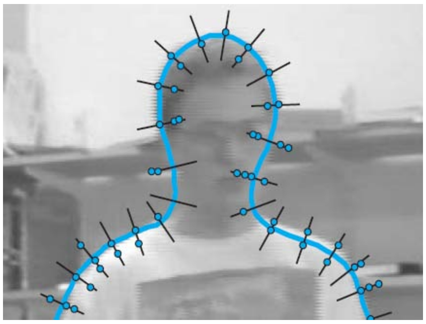
For this week, we are moving from Images as Points to Images as Graphs. Graph-based representations are critical whenever you need to consider the spatial relationship between elements in the image; i.e., operate on the domain in a local manner. The lecture goes through a number of motivational examples followed by mathematical foundations and visual psychophysical foundations. Finally, we discuss graph representations and a pair of algorithms that use it. The slides have more background materials as well.

The reading is to further substantiate the background in graph-based representations and image segmentation problems.

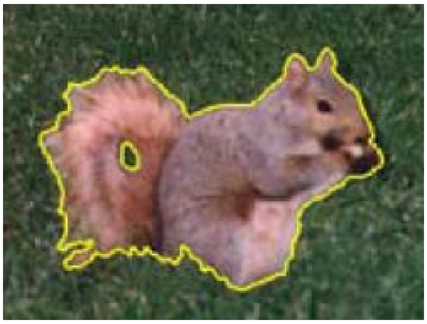
Only a subset of these sections are included.

Chapter 5

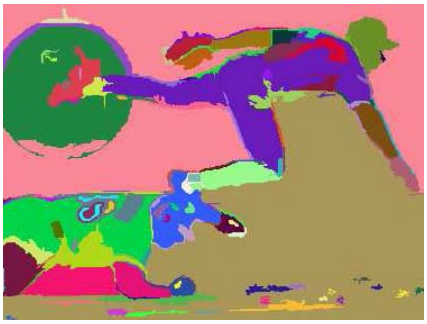
Segmentation



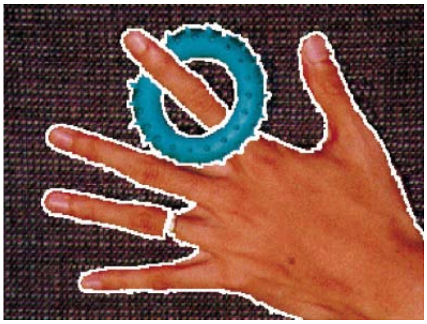
(a)



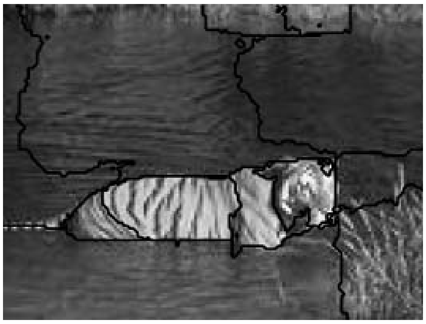
(b)



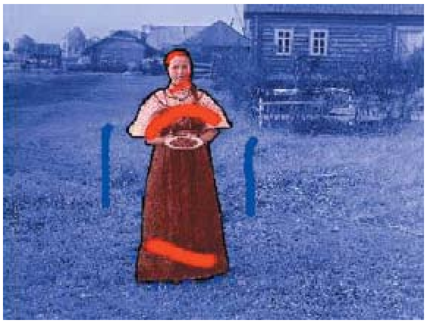
(c)



(d)



(e)



(f)

Figure 5.1 Some popular image segmentation techniques: (a) active contours (Isard and Blake 1998) © 1998 Springer; (b) level sets (Cremers, Rousson, and Deriche 2007) © 2007 Springer; (c) graph-based merging (Felzenszwalb and Huttenlocher 2004b) © 2004 Springer; (d) mean shift (Comaniciu and Meer 2002) © 2002 IEEE; (e) texture and intervening contour-based normalized cuts (Malik, Belongie, Leung *et al.* 2001) © 2001 Springer; (f) binary MRF solved using graph cuts (Boykov and Funka-Lea 2006) © 2006 Springer.

Image segmentation is the task of finding groups of pixels that "go together". In statistics, this problem is known as *cluster analysis* and is a widely studied area with hundreds of different algorithms (Jain and Dubes 1988; Kaufman and Rousseeuw 1990; Jain, Duin, and Mao 2000; Jain, Topchy, Law *et al.* 2004).

