Seeded Segmentation Methods for Medical Image Analysis

Camille Couprie, Laurent Najman, Hugues Talbot

Chapter 3
Seeded Segmentation Methods for Medical Image Analysis

Camille Couprie, Laurent Najman, and Hugues Talbot

Segmentation is one of the key tools in medical image analysis. The objective of segmentation is to provide reliable, fast, and effective organ delineation. While traditionally, particularly in computer vision, segmentation is seen as an early vision tool used for subsequent recognition, in medical imaging the opposite is often true. Recognition can be performed interactively by clinicians or automatically using robust techniques, while the objective of segmentation is to precisely delineate contours and surfaces. This can lead to effective techniques known as "intelligent scissors" in 2D and their equivalent in 3D.

This chapter is divided as follows. Section 3.1 starts off with a more "philosophical" section setting the background for this study. We argue for a segmentation context where high-level knowledge, object information, and segmentation method are all separate.

In Sect. 3.2, we survey in some detail a number of segmentation methods that are well-suited to image analysis, in particular of medical images. We illustrate this, make some comparisons and some recommendations.

In Sect. 3.3, we introduce very recent methods that unify many popular discrete segmentation methods and we introduce a new technique. In Sect. 3.4, we give some remarks about recent advances in seeded, globally optimal active contour methods that are of interest for this study.

In Sect. 3.5, we compare all presented methods qualitatively. We then conclude and give some indications for future work.

H. Talbot (✉)
Université Paris-Est, Paris, France

G. Dougherty (ed.), *Medical Image Processing: Techniques and Applications*, Biological and Medical Physics, Biomedical Engineering, DOI 10.1007/978-1-4419-9779-1_3,

3.1 The Need for Seed-Driven Segmentation

Segmentation is a fundamental operation in computer vision and image analysis. It consists of identifying regions of interests in images that are semantically consistent. Practically, this may mean finding individual white blood cells amongst red blood cells; identifying tumors in lungs; computing the 4D hyper-surface of a beating heart, and so on.

Applications of segmentation methods are numerous. Being able to reliably and readily characterize organs and objects allows practitioners to measure them, count them and identify them. Many images analysis problems begin by a segmentation step, and so this step conditions the quality of the end results. Speed and ease of use are essential to clinical practice.

This has been known for quite some time, and so *numerous* segmentation methods have been proposed in the literature [57]. However, segmentation is a difficult problem. It usually requires high-level knowledge about the objects under study. In fact, semantically consistent, high-quality segmentation, in general, is a problem that is indistinguishable from strong Artificial Intelligence and has probably no exact or even generally agreeable solution. In medical imaging, experts often disagree amongst themselves on the placement of the 2D contours of normal organs, not to mention lesions. In 3D, obtaining expert opinion is typically difficult, and almost impossible if the object under study is thin, noisy and convoluted, such as in the case of vascular systems. At any rate, segmentation is, even for humans, a difficult, time-consuming and error-prone procedure.

3.1.1 Image Analysis and Computer Vision

Segmentation can be studied from many angles. In computer vision, the segmentation task is often seen as a low-level operation, which consists of separating an arbitrary scene into reasonably alike components (such as regions that are consistent in terms of color, texture and so on). The task of grouping such component into semantic objects is considered a different task altogether. In contrast, in image analysis, segmentation is a high-level task that embeds high-level knowledge about the object.

This methodological difference is due to the application field. In computer vision, the objective of segmentation (and grouping) is to recognize objects in an arbitrary scene, such as persons, walls, doors, sky, etc. This is obviously extremely difficult for a computer, because of the generality of the context, although humans do generally manage it quite well. In contrast, in image analysis, the task is often to *precisely* delineate some objects sought in a particular setting known in advance. It might be for instance to find the contours of lungs in an X-ray photograph.

The segmentation task in image analysis is still a difficult problem, but not to the same extent as in the general vision case. In contrast to the vision case, experts might agree that a lesion is present on a person's skin, but may disagree on its exact contours [45]. Here, the problem is that the boundary between normal skin and lesion might be objectively difficult to specify. In addition, sometimes there does exist an object with a definite physical contour (such as the inner volume of the left ventricle of the heart). However, imaging modalities may be corrupted by noise and partial volume effects to an extent that delineating the precise contours of this physical object in an image is also objectively difficult.

3.1.2 Objects Are Semantically Consistent

However, in spite of these difficulty, we may assume that, up to some level of ambiguity, an object (organ, lesion, etc) may still be specified somehow. This means that semantically, an object possess some consistency. When we point at a particular area on an image, we expect to be, again with some fuzziness, either inside or outside the object

This leads us to the realize that there must exist some mathematical indicator function, that denotes whether we are inside or outside of the object with high probability. This indicator function can be considered like a series of constraints, or labels. They are sometimes called *seeds* or *markers*, as they provide starting points for segmentation procedures, and they mark where objects are and are not.

In addition, a *metric* that expresses the consistency of the object is likely to exist. A gradient on this metric may therefore provide object contour information. Contours may be weak in places where there is some uncertainty, but we assume they are not weak everywhere (else we have an ambiguity problem, and our segmentation cannot be precise). The metric may simply be the image intensity or color, but it may express other information like consistency of texture for instance. Even though this metric may contain many descriptive elements (as a vector of descriptors for instance), we assume that we are still able to compute a gradient on this metric [61].

This is the reason why many segmentation methods focus on contours, which are essentially discontinuities in the metric. Those that focus on regions do so by defining and utilizing some consistency metric, which is the same problem expressed differently.

The next and final step for segmentation is the actual contour placement, which is equivalent to object delineation. This step can be considered as an optimization problem, and this is the step on which segmentation methods in the literature focus the most. We will say more about this in Sect. 3.2 listing some image segmentation categories.

3.1.3 A Separation of Powers

In summary, to achieve segmentation in the analysis framework, we need three ingredients: (1) an indicator function that denotes whether we are inside or outside of the object of interest; (2) a metric from which we may derive contour information, and (3) an optimization method for placing the contour accurately.

To achieve accuracy, we need flexibility and robustness. Some have argued that it is useful to treat these three steps separately. This was first described in [47]) as the *morphological* method, but is also called by others *interactive* or *seeded* segmentation [31]. In this context, this does not mean that user interaction is required, only that object identification is provided by some means, and contour extraction separately by a segmentation operator.

The first ingredient, the object identification, or our indicator function, is of course essential and it is frustrating to be obliged to only write here "by some means". Accurate content identification can simplify the requirements on the segmentation operator greatly. Unfortunately, the means in question for contents identification are problem-dependent and sometimes difficult to publish, because they are often seen as *ad hoc* and of limited interest beyond their immediate use in the problem at hand. Fortunately, some journals accept such publications, such as the *Journal of Image Analysis and Stereology* and applications journals (e.g. *Journal of Microscopy*, materials, etc). There are also a few recent books on the matter [23,52]. Software libraries are also important but not many are freely available for training, although the situation is improving.

Also, whereas in computer vision a fully automated solution is required, in medical imaging a semi-automated method might be sufficient. In biomedical imaging, a large number of objects are typically measured (such as cells, organelles, etc.), and a fully automated method is often desirable. However, in medical imaging, typically a relatively small number of patients is being monitored, treated or surveyed, and so human-guided segmentation can be sufficient. The objective of the segmentation method in this context is to provide reasonable contours quickly, which can be adjusted easily by an operator.

In this variety of contexts, is it possible to define precisely the segmentation problem? The answer is probably no, at this stage at least in image analysis research. However, it is possible to provide *formulations* of the problem. While this may sound strange or even suspicious, the reason is that there exists a real need for automated or semi-automated segmentation procedures for both image analysis and computer vision, and so solutions have been proposed. They can still be explained, compared and evaluated.

3.1.4 Desirable Properties of Seeded Segmentation Methods

We come to the first conclusion that to provide reliable and accurate results, we must rely on a segmentation procedure and not just an operator. Object identification

and constraints analysis will set us in good stead to achieve our results, but not all segmentation operators are equivalent. We can list here some desirable properties of interactive segmentation operators.

- It is useful if the operator can be expressed in an energy or cost optimization formulation. It is then amenable to existing optimization methods, and this entails a number of benefits. Lowering the cost or the energy of the formulation can be done in several ways (e.g. continuous or discrete optimization), which results in different characteristics and compromises, say between memory resources and time. Optimization methods improve all the time through the work of researchers, and so our formulations will benefit too.
- It is desirable if the optimization formulation can provide a solution that is at least locally optimal, and if possible globally optimal, otherwise noise will almost certainly corrupt the result.
- The operator should be fast, and provide guaranteed convergence, because it will be most likely restarted several times, in order to adjust parameters. Together with this requirement, the ability to segment many objects at once is also desirable, otherwise the operator will need to be restarted as many time as there are objects in the image. This may not be a big problem if objects do not overlap and if bounding boxes can be drawn around them, because the operator can then be run only within the bounding box, but this is not the general case.
- The operator should be bias-free: e.g. with respect to objects size or to the discretization grid or with respect to initialization.
- The operator should be flexible: it is useful if it can be coupled with topology information for instance, or with multi-scale information.
- It should be generic, not tied to particular data or image types.
- It should be easy to use. This in practice means possessing as few parameters as possible. Of course one can view constraints setting as an enormous parameter list, but this is the reason why we consider this step as separate.

Such a method certainly does not yet exist to our knowledge, although some might be considered to come close. We describe some of them in the next section.

3.2 A Review of Segmentation Techniques

Here, we list and detail some segmentation categories that are compatible with the image analysis viewpoint, although cannot hope to present a complete description of this field.

3.2.1 Pixel Selection

Pixel selection is likely the oldest segmentation method. It consists of selecting pixels solely based on their values and irrespective of their spatial neighborhood.

