path by recursively extending these M paths. Thus the computational load is linear with respect to the length of the trellis, $O(MN)$, rather than exponential, $O(M^N)$. As for Dijkstra's algorithm, we maintain a list of backward pointers so that we can eventually recover the shortest paths.

The pseudocode for the complete Viterbi algorithm is shown in figure 1.12. The algorithm progresses along the trellis one hop at a time maintaining a record of the shortest path to each destination node from the starting node. After a hop, the algorithm extends the shortest paths to the previous set of destination nodes in all directions to find the shortest paths to each of the current set of destination nodes, and then it hops again. However, unlike Dijkstra's algorithm which can extend known paths in any direction, the Viterbi algorithm marches through the trellis column by column updating the paths. If it was useful to visualise the evolution of Dijkstra's algorithm as an expanding wavefront, perhaps the Viterbi algorithm is more akin to an electromagnetic wave travelling down a waveguide.

Note that in the Viterbi algorithm pseudocode of figure 1.12, we have chosen to initialise the algorithm with the distance to the starting node set to 0 and the distance to all other nodes set to infinity (*i.e.,* unreachable) to

[4] here we use the word node to refer to a junction in the trellis which corresponds to a particular state at a particular hop count

[5] *i.e.,* where a state can transition to any other state.

force all paths to start at the starting node. Nevertheless there are situations where we may not know the best starting node or we may have a choice of several starting nodes. In these cases, we could initialise the distances to all possible starting nodes to 0 and the Viterbi algorithm would then calculate the shortest path to the best choice from the set of starting nodes. If we follow the backward pointers back from the destination node we can determine which of the stating nodes was actually chosen for the particular shortest path at hand.

Following the same reasoning, when the Viterbi algorithm is used with discrete hidden Markov models (HMMs) [10] for, say, speech recognition it is common practice to assign a particular probability to each of the starting states based on *a priori* knowledge. Indeed, when used with HMMs, the costs of transitioning between the states are actually state transition probabilities representing the likelihood of transitioning from one state to the next. In this application, the multiplication of these very small numbers leads to a risk of arithmetic underflow in the calculation of the Viterbi algorithm which is overcome by using the logarithm of probability and other scaling techniques.

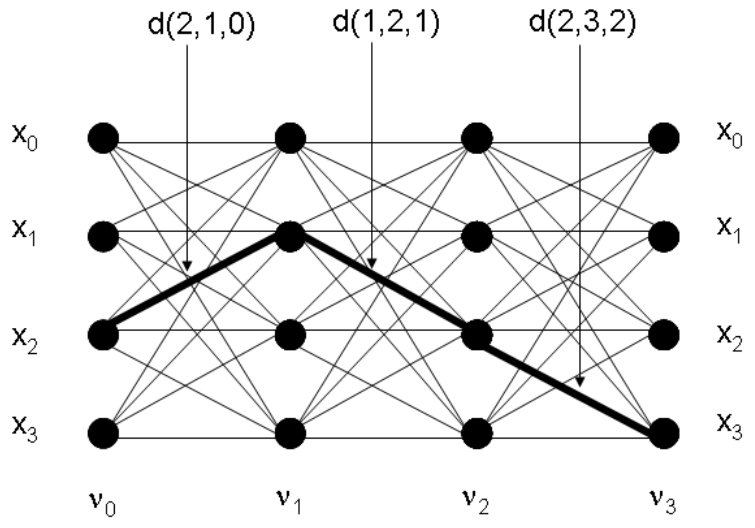

Fig. 1.11. Example path in a trellis.

Dynamic Programming for open snakes

Returning now to the calculation of evolving snakes, the usage of the Viterbi algorithm is quite straight-forward. For each control point, we determine a set of candidate points for the next evolution of the snake spread along a short

```
function Viterbi(T, w, s)                       // T = trellis
                                                // w = costs
                                                // s = start state
  for each state x in X[T]                      // Initializations
        d[x] := infinity
        previous[x,0] := undefined              // Index on state,hop
  d[s] := 0
  for each hop h in H[T]                        // H[T] is hops
       for each state v in X[T]                 // Destination states
           dist[v] := infinity                  // Flag v as trial
           for each state u in X[T]             // Source states
               for each edge (u,v) outgoing from u
                       if d[u] + w(u,v) < dist[v]
                           dist[v] := d[u] + w(u,v)
                           previous[v,h] := u
           d[v] := dist[v]                     // Flag v as known
```

Fig. 1.12. Pseudocode for Viterbi Algorithm for finding the shortest path on a trellis.

distance in a direction perpendicular to the snake as illustrated in figure 1.13. Then the problem of minimising snake energy becomes the problem of finding the minimum cost path on this (distorted) trellis. We can allow the starting node to remain fixed, or by using the initialisation trick above, we can allow both ends of the snake to move freely to a new minimal energy configuration. This algorithm can be applied iteratively to allow the snake to evolve over a larger area.

A problem that can be encountered is that the control points of the snake may bunch up and become uneven after several iterations. Thus it becomes necessary to reinterpolate the control points from time to time. Another problem common to all snake techniques is the risk of the snake crossing itself or forming loops that may be undesirable for the purposes of image segmentation. While many researchers have tackled this problem [14, 15, 16], most solutions are somewhat inelegant and add significant complexity to the approach.

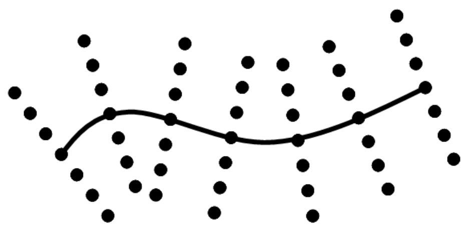

Fig. 1.13. An open snake converted to a trellis

Dynamic Programming for closed snakes

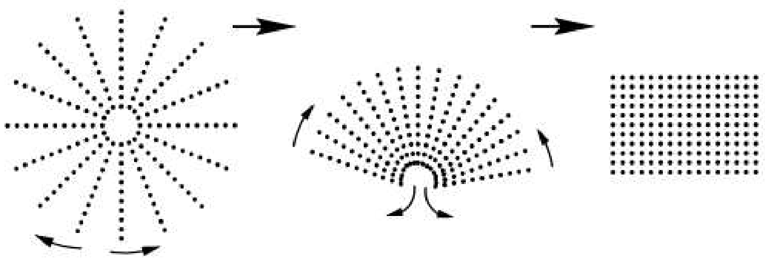
Fig. 1.14. A closed snake search space converted to a trellis

Closed snake energy minimisation presents a challenge because the efficient application of dynamic programming is not entirely straight forward. Geiger *et al* [7] address this problem in the context of dynamic programming for energy minimisation. Their approach is to unwrap the circular domain to form a linear trellis as shown in Figure 1.14. To ensure that the solution is indeed a closed contour, they examine shortest paths where the start and finish nodes have the same index and select the global minimum contour. Thus if m candidate points are to be examined, this method would require the Viterbi algorithm to be run m times for each possible starting node instead of just once. So with a value of, say, $m = 30$ the optimal closed snake evolution would run 30 times slower than the open snake evolution.

As this would be unacceptably slow, Geiger *et al* suggest a heuristic speed-up which requires only two passes of the algorithm. The Viterbi algorithm is run once using an arbitrary start point — in practice it is best to choose the candidate point with highest image gradient which is likely to be on the optimal contour. Then for the second run the contour is reordered so as to start and terminate from the second point of the trellis. The argument for this procedure would be that the second point is more likely to be on the optimal contour that the arbitrary first point since the snake energy minimisation process will have 'pulled' it toward the optimal contour. Now assuming our trellis has N control points, if the second point is more likely to be on the optimal contour than the first, why not unwrap the trellis at the $N/2^{th}$point on the other side of the circular search space?

This idea leads us to the mid-point heuristic. The mid-point heuristic can be stated as *the optimal positions of the mid-points of a snake are generally independent of the positions of the end-points.* This led Gunn and Nixon [17] to propose a similar two-pass technique to use dynamic programming to solve the closed snake problem using two open snakes. The closed snake is converted to an open snake problem by unwrapping about an arbitrary cut point as before. First an open snake minimisation is performed using no smoothness or continuity constraints on the endpoints. The two points at the mid-point of this contour are then taken as the start and end points for the closed contour. The Viterbi algorithm is run again with the start and end points fixed. Thus we only require two runs of the Viterbi algorithm instead of the m runs required for the optimal method.

Although these heuristics work well in practice, there is a theoretical possibility that they may fail to find the true optimal contour. We address this issue in section 2.2 and describe a fast and optimal minimisation method using branch and bound techniques.

1.6 Problems and Pitfalls

Traditional snakes minimise energy within a local search space only. This leads to difficulties in many applications because the snakes become stuck on local minima rather than finding global solutions which may be preferable for fully automated image segmentation. As a result of the gradient descent nature of the traditional snake, the answer obtained is very dependent on initialisation and stopping criteria and these criteria may be very difficult to determine in general.

An example of this difficulty with a traditional closed snake or balloon on the cell image segmentation problem is shown in figure 1.15 where the contour is stuck in a local minimum. If we increase the deflationary force, we may be able to contract the contour down on to the nuclear membrane. Unfortunately, we may also run the very real risk of the contour being pushed right inside the membrane — especially if the image gradient on the nuclear membrane is less than on the surrounding artefacts.