In computer vision, image segmentation is one of the oldest and most widely studied problems (Brice and Fennema 1970; Pavlidis 1977; Riseman and Arbib 1977; Ohlander, Price, and Reddy 1978; Rosenfeld and Davis 1979; Haralick and Shapiro 1985). Early techniques tend to use region splitting or merging (Brice and Fennema 1970; Horowitz and Pavlidis 1976; Ohlander, Price, and Reddy 1978; Pavlidis and Liow 1990), which correspond to *divisive* and *agglomerative* algorithms in the clustering literature (Jain, Topchy, Law *et al.* 2004). More recent algorithms often optimize some global criterion, such as intra-region consistency and inter-region boundary lengths or dissimilarity (Leclerc 1989; Mumford and Shah 1989; Shi and Malik 2000; Comaniciu and Meer 2002; Felzenszwalb and Huttenlocher 2004b; Cremers, Rousson, and Deriche 2007).

We have already seen examples of image segmentation in Sections 3.3.2 and 3.7.2. In this chapter, we review some additional techniques that have been developed for image segmentation. These include algorithms based on active contours (Section 5.1) and level sets (Section 5.1.4), region splitting and merging (Section 5.2), *mean shift* (mode finding) (Section 5.3), *normalized cuts* (splitting based on pixel similarity metrics) (Section 5.4), and binary Markov random fields solved using graph cuts (Section 5.5). Figure 5.1 shows some examples of these techniques applied to different images.

Since the literature on image segmentation is so vast, a good way to get a handle on some of the better performing algorithms is to look at experimental comparisons on human-labeled databases (Arbeláez, Maire, Fowlkes *et al.* 2010). The best known of these is the Berkeley Segmentation Dataset and Benchmark[1] (Martin, Fowlkes, Tal *et al.* 2001), which consists of 1000 images from a Corel image dataset that were hand-labeled by 30 human subjects. Many of the more recent image segmentation algorithms report comparative results on this database. For example, Unnikrishnan, Pantofaru, and Hebert (2007) propose new metrics for comparing such algorithms. Estrada and Jepson (2009) compare four well-known segmentation algorithms on the Berkeley data set and conclude that while their own SE-MinCut algorithm (Estrada, Jepson, and Chennubhotla 2004) algorithm outperforms the others by a small margin, there still exists a wide gap between automated and human segmentation performance.[2] A new database of foreground and background segmentations, used by Alpert, Galun, Basri *et al.* (2007), is also available.[3]

[1] http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

[2] An interesting observation about their ROC plots is that automated techniques cluster tightly along similar curves, but human performance is all over the map.

[3] http://www.wisdom.weizmann.ac.il/~vision/Seg_Evaluation_DB/index.html

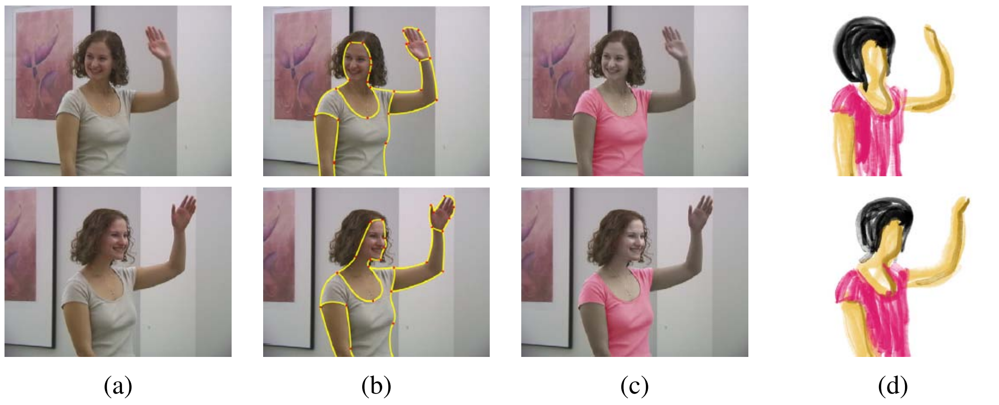


Figure 5.12 Keyframe-based rotoscoping (Agarwala, Hertzmann, Seitz *et al.* 2004) © 2004 ACM: (a) original frames; (b) rotoscoped contours; (c) re-colored blouse; (d) rotoscoped hand-drawn animation.

Additional applications of rotoscoping (object contour detection and segmentation), such as cutting and pasting objects from one photograph into another, are presented in Section 10.4.

5.2 Split and merge

As mentioned in the introduction to this chapter, the simplest possible technique for segmenting a grayscale image is to select a threshold and then compute connected components (Section 3.3.2). Unfortunately, a single threshold is rarely sufficient for the whole image because of lighting and intra-object statistical variations.

In this section, we describe a number of algorithms that proceed either by recursively splitting the whole image into pieces based on region statistics or, conversely, merging pixels and regions together in a hierarchical fashion. It is also possible to combine both splitting and merging by starting with a medium-grain segmentation (in a quadtree representation) and then allowing both merging and splitting operations (Horowitz and Pavlidis 1976; Pavlidis and Liow 1990).