The simplest pixel selection method is humble thresholding, where we select pixels that have a gray-level value greater or smaller than some threshold value. This particular method is of course very crude, but is used frequently nonetheless. Multiple thresholding uses several values instead of a single value; color and multi-spectral thresholding using vectors of values and not just scalars. By definition all histogram-based methods for finding the parameters of the thresholding, including those that optimize a metric to achieve this [54], are pixel selection methods. Statistical methods (e.g. spectral classification methods) that include no spatial regularization fall into this category as well. This is therefore a veritable plethora of methods that we are including here, and research is still active in this domain.

Of course, thresholding and related methods are usually very fast and easily made interactive, which is why they are still used so much. By properly pre-processing noisy, unevenly illuminated images, or by other transforms, it is surprising how many problems can be solved by interactive or automated thresholding. However, this is of course not always the case, hence the need for more sophisticated methods.

3.2.2 Contour Tracking

It was realized early on that (1) human vision is sensitive to contours and (2) there is a duality between simple closed contours and objects. A simple closed contour (or surface) is one that is closed and does not self-intersect. By the Jordan theorem, in the Euclidean space, any such contour or surface delineates a single object of finite extent. There are some classical difficulties with the Jordan theorem in the discrete setting [52], but they can be solved by selecting proper object/background connectivities, or by using a suitable graph, for instance, the 6-connected hexagonal grid or the Khalimsky topology [22, 40].

A contour can be defined locally (it is a frontier separating two objects (or an object and its background in the binary case)), while an object usually cannot (an object can have an arbitrary extent). A gradient (first derivative) or a Laplacian (second derivative) operator can be used to define an object border in many cases, and gradients are less sensitive to illumination conditions than pixel values. As a result, contour detection through the use of gradient or Laplacian operators became popular, and eventually led to the Marr–Hildreth theory [44].

Given this, it is only natural that most segmentation method use contour information directly in some ways, and we will revisit this shortly. Early methods used *only* this information to detect contours and then tried to combine them in some way. By far the most popular and successful version of this approach is the Canny edge detector [9]. In his classical paper, Canny proposed a closed-form optimal 1D edge detector assuming the presence of additive white Gaussian noise, and successfully proposed a 2D extension involving edge tracking using non-maxima suppression with hysteresis.

One problem with this approach is that there is no optimality condition in 2D, no topology or connectivity constraints and no way to impose markers in the final result. All we get is a series of contours, which may or may not be helpful. Finding a suitable combination of detected contours (which can be incomplete) to define objects is then a combinatorial problem of high complexity. Finally, this approach extends even less to 3D.

Overall, in practical terms, these contour tracking methods have been superseded by more recent methods and should not be used without good reasons. For instance, more recent minimal-path methods can be applied to contour tracking methods, although they are much more sophisticated in principle [3, 14]. In this class of methods belongs also the "intelligent scissors" types. There were many attempts in previous decades to provide automated delineating tools in various image processing software packages, but a useful contribution was provided relatively recently by Mortensen [48]. This method is strictly interactive, in the sense that it is designed for human interaction and feedback. "Intelligent scissor" methods are useful to clinicians for providing ground truth data for instance. Such methods are still strictly 2D. As far as we know, no really satisfying 3D live-wire/intelligent scissor method is in broad use today [5]. However, minimal surfaces methods, which we will describe shortly in Sect. 3.4.3, in some ways do perform this extension to nD [30].

3.2.3 Statistical Methods

The opposite approach to contour detection is to work on the objects, or regions themselves. An early and intuitive approach has been to try to divide (the *splitting* step) an image into uniform regions, for example using a hierarchical representation of an image in the form of quadtrees (in 2D) and octrees (in 3D). Uniformity can be defined by statistical parameters and/or tests. Subsequently, a *merging* step considering neighboring and statistical region information is performed [36]. Initial considered statistics were color and intensity, but other region descriptors can be used as well, for instance including texture, motion and so on. In this approach, even though regions statistics are used, they are inevitably derived at the pixel level. The split and merge approach consists of acquiring all the statistics first and basing a decision on them.

A different approach, which is also productive, consists of building a *model* first. One way is to consider an image as a 2D or 3D graph of pixels, to start from a vast over-segmentation at the pixel level, and to evolve cliques of pixels (e.g. sets of one, two or more pixels that are fully-connected, respectively called unary, binary or higher-level cliques) to fit that model. This is the *Markov Random Field* (MRF) model, named in this way by comparison to classical one-dimensional Markov chains, for which only immediate neighboring relationships matter. Models that can be written using these cliques turn out to corresponds to energies featuring

weighted finite sums with as many terms as there are different kinds of cliques. In [26] Geman and Geman proposed to optimize these sums using Gibbs sampling (a form of Monte-Carlo Markov Chain algorithm) and simulated annealing. This was first used for image restoration, but can be readily applied to segmentation as well. This approach was very successful because it is very flexible. Markers and texture terms can be added in, and many algorithmic improvement were proposed over the years. However, it remains a relatively costly and slow approach. Even though Geman and Geman showed that their simulated annealing strategy converges, it only does so under conditions that make the algorithm extremely slow, and so usually only a non-converged or approximate result is used. More recently, it was realized that Graph-Cut (GC) methods were well-suited to optimized some MRF energies very efficiently. We will give more details in the corresponding section.

MRFs belong to the larger class of Bayesian methods. Information-theoretic perspectives and formulations, such as following the Minimum Description Length principle, also exist. These frameworks are also very flexible, allowing for example region competition [69]. However, the corresponding models might be complicated both to understand and run, and sometimes possess many parameters that are not obvious to tune. Well-designed methods are guaranteed to converge to at least a local minimum.

In general, when dealing with regions that have complex content (for instance, textures, or multispectral content), statistical methods can be a very good choice although they cannot be recommended for general work, since simpler and faster methods often are sufficient.

3.2.4 Continuous Optimization Methods

In the late 1980s, it was realized that contour tracking methods were too limited for practical use. Indeed, getting closed contours around objects were difficult to obtain with contour tracking. This meant that detecting actual objects was difficult except in the simplest cases.

3.2.4.1 Active Contours

Researchers, therefore, proposed to start from already-closed loops, and to make them evolve in such a way that they would converge towards the true contours of the image. Thus were introduced *active contours*, or *snakes* [39]. The formulation of snakes takes the following continuous-domain shape:

$$E_{\text{snake}} = \int_0^1 \{E_{\text{internal}}(\mathbf{v}(s)) + E_{\text{data}}(\mathbf{v}(s)) + E_{\text{constraints}}(\mathbf{v}(s))\}\, \mathrm{d}s. \qquad (3.1)$$

where $\mathbf{v}(s)$ is a parametric representation of the contour.

This model is very flexible. It contains internal terms, image data terms and constraints terms (see Chapter 4 for more details):

- The first term, the internal energy, contains a curvature term and a "rubber band" energy. The former tends to smooth the resulting contour following a thin plate, while the latter tends to make it shrink around features of interest. Other terms such as kinetic energy can be added too, which makes it possible for the snake to avoid noisy zones and flat areas.
- The second term, the data energy, attracts the active contours towards points of interest in the image: typically, image contours (zones of high gradient), lines or termination points.
- The last term, the constraint term, is optional, but allows interaction with the snake by defining zones of attraction and repulsion.

To solve this equation, the Euler–Lagrange of (3.1) is worked out (typically in closed form), and a gradient descent algorithm is used. All the terms are combined in a linear combination, allowing them to be balanced according to the needs of the user. Due to its flexibility, the active contour model was very popular in the literature as well as in applications. It fits very well into the interactive segmentation paradigm because constraints can be added very easily, and it can be quite fast because it uses a so-called Lagrangian framework. The contour itself is discretized at regular interval points and evolves according to (3.1). Convergence towards a local minimum of the energy is guaranteed, but may require many iterations.

In practice, there are some difficulties: the snake energy is flexible but difficult to tune. Because of the contour evolution, points along the contour tend to spread out or bunch up, requiring regular and frequent resampling. There can also be topological difficulties, for instance causing the snake to self-intersect. The snake is also sensitive to its parametrization and to initialization. Finally, even though a local optimum is guaranteed, in practice, it may not be of good quality due to noise sensitivity.

One major difficulty with snakes is that they can be extended to 3D via triangulation, but such extensions can be complicated, and topological problems plaguing snakes in 2D are usually more difficult to avoid in 3D. However, 3D active surfaces are still widely used, because they make it easy to improve or regularize a triangulated surface obtained by other means. For instance, the brain segmentation software FreeSurfer includes such a method. To distinguish them from other models we are going to introduce now, snake-like active contours or surfaces are sometimes called *parametric deformable models*.

3.2.4.2 Level Sets

One way to avoid altogether some of the problems brought about by the way parametric deformable models are discretized, is to embed the contour into a higher-dimensional manifold. This idea gave rise to *level sets*, proposed by Osher and Sethian in 1988 [53]. Remarkably, this is around the same time when active contours

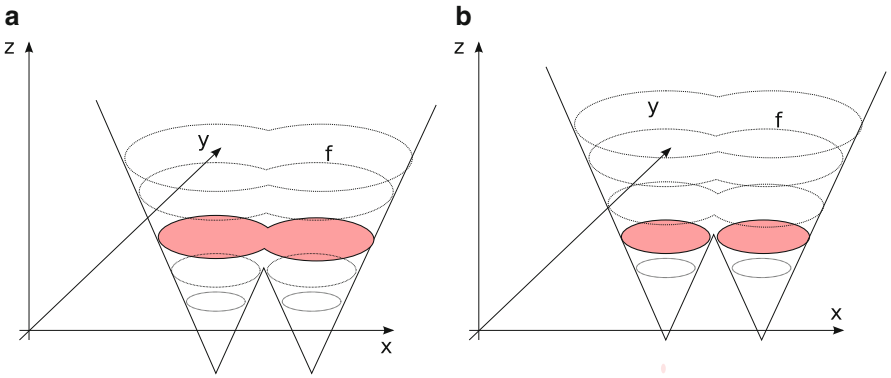

Fig. 3.1 Embedding and evolving a curve as a level set of a higher-dimension function. The zero-level of function ψ is shown in color, representing a 2D contour. To evolve the contour, the whole function evolves. Note that topology changes can occur in the contour, while the embedding surface shows no such effect

were proposed. However level sets were initially proposed for computational fluid dynamics and numerical simulations. They were applied to imaging somewhat later [43, 62]. A contour is represented on the surface S of an evolving regular function ψ by its zero level-set, which is simply the threshold of the function ψ at zero. By using sufficiently regular embedding functions ψ, namely signed distance transforms from an initial contour, it was possible to propose effective evolution equations to solve similar problems to Lagrangian active contours.