Gunn and Nixon [18] attempt to address this issue by using a dual active contour model. Their idea is to initialise balloons both inside and outside the object of interest. The inner balloon would then expand and the outer balloon would contract. If the two balloons did not meet, the inflationary and deflationary forces would be increased until the balloons were forced together. This approach has the advantage of clear initialisation and stopping criteria, but does not necessarily yield the optimal minimal energy solution in general. The dual active contour process is illustrated in figure 1.16 on the cell image problem.

1.7 Connected Snakes for Advanced Segmentation

Snakes can be used to segment quite complicated images with a little guidance from a human expert. Usually only one snake is evolved, but some situations call for a far more complex segmentation where many snakes must be evolved simultaneously. Figure 1.17 shows a set of connected objects and an initial hand drawn rough segmentation. Our goal is to use snakes to refine the rough

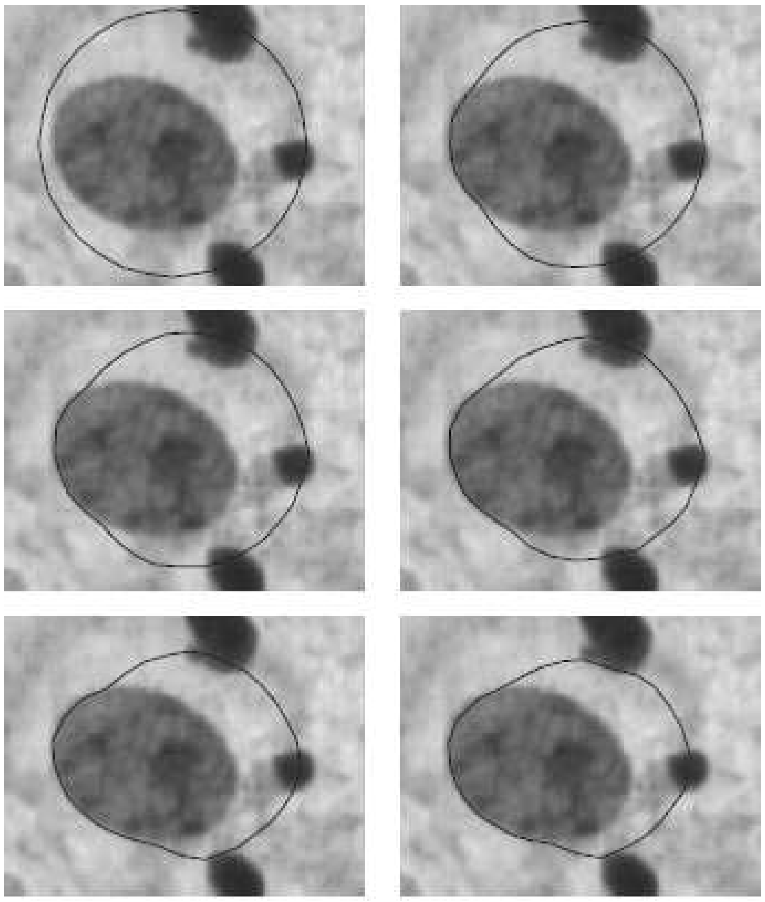

Fig. 1.15. Closed snake being prematurely stopped by a local minimum when applied to cell image segmentation problem (from [3]).

object segmentation into an acceptable segmentation with good boundary delineation. This approach was developed by Walford [19] to fuse spatial LIDAR information with image data for the automatic analysis of rock wall faces in a mine.

We treat each section of the boundary between the joins as an open-ended snake as illustrated in figure 1.18. Now our problem is to find the minimum energy configuration of snakes by evolving all snakes simultaneously. At first glance, this appears to be a very challenging problem. Nevertheless a good solution can be found if we decouple the problem by taking advantage of the mid-point heuristic as described in section 1.5.

Each snake in the network is evolved within its search space as a open ended snake, without regard to its connectivity to other snakes in the network

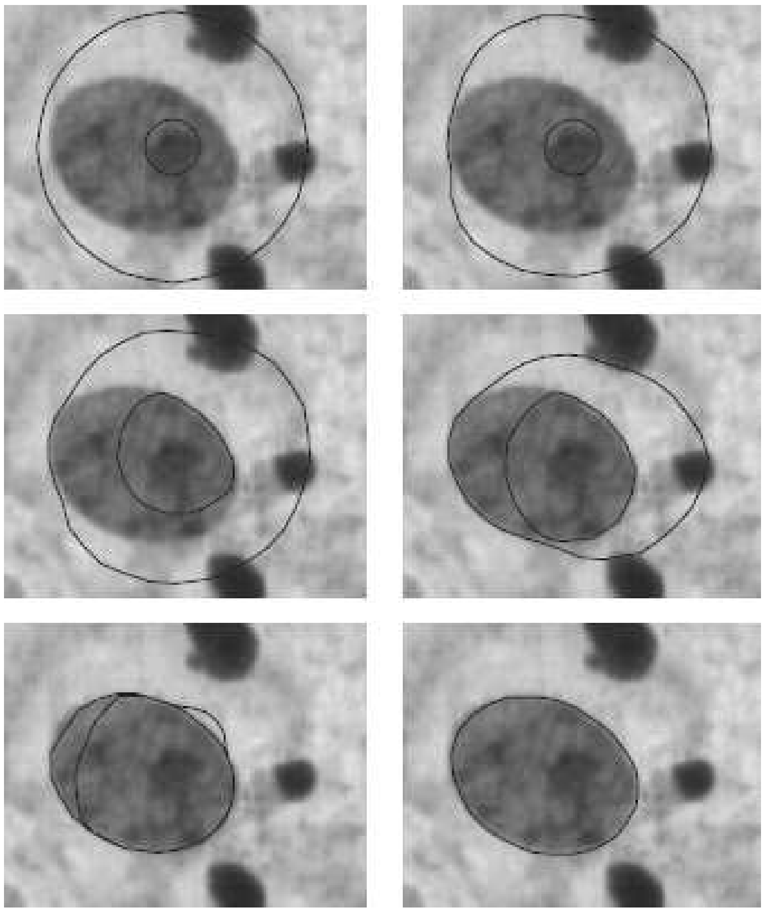

Fig. 1.16. The Gunn and Nixon dual active contour approach to handle local minimum problem applied to cell image segmentation problem (from [3]).

to estimate the optimal position of its mid-points. Next each snake is split in two at the mid-points to create two new half-snakes. We fix the location of the midpoint end of each half-snake and perform a forward pass of the Viterbi algorithm. This yields the minimum energy of each half-snake for all m possible positions of its joint-end. All half-snakes that are connected by a joint have the same m possible positions for their joint-end so we can then determine the common joint-end position that minimises the total energy of all half-snakes that meet at that joint. Once we know the common joint-end position, we can follow the backward pointers of the Viterbi algorithm to determine the position of all remaining points on the half-snake. The refined half-snakes are then reconnected to form the final result as shown in figure 1.19.

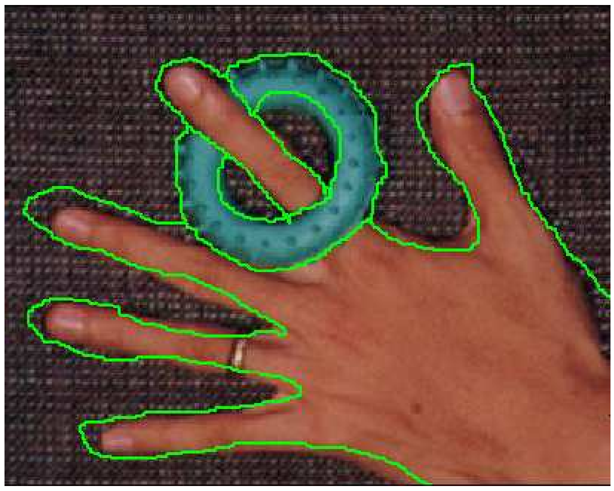

Fig. 1.17. Hand drawn segmentation of a connected object (from [19])

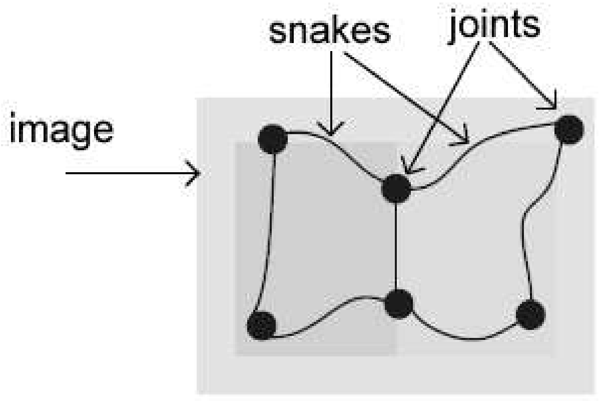

Fig. 1.18. Viewing segmentation boundaries as a network of connected snakes (from [19])

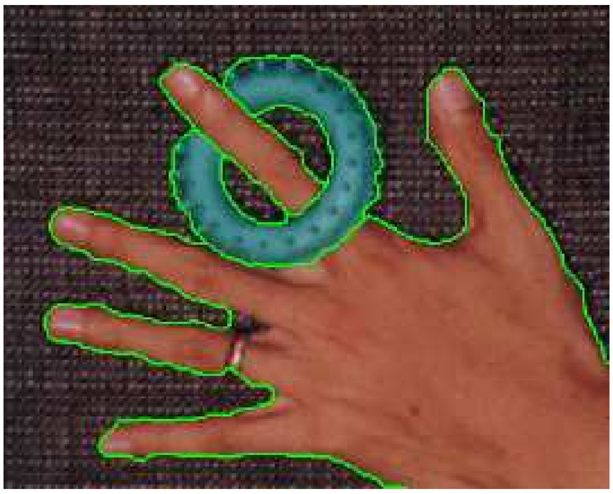

Fig. 1.19. Refined segmentation of the connected object using a connected snake network (from [19])

1.8 Conclusions

Snakes use energy minimisation techniques to form smooth curves. However, snakes are mainly used for image segmentation and interpretation rather than mathematical interpolation *per se*. Rather than interpolating between known control points as is the case with Bernstein-Bézier splines, snakes find their own control points using image features such as edges, lines, and line terminations in an image under analysis. The formulation of internal snake energy has a membrane term that provides a form of elasticity similar to an elastic band, and a thin-plate term that provides a form of stiffness like a traditional wooden spline.