5.2.1 Watershed

A technique related to thresholding, since it operates on a grayscale image, is *watershed* computation (Vincent and Soille 1991). This technique segments an image into several *catchment basins*, which are the regions of an image (interpreted as a height field or landscape) where

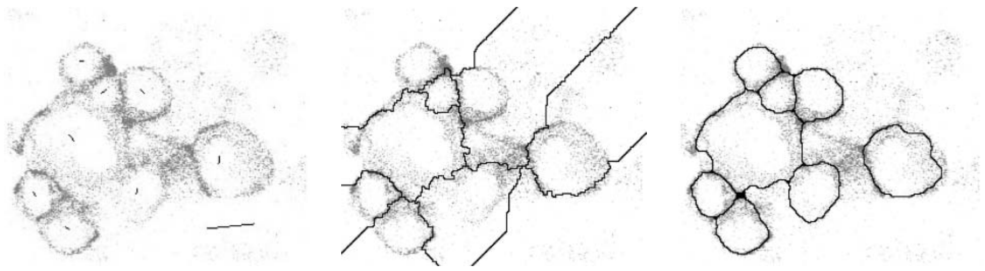


Figure 5.13 Locally constrained watershed segmentation (Beare 2006) © 2006 IEEE: (a) original confocal microscopy image with marked seeds (line segments); (b) standard watershed segmentation; (c) locally constrained watershed segmentation.

rain would flow into the same lake. An efficient way to compute such regions is to start flooding the landscape at all of the local minima and to label ridges wherever differently evolving components meet. The whole algorithm can be implemented using a priority queue of pixels and breadth-first search (Vincent and Soille 1991).[8]

Since images rarely have dark regions separated by lighter ridges, watershed segmentation is usually applied to a smoothed version of the gradient magnitude image, which also makes it usable with color images. As an alternative, the maximum oriented energy in a steerable filter (3.28–3.29) (Freeman and Adelson 1991) can be used as the basis of the *oriented watershed transform* developed by Arbeláez, Maire, Fowlkes *et al.* (2010). Such techniques end up finding smooth regions separated by visible (higher gradient) boundaries. Since such boundaries are what active contours usually follow, active contour algorithms (Mortensen and Barrett 1999; Li, Sun, Tang *et al.* 2004) often precompute such a segmentation using either the watershed or the related *tobogganing* technique (Section 5.1.3).

Unfortunately, watershed segmentation associates a unique region with each local minimum, which can lead to over-segmentation. Watershed segmentation is therefore often used as part of an interactive system, where the user first marks seed locations (with a click or a short stroke) that correspond to the centers of different desired components. Figure 5.13 shows the results of running the watershed algorithm with some manually placed markers on a confocal microscopy image. It also shows the result for an improved version of watershed that uses local morphology to smooth out and optimize the boundaries separating the regions (Beare 2006).

[8] A related algorithm can be used to compute maximally stable extremal regions (MSERs) efficiently (Section 4.1.1) (Nistér and Stewénius 2008).

5.2.2 Region splitting (divisive clustering)

Splitting the image into successively finer regions is one of the oldest techniques in computer vision. Ohlander, Price, and Reddy (1978) present such a technique, which first computes a histogram for the whole image and then finds a threshold that best separates the large peaks in the histogram. This process is repeated until regions are either fairly uniform or below a certain size.

More recent splitting algorithms often optimize some metric of intra-region similarity and inter-region dissimilarity. These are covered in Sections 5.4 and 5.5.

5.2.3 Region merging (agglomerative clustering)

Region merging techniques also date back to the beginnings of computer vision. Brice and Fennema (1970) use a dual grid for representing boundaries between pixels and merge regions based on their relative boundary lengths and the strength of the visible edges at these boundaries.

In data clustering, algorithms can link clusters together based on the distance between their closest points (single-link clustering), their farthest points (complete-link clustering), or something in between (Jain, Topchy, Law *et al.* 2004). Kamvar, Klein, and Manning (2002) provide a probabilistic interpretation of these algorithms and show how additional models can be incorporated within this framework.

A very simple version of pixel-based merging combines adjacent regions whose average color difference is below a threshold or whose regions are too small. Segmenting the image into such *superpixels* (Mori, Ren, Efros *et al.* 2004), which are not semantically meaningful, can be a useful pre-processing stage to make higher-level algorithms such as stereo matching (Zitnick, Kang, Uyttendaele *et al.* 2004; Taguchi, Wilburn, and Zitnick 2008), optic flow (Zitnick, Jojic, and Kang 2005; Brox, Bregler, and Malik 2009), and recognition (Mori, Ren, Efros *et al.* 2004; Mori 2005; Gu, Lim, Arbelaez *et al.* 2009; Lim, Arbeláez, Gu *et al.* 2009) both faster and more robust.