The main advantages of the level-sets method were that contour resampling was no longer necessary, and contour self-intersection (shock solutions) was avoided because level sets were able to change topology easily (see Fig. 3.1b). This means practically that it was possible at least in theory to initialize a segmentation by drawing a box around a series of object of interest, and the level set could find a contour around each of them. This was seen as a major benefit by the vision community. The level set Eulerian formulation (where the whole space is discretized) is thought to offer better theoretical guarantees than the Lagrangian framework of previous non-embedded formulations, and the simulation of function evolution is a well-researched topic with many usable and interesting results. Finally, the formulation is dimension independent. Level sets work virtually unchanged in 3D or more, which is a major benefit.

There are also a number of drawbacks. First, the level set formulation is more expensive than earlier active contour formulations. It requires the iterative solving of PDEs in the whole space, which is expensive. In practice, it is possible to limit the computation in a narrow band around the contour, but this is still more costly than if they were limited to the contour itself, and requires the resampling that was sought to be avoided. The surface S of function ψ is implicitly represented by the function itself, but it requires more space than the contour. In 3D or more,

this may be prohibitive. Some contour motions are not representable (e.g. contour rotation), but this is a minor problem. More importantly, the fact that level-sets can undergo topology changes is actually a problem in image analysis, where it is useful to know that a contour initialized somewhere will converge to a single simple closed contour. In some cases, a contour can split or even disappear completely, leading to undesirable results.

Nonetheless, level-set formulations are even more flexible than active contours, and very complex energies solving equally complex problems have been proposed in the literature. Solving problem involving texture, motion, competing surfaces and so on is relatively easy to formulate in this context [55, 56]. For this reason, they were and remain popular. Complex level-set formulation tend to be sensitive to noise and can converge to a poor locally optimal solution. On the other hand, more robust, closer to convex solutions can now be solved via other means. An example of relatively simple PDE that can be solved by level sets is the following:

$$\psi_t + F|\nabla\psi_t| = 0, \tag{3.2}$$

where F is the so-called speed function. Malladi and Sethian proposed the following for F:

$$F = \frac{1-\varepsilon\kappa}{1+|\nabla I|} + \beta(\nabla\psi.\nabla|\nabla I|). \tag{3.3}$$

The first part of the equation is a term driving the embedding function ψ towards contours of the image with some regularity and smoothing controlled by the curvature κ. The amount of smoothing is controlled by the parameter ε. The second term is a "balloon" force that tends to expand the contour. It is expected that the contour initially be placed inside the object of interest, and that this balloon force should be reduced or eliminated after some iterations, controlled by the parameter β. We see here that even though this model is relatively simple for a level-set one, it already has a few parameters that are not obvious to set or optimize.

3.2.4.3 Geodesic Active Contours

An interesting attempt to solve some of the problems posed by overly general level sets was to go back and simplify the problem, arguing for consistency and a geometric interpretation of the contour obtained. The result was the geodesic active contour (GAC), proposed by Caselles et al. in 1997 [10]. The level set formulation is the following:

$$\psi_t = |\nabla\psi|\,\mathrm{div}\left(g(I)\frac{\nabla\psi}{|\nabla\psi|}\right). \tag{3.4}$$

This equation is virtually parameter-free, with only a g function required. This function is a *metric* and has a simple interpretation: it defines at point x the cost of a contour going through x. This metric is expected to be positive definite, and in

most cases is set to be a scalar functional with values in $\mathbb{R}^+$. In other words, the GAC equation finds the solution of:

$$\text{argmin}_C \int_C g(s)\text{d}s, \tag{3.5}$$

where C is a closed contour or surface. This is the minimal closed path or minimal closed surface problem, i.e. finding the closed contour (or surface) with minimum weight defined by g. In addition to simplified understanding and improved consistency, (3.4) has the required form for Weickert's PDE operator splitting [28, 68], allowed PDEs to be solved using separated semi-implicit schemes for improved efficiency. These advances made GAC a reference method for segmentation, which is now widely used and implemented in many software packages such as ITK. The GAC is an important interactive segmentation method due to the importance of initial contour placement, as with all level-sets methods. Constraints such as forbidden or attracting zones can all be set through the control of function g, which has an easy interpretation.

As an example, to attract the GAC towards zones of actual image contours, we could set

$$g \equiv \frac{1}{1+|\nabla I|^p}, \tag{3.6}$$

With $p = 1$ or 2. We see that for this function, g is small (costs little) for zones where the gradient is high. Many other functions, monotonically decreasing for increasing values for ∇I, can be used instead. One point to note, is that GAC has a so-called *shrinking bias*, due to the fact that the globally optimal solution for (3.5) is simply the null contour (the energy is then zero). In practice, this can be avoided with balloon forces but the model is again non-geometric. Because GAC can only find a local optimum, this is not a serious problem, but this does mean that contours are biased towards smaller solutions.

3.2.5 Graph-Based Methods

The solution to (3.5) proposed in the previous section was in fact inspired by preexisting discrete solutions to the same problem. On computers, talking about continuous-form solutions is a bit of a misnomer. Only the mathematical formulation is continuous, the computations and the algorithms are all necessarily discrete to be computable. The idea behind discrete algorithm is to embrace this constraint and embed the discrete nature of numerical images in the formulation itself.

3.2.5.1 Graph Cuts

We consider an image as a graph $\Gamma(\mathcal{V},\mathcal{E})$ composed of n vertices $\mathcal{V}$ and m edges $\mathcal{E}$. For instance, a 2D $N \times N$ 4-connected square grid image will have $n = N^2$ vertices

and $m = 2 \times N \times (N-1)$ edges.[1] We assume that both the edges and the vertices are weighted. The vertices will typically hold image pixel values and the edge values relate to the gradient between their corresponding adjacent pixels, but this is not necessary. We assume furthermore that a segmentation of the graph can be represented as a graph *partition*, i.e:

$$V = \bigcup_{V_i \in \Gamma} V_i; \forall i \neq j, V_j \cap V_i = \emptyset. \tag{3.7}$$

Then $E^\star$ is the set of edges that are such that their corresponding vertices are in different partitions.

$$E^\star = \{e = \{p_i, p_j\} \in E, p_i \in V_i; p_j \in V_j, i \neq j\}. \tag{3.8}$$

The set $E^\star$ is called the *cut*, and the cost of the cut is the sum of the edge weights that belong to the cut:

$$C(E^\star) = \sum_{e \in E^\star} w_e, \tag{3.9}$$

where w_e is the weight of individual edge e. We assume these weights to be positive. Reinterpreting these weights as *capacities*, and specifying a set of vertices as connected to a *source* s and a distinct set connected to a *sink* t, the celebrated 1962 Ford and Fulkerson result [25] is the following:

Theorem 3.1. *Let P be a path in Γ from s to t. A flow through that path is a quantity which is constrained by the minimum capacity along the path. The edges with this capacity are said to be saturated, i.e. the flow that goes through them is equal to their capacity. For a finite graph, there exists a maximum flow that can go through the whole graph Γ. This maximum flow saturates a set of edges E^s. This set of edges define a cut between s and t, and this cut has minimal weight.*

This theorem is illustrated in Fig. 3.2.

In 2D and if Γ is planar, this duality essentially says that the Ford and Fulkerson minimum cut can be interpreted as a shortest path in a suitable dual graph to Γ [2]. In arbitrary dimension, the maxflow – mincut duality allows us to compute discrete minimal hypersurfaces by optimizing a discrete version of (3.4).

There exist many algorithms that can be used to compute the maximum flow in a graph (also called network in this framework), but none with a linear complexity. Augmenting paths algorithms [7] are effective in 2D where the number of vertices is relatively high compared to the number of edges. In 3D and above, where the reverse is true, push-relabel algorithms [27] are more efficient. These algorithms can only be used when there is one source and one sink. The case where there are multiple sources or sinks is known to be NP-hard. To compute energies comprising several sources or sinks and leading to multi-label segmentation, approximations

[1] This particular computation is left as an exercise to the reader.