Traditionally a local gradient descent method is used to determine the minimum energy contour. This leads to the well-known pitfalls in the application of conventional snakes due to the inability to find satisfactory answers to the following problems.

- How do we intitialise the snake to find the best solution?
- When do we stop the snake evolving?
- How do we avoid unsatisfactory local minima?

Gunn and Nixon [17] argue that, “A weakness of the evolutionary, or local minimum, approach is the sensitivity to initialization and difficulty in determining suitable parameters. This can be exaggerated by noise.” They then advocate techniques based on global energy minimisation rather than local minimisation.

Techniques to find optimal global minimum energy solutions may be preferable for fully automated image segmentation applications because they will usually lead to a unique answer for a given search space. Moreover there is no need to specify initialisation and starting criteria for the search. Note that the global minimum is not always the best solution for a given segmentation problem but, in our experience, it can work surprisingly well if the search space is well chosen.

Henceforth we will concentrate on methods for finding globally optimal solutions to the energy minimisation problem and then apply this to the optimal image segmentation problem. In the next chapter we will relate the development of this theme over a number of years through case studies from several research projects.

2

Globally Optimal Energy Minimisation Techniques

2.1 Introduction and Time-Line

In 1992 we began a research project to automatically segment cell images from Pap Smear slides for the detection of cancer of the cervix. We investigated simple low level techniques based on edge detection, grayscale thresholding, and grayscale morphology (*e.g.,* watersheds), but could only achieve accurate segmentation on about 60% of cell images. In 1997 we started looking at dual active contour segmentation techniques as proposed by Gunn and Nixon [18], but this method suffered from poor robustness on our images. However Gunn [20] also suggested a fast globally optimal method based on converting the problem of finding the best circular contour into a linear trellis and then applying the Viterbi algorithm to determine the minimum energy path. This approach worked remarkably well as reported by Bamford and Lovell [21] in 1998 and yielded 99.5% correct segmentation on a cell database of nearly 20,000 cell images.

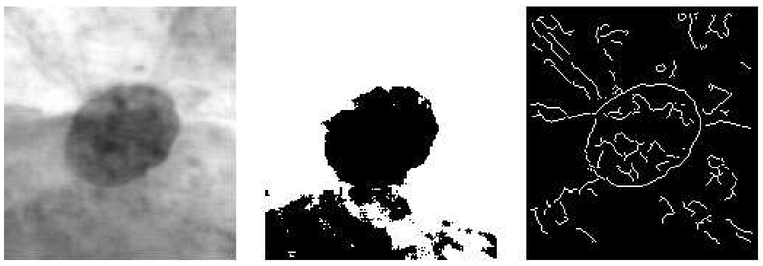
(a) (b) (c)

Fig. 2.1. Traditional bottom-up approach to cell image segmentation. (a) Original gray-level image, (b) thresholded image showing voids and artefacts, and (c) Canny edge map showing a partially complete border and other spurious edges (from [3]).

As this method was so remarkably effective on cell images there was little incentive to improve the method for the Pap Smear problem itself, but we still held a desire to develop more powerful global energy minimisation techniques which could be applied to a general class of objects. In particular the Viterbi algorithm based method would only work for objects that were convex and two-dimensional.

In 2002, Appleton and Sun [22] put the problem of minimising the energy of closed contours unwrapped onto linear trellis on to a firm mathematical basis. Then, in 2003, Appleton and Talbot [23, 24] extended and generalized the energy minimisation approach to handle the optimal segmentation of planar concave objects as well as convex images such as cells. This extension avoided dependence on a coarse discretization grid so that grid-bias could be removed. The extension to 3D was achieved in late 2003 by Appleton and Talbot [25] by converting the shortest path techniques into an equivalent continuous maximum flow/minimal surface problem.

In this chapter we briefly describe the various energy minimisation segmentation techniques and show how they can be applied to solve quite difficult segmentation and reconstruction problems in diverse domains from volumetric medical imaging to multiview reconstruction.

2.2 Cell Image Segmentation using Dynamic Programming

Although the use of active contours [1] is well established, it is well known that these methods tend to suffer from local minima, initialisation, and stopping criteria problems [26]. Fortunately global minimum energy, or equivalently shortest-path, searching methods have been found which are particularly effective in avoiding such local minima problems due to the presence of the many artefacts often associated with medical images [27, 7].

The energy minimization method employed was based on a suggestion in Gunn's dissertation [20]. A circular search space is first defined within the image, bounded by two concentric circles centralised upon the approximate centre of the nucleus found by an initial rough segmentation technique (*e.g.,* converging squares algorithm). This search space is sampled to form a circular trellis by discretising both the circles and a grid of evenly-spaced radial lines joining them (figure 2.2). This circular trellis is then unwrapped in a polar to rectangular transformation yielding a conventional linear trellis.

Every possible contour that lies upon the nodes of the search space is then evaluated and an associated energy or cost function is calculated. As with the snake energy formulation of (1.3), this cost is a function of both the contour's smoothness and how closely it follows image edges. The energy [21] is defined by:

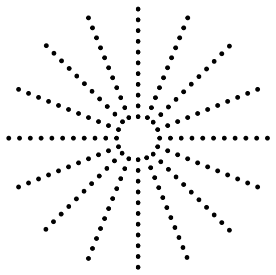

Fig. 2.2. Discrete search space

$$E_{snake} = \int_0^1 E_{int}(\nu(s)) + E_{image}(\nu(s)). \tag{2.1}$$

Using the discrete notation from chapter 1, we have

$$E_{int} = \left(\frac{\nu_{i+1} - 2\nu_i + \nu_{i-1}}{\nu_{i+1} - \nu_{i-1}} \right) \tag{2.2}$$

and

$$E_{image} = -|\nabla I(x,y)|^2. \tag{2.3}$$

The internal energy consists of a thin-plate term only. The relative weighting of the cost components is controlled by a single regularization parameter, $\lambda \in [0,1]$. By choosing a high value of λ, the thin-plate or stiffness term dominates, which may lead to smooth contours that tend to ignore important image edges. On the other hand, low values of λ allow contours to develop sharp corners as they attempt to follow all high gradient edges, even those which may not necessarily be on the desired object's boundary. Once every contour has been evaluated, the single contour with least cost is the global solution. The Viterbi algorithm provides a very efficient method to find this global solution as described in section 1.5.

A data set of 19946 Pap stained cervical cell images was available for testing. The single regularisation parameter λ was empirically chosen to be 0.7 after trial runs on a small sub-set of the images. The effect of the choice of λ on segmentation accuracy on this trial set is shown by the graph of figure 2.3. This figure shows a value of $\lambda = 0.7$ as being the most suitable for these particular images. It further shows that acceptable segmentation performance can be obtained with λ ranging from 0.1 to 0.9 — an enormous range which demonstrates the robustness and suitability of the approach. Every image in the data set was then segmented at $\lambda = 0.7$ and the results verified by eye. Of the 19946 images, 99.5% were found to be correctly segmented.

With λ set at 0.0, the smoothness constraint from the thin-plate term is completely ignored and the point of greatest gradient is chosen along each of the search space radii. Previous studies [28] have shown that for approximately

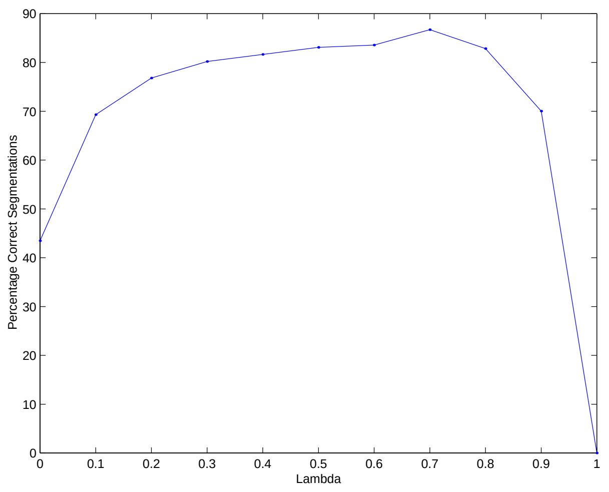

Fig. 2.3. Plot of percentage of correct segmentations against λ for a set of images consisting of known 'difficult' images and randomly selected images.

65% of images, all points of greatest gradient actually lie upon the nucleus-cytoplasm border, so these "easy" cell images will be correctly segmented with $\lambda = 0$. For the remaining 35% of images, a large gradient due to an artefact or darkly stained chromatin will draw the contour away from the desired border. As λ increases, the large curvatures necessary to admit these incorrect configurations become less probable as shown in figure 2.4.