5.2.4 Graph-based segmentation

While many merging algorithms simply apply a fixed rule that groups pixels and regions together, Felzenszwalb and Huttenlocher (2004b) present a merging algorithm that uses *relative dissimilarities* between regions to determine which ones should be merged; it produces an algorithm that provably optimizes a global grouping metric. They start with a pixel-to-pixel dissimilarity measure $w(e)$ that measures, for example, intensity differences between $\mathcal{N}_8$ neighbors. (Alternatively, they can use the *joint feature space* distances (5.42) introduced by Comaniciu and Meer (2002), which we discuss in Section 5.3.2.)

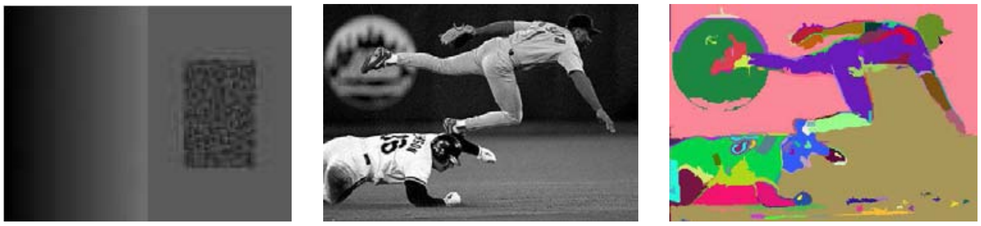

(a) (b) (c)

Figure 5.14 Graph-based merging segmentation (Felzenszwalb and Huttenlocher 2004b) © 2004 Springer: (a) input grayscale image that is successfully segmented into three regions even though the variation inside the smaller rectangle is larger than the variation across the middle edge; (b) input grayscale image; (c) resulting segmentation using an $\mathcal{N}_8$ pixel neighborhood.

For any region R, its *internal difference* is defined as the largest edge weight in the region's minimum spanning tree,

$$Int(R) = \min_{e \in MST(R)} w(e). \tag{5.20}$$

For any two adjacent regions with at least one edge connecting their vertices, the difference between these regions is defined as the minimum weight edge connecting the two regions,

$$Dif(R_1, R_2) = \min_{e=(v_1,v_2)|v_1 \in R_1, v_2 \in R_2} w(e). \tag{5.21}$$

Their algorithm merges any two adjacent regions whose difference is smaller than the minimum internal difference of these two regions,

$$MInt(R_1, R_2) = \min(Int(R_1) + \tau(R_1), Int(R_2) + \tau(R_2)), \tag{5.22}$$

where $\tau(R)$ is a heuristic region penalty that Felzenszwalb and Huttenlocher (2004b) set to $k/|R|$, but which can be set to any application-specific measure of region goodness.

By merging regions in decreasing order of the edges separating them (which can be efficiently evaluated using a variant of Kruskal's minimum spanning tree algorithm), they provably produce segmentations that are neither too fine (there exist regions that could have been merged) nor too coarse (there are regions that could be split without being mergeable). For fixed-size pixel neighborhoods, the running time for this algorithm is $O(N \log N)$, where N is the number of image pixels, which makes it one of the fastest segmentation algorithms (Paris and Durand 2007). Figure 5.14 shows two examples of images segmented using their technique.

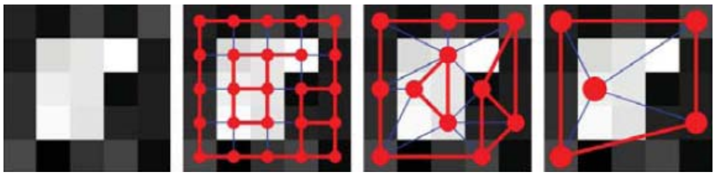


Figure 5.15 Coarse to fine node aggregation in segmentation by weighted aggregation (SWA) (Sharon, Galun, Sharon *et al.* 2006) © 2006 Macmillan Publishers Ltd [Nature]: (a) original gray-level pixel grid; (b) inter-pixel couplings, where thicker lines indicate stronger couplings; (c) after one level of coarsening, where each original pixel is strongly coupled to one of the coarse-level nodes; (d) after two levels of coarsening.

5.2.5 Probabilistic aggregation

Alpert, Galun, Basri *et al.* (2007) develop a probabilistic merging algorithm based on two cues, namely gray-level similarity and texture similarity. The gray-level similarity between regions R_i and R_j is based on the *minimal external difference* from other neighboring regions,

$$\sigma^+_{local} = \min(\Delta^+_i, \Delta^+_j), \tag{5.23}$$

where $\Delta^+_i = \min_k |\Delta_{ik}|$ and Δ_{ik} is the difference in average intensities between regions R_i and R_k. This is compared to the *average intensity difference*,

$$\sigma^-_{local} = \frac{\Delta^-_i + \Delta^-_j}{2}, \tag{5.24}$$

where $\Delta^-_i = \sum_k(\tau_{ik}\Delta_{ik})/\sum_k(\tau_{ik})$ and τ_{ik} is the boundary length between regions R_i and R_k. The texture similarity is defined using relative differences between histogram bins of simple oriented Sobel filter responses. The pairwise statistics σ^+_{local} and σ^-_{local} are used to compute the likelihoods p_{ij} that two regions should be merged. (See the paper by Alpert, Galun, Basri *et al.* (2007) for more details.)