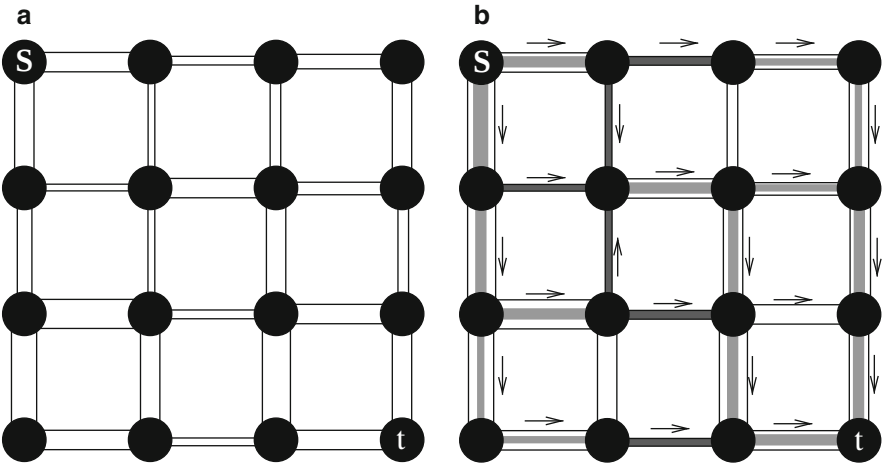

Fig. 3.2 (**a**) A graph with edge weights interpreted as capacities, shown as varying diameters in this case. (**b**) A maximum flow on this graph. We see that the saturated vertices (in *black*) separate *s* from *t*, and they form a cut of minimum weight

can be used, such as α-expansions. These can be used to formulate and optimize complex discrete energies with MRF interpretations [8, 66], but the solution is only approximate. Under some conditions, the result is not necessarily a local minimum of the energy, but can be guaranteed not to be too far from the globally optimal energy (within a known factor, often 2).

In the last 10 years, GC methods have become extremely popular due to their ability to solve a large number of problems in computer vision, particularly in stereo-vision and image restoration. In image analysis, their ability to form a globally optimal binary partition with a geometric interpretation is very useful. However, GC do have some drawbacks. They are not easy to parallelize, they are not very efficient in 3D, they have a so-called *shrinking bias*, just as GAC and continuous maxflow have as well. In addition, they have a *grid bias*, meaning that they tend to find contours and surfaces that follow the principal directions of the underlying graph. This results in "blocky" artifacts, which may or may not be problematic.

Due to their relationship with sources and sinks, which can be seen as internal and external markers, as well as their ability to modify the weights in the graph to select or exclude zones, GC are at least as interactive as the continuous methods of previous sections.

3.2.5.2 Random Walkers

In order to correct some of the problems inherent to graph cuts, Grady introduced the Random Walker (RW) in 2004 [29, 32]. We set ourselves in the same framework as in the Graph Cuts case with a weighted graph, but we consider from the start

a multilabel problem, and, without loss of generality, we assume that the edge weights are all normalized between 0 and 1. This way, they represent the probability that a random particle may cross a particular edge to move from a vertex to a neighboring one. Given a set of starting points on this graph for each label, the algorithm considers the probability for a particle moving freely and randomly on this weighted graph to reach any arbitrary unlabelled vertex in the graph before any other coming from the other labels. A vector of probabilities, one for each label, is therefore computed at each unlabeled vertex. The algorithm considers the computed probabilities at each vertex and assigns the label of the highest probability to that vertex.

Intuitively, if close to a label starting point the edge weights are close to 1, then its corresponding "random walker" will indeed walk around freely, and the probability to encounter it will be high. So the label is likely to spread unless some other labels are nearby. Conversely, if somewhere edge weights are low, then the RW will have trouble crossing these edges. To relate these observations to segmentation, let us assume that edge weights are high within objects and low near edge boundaries. Furthermore, suppose that a label starting point is set within an object of interest while some other labels are set outside of it. In this situation, the RW is likely to assign the same label to the entire object and no further, because it spreads quickly within the object but is essentially stopped a the boundary. Conversely, the RW spreads the other labels outside the object, which are also stopped at the boundary. Eventually, the whole image is labeled with the object of interest consistently labeled with a single value.

This process is similar in some way to classical segmentation procedures like seeded region growing [1], but has some interesting differentiating properties and characteristics. First, even though the RW explanation sounds stochastic, in reality the probability computations are deterministic. Indeed, there is a strong relationship between random walks on discrete graphs and various physical interpretations. For instance, if we equate an edge weight with an electrical resistance with the same value, thereby forming a resistance lattice, and if we set a starting label at 1 V and all the other labels to zero volt, then the probability of the RW to reach a particular vertex will be the same as its voltage calculated by the classical Kirchhoff's laws on the resistance lattice [24]. The problem of computing these voltages or probability is also the same as solving the discrete Dirichlet problem for the Laplace equation, i.e. the equivalent of solving $\nabla^2 \varphi = 0$ in the continuous domain with some suitable boundary conditions [38]. To solve the discrete version of this equation, discrete calculus can be used [33], which in this case boils down to inverting the graph Laplacian matrix. This is not too costly as it is large but very sparse. Typically calculating the RW is less costly and more easily parallelizable than GC, as it exploits the many advances realized in numerical analysis and linear algebra over the past few decades.

The RW method has some interesting properties with respect to segmentation. It is quite robust to noise and can cope well with weak boundaries (see Fig. 3.3). Remarkably, in spite of the RW being a purely discrete process, it exhibits no grid bias. This is due to the fact that level lines of the resistance distance (i.e. the

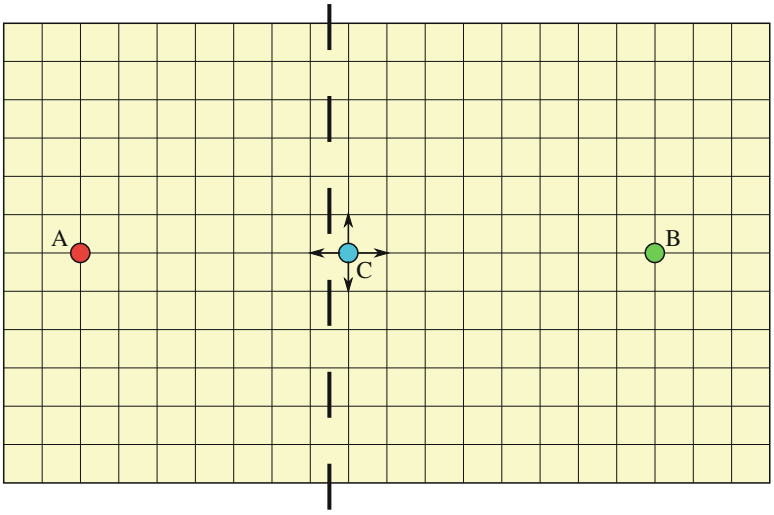

Fig. 3.3 An intuitive explanation of why the Random Walker copes well with weak boundaries. We assuming constant, high probabilities everywhere on this graph, except where *thick vertical lines* cross an edge, where the probabilities are low. A and B represent labels, and we estimate the probability of a random walker in C to move to the left as opposed to all the other directions (north, south, or east). We see that locally the probabilities are identical, but globally, there are many ways for a random walker to come from B to the north, east or south position from C. However, there is only one way to move to the west of C, and that is to go through C. Therefore, Random walker probabilities must be high up to C, and then drop precipitously. Since the situation is symmetrical with respect to A, it is likely that the region left of he thick lines will be labelled with A, and the region right to it are going to be labelled with B. This is in spite of the fact that the boundary defined by the *thick vertical lines* is weak and closer to A than B

resistance between a fixed node and all the others) in an infinite graph with constant edge weights are asymptotically isotropic [21]. RW exhibit a shrinking bias but not as strong as GC.

3.2.5.3 Watershed

While there are many variations on discrete segmentation methods, we will consider one last method: the Watershed Transform (WT). It was introduced in 1979 by Beucher and Lantuéjoul [6] by analogy to the topography feature in geography. It can be explained intuitively in the following manner: consider a gray-level image to be a 3D topographical surface or terrain. A drop of water falling onto this surface would follow a descending path towards a local minimum of the terrain. The set of points, such that drops falling onto them would flow into the same minimum, is called a *catchment basin*. The set of points that separate catchment basins form the *watershed line*. Finally, the transform that takes an image as input and produces its set of watershed lines is called the *Watershed Transform*. To use

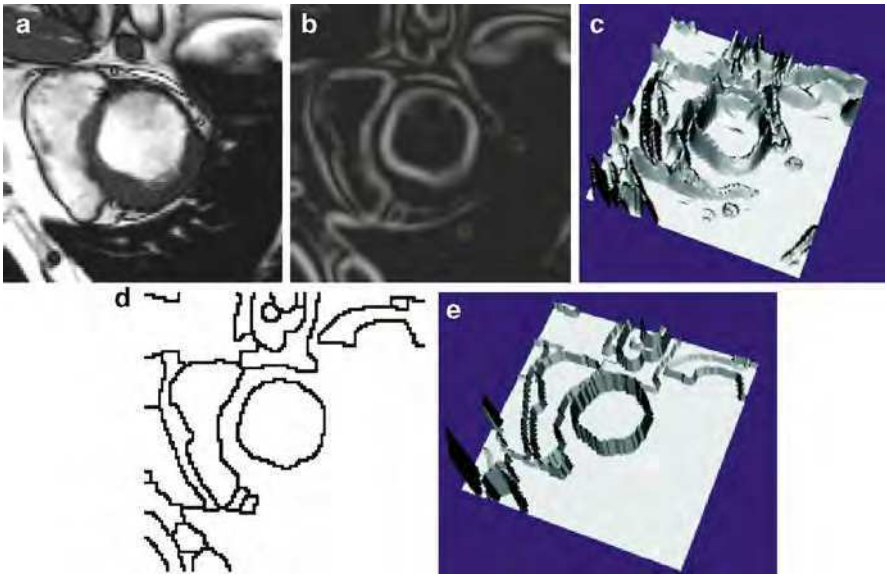

Fig. 3.4 Watershed segmentation: (**a**) an MRI image of the heart, (**b**) its smoothed gradient, (**c**) the gradient seen as a topographic surface, (**d**) Watershed of the gradient, and (**e**) topographical view of the watershed

this transform in practical segmentation settings, we must reverse the point of view somewhat. Assume now that labels are represented by lakes on this terrain and that by some flooding process, the water level rises evenly. The set of points that are such that waters from different lakes meet is also called the watershed line. Now this watershed line is more constrained, because there are only as many lines as necessary to separate all the lakes.