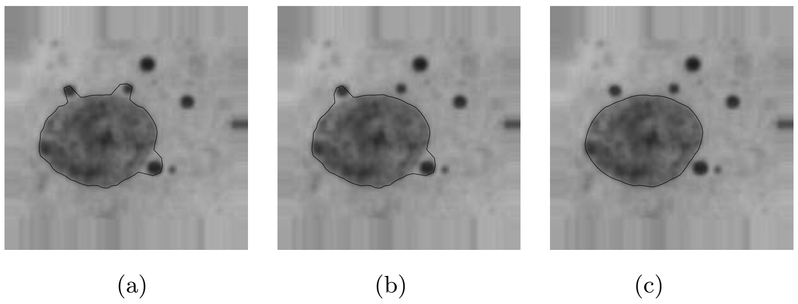

Fig. 2.4. The effect of increasing λ. (a) $\lambda = 0.1$, (b) $\lambda = 0.2$, and (c) $\lambda = 0.5$.

Comments on the Dynamic Programming Method

We show in [29] that the above segmentation method can be viewed as the application of hidden Markov model techniques [10] where the transition matrix is determined by the curvature constraints and the observation matrix is determined by the gradient image.

A simple way to find shortest paths on the linear trellis that corresponds to the closed contours in the image domain is to replicate the m nodes. Specifically we unwrap the circular domain such that the last column of nodes in the trellis is a copy of the first column. Then if there are m such nodes in a column, we would need to evaluate each of the m paths starting and finishing on the same node $i \in [0 \ldots M-1]$. This would require m evaluations of the Viterbi algorithm as described in chapter section 1.5. So we use the two pass method of Gunn and Nixon [17] based on the midpoint heuristic to find the minimum energy.

Although this heuristic works very well in practice, in theory there are clearly situations where it could fail to find the optimal solution.

Circular Shortest Path Algorithm (CSP)

Appleton and Sun [22] investigated this general problem of optimal circular shortest paths to address the theoretical deficiencies of the mid-point heuristic. Their circular shortest path algorithm is guaranteed to find the shortest circular path and uses a branch and bound technique [30, 31] to efficiently locate it.

The need for circular shortest paths arises when the search space is naturally periodic. Here we must satisfy the constraint that the end points of the shortest path must be connected in the periodic extension of the trellis. This constraint creates a cyclic dependency in the computation of the path cost which prevents us from applying the standard shortest path algorithms of section 1.5 based on dynamic programming. However this dependency is quite simply overcome by periodically extending the trellis as follows.

We perform a rectangular to polar mapping to convert the circular search space into a linear one as before. However, in this case, the column at the cut point is replicated as the last column to provide a periodic extension. In other words, a circular search space with N nodes is represented by a linear trellis of length $N+1$ where the last column is a replica of the first. Then a contour is circular if and only if the row index of the first node ν_0 is the same as the row index of the last node ν_N.

The root of the branch and bound search tree consists of the entire first column of nodes. This set of nodes is recursively split in two to form the binary search tree as depicted in figure 2.5. The Viterbi algorithm allows us to treat a set of nodes in the first column of the trellis as the source rather than using a single node as the source as described in section 1.5. The circular shortest path algorithm progresses as follows.

The shortest path to the other end of the trellis is found from the root node (*i.e.,* the entire first column) and this forms a lower bound on the cost of the circular shortest path. We find the destination node corresponding to the shortest path and follow the backward pointers to determine the corresponding

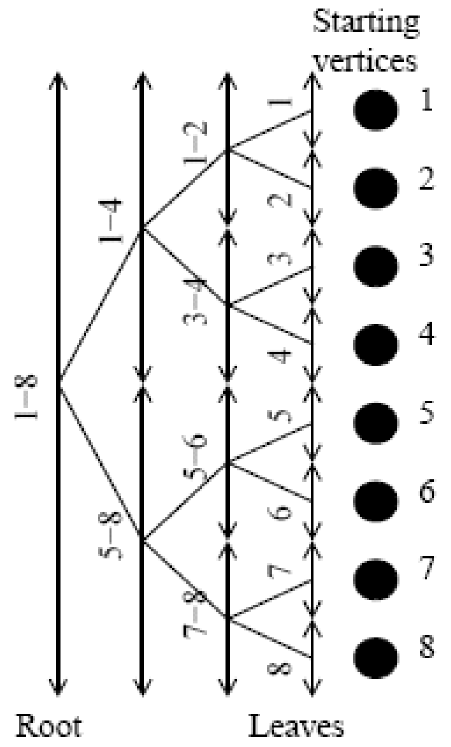

Fig. 2.5. The binary search tree for m=8. Only the first column of the trellis containing the potential source vertices is shown (from [32]).

source node. If the source node ν_0 is the same as the destination node ν_N, then we have found the circular shortest path and the algorithm terminates.

Otherwise the source node is split in two and the Viterbi algorithm is then run on the upper and lower subproblems. Since we know that a circular shortest path must start and finish on the same node, a new bound on the circular shortest path is obtained by examining the shortest path length to the corresponding upper and lower half of the destination nodes. For example, if $m = 8$ say, we would look for shortest paths between source and destination nodes with row indices 0–3 and 4–7 respectively. As before, if the shortest path found is circular, the algorithm terminates.

Otherwise we recursively split the node with the lowest circular shortest path bound and continue the search. The complete algorithm is given in [32] and an example segmentation of a diatom is shown in figure 2.6. On typical images the CSP algortihm often identifies the optimum circular shortest path with just one run of the Viterbi algorithm, although some pathological examples may take considerably longer to compute.

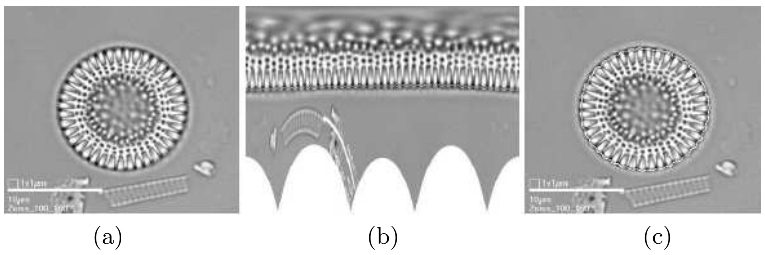

Fig. 2.6. Segmentation of *Cyclostephanos Dubius* by circular shortest path method. (a) The original microscope image of the diatom, (b) The polar unwrapping with circular shortest path overlaid, and (c) The corresponding segmentation contour (from [32]).

However despite this improvement a major shortcoming of all methods based on a polar to rectangular mapping is the inability to handle concave contours and higher dimensional objects, thus severely limiting their application domain.

2.3 Globally Optimal Geodesic Active Contours (GOGAC)

The classic active contour or snake model proposed by Kass [1] modelled a segmentation boundary by a series of point masses connected by springs. This explicit view of curves as a polygon was replaced by an implicit view of curves as the level set of some 3D surface by Osher and Sethian [33]. Level sets offer significant advantages over traditional snakes including improved stability and much better handling of topology (*e.g.,* segmentation of multiple objects with just one contour). Another advance came in the form of geodesic active contours as proposed by Caselles *et al* [34]. They demonstrated the equivalence of their energy function to the length of a geodesic (*i.e.,* path of least cost, path of least time) in an isotropic space. A problem with traditional geodesic active contours is that they are a gradient descent method and thus have all the usual problems of initialisation, termination, and local minima associated with such methods. They simply do not have the stability and simplicity of application of globally optimal segmentation methods.

The globally optimal GOGAC method we outline here finds closed contours in the image domain itself rather than unwrapping the image through polar to rectangular transformation. Working in the image domain means that we cannot find simple shortest paths as this would cause a bias towards small contours that wrap tightly around the origin. Instead we use a contour energy of the form [32]

$$E[C] = \oint_C \frac{g}{r}\, ds \tag{2.4}$$

where g is a measure of probability of being on the boundary (*e.g.,* image gradient) and r is the radius of the contour C. Thus all circles centred on the origin would have the same contour energy.

Now we cut the image plane with an arbitrary cut line as depicted in figure 2.7. Let us now consider a point on the cut line, p_{cut}, which is mapped to two equivalent points p_{start} and p_{end} in the cut plane S'. Now to find the shortest circular path beginning and ending at p_{cut}, we just solve the equivalent problem of finding the shortest path from p_{start} to p_{end} in the cut plane S' (using the Fast Marching algorithm described in the next section).

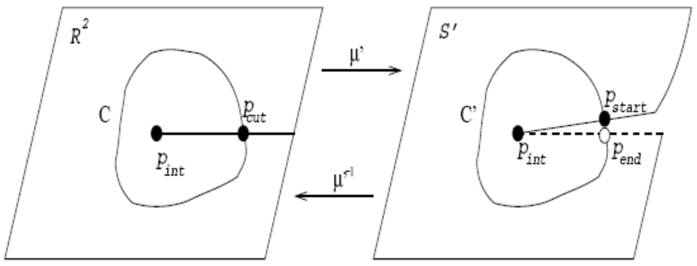

Fig. 2.7. A minimal closed geodesic in the image plane passing through p_{cut} and the corresponding open shortest path (*i.e.,* geodesic) in the cut plane between p_{start} and p_{end}. (from [32]).

A problem with the above approach is that it would not allow the shortest path to cross the cut line. This would once again restrict the algorithm to convex shapes only. However Appleton [32] shows that if we represent the open search space in an augmented helicoidal representation, it allows us to represent concave contours that cross the cut line (*i.e.,* unwrapping line) multiple times as illustrated in figure 2.8. Thus we can now find the shortest closed path passing through p_{cut} even if the contour wraps around the origin several times. Thus the arbitrary choice of the cut line does not influence nor restrict the range of image topology the GOGAC algorithm can handle.