Merging proceeds in a hierarchical fashion inspired by algebraic multigrid techniques (Brandt 1986; Briggs, Henson, and McCormick 2000) and previously used by Alpert, Galun, Basri *et al.* (2007) in their segmentation by weighted aggregation (SWA) algorithm (Sharon, Galun, Sharon *et al.* 2006), which we discuss in Section 5.4. A subset of the nodes $C \subset V$ that are (collectively) *strongly coupled* to all of the original nodes (regions) are used to define the problem at a coarser scale (Figure 5.15), where strong coupling is defined as

$$\frac{\sum_{j\in C} p_{ij}}{\sum_{j\in V} p_{ij}} > \phi, \tag{5.25}$$

with ϕ usually set to 0.2. The intensity and texture similarity statistics for the coarser nodes are recursively computed using weighted averaging, where the relative strengths (couplings) between coarse- and fine-level nodes are based on their merge probabilities p_{ij}. This allows the algorithm to run in essentially $O(N)$ time, using the same kind of hierarchical aggregation operations that are used in pyramid-based filtering or preconditioning algorithms. After a segmentation has been identified at a coarser level, the exact memberships of each pixel are computed by propagating coarse-level assignments to their finer-level "children" (Sharon, Galun, Sharon *et al.* 2006; Alpert, Galun, Basri *et al.* 2007). Figure 5.22 shows the segmentations produced by this algorithm compared to other popular segmentation algorithms.

5.3 Mean shift and mode finding

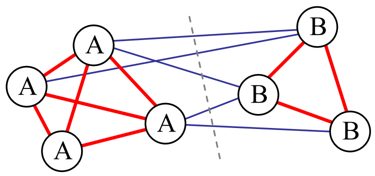
Mean-shift and mode finding techniques, such as k-means and mixtures of Gaussians, model the feature vectors associated with each pixel (e.g., color and position) as samples from an unknown probability density function and then try to find clusters (modes) in this distribution.

Consider the color image shown in Figure 5.16a. How would you segment this image based on color alone? Figure 5.16b shows the distribution of pixels in L*u*v* space, which is equivalent to what a vision algorithm that ignores spatial location would see. To make the visualization simpler, let us only consider the L*u* coordinates, as shown in Figure 5.16c. How many obvious (elongated) clusters do you see? How would you go about finding these clusters?

The k-means and mixtures of Gaussians techniques use a *parametric* model of the density function to answer this question, i.e., they assume the density is the superposition of a small number of simpler distributions (e.g., Gaussians) whose locations (centers) and shape (covariance) can be estimated. Mean shift, on the other hand, smoothes the distribution and finds its peaks as well as the regions of feature space that correspond to each peak. Since a complete density is being modeled, this approach is called *non-parametric* (Bishop 2006). Let us look at these techniques in more detail.

5.3.1 K-means and mixtures of Gaussians

While k-means implicitly models the probability density as a superposition of spherically symmetric distributions, it does not require any probabilistic reasoning or modeling (Bishop 2006). Instead, the algorithm is given the number of clusters k it is supposed to find; it then iteratively updates the cluster center location based on the samples that are closest to each center. The algorithm can be initialized by randomly sampling k centers from the input feature vectors. Techniques have also been developed for splitting or merging cluster centers



(a)

	A	B	sum
A	$assoc(A,A)$	$cut(A,B)$	$assoc(A,V)$
B	$cut(B,A)$	$assoc(B,B)$	$assoc(B,V)$
sum	$assoc(A,V)$	$assoc(B,v)$	

(b)

Figure 5.19 Sample weighted graph and its normalized cut: (a) a small sample graph and its smallest normalized cut; (b) tabular form of the associations and cuts for this graph. The $assoc$ and cut entries are computed as area sums of the associated weight matrix $\boldsymbol{W}$ (Figure 5.20). Normalizing the table entries by the row or column sums produces normalized associations and cuts $Nassoc$ and $Ncut$.

5.4 Normalized cuts

While bottom-up merging techniques aggregate regions into coherent wholes and mean-shift techniques try to find clusters of similar pixels using mode finding, the normalized cuts technique introduced by Shi and Malik (2000) examines the *affinities* (similarities) between nearby pixels and tries to separate groups that are connected by weak affinities.

Consider the simple graph shown in Figure 5.19a. The pixels in group A are all strongly connected with high affinities, shown as thick red lines, as are the pixels in group B. The connections between these two groups, shown as thinner blue lines, are much weaker. A *normalized cut* between the two groups, shown as a dashed line, separates them into two clusters.