This intuitive presentation is useful but does not explain why the WT is useful for segmentation. As the "terrain", it is useful to consider the magnitude of the gradient of the image. On this gradient image, interior objects will have values close to zero and will be surrounded by zones of high values: the contours of the objects. They can therefore be assimilated to catchment basins, and the WT can delineate them well (see Fig. 3.4).

The WT is a seeded segmentation method, and has many interesting interpretations. If we consider the image again as a graph as in the GC setting, then on this graph the set of watershed lines from the WT form a graph cut. The edges of this tree can be weighted with a functional derived from a gradient exactly as in the GC case. Computing the WT can be performed in many efficient ways [46, 67], but an interesting one is to consider the Maximum Spanning Forest (MSF) algorithm [19]. In this algorithm, the classical graph algorithm for maximum spanning tree (MST) is run on the graph of the image, following for instance Kruskal's algorithm [41, 59], with the following difference: when an edge selected by the MST algorithm is connected with a seed, then all vertices that are connected with it become also

labeled with this seed, and so on recursively. However, when an edge selected by the MST algorithm would be connecting two different seeds, the connection is simply not performed. It is easy to show that (1) eventually all edges of the graph are labeled with this algorithm; (2) the set of edge that are left connected form a graph cut separating all the seeds; and (3) the labels are connected to the seeds by subtrees. The result is a MSF, and the set of unconnected edges form a watershed line. The MSF algorithm can be run in quasi-linear time [20].

3.2.6 Generic Models for Segmentation

Even though seeded models are the focus of this chapter, we say here a few words about generic models that are not seeded by default, because they contain powerful ideas for the future of seeded models.

3.2.6.1 Continuous Models

Over the years, several now widely cited formulations of the segmentation problem have been proposed, including for instance the Mumford–Shah functional [49] or the Chan–Vese active contour without edges (AWE) [13]. They generally seek to solve the segmentation problem in the vision setting, and can be used for image restoration as well (denoising, inpainting, etc).

In particular, the Mumford–Shah functional is the following:

$$E(\mathbf{f},C) = \beta \int_{\Omega} (\mathbf{f}-\mathbf{g})^2 dA + \alpha \int_{\Omega \setminus C} |\nabla \mathbf{f}|^2 dA + \gamma \int_{C} ds. \tag{3.10}$$

This formulation is very interesting because it has been an inspiration to many. In this expression, **g** is the original image, **f** a piecewise smooth approximation of **g** and C a collection of contours where **f** is discontinuous. In essence, C represents the segmentation of $\eth$ and **f** is a restored (denoised, etc) model of $\eth$. The first term in (3.10) is a data fidelity term; the second is a total variation term (TV), and the last optimizes an unweighted contour length.

Both MS and AWE initially were solved using level-sets methods, but more recently convex methods have been used. The MS functional is NP-hard in general, but convex relaxations are computable, and can be exact in the binary case. In particular, the ROF model is convex, and correspond to the MS model without the last term [12]. From the image analysis point of view, these models are not readily usable, because they correspond to simplistic models of vision, and if markers or shape constraints are added, they tend to dominate the model, which then does not help very much.

3.2.6.2 Hierarchical Models

Hierarchies of segmentations are a powerful way to deal with the multi-resolution inherent to nature. Many images contain objects at different scales. In medical imaging a vascular network is a typical example. It is very difficult to come up with a seeded strategy to solve this case. One general idea is to perform many segmentations at once or in sequence, taking into account various scales. This is not as easy to do as it sounds, because simply repeating a segmentation procedure with different parameters will not yield compatible segmentations, in the sense that contours are not likely to remain stable as the scale increases or decreases. One way of dealing with this is to offer a measure to the strength of a particular piece of contour, and as the scale increase, remove pieces of contours with weak strength first. This *saliency* idea was proposed by Najman and Schmitt in [51] in the context of watershed segmentation, but more work has been done on this idea since, for example, on ultrametric watershed and connections [50, 63]. A saliency map or ultrametric watershed is an interactive segmentation because edge strength can be selected by interactive thresholding for instance, but it is not always obvious how to combine this with seeded segmentation.

Hierarchical methods do offer some other benefits, such as the ability to efficiently optimize Mumford–Shah-like functionals on a saliency map [35]. Other functionals are also possible, such as optimizing minimum ratio costs [34]. There are some drawbacks as well, such as decreased speed and extra memory requirement, and again the question of compatibility with other constraints. This is at present a very interesting area of research.

3.2.6.3 Combinations

Many segmentation algorithms can be combined to provide different sets of compromises or extensions. For instance, Yuille proposed an interesting model combining Bayesian methods with level-sets [69]. An active area of research today are so-called *turbopixels*, where a first-level over-segmentation is performed in order to group pixels into consistent regions of similar size. Then these regions are linked in a graph and a discrete segmentation is performed over them [42]. This two-level segmentation procedure has some advantages in terms of speed and resource allocations. Final segmentation can still be precise if the first-order grouping is done well, and these methods are compatible with seeded segmentation. However, segmentation quality may be poor in the presence of weak edges [64].

3.3 A Unifying Framework for Discrete Seeded Segmentation

In many early segmentation methods, the focus was on the values of the pixels themselves, or in graph terms the values of the vertices. Since the advent of GC methods, it was realized that focusing instead on the edges was useful. In particular,

defining a gradient function on the edges is easy. Let p and q be two vertices in the graph $\Gamma(\mathscr{V},\mathscr{E})$ of image I, that we have been using so far (see Sect. 3.2.5), then we can set as weight $w_{p,q}$ for the edge linking p and q any value depending on the discrete gradient $I_q - I_p$, where I_q represents the value of I at vertex q. For instance, we can use $w_{p,q} = \exp(-\beta|I_q - I_p|^2)$, with β a positive scalar parameter. This is a monotonically decreasing function of the gradient, recommended by several authors. In addition, there are topological advantages, as a cut in such a graph obeys the Jordan property in arbitrary dimension. In addition, there is a fundamental difference between regions, formed of uniformly labeled vertices, and cuts formed of edges. In former pixel-based segmentation procedures, the contours were themselves made of pixels, which created problems [19]. The only significant drawback is that storing edge weights rather than pixels costs roughly twice as much memory in 2D, or three times as much in 3D for the simplest nearest-neighbor connectivity. This extra cost increases with the connectivity, and may be indeed be a problem in some applications.

3.3.1 Discrete Optimization

Assuming then this simple model of discrete images, the segmentation problem can be viewed as an optimization problem over cliques of one or two pixels, like in the MRF setting. For instance, classical graph cut can optimize the following problem exactly:

$$\text{argmin}_x E(x) = \sum_{u\in\mathscr{V}} w_u|x_u - y_u| + \sum_{(u,v)\in\mathscr{E}} w_{u,v}|x_u - x_v|, \tag{3.11}$$

in the case where x is a binary vertex labeling, y a reference binary image that can, for instance, represent seeds, and w_u and $w_{u,v}$ positive unary weights and binary weights respectively. The seeded segmentation case corresponds to an image y containing some vertices labelled with 0, others with 1 (the seeds) and unlabelled ones as well. The w_u for the labelled vertices in y have infinite weights, and the unlabeled one zero. Using the same notation, the Random Walker optimizes the following energy:

$$\text{argmin}_x E(x) = \sum_{u\in\mathscr{V}} w_u(x_u - y_u)^2 + \sum_{(u,v)\in\mathscr{E}} w_{u,v}(x_u - x_v)^2. \tag{3.12}$$

In this case, the optimal labelling $x^\star$ is not binary even if y is binary. It expresses the probability of a vertex belonging to label 0 or label 1. To reach a unique solution, we must threshold the result:

$$s_u = 0 \text{ if } x_u < \frac{1}{2}, s_u = 1 \text{ otherwise.} \tag{3.13}$$

In this case, the binary result s represents the segmentation. There is a striking similarity between (3.11) and (3.12), which leads us to propose a unifying framework.

Table 3.1 Our generalized scheme for image segmentation includes several popular segmentation algorithms as special cases of the parameters p and q. The power watershed are previously unknown in the literature, but may be optimized efficiently with a MSF calculation

$q \backslash p$	0	Finite	∞
1	Collapse to seeds	Graph cuts	Power watershed $q = 1$
2	ℓ_2 norm Voronoi	Random walker	Power watershed $q = 2$
∞	ℓ_1 norm Voronoi	ℓ_1 norm Voronoi	Shortest path forest

3.3.2 A Unifying Framework

We propose to optimize the following general discrete energy:

$$\text{argmin}_x E(x) = \sum_{u \in \mathcal{V}} w_u^p |x_u - y_u|^q + \sum_{(u,v) \in \mathcal{E}} w_{u,v}^p |x_u - x_v|^q, \tag{3.14}$$

The p and q terms are integer exponents. In cases where the optimal $x^\star$ is not binary, we threshold it in the end as in (3.13). An analysis of the influence of p and q provides us with Table 3.1.

In this table, we find some well-known algorithms, such as previously mentioned GR and RW, in addition to the Shortest Path Forests algorithm [20], that uses forests of shortest path leading to seeds as segmentation criteria. Most of the other cases are not interesting (Voronoi diagrams, for instance), but the case $q = 1$ or 2 and $p \to \infty$ is novel and interesting: this is the Power Watershed algorithm [15].

3.3.3 Power Watershed

Among the drawbacks of traditional watershed as described in Sect. 3.2.5.3 are the following: (1) watershed has no energy interpretation and is purely a segmentation algorithm; (2) watershed segmentations are not unique: for the same seed placement and edge weights, the same definition can provide different results; (3) watershed results tend to leak in the presence of weak boundaries. We intend to solve all three problems.

An analysis of the convergence of (3.14) in the case $q = 1$ or 2 and $p \to \infty$ led us to the algorithm shown below.