In the above development it was assumed that p_{cut}, the intersection of the shortest path and the cut line, is known in advance. This is not the case, and an exhaustive search of all possible values of p_{cut} would be quite inefficient. Instead, we use the branch and bound approach of the CSP algorithm in section 2.2. Referring to figure 2.7, we use the set of all possible points for

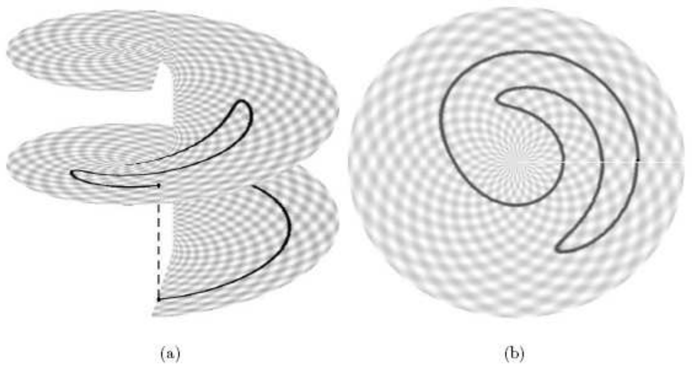

Fig. 2.8. The helicoidal representation of the cut-concave shape . (a) open curve (b) closed curve (from [32]).

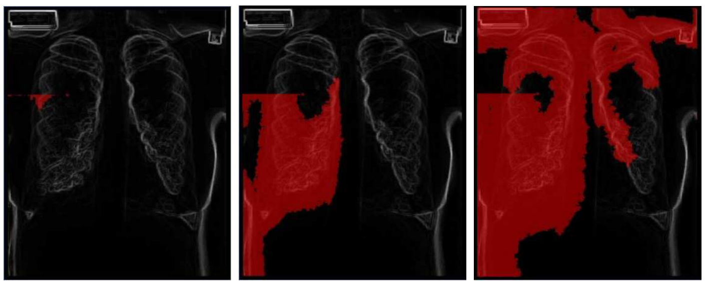

Fig. 2.9. Application of GOGAC to lung Xray image segmentation. Evolution of the fast marching wavefront from the cut line. (images provided by Ben Appleton).

p_{cut} along the cut line as the source and the periodic replica of these points in the cutplane S' as the destination and then proceed with the binary search in the same manner as the CSP algorithm. Figure 2.9 shows the evolution of the fast marching wavefront emanating from the cut line as it segments a lung using GOGAC. Figures 2.10 and 2.11 show other segmentation results from the GOGAC method.

2.3.1 Fast Marching Algorithm

We use the Fast Marching Algorithm [12] to find the surfaces of minimal action whose gradient curves form shortest paths, also known as geodesics. A geodesic is a generalization of the concept of a "straight line" to "curved spaces" (*i.e.,* Riemannian spaces) such as the surface of the earth. In the case of a sphere such as the Earth, a geodesic is a great circle. With respect to a given metric, geodesics are defined to be the shortest path between points on the space. A shortest path between two points in a curved space can be

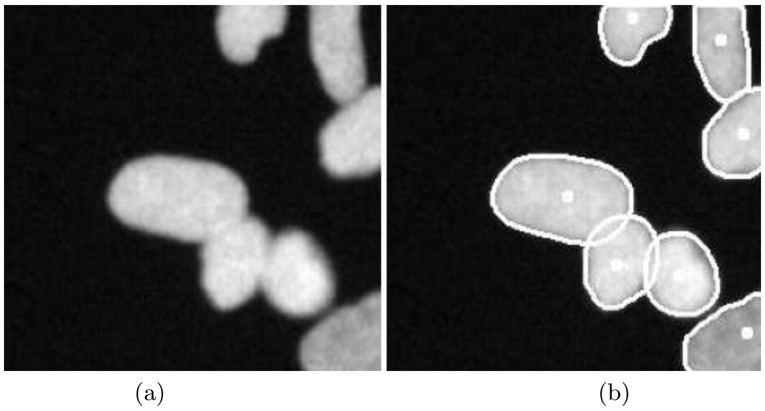

Fig. 2.10. Globally optimal geodesic active contours applied to overlapping objects. The cells (a) are separated despite the weak intensity gradient between them (b) (from [32]).

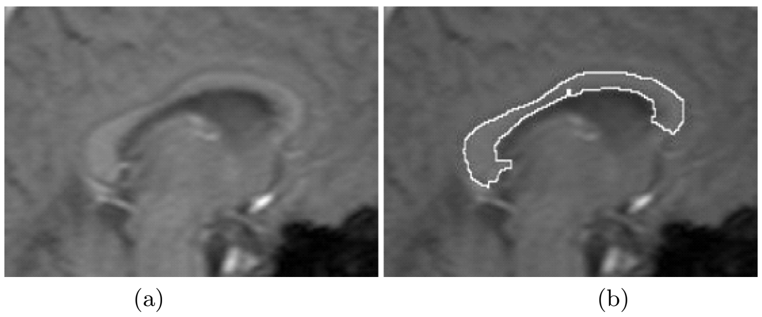

Fig. 2.11. Segmentation of MRI image of a concave contour, the corpus callosum in a human brain, from [32]. Image (a) is the original and (b) is the segmentation via GOGAC.

found by writing the equation for the length of a curve, and then minimizing the length using techniques from calculus of variations. An entirely equivalent approach is to define the energy of a curve; then minimizing the energy leads to the same equations for a geodesic. This latter formulation can better be understood when we consider how an elastic band stretched between two points will contract in length to minimize its energy — the final shape of the band is a geodesic. Thus there is an intimate relationship between the mathematical formalism of geodesics and the concepts underpinning snakes as proposed by Kass *et al.*

The globally minimal geodesic between two sets of points in an isotropic Riemannian space can be calculated with the fast marching method [35]. This method computes the surface of minimal action, also known as a distance

```
Fast Marching Algorithm

Initialisation:

For all grid points $x$ in $P_0$:
  • Set $U(x) = 0$
  • Label $x$ as Trial and insert into $Q$
For all other grid points $x$:
  • Set $U(x) = \infty$
  • Label x as Far
Main loop:
  • If Q is empty, halt
  • Otherwise remove the Trial point of minimum value from the priority queue:
        $x = \underset{x'}{\operatorname{argmin}} \{U(x')|x' \in Q\}$
  • Label x as Known
  • For each neighbour $n$ of $x$ in the grid:
      – If $n$ is Known, continue to next neighbour
      – If $n$ is Far, change label to Trial and insert into $Q$
      – Update $U(n)$ by solving (2.5). Only use the values at neighbouring grid points
        which are labelled Known.
```

Fig. 2.12. Pseudocode for Fast Marching Algorithm to find the surfaces of minimal action and geodesics (*i.e.,* shortest paths)(from [32]).

function, from the starting set P_0 to all points in the space. It finds the surface of minimal action by considering it as the first time-of-arrival of a wavefront emanating from the starting set P_0 and travelling with speed $\frac{1}{g}$, where g is usually the image gradient as before. This wavefront sweeps the grid beginning with the starting set P_0 and proceeds in order of arrival time U.

The algorithm is identical to Dijkstra's shortest distance algorithm [8] from section 1.5 apart from the need to update g. On a rectangular two-dimensional grid in with grid step h we may use the discrete gradient operator defined by

$$\begin{aligned} g^2(i,j) &= \frac{1}{h^2}\max\{U[i,j]-U[i-1,j],U[i,j]-U[i+1,j],0\}^2 \\ &+ \frac{1}{h^2}\max\{U[i,j]-U[i,j-1],U[i,j]-U[i,j+1],0\}^2. \end{aligned} \quad (2.5)$$

During the course of the algorithm, points which have not been considered yet are labelled *Far.* Points which have been assigned a temporary value for U are labelled *Trial.* Points for which U has been finalised are labelled *Known.*

The algorithm makes use of a priority queue[1] Q of Trial points in order to maintain efficient access to the minimum distant point. Pseudocode for the complete algorithm is provided in figure 2.12.

Figure 2.13 shows that the fast marching algorithm calculates distance functions that behave similarly to propagating electromagnetic radiation. Indeed this example shows that the shortest path calculated via the fast marching algorithm follows Snell's Law of Refraction from the field of optics. This formula relates the angles of incidence and refraction where a ray of light crosses a boundary between different media, such as air and glass. Snell's law can be derived from Fermat's principle of least time, which states that the path taken between two points by a ray of light is the path that can be traversed in the least time - that is, a geodesic. Figure 2.14 shows a similar computation on an inverse velocity cost function.

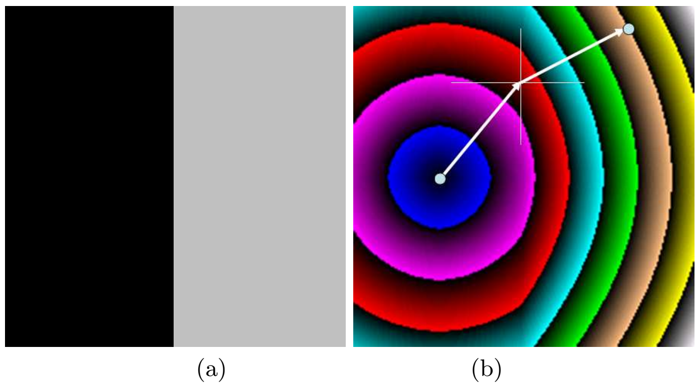

Fig. 2.13. The fast marching algorithm calculates the surfaces of minimal action (b) for the two-valued cost function of (a). Note that the geodesic shown in (b) follows Snells' Law of refraction for optical and electromagnetic waves — that is, the sine of the angle of incidence divided by the sine of the angle of refraction is a constant determined by the properties of the two media at the interface (images provided courtesy of Ben Appleton).