The cut between two groups A and B is defined as the sum of all the weights being cut,

$$cut(A,B) = \sum_{i \in A, j \in B} w_{ij}, \tag{5.43}$$

where the weights between two pixels (or regions) i and j measure their similarity. Using a minimum cut as a segmentation criterion, however, does not result in reasonable clusters, since the smallest cuts usually involve isolating a single pixel.

A better measure of segmentation is the normalized cut, which is defined as

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}, \tag{5.44}$$

where $assoc(A,A) = \sum_{i \in A, j \in A} w_{ij}$ is the *association* (sum of all the weights) within a cluster and $assoc(A,V) = assoc(A,A) + cut(A,B)$ is the sum of *all* the weights associated with nodes in A. Figure 5.19b shows how the cuts and associations can be thought of as area sums in the weight matrix $\boldsymbol{W} = [w_{ij}]$, where the entries of the matrix have been arranged so that the nodes in A come first and the nodes in B come second. Figure 5.20 shows an actual weight matrix for which these area sums can be computed. Dividing each of these areas by the corresponding row sum (the rightmost column of Figure 5.19b) results in the normalized cut and association values. These normalized values better reflect the fitness of a particular segmentation, since they look for collections of edges that are weak relative to all of the edges both inside and emanating from a particular region.

Unfortunately, computing the optimal normalized cut is NP-complete. Instead, Shi and Malik (2000) suggest computing a real-valued assignment of nodes to groups. Let $\boldsymbol{x}$ be the *indicator vector* where $x_i = +1$ iff $i \in A$ and $x_i = -1$ iff $i \in B$. Let $\boldsymbol{d} = \boldsymbol{W}\boldsymbol{1}$ be the row sums of the symmetric matrix $\boldsymbol{W}$ and $\boldsymbol{D} = \operatorname{diag}(\boldsymbol{d})$ be the corresponding diagonal matrix. Shi and Malik (2000) show that minimizing the normalized cut over all possible indicator vectors $\boldsymbol{x}$ is equivalent to minimizing

$$\min_{\boldsymbol{y}} \frac{\boldsymbol{y}^T(\boldsymbol{D}-\boldsymbol{W})\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{D}\boldsymbol{y}}, \tag{5.45}$$

where $\boldsymbol{y} = ((\boldsymbol{1}+\boldsymbol{x}) - b(\boldsymbol{1}-\boldsymbol{x}))/2$ is a vector consisting of all 1s and $-b$s such that $\boldsymbol{y} \cdot \boldsymbol{d} = 0$. Minimizing this *Rayleigh quotient* is equivalent to solving the generalized eigenvalue system

$$(\boldsymbol{D}-\boldsymbol{W})\boldsymbol{y} = \lambda \boldsymbol{D}\boldsymbol{y}, \tag{5.46}$$

which can be turned into a regular eigenvalue problem

$$(\boldsymbol{I}-\boldsymbol{N})\boldsymbol{z} = \lambda \boldsymbol{z}, \tag{5.47}$$

where $\boldsymbol{N} = \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$ is the *normalized* affinity matrix (Weiss 1999) and $\boldsymbol{z} = \boldsymbol{D}^{1/2}\boldsymbol{y}$. Because these eigenvectors can be interpreted as the large modes of vibration in a spring-mass system, normalized cuts is an example of a *spectral method* for image segmentation.

Extending an idea originally proposed by Scott and Longuet-Higgins (1990), Weiss (1999) suggests normalizing the affinity matrix and then using the top k eigenvectors to reconstitute a $\boldsymbol{Q}$ matrix. Other papers have extended the basic normalized cuts framework by modifying the affinity matrix in different ways, finding better discrete solutions to the minimization problem, or applying multi-scale techniques (Meilă and Shi 2000, 2001; Ng, Jordan, and Weiss 2001; Yu and Shi 2003; Cour, Bénézit, and Shi 2005; Tolliver and Miller 2006).

Figure 5.20b shows the second smallest (real-valued) eigenvector corresponding to the weight matrix shown in Figure 5.20a. (Here, the rows have been permuted to separate the two groups of variables that belong to the different components of this eigenvector.) After this real-valued vector is computed, the variables corresponding to positive and negative

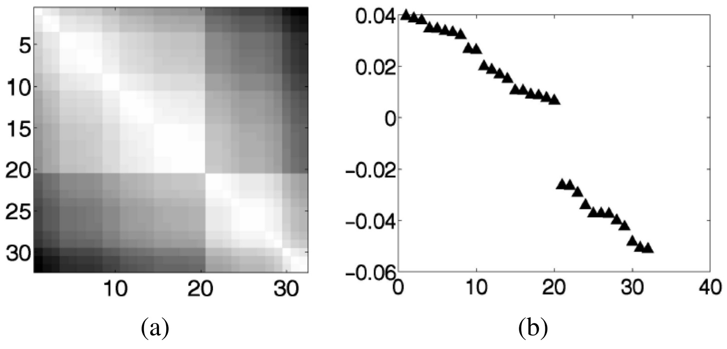


Figure 5.20 Sample weight table and its second smallest eigenvector (Shi and Malik 2000) © 2000 IEEE: (a) sample 32×32 weight matrix $\boldsymbol{W}$; (b) eigenvector corresponding to the second smallest eigenvalue of the generalized eigenvalue problem $(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{y} = \lambda \boldsymbol{D}\boldsymbol{y}$.

eigenvector values are associated with the two cut components. This process can be further repeated to hierarchically subdivide an image, as shown in Figure 5.21.