This algorithm is illustrated in Fig. 3.5. We also show some pictorial results in Fig. 3.6, where we compare qualitatively the results of PW with the other classical discrete segmentation algorithms, namely GC, RW, SPF and the classical WT in the form of a MSF.

More details on the Power Watershed algorithm can be found in [16]. We show the PW algorithm performs very well in terms of quantitative results, that qualitatively PW is devoid of size bias and grid artifacts, while being only slightly

Algorithm: power watershed algorithm, optimizing $p \to \infty, q \geq 1$

Data: A weighted graph $\Gamma(\mathcal{V}, \mathcal{E})$ and a reference image y containing seed information

Result: A **potential function** x and a labeling s associating a label to each vertex.

Set x values as unknown except seed values.
Sort the edges of E by decreasing order of weight.
while *any node has an unknown potential* **do**
  Find an edge (or a plateau) E_{MAX} in E of maximal weight; denote by S the set of nodes connected by E_{MAX}.
  if *S contains any nodes with known potential* **then**
    Find x_S minimizing (3.14) (using the input value of q) on the subset S with the weights in E_{MAX} set to $w_{ij} = 1$, all other weights set to $w_{ij} = 0$ and the known values of x within S fixed to their known values.
    Consider all x_S values produced by this operation as known.
  else
    Merge all of the nodes in S into a single node, such that when the value of x for this merged node becomes known, all merged nodes are assigned the same value of x and considered known.
Set $s_i = 1$ if $x_i \geq \frac{1}{2}$ and $s_i = 0$ otherwise.

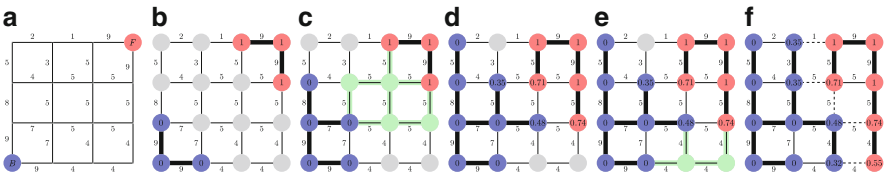

Fig. 3.5 Illustration of the different steps for Algorithm in the case $q = 2$. The values on the nodes correspond to x, their color to s. The bold edges represents edges belonging to a Maximum Spanning Forest. (**a**) A weighted graph with two seeds, all maxima of the weight function are seeded, (**b**) First step, the edges of maximum weight are added to the forest, (**c**) After several steps, the next largest edge set belongs to a plateau connected to two labeled trees, (**d**) Minimize (3.14) on the subset (considering the merged nodes as a unique node) with $q = 2$ (i.e., solution of the Random Walker problem), (**e**) Another plateau connected to three labeled vertices is encountered, and (**f**) Final solutions x and s obtained after few more steps. The q-cut, which is also an MSF cut, is represented in *dashed lines*

slower than standard watershed and much faster than either GC or RW, particularly in 3D. The PW algorithm provides a unique unambiguous result, and an energy interpretation for watershed, which allows it to be used in wider contexts as a solver, for instance in filtering [17] and surface reconstruction. One chief advantage of PW with respect with GC for instance, is its ability to compute a globally optimal result in the presence of multiple labels. When segmenting multiple objects this can be important.

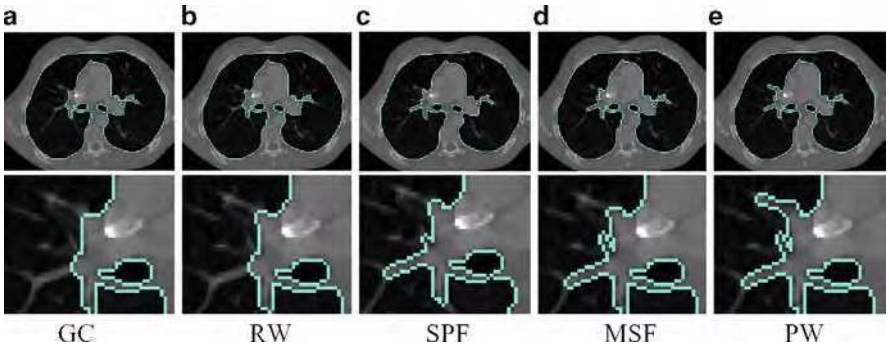

Fig. 3.6 Slides of a 3D lung segmentation. The foreground seed used for this image is a small rectangle in one slice of each lung, and the background seed is the frame of the image (**a**) GC (**b**) RW (**c**) SPF (**d**) MSF (**e**) PW

3.4 Globally Optimum Continuous Segmentation Methods

Here, we provide some arguments for globally optimal segmentation in the context of continuous-domain optimization.

3.4.1 Dealing with Noise and Artifacts

Even assuming we can construct a contents metric as explained in the first section, there are several sources of artifacts in segmentation: (1) weak edges cause uncertainty in the result; (2) noise tends to corrupt boundaries, and for some methods tend to lead to wrong results; (3) method artifacts, such as a size bias or blockiness artifacts can cause undesirable results. Of course all these artifacts are linked and essentially due to the contents metric, reflecting insufficient knowledge about the content, but it is precisely to solve this problem that we require segmentation.

Weak edges are a fact of life in medical imaging. Most often in CT for example it is difficult to delineate a lesion because it has a similar radiation absorption profile to surrounding tissues. In this case, it is better to use methods that interpolate contours and surfaces well. The GAC is very useful in this context because of its geometric formulation and shortest path/minimal surface interpretation. In addition, it is straightforward to add simple shape information, such as elliptical or spherical shape priors.

Many iterative methods do not cope well with noise. One reason might be that the formulation of the corresponding energy is not convex, which implies that it would probably not have a single global optimum. This is unfortunately the case with most active contours and level set formulations, including the classic formulation of GACs. In addition, these methods make it easy to add terms to the energy and make it look like it can be optimized. The reality is that in most cases, these methods

get stuck into a poor quality local minimum. If models are complex, tweaking their parameters is difficult and error-prone. This is the reason why most recent segmentation models feature few parameters and tend to propose formulations that can be optimized globally.

Finally, all segmentation methods present artifacts. Graph Cuts for instance tend to both produce blocky results (grid bias) and favour small objects (shrinking bias). They can be coped with by augmenting the connectivity of the graph and by metric manipulation knowing the position of the seeds. However, it is preferable to use formulations that are isotropic in nature, such as continuous-domain ones.

These are some of the reasons that motivate us to mention continuous, isotropic, efficient formulations for finding the global solution to the GAC equation exactly.

3.4.2 Globally Optimal Geodesic Active Contour

In spite of advances in GAC optimization, more efficient ways of solving equation (3.5) do exist. In particular, in 2D, this equation can be solved by a continuous-domain, non point-convex circular shortest path [3]. The solution, called the globally optimal geodesic active contour (GOGAC) is globally optimal and extremely efficient [4], although it can only find a single contour at a time. The GOGAC solution is as flexible as the original GAC, but due to its formulation and algorithm, it is significantly less affected by noise.

This GOGAC has no shrinking bias and no grid bias, however, it tends to favor circular boundaries due to its polar coordinate equivalence. This may be desirable in some applications, but not in others. This can be avoided by using a different weighting than the $1/r$ given in the original article. A flat weighting can be used if small solutions are forbidden for instance.

3.4.3 Maximal Continuous Flows and Total Variation

The GOGAC solution is extremely efficient but does not extend to 3D and higher, but in 2006, Appleton and Talbot proposed a continuous maximum flow (CMF) solution to solve this problem. Their solution, inspired by local solutions for discrete graph cuts, consists of simulating a flow originating from a source s and ending in a sink t, and a pressure field, linked by a PDE system forming a propagation equation and constrained by the metric g:

$$\begin{aligned} \frac{\partial \vec{F}}{\partial t} &= -\nabla P \\ \frac{\partial P}{\partial t} &= -\operatorname{div} \vec{F} \\ \|\vec{F}\| &\leq g. \end{aligned} \tag{3.15}$$

This unusual system, at convergence, produces a scalar field P that acts as an indicator function for the interior of the contour C of (3.5). It solves the closed minimal surface problem exactly and efficiently, and so this represents a better way to solve it than (3.4). The result in 2D is exactly the same as that obtained with GOGAC. This CMF result provides a direct algorithm for solving the problem posed by Iri and Strang in [37, 65]. Interestingly, researchers in image restoration had proposed over the years solutions to Strang's dual problem, that of minimizing the *total variation* (TV) of a functional. Initial solutions used level-set formulations [60], and later ones convex optimization methods [11, 58]. Nonetheless, it is thought that primal maximum flow methods are better suited to segmentation than TV formulations [18]. Note that CMF are also biased towards small contours, and because they find a global optimum, this is a more serious problem than with standard GAC. However, there exist ways to remove this shrinking bias for an arbitrary collection of sources and sinks [2], and the bias is less strong in 3D and can be ignored, as long as small solutions are forbidden, using for instance large enough inner seeds. CMFs are about as fast as GC, but can be parallelized easily.

3.5 Comparison and Discussion

In the space of a single chapter it is not possible to present a thorough, quantitative assessment of the most popular segmentation methods. However, in Table 3.2, we present a qualitative comparison.

In this table, we have presented all the methods discussed in the chapter. A score of 1 indicates a low, undesirable score and the highest score is 5. These scores are potentially controversial and represent experience and opinion rather than hard fact. We have ranked all methods according to some desirable features. In the following discussion, we present robustness as the ability of a method to cope with noise and weak edges. Flexibility denotes the ability of a method to be used in different contexts: seeded or non-seeded segmentation, and the possibility to optimize different models, for instance with texture.