2.4 Globally Minimal Surfaces (GMS)

The planar segmentation technique outlined in the last section cannot be extended to higher dimensions, so we need an entirely new approach. Minimum cuts and maximum flow techniques are naturally suited to globally optimal

[1] the implemention employs a heap data structure and heapsort for efficient location of the minimum.

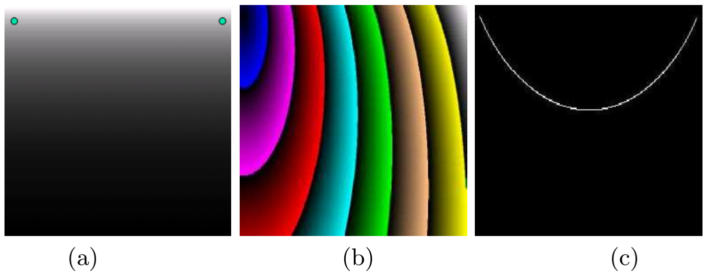

Fig. 2.14. The fast marching algorithm is used to compute the path of a ball rolling on an inclined plane under the influence of gravity. The path of the ball will always be a geodesic. (a) Inverse velocity metric, (b) arrival time (surfaces of minimal action), and (c) geodesic (shortest path).

segmentation in higher dimensions. Although this has been tried in the past with discrete approximations, Appleton and Talbot [25] proposed a method based on continuous maximal flows by solving a system of partial differential equations. It is shown [32] that this method gives identical results to the previous GOGAC method in the case of planar images.

2.4.1 Minimum Cuts and Maximum Flows

Minimum cuts are another concept from graph theory that are related to shortest paths, although the computation is often slower and more complicated. Graph cuts may be used to determine the capacity of a communications network or to determine the minimum number of links that must fail before a network becomes disconnected — an important measure of the reliability of a network. In image analysis they have been proposed for optimally partitioning an image or volume into two regions. For example, this technique could be used to determine the most likely shape of a 3D object in an ultrasound or Magnetic Resonance Imaging (MRI) image.

Consider a finite directed graph G where every edge (u, v) has a capacity of $c(u, v)$ which is a non-negative real number. We identify two vertices, known as the source s and the sink t. A cut is a partition of the nodes into two sets S and T, such that $s \in S$ and $t \in T$. The capacity of a cut (S,T) is

$$c(S, T) = \sum_{u \in S, v \in T | (u,v) \text{ is an edge}} c(u, v),$$

which is just the sum of the capacity of all edges crossing the cut from region S to T.

The *max-flow min-cut theorem* [36, 37] indexmax-flow min-cut theorem states that, the maximal amount of flow in a network is equal to the capacity of a minimal cut. In other words, the theorem states that the maximum flow in a network is dictated by its bottleneck — the minimum cut surface. It turns out that the maximum flow problem is convex and is consequently easier to solve than the dual problem of finding the minimum cut.

Augmenting Path Algorithm

The best known algorithm for solving the maximum flow problem is the famous Ford-Fulkerson [37] augmenting path algorithm. This algorithm successively increases the maximum flow from source s to sink t by continually locating paths along which more flow may be pushed. Once all paths from source to sink are saturated, the flow is maximal. The pseudocode for the Ford-Fulkerson algorithm is given in figure 2.15.

Ford Fulkerson Augmenting Path Algorithm

Initialisation:

```
Set $F = 0$ on each edge
```

Main loop:

```
- Search for an s-t path along which more flow may be pushed
- If no such path exists, halt
- Otherwise, increase the flow uniformly along this path until at least one edge
    becomes saturated
```

Fig. 2.15. Pseudocode for Ford-Fulkerson Augmenting Path algorithm (from [32]).

Pre-Flow Push Algorithm

An alternative to the augmenting path algorithm is the more recent pre-flow push algorithm of Goldberg and Tarjan [38]. One advantage of this formulation is that it is highly parallelisable compared to the Ford-Fulkerson algorithm. A preflow is like a flow, except that the total amount flowing into a vertex is allowed to exceed the total amount flowing out. The algorithm maintains a preflow in the original network and then pushes excess local flow toward the sink along what are estimated to be shortest paths.

A vertex which has greater inward flow than outward flow is called an *active* vertex — the excess being the positive difference between the two. The algorithm repeatedly pushes flow outwards from active vertices toward the sink. A height function H is introduced on the vertices to guide the flow along the shortest unsaturated path toward the sink. The source and sink have fixed heights of $|V|$ and 0 respectively and may never become active. Active vertices are stored in a queue, Q. The pseudocode for the Goldberg-Tarjan algorithm is given in figure 2.16.

Goldberg-Tarjan Pre-Flow Push Algorithm

```
Initialisation:
• Set $F = 0$ on each edge
• Set $H$ to be the length of the shortest (unweighted) path to the sink $t$, and
  $H(s) = |V|$
• Set the source $s$ as active and place it in the $Q$
Main loop:
• If $Q$ is empty, halt
• Otherwise, retrieve an active vertex $v$ from $Q$
• For all neighbouring vertices u of v:
  – If the edge $(v, u)$ is unsaturated and $H(v) = H(u) + 1$, push more flow along
    edge $(v, u)$ until it is saturated or $v$ has excess 0
  – If this increased the flow to $u$, set $u$ as active and place in $Q$. Note: $u$ may
    already be active
  – If $v$ still has positive excess, increment $H(v)$ and place $v$ in $Q$
• Otherwise, set v as inactive
```

Fig. 2.16. Pseudocode for Goldberg-Tarjan Pre-Flow Push Algorithm (from [32]).

2.4.2 Development of the GMS Algorithm

It is well-known that maximum flow techniques work well in a discrete domain of a network, but the imposition of a coarse discretisation grid on a natural image leads to quite unnatural grid-biases in the segmentation. The segmentaiton contours tend to follow the artificially imposed discretisation grid rather than the following smooth curves in image itself leading to unacceptable staircase artefacts. The goal here is to develop an algorithm that works directly in the continuous image domain.

It is not at all clear how the augmenting path algorithm can be extended from the continuous to the discrete domain. On the other hand, the pre-flow push method is much better suited to the problem. One advantage is that the updates on vertices require only local information from the neighbours rather than global knowledge of the image. This suggests a method based on solving a system of partial differential equations — indeed in much the same way, the solution of Maxwell's equations leads to the solutions for electromagnetic fields and travelling electromagnetic waves such as light.

We relax the flow conservation constraint at each vertex by adding an additional variable at each point. This results in a scalar potential field, P, which will keep track of the inflow-outflow imbalance (*i.e.,* divergence) in the (compressible) flow and provides a restoring force to drive this imbalance to zero at convergence.

One way to visualize the potential function is to think of a network of water pipes connected to a underground junction. When water initially surges down the pipes and meets at the junction, enormous pressures are generated which

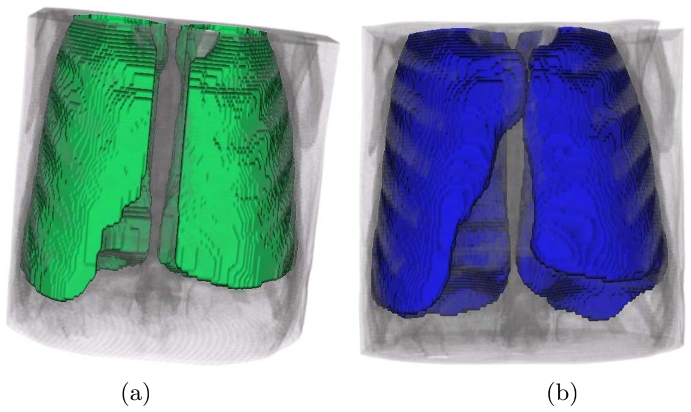

Fig. 2.17. Comparison of 3D lung MRI image segmentation using (a) discrete min-cut, and (b) continuous GMS. Note the unnatural staircase effect in the segmentation of the lower left lung due to grid bias. Computation time was 2 minutes for min-cut and 30 seconds for GMS using a 1Ghz Pentium$^{©}$ computer. (images provided by Ben Appleton).

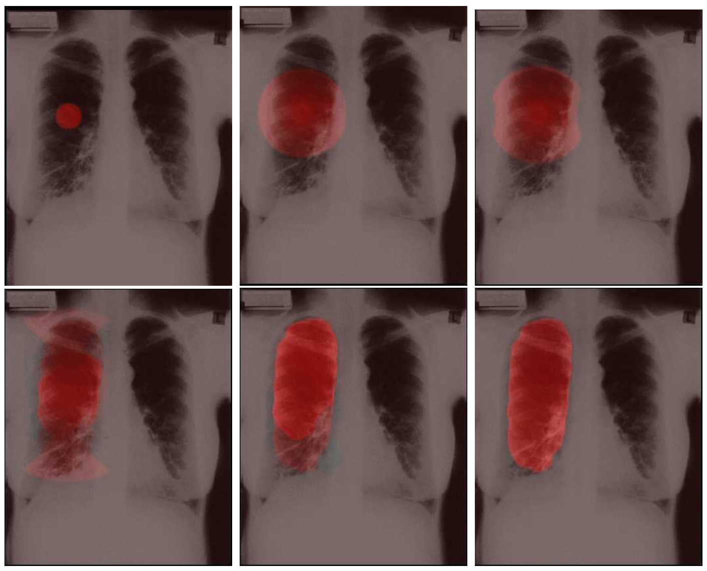

Fig. 2.18. Application of GMS to 2D lung Xray image segmentation. Evolution of the potential function used to find the global minimal surface (images provided by Ben Appleton).

could split the pipes unless the junction box is vented. The ancient Romans knew of this problem and their solution was to relieve the pressures in their underground aquaducts with a series of vertical vents and fountains. Vents allows the excess water to rise up the vent pipe providing a restoring force to balance the flow. Thus the water level in the vent pipe is equivalent to the potential function in the GMS algorithm.