The original algorithm proposed by Shi and Malik (2000) used spatial position and image feature differences to compute the pixel-wise affinities,

$$w_{ij} = \exp\left(-\frac{\|\boldsymbol{F}_i - \boldsymbol{F}_j\|^2}{\sigma_F^2} - \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{\sigma_s^2}\right), \tag{5.48}$$

for pixels within a radius $\|\boldsymbol{x}_i - \boldsymbol{x}_j\| < r$, where $\boldsymbol{F}$ is a feature vector that consists of intensities, colors, or oriented filter histograms. (Note how (5.48) is the negative exponential of the joint feature space distance (5.42).)

In subsequent work, Malik, Belongie, Leung *et al.* (2001) look for *intervening contours* between pixels i and j and define an intervening contour weight

$$w_{ij}^{IC} = 1 - \max_{\boldsymbol{x} \in l_{ij}} p_{con}(\boldsymbol{x}), \tag{5.49}$$

where l_{ij} is the image line joining pixels i and j and $p_{con}(\boldsymbol{x})$ is the probability of an intervening contour perpendicular to this line, which is defined as the negative exponential of the oriented energy in the perpendicular direction. They multiply these weights with a texton-based texture similarity metric and use an initial over-segmentation based purely on local pixel-wise features to re-estimate intervening contours and texture statistics in a region-based manner. Figure 5.22 shows the results of running this improved algorithm on a number of test images.

Because it requires the solution of large sparse eigenvalue problems, normalized cuts can be quite slow. Sharon, Galun, Sharon *et al.* (2006) present a way to accelerate the computation of the normalized cuts using an approach inspired by algebraic multigrid (Brandt

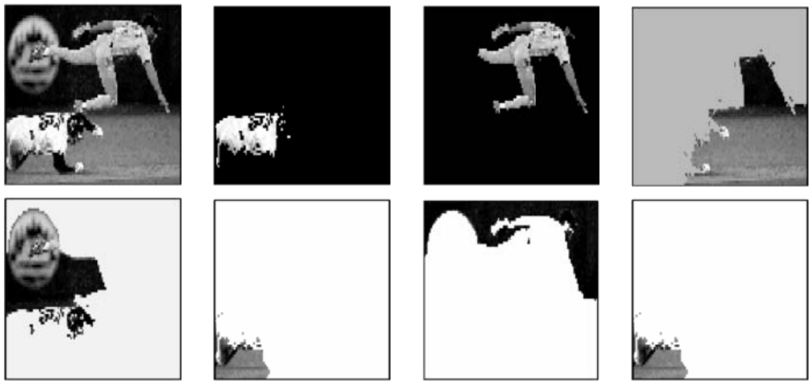


Figure 5.21 Normalized cuts segmentation (Shi and Malik 2000) © 2000 IEEE: The input image and the components returned by the normalized cuts algorithm.

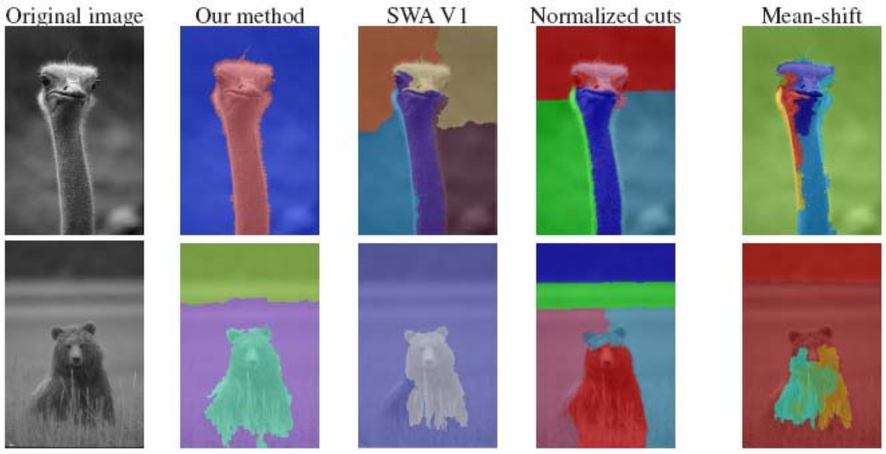


Figure 5.22 Comparative segmentation results (Alpert, Galun, Basri *et al.* 2007) © 2007 IEEE. "Our method" refers to the probabilistic bottom-up merging algorithm developed by Alpert *et al*.