Taking the methods in order, we see that (1) Pixel selection uses low resources but is extremely simplistic; (2) Contour tracking has some speed and flexibility advantages but is limited to 2D; (3) Split-and-merge methods generally have high scores but are not robust and not flexible; (4) MRFs and Bayesian methods optimized by simulated annealing feature a lot of flexibility but are very slow; (5) Active contours are fast and flexible but not robust, they find only one object at a time, and cannot be extended easily to 3D; (6) Level sets (LS) are similar in some ways but are quite slow, require lots of resources and are not robust. They do extend to 3D readily; (7) GAC are a particular case of LS methods, which are popular in medical imaging because they are faster and more robust but less flexible. However standard GAC is slow compared to many other methods and still not robust enough; (8) Graph cuts are a very popular recent method, which feature relatively high scores across the board, in particular they are among the most flexible and

Table 3.2 A qualitative assessment of many popular segmentation methods. See text for details

	High Speed	Low memory	Multi-label	Flexibility	Robustness	No bias	3D and more	Parallelizable	Multi-resolution	score
Pixel selection	5	5	1	1	1	5	5	5	1	29
Contour tracking	5	5	1	4	3	4	1	1	2	26
Split and merge	4	4	5	2	2	3	4	3	5	32
MRF – SA	1	3	4	4	3	3	3	5	2	28
Active contours	4	4	1	5	2	2	2	2	2	24
Level sets	1	2	2	5	2	3	5	4	3	27
GAC	2	2	2	3	3	3	5	4	3	27
Graph cuts	2	3	2	4	5	2	4	2	3	27
Watershed	4	4	5	2	3	5	5	3	4	35
Random Walker	3	3	5	3	4	4	4	4	3	33
GOGAC	5	3	1	1	5	3	1	1	2	22
CMF	3	2	1	2	5	3	5	4	3	28
Power watershed	4	3	5	3	4	5	5	2	4	35

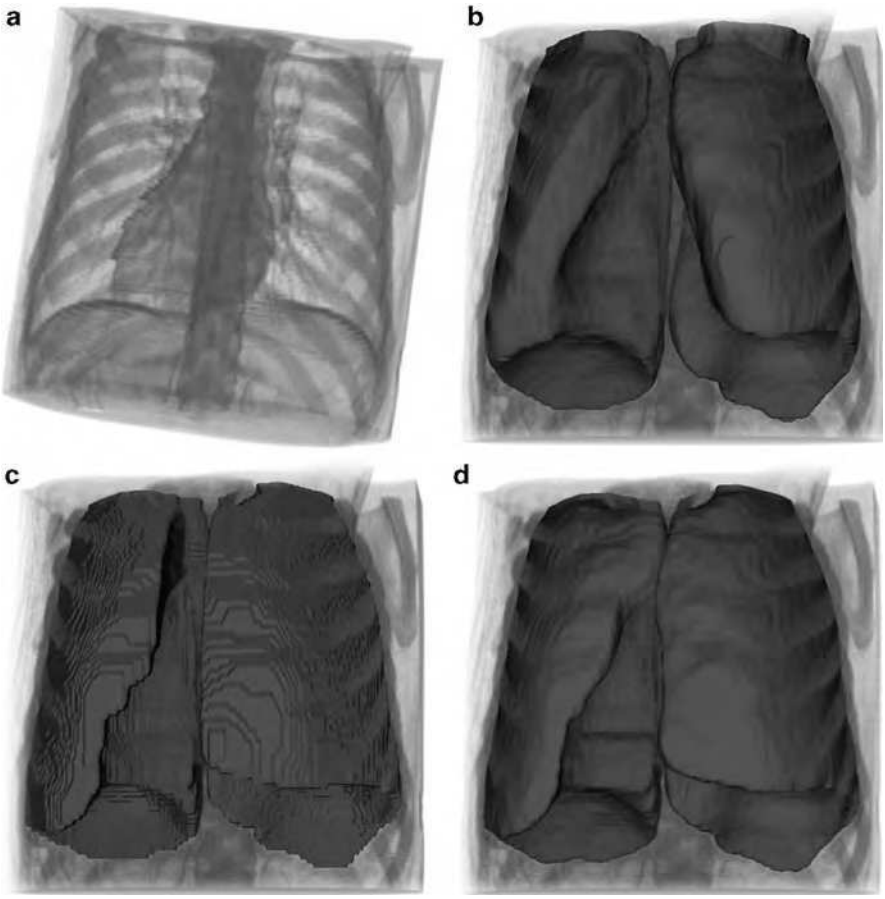

Fig. 3.7 Segmentation of the lungs in a chest CT image. (**a**) The CT image. (**b**) Segmentation using 3D standard Geodesic Active Contours. The surfaces fail to fill the base of the lung. (**c**) Segmentation using a discrete maximal flow algorithm. Observe the directional bias due to the grid. (**d**) Segmentation from identical input using continuous maximal flows

robust methods. However, they are slow, particularly in 3D, not parallelizable easily and feature much bias; (9) Watershed is an old method but has definite advantages for segmentation: it is fast, bias-free and multi-label (it can segment many objects at once). However, it is not flexible or very robust. Watershed can be extended readily for multi-resolution, and due to its age, many parallel implementations exist, including hardware ones; (10) The Random Walker is a recent method which is similar in some ways to Watershed, but is significantly more robust. It requires more resources however.

Among the newer methods presented in this chapter, (11) GOGAC solves GAC exactly and quickly in 2D, and so provides a quick robust solution, which is good for 2D interactive segmentation of single objects. However, is not flexible in its model;

(12) CMF is probably among the most robust segmentation method in the literature for 3D segmentation, but it segments only one object at a time, is not very flexible, and has no grid bias but does feature a shrinking bias. Finally, (13) Power watershed fits in between standard watershed and random walker. It is significantly more flexible and robust than standard watershed. Its speed is also comparable, but it uses more memory, and is less parallelizable.

The global score is probably even more subject to controversy than the individual ones, but it would tend to show that active contour methods should not be tried as a first choice method. For medical imaging, Random Walker and watershed-based methods are probably a good first choice, particularly for ease of use. It is comforting to realize that more modern methods suitable for 3D medical imaging (GC, RW, PW and CMF) are all very robust.

Many advantages presented in the literature, such as purported sub-pixel accuracy of segmentation, are not listed here because they are an illusion. The reported ability of some methods to control topology or on the contrary to allow it to change is not necessarily a drawback or advantage either way, so we do not include it as well.

3.6 Conclusion and Future Work

In conclusion, we argue that seeded or interactive segmentation is useful in medical imaging. Compared with model-based segmentation, seeded segmentation is more robust in actual image analysis applications, as opposed to computer vision. The ability to separate seeds/markers, use contour information, and perform contour optimization are very useful, as these elements generally result in a higher likelihood of good results. From this point of view, we argue that segmentation is a process and not merely an operator.

In general, the literature focuses on contour placement optimization at the expense of the other two components, with some rare exceptions. This is unfortunate but understandable with respect to seeds/markers, because they are highly application dependent. The choice of methods for obtaining contour information is also limited, and this is probably a good area for future research. One conclusion of this study is that contour placement optimization methods are important. More recent methods focus on optimization robustness, which is important. For someone not yet experienced in medical segmentation, simpler, more robust methods should be preferred over complex ones. Among those, power-watershed is a good candidate because of its combination of speed, relative robustness, ability to cope with multiple labels, absence of bias and availability (the code is easily available online). The random walker is also a very good solution, but is not generally and freely available.

We have not surveyed or compared methods that encompass shape constraints. We recognize that this is important in some medical segmentation methods, but this would require another study altogether.

Finally, at present there exists a dichotomy between the way discrete and continuous-domain work and are presented. In the near future, it is likely we will see methods unifying both aspects to great advantage.

References

1. Adams, R., Bischof, L.: Seeded region growing. IEEE Trans. Pattern Anal. Mach. Intell. **16**(6), 641–647 (1994)
2. Appleton, B.: Globally minimal contours and surfaces for image segmentation. Ph.D. thesis, University of Queensland (2004). Http://espace.library.uq.edu.au/eserv/UQ:9759/ba_thesis.pdf
3. Appleton, B., Sun, C.: Circular shortest paths by branch and bound. Pattern Recognit. **36**(11), 2513–2520 (2003)
4. Appleton, B., Talbot, H.: Globally optimal geodesic active contours. J. Math. Imaging Vis. **23**, 67–86 (2005)
5. Ardon, R., Cohen, L.: Fast constrained surface extraction by minimal paths. Int. J. Comput. Vis. **69**(1), 127–136 (2006)
6. Beucher, S., Lantuéjoul, C.: Use of watersheds in contour detection. In: International Workshop on Image Processing. CCETT/IRISA, Rennes, France (1979)
7. Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. PAMI **26**(9), 1124–1137 (2004)
8. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **23**(11), 1222–1239 (2001)
9. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **8**(6), 679–698 (1986)
10. Caselles, V., Kimmel, R., Sapiro, G.: Geodesic active contours. Int. J. Comput. Vis. **22**(1), 61–79 (1997)
11. Chambolle, A.: An algorithm for total variation minimization and applications. J. Math. Imaging Vis. **20**(1–2), 89–97 (2004)
12. Chan, T., Bresson, X.: Continuous convex relaxation methods for image processing. In: Proceedings of ICIP 2010 (2010). Keynote talk, http://www.icip2010.org/file/Keynote/ICIP
13. Chan, T., Vese, L.: Active contours without edges. IEEE Trans. Image Process. **10**(2), 266–277 (2001)
14. Cohen, L.D., Kimmel, R.: Global minimum for active contour models: A minimal path approach. Int. J. Comput. Vis. **24**(1), 57–78 (1997). URL citeseer.nj.nec.com/cohen97global.html
15. Couprie, C., Grady, L., Najman, L., Talbot, H.: Power watersheds: A new image segmentation framework extending graph cuts, random walker and optimal spanning forest. In: Proceedings of ICCV 2009, pp. 731–738. IEEE, Kyoto, Japan (2009)
16. Couprie, C., Grady, L., Najman, L., Talbot, H.: Power watersheds: A unifying graph-based optimization framework. IEEE Transactions on Pattern Analysis and Machine Intelligence, **33**(7), 1384–1399 (2011)
17. Couprie, C., Grady, L., Talbot, H., Najman, L.: Anisotropic diffusion using power watersheds. In: Proceedings of the International Conference on Image Processing (ICIP), pp. 4153–4156. Honk-Kong (2010)
18. Couprie, C., Grady, L., Talbot, H., Najman, L.: Combinatorial continuous maximum flows. SIAM J. Imaging Sci. (2010). URL http://arxiv.org/abs/1010.2733. In revision
19. Cousty, J., Bertrand, G., Najman, L., Couprie, M.: Watershed cuts: Minimum spanning forests and the drop of water principle. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1362–1374. (2008)