Now consider the following system of differential equations.

$$\frac{\partial P}{\partial t} = -\mathrm{div}\vec{F}, \tag{2.6}$$

$$\frac{\partial \vec{F}}{\partial t} = -\nabla P, \tag{2.7}$$

$$\|\vec{F}\|_2 \leq g. \tag{2.8}$$

These first two equations taken together form a simple system of wave equations. They may be interpreted as a linear model of the dynamics of an idealised fluid with pressure P and velocity $\vec{F}$. Without loss of generality and to maintain symmetry between source and sink, we fix the scalar potential field P at the source s and sink t such that $P_s = 1$ and $P_t = -1$.

It can be shown [32] that at convergence the potential field is an isosurface of value +1 in the region connected to the source and -1 in the region connected to the sink. Thus the potential field becomes an indicator function which tells us whether we are inside or outside the minimal surface. Without loss of generality, we choose level set 0 as the minimal surface.

2.4.3 Applications of the GMS Algorithm

The evolution of the potential function to determine the minimal surface corresponding to a human lung is shown in figure 2.18. Note how the potential function evolves to an indicator function separating the interior region of the lung from the exterior. Figure 2.19 shows the segmentation of volumetric MRI data to segment the hippocampus.

A loss obvious application is the use of GMS is to find the optimal 3D reconstruction from multiview images. Now the use of a stereo pair of images to determine ground elevation from image disparity is a well-known technique from aerial photogrammetry. Unfortunately, stereo image pair photogrammetry can only provides so-called 2 1/2 D rather than true 3D reconstruction — with just two frontal images it is impossible to reconstruct the back of an object. So true 3D model reconstruction requires multiple images — hence the term multiview reconstruction.

Leung [40] developed a technique called Embedded Voxel Colouring (EVC) which employed space carving and photoconsistency contraints to the 3D reconstruction problem. He determines the 3D surface which optimally satified

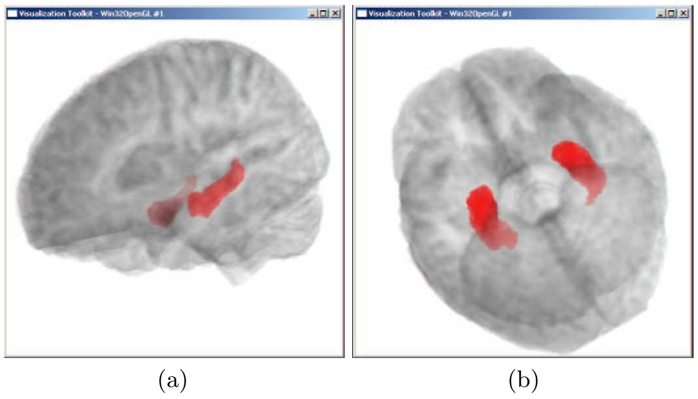

Fig. 2.19. Segmentation of the hippocampi from an MRI dataset using GMS. Image (a) is the view from the side and (b) is the view from below from [32].

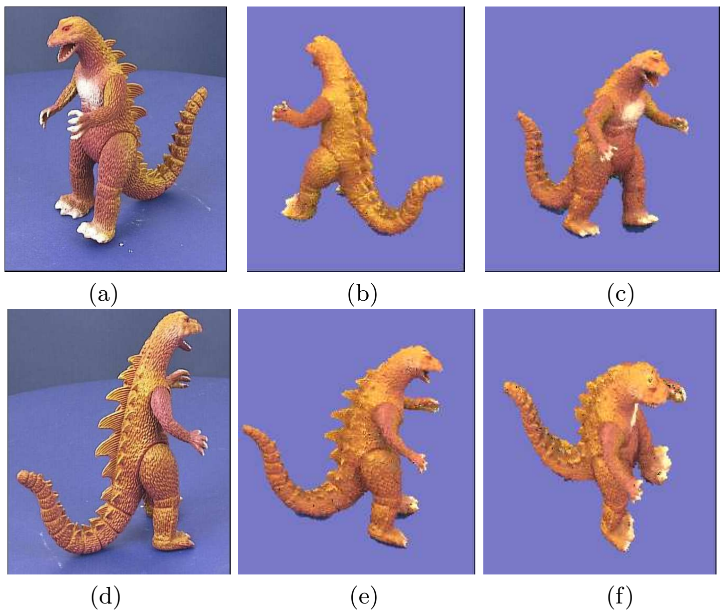

Fig. 2.20. Reconstruction of the dinosaur image sequence using Embedded Voxel Colouring (EVC) and adaptive thresholding via GMS. Images (a) and (d) are selected images from the dinosaur image set. Images (b), (c), (e), and (f) are new views generated from the 3D reconstruction (from [39]).

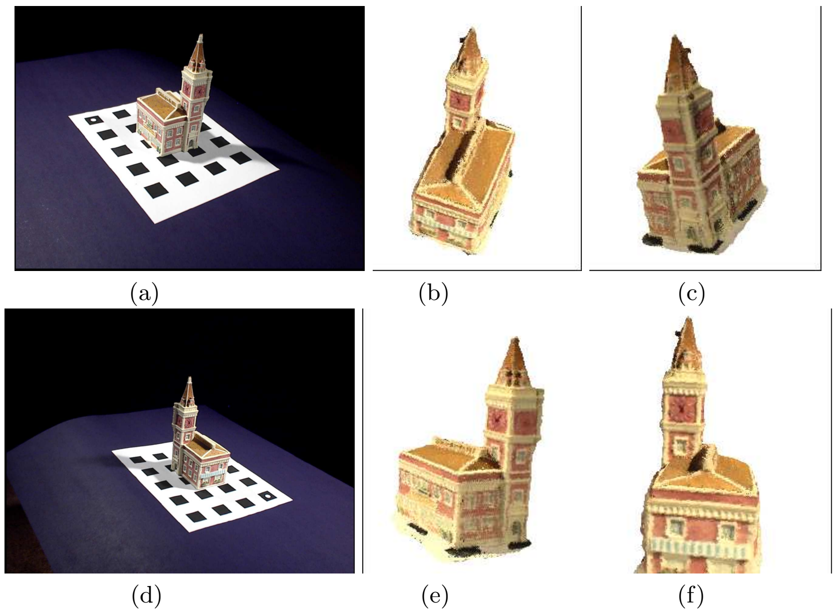

Fig. 2.21. Reconstruction of the Ghirardelli image sequence using Embedded Voxel Colouring (EVC) and adaptive thresholding via GMS. Images (a) and (d) are selected images from the Ghirardelli image set. Images (b), (c), (e), and (f) are new views generated from the 3D reconstruction (from [40, 39]).

all the reconstruction constraints using the GMS algorithm. Figure 2.21 shows a 3D reconstruction from multiview images using GMS as a postprocessor.

One advantage of the GMS algorithm for extraordinarily difficult segmentation tasks, such as extracting the hippocampus from MRI images, is the ability to define multiple sources and sinks to mark points which are definitely interior and exterior to the object undergoing segmentation as shown in figure 2.22. Franklin [41] used this approach to guide the GMS algorithm so that the hippocampi of sets of human brains could be labelled fully automatically as shown in figure 2.23. This study has now been completed on a small set of brains yielding quite good results. It will be extended to a much larger set in the near future.

This latter work is important because there is evidence that changes in the shape of the hippocampi may be an early indicator of the onset of Alzheimer's disease (also known as dementia). The economic and social cost of Alzheimer's disease is growing rapidly due to the aging population in the western world. Indeed Access Economics estimates that the cost of dementia to the Australian economy alone in 2004 was approximately USD 4 billion [42]. At present the detection of Alzheimer's disease is largely performed through psychological tests that detect loss of cognitive ability once the brain is damaged. What

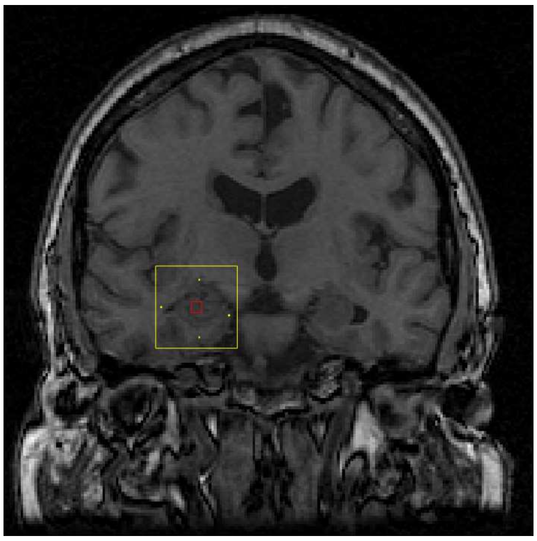

Fig. 2.22. The usage of multiple sources and sinks to control the evolution of the GMS algorithm for the fully automated segmentation of the hippocampus in the human brain (from [41]).