1986; Briggs, Henson, and McCormick 2000). To coarsen the original problem, they select a smaller number of variables such that the remaining fine-level variables are *strongly coupled* to at least one coarse-level variable. Figure 5.15 shows this process schematically, while (5.25) gives the definition for strong coupling except that, in this case, the original weights w_{ij} in the normalized cut are used instead of merge probabilities p_{ij}.

Once a set of coarse variables has been selected, an inter-level interpolation matrix with elements similar to the left hand side of (5.25) is used to define a reduced version of the normalized cuts problem. In addition to computing the weight matrix using interpolation-based coarsening, additional region statistics are used to modulate the weights. After a normalized cut has been computed at the coarsest level of analysis, the membership values of finer-level nodes are computed by interpolating parent values and mapping values within $\epsilon = 0.1$ of 0 and 1 to pure Boolean values.

An example of the segmentation produced by weighted aggregation (SWA) is shown in Figure 5.22, along with the most recent probabilistic bottom-up merging algorithm by Alpert, Galun, Basri *et al.* (2007), which was described in Section 5.2. In even more recent work, Wang and Oliensis (2010) show how to estimate statistics over segmentations (e.g., mean region size) directly from the affinity graph. They use this to produce segmentations that are more *central* with respect to other possible segmentations.

5.5 Graph cuts and energy-based methods

A common theme in image segmentation algorithms is the desire to group pixels that have similar appearance (statistics) and to have the boundaries between pixels in different regions be of short length and across visible discontinuities. If we restrict the boundary measurements to be between immediate neighbors and compute region membership statistics by summing over pixels, we can formulate this as a classic pixel-based energy function using either a *variational formulation* (regularization, see Section 3.7.1) or as a binary Markov random field (Section 3.7.2).

Examples of the continuous approach include (Mumford and Shah 1989; Chan and Vese 1992; Zhu and Yuille 1996; Tabb and Ahuja 1997) along with the level set approaches discussed in Section 5.1.4. An early example of a discrete labeling problem that combines both region-based and boundary-based energy terms is the work of Leclerc (1989), who used minimum description length (MDL) coding to derive the energy function being minimized. Boykov and Funka-Lea (2006) present a wonderful survey of various energy-based techniques for binary object segmentation, some of which we discuss below.

As we saw in Section 3.7.2, the energy corresponding to a segmentation problem can be written (c.f. Equations (3.100) and (3.108–3.113)) as

$$E(f) = \sum_{i,j} E_r(i,j) + E_b(i,j), \tag{5.50}$$

where the region term

$$E_r(i,j) = E_S(I(i,j); R(f(i,j))) \tag{5.51}$$

is the negative log likelihood that pixel intensity (or color) $I(i,j)$ is consistent with the statistics of region $R(f(i,j))$ and the boundary term

$$E_b(i,j) = s_x(i,j)\delta(f(i,j) - f(i+1,j)) + s_y(i,j)\delta(f(i,j) - f(i,j+1)) \tag{5.52}$$

measures the inconsistency between $\mathcal{N}_4$ neighbors modulated by local horizontal and vertical smoothness terms $s_x(i,j)$ and $s_y(i,j)$.

Region statistics can be something as simple as the mean gray level or color (Leclerc 1989), in which case

$$E_S(I;\mu_k) = \|I - \mu_k\|^2. \tag{5.53}$$

Alternatively, they can be more complex, such as region intensity histograms (Boykov and Jolly 2001) or color Gaussian mixture models (Rother, Kolmogorov, and Blake 2004). For smoothness (boundary) terms, it is common to make the strength of the smoothness $s_x(i,j)$ inversely proportional to the local edge strength (Boykov, Veksler, and Zabih 2001).

Originally, energy-based segmentation problems were optimized using iterative gradient descent techniques, which were slow and prone to getting trapped in local minima. Boykov and Jolly (2001) were the first to apply the binary MRF optimization algorithm developed by Greig, Porteous, and Seheult (1989) to binary object segmentation.

In this approach, the user first delineates pixels in the background and foreground regions using a few strokes of an image brush (Figure 3.61). These pixels then become the *seeds* that tie nodes in the *S–T graph* to the source and sink labels S and T (Figure 5.23a). Seed pixels can also be used to estimate foreground and background region statistics (intensity or color histograms).

The capacities of the other edges in the graph are derived from the region and boundary energy terms, i.e., pixels that are more compatible with the foreground or background region get stronger connections to the respective source or sink; adjacent pixels with greater smoothness also get stronger links. Once the minimum-cut/maximum-flow problem has been solved using a polynomial time algorithm (Goldberg and Tarjan 1988; Boykov and Kolmogorov 2004), pixels