20. Cousty, J., Bertrand, G., Najman, L., Couprie, M.: Watershed cuts: Thinnings, shortest-path forests and topological watersheds. IEEE Trans. Pattern Anal. Mach. Intell. **32**(5), 925–939 (2010)
21. Cserti, J.: Application of the lattice Green's function for calculating the resistance of an infinite network of resistors. Am. J. Phys. **68**, 896 (2000)
22. Daragon, X., Couprie, M., Bertrand, G.: Marching chains algorithm for Alexandroff-Khalimsky spaces. In: SPIE Vision Geometry XI, vol. 4794, pp. 51–62 (2002)
23. Dougherty, E., Lotufo, R.: Hands-on Morphological Image Processing. SPIE press, Bellingham (2003)
24. Doyle, P., Snell, J.: Random Walks and Electric Networks. Carus Mathematical Monographs, vol. 22, p. 52. Mathematical Association of America, Washington, DC (1984)
25. Ford, J.L.R., Fulkerson, D.R.: Flows in Networks. Princeton University Press, Princeton, NJ (1962)
26. Geman, S., Geman, D.: Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. PAMI **6**, 721–741 (1984)
27. Goldberg, A., Tarjan, R.: A new approach to the maximum-flow problem. J. ACM **35**, 921–940 (1988)
28. Goldenberg, R., Kimmel, R., Rivlin, E., Rudzsky, M.: Fast geodesic active contours. IEEE Trans. Image Process. **10**(10), 1467–1475 (2001)
29. Grady, L.: Multilabel random walker image segmentation using prior models. In: Computer Vision and Pattern Recognition, IEEE Computer Society Conference, vol. 1, pp. 763–770 (2005). DOI http://doi.ieeecomputersociety.org/10.1109/CVPR.2005.239
30. Grady, L.: Computing exact discrete minimal surfaces: Extending and solving the shortest path problem in 3D with application to segmentation. In: Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference, vol. 1, pp. 69–78. IEEE (2006)
31. Grady, L.: Random walks for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **28**(11), 1768–1783 (2006)
32. Grady, L., Funka-Lea, G.: Multi-label image segmentation for medical applications based on graph-theoretic electrical potentials. In: Computer Vision and Mathematical Methods in Medical and Biomedical Image Analysis, pp. 230–245. (2004)
33. Grady, L., Polimeni, J.: Discrete Calculus: Applied Analysis on Graphs for Computational Science. Springer Publishing Company, Incorporated, New York (2010)
34. Grady, L., Schwartz, E.: Isoperimetric graph partitioning for image segmentation. Pattern Anal. Mach. Intell. IEEE Trans. **28**(3), 469–475 (2006)
35. Guigues, L., Cocquerez, J., Le Men, H.: Scale-sets image analysis. Int. J. Comput. Vis. **68**(3), 289–317 (2006)
36. Horowitz, S., Pavlidis, T.: Picture segmentation by a directed split-and-merge procedure. In: Proceedings of the Second International Joint Conference on Pattern Recognition, vol. 424, p. 433 (1974)
37. Iri, M.: Survey of Mathematical Programming. North-Holland, Amsterdam (1979)
38. Kakutani, S.: Markov processes and the Dirichlet problem. In: Proceedings of the Japan Academy, vol. 21, pp. 227–233 (1945)
39. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models. Int. J. Comput. Vis. **1**, 321–331 (1988)
40. Khalimsky, E., Kopperman, R., Meyer, P.: Computer graphics and connected topologies on finite ordered sets. Topol. Appl. **36**(1), 1–17 (1990)
41. Kruskal, J.J.: On the shortest spanning subtree of a graph and the travelling salesman problem. Proc. AMS **7**(1) (1956)
42. Levinshtein, A., Stere, A., Kutulakos, K., Fleet, D., Dickinson, S., Siddiqi, K.: Turbopixels: Fast superpixels using geometric flows. Pattern Anal. Mach. Intell. IEEE Trans. **31**(12), 2290–2297 (2009)
43. Malladi, R., Sethian, J., Vemuri, B.: Shape modelling with front propagation: A level set approach. IEEE Trans. Pattern Anal. Mach. Intell. **17**(2), 158–175 (1995)

44. Marr, D., Hildreth, E.: Theory of edge detection. Proc. R. Soc. Lond. Ser. B Biol. Sci. **207**, 187–217 (1980)
45. Menzies, S.W., Crotty, K.A., Ingvar, C., McCarthy, W.H.: An Atlas of Surface Microscopy of Pigmented Skin Lesions. McGraw-Hill, Roseville, Australia (1996). ISBN 0 07 470206 8
46. Meyer, F.: Topographic distance and watershed lines. Signal Process. **38**(1), 113–125 (1994)
47. Meyer, F., Beucher, S.: Morphological segmentation. J. Vis. Commun. Image Represent. **1**(1), 21–46 (1990)
48. Mortensen, E., Barrett, W.: Interactive segmentation with intelligent scissors. Graph. Models Image Process. **60**(5), 349–384 (1998)
49. Mumford, D., Shah, J.: Optimal approximations by piecewise smooth functions and associated variational problems. Commun. Pure Appl. Math. **42**(5), 577–685 (1989)
50. Najman, L.: On the equivalence between hierarchical segmentations and ultrametric watersheds. Journal of Mathematical Imaging and Vision, **40**(3), 231–247 (2011)
51. Najman, L., Schmitt, M.: Geodesic saliency of watershed contours and hierarchical segmentation. Pattern Anal. Mach. Intell. IEEE Trans. **18**(12), 1163–1173 (2002)
52. Najman, L., Talbot, H. (eds.): Mathematical Morphology: From theory to applications. ISTE-Wiley, London, UK (2010)
53. Osher, S., Sethian, J.: Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. J. Comput. Phys. **79**(1), 12–49 (1988)
54. Otsu, N.: A threshold selection method from gray-level histograms. Automatica **11**, 285–296 (1975)
55. Paragios, N., Deriche, R.: Geodesic active contours and level sets for the detection and tracking of moving objects. Pattern Anal. Mach. Intell. IEEE Trans. **22**(3), 266–280 (2002)
56. Paragios, N., Deriche, R.: Geodesic active regions and level set methods for supervised texture segmentation. Int. J. Comput. Vis. **46**(3), 223–247 (2002)
57. Pham, D., Xu, C., Prince, J.: Current methods in medical image segmentation1. Biomed. Eng. **2**(1), 315 (2000)
58. Pock, T., Cremers, D., Bischof, H., Chambolle, A.: An algorithm for minimizing the Mumford-Shah functional. In: 12th International Conference on Computer Vision, pp. 1133–1140. IEEE (2009)
59. Prim, R.: Shortest connection networks and some generalizations. Bell Syst. Techn. J. **36**(6), 1389–1401 (1957)
60. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Phys. D **60**(1-4), 259–268 (1992). DOI http://dx.doi.org/10.1016/0167-2789(92)90242-F
61. Sagiv, C., Sochen, N., Zeevi, Y.: Integrated active contours for texture segmentation. Image Process. IEEE Trans. **15**(6), 1633–1646 (2006)
62. Sethian, J.: Level set methods and fast marching methods. Cambridge University Press, Cambridge (1999). ISBN 0-521-64204-3
63. Soille, P.: Constrained connectivity for hierarchical image partitioning and simplification. IEEE Trans. Pattern Anal. Mach. Intell. **30**(7), 1132–1145 (2008)
64. Stawiaski, J., Decencière, E., Bidault, F.: Computing approximate geodesics and minimal surfaces using watershed and graph cuts. In: Banon, G.J.F., Barrera, J., Braga-Neto, U.d.M., Hirata, N.S.T. (eds.) Proceedings of the 8th International Symposium on Mathematical Morphology, vol. 1, pp. 349–360. Instituto Nacional de Pesquisas Espaciais (INPE) (2007). URL http://urlib.net/dpi.inpe.br/ismm@80/2007/03.19.13.04
65. Strang, G.: Maximal flow through a domain. Math. Program. **26**, 123–143 (1983)
66. Veksler, O.: Efficient graph-based energy minimization. Ph.D. thesis, Cornell University (1999)
67. Vincent, L., Soille, P.: Watersheds in digital spaces: An efficient algorithm based on immersion simulations. IEEE Trans. Pattern Anal. Mach. Intell. **13**(6), 583–598 (1991)
68. Weickert, J., Romeny, B., Viergever, M.: Efficient and reliable schemes for nonlinear diffusion filtering. Image Process. IEEE Trans. **7**(3), 398–410 (2002)
69. Zhu, S., Yuille, A.: Region competition: Unifying snakes, region growing, and Bayes/MDL for multiband image segmentation. Pattern Anal. Mach. Intell. IEEE Trans. **18**(9), 884–900 (2002)