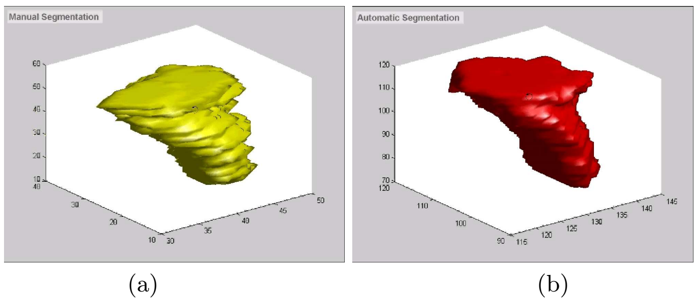

Fig. 2.23. Comparison of manual and automatic segmentation of the hippocampus in the human brain. Image (a) is a manual segmentation by a clinician which required about 2 hours of labelling and (b) is a fully-automated segmentation via GMS using multiple sources and sinks positioned by cross-validated training on labelled images which required just 2 minutes of computation (from [41]).

is needed is a fast, cheap, and reliable method to extract the shape of the hippocampi from brain MRI which could be used as a screening test for early Alzheimer's disease. Such a test may allow health workers to intervene before serious brain damage occurs.

2.5 Conclusions

These globally optimal energy minimisation methods are fast, easy to apply, and tend to yield robust solutions. When using conventional active contours based on local energy minimisation, a great deal of effort is expended in developing techniques for choosing the initial position of the contour, escaping local minima, and determining stopping criteria. It is certainly true that some effort must be expended on determining the search space and the energy function when using global energy minimisation techniques. Yet, in our experience, these techniques are much simpler to apply in practice and yield more robust and accurate results. Note further that by carefully positioning the search space, global energy minimisation techniques can always find locally minimal energy solutions. In particular, for the globally minimal surface approach multiple sources and sinks can be used to guide the solution providing many of the purported advantages of the original snakes of Kass *et al.* The converse however is not true — local energy minimisation techniques are never guaranteed to find global solutions.

Future work is focussed on integrating these techniques with statistical shape models to develop an 3D Expectation Maximization algorithm incorporating prior shape knowledge for detection and segmentation of known shapes.

2.6 Acknowledgements

This chapter outlines a number of research themes currently being pursued within the Intelligent Real-Time Imaging and Sensing Group and National ICT Australia. I would like to acknowledge the contributions of Terry Caelli, Hugues Talbot, Peter Kootsookos and my current and former students Pascal Bamford, Ben Appleton, Carlos Leung, David McKinnon, Christian Walder, Stephen Franklin, and Daniel Walford. We would also like to acknowledge the ANU Centre for Mental Health for providing the labelled brain images.

References

1. Kass, M., Witten, A., Terzopoulos, D.: Snakes: Active contour models. In: Proc. International Conf. on Computer Vision, IEEE (1987) 259–268
2. Kass, M., Witten, A., Terzopoulos, D.: Snakes: Active contour models. International Journal of Computer Vision **1** (1988) 321–331
3. Bamford, P.: Segmentation of Cell Images with Application to Cervical Cancer Screening. PhD thesis, The University of Queensland (1999)
4. Amini, A.A., Weymouth, T.E., Jain, R.C.: Using dynamic programming for solving variational problems in vision. IEEE Transactions on Pattern Analysis and Machine Intelligence **12** (1990) 855–867
5. Cohen, L.: On active contour models and balloons. Computer Vision, Graphics and Image Processing: Image Understanding **53** (1991) 211–218
6. Viterbi, A.J.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory **13** (1967) 260–269
7. Geiger, D., Gupta, A., Costa, A., Vlontzos, J.: Dynamical programing for detecting, tracking, and matching deformable contours. Pattern Analysis and Machine Intelligence **17** (1995) 294–302
8. Dijkstra, E.: A note on two problems in connexion with graphs. Numerische Mathematik **1** (1959) 269–271
9. Blahut, R.E.: Fast Algorithms for Digital Signal Processing. Addison-Wesley (1987)
10. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE **77** (1989) 257–286
11. Bellman, R.E.: Dynamic Programming. Princeton University Press (1957)
12. Sethian, J.A.: A fast marching level set method for monotonically advancing fronts. Proceedings of the National Academy of Sciences **93** (1996) 1591–1595
13. Sethian, J.A.: Level Set Methods and Fast Marching Methods — Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science. Cambridge University Press (1999)
14. Ji, L., Yan, H.: Loop-free snakes for image segmentation. In: International Conference on Image Processing. Volume 3. (1999) 193–197
15. Oliveira, A., Ribeiro, S., Esperanca, C., Giraldi, G.: Loop snakes: the generalized model. In: International Conference on Information Visualisation. (2005) 975 – 980

16. McInerney, T., Terzopoulos, D.: Topologically adaptable snakes. In: International Conference on Computer Vision, IEEE (1995) 840 – 845
17. Gunn, S.R., Nixon, M.: Snake head boundary extraction using global and local energy minimization. In: Proceedings of 13th International Conference on Pattern Recognition, IAPR, IEEE (1996) 25–29
18. Gunn, S.R., Nixon, M.S.: A robust snake implementation; a dual active contour. IEEE Transactions on Pattern Analysis and Machine Intelligence **19** (1997) 63–68
19. Walford, D.: Rock wall segmentation using spatial and image information fusion. Technical report, The University of Queensland (2006)
20. Gunn, S.R.: Dual Active Contour Models for Image Feature Extraction. University of Southampton (1996) PhD Thesis.
21. Bamford, P., Lovell, B.: Unsupervised cell nucleus segmentation with active contours. Signal Processing Special Issue: Deformable Models and Techniques for Image and Signal Processing **71** (1998) 203–213
22. Appleton, B., Sun, C.: Circular shortest paths by branch and bound. Pattern Recognition **36** (2003) 2513–2520
23. Appleton, B.: Optimal geodesic active contours: application to heart segmentation. In Lovell, B.C., Maeder, A.J., eds.: APRS Workshop on Digital Image Computing. Volume 1., Brisbane, APRS (2003) 27–32
24. Appleton, B., Talbot, H.: Globally optimal geodesic active contours. Journal of Mathematical Imaging and Vision (2005)
25. Appleton, B., Talbot, H.: Globally optimal surfaces by continuous maximal flows. In Sun, C., Talbot, H., Ourselin, S., Adriaansen, T., eds.: Digital Image Computing: Techniques and Applications. Volume 2., Sydney, CSIRO Publishing (2003) 987–996
26. Cohen, L.D., Cohen, I.: Finite-element methods for active contour models and balloons for 2-D and 3-D images. IEEE Transactions on Pattern Analysis and Machine Intelligence **15** (1993) 1131–1147
27. Davatzikos, C.A., Prince, J.L.: An active contour model for mapping the cortex. IEEE Transactions on Medical Imaging **14** (1995) 65–80
28. Bamford, P., Lovell, B.: Improving the robustness of cell nucleus segmentation. In Lewis, P.H., Nixon, M.S., eds.: Proceedings of the Ninth British Machine Vision Conference, BMVC '98, University of Southampton (1998) 518–524
29. Lovell, B.C.: Hidden markov models for spatio-temporal pattern recognition and image segmentation. In Mukherjee, D.P., Pal, S., eds.: International Conference on Advances in Pattern Recognition. Volume 1., Calcutta (2003) 60–65
30. Winston, P.H.: Artificial Intelliegence. Addison-Wesley Publishing Company Inc (1984)
31. Land, A., Doig, A.: An automatic method of solving discrete programming problems. Econometrica **28** (1960) 497–520
32. Appleton, B.C.: Globally Minimal Contours and Surfaces for Image Segmentation. The University of Queensland (2004)
33. Osher, S., Sethian, J.A.: Fronts propagating with curvature dependent speeed: Algorithms based on hamilton-jacobi formulations. Journal of Computational Physics **79** (1988) 12–49
34. Caselles, V., Kimmel, R., Sapiro, G.: Geodesic active contours. International Journal of Computer Vision **22** (1997) 61–79
35. Adalsteinsson, D., Sethian, J.A.: A fast level set method for propagating interfaces. Journal of Computational Physics **118** (1995) 269277

36. Elias, P., Feinstein, A., Shannon, C.E.: Note on maximum flow through a network. IRE Transactions on Information Theory **IT-2** (1956) 117–119
37. Ford, L.R., Fulkerson, D.R.: Flows in Networks. Princeton University Press (1962)
38. Goldberg, A.V., Targan, R.E.: A new approach to the maximum-flow problem. Journal of the ACM **35** (1988) 921 – 940
39. Leung, C.: Efficient Methods For 3D Reconstruction From Multiple Images. PhD thesis, The University of Queensland (2006)
40. Leung, C., Appleton, B., Lovell, B.C., Sun, C.: An energy minimisation approach to stereo-temporal dense reconstruction. In: International Conference on Pattern Recognition. Volume 1., Cambridge (2004) 72–75
41. Franklin, S.: Automatic segmentation of mri brain images. Master's thesis, The University of Queensland (2006)
42. Khachaturian, Z., Brodaty, H., Broe, T., Jorm, T., Masters, C., Nay, R., Haikerwal, M., Rees, G., Low, L.F.: Dementia research: A vision for australia. Technical report, Alzheimers Australia (2004)

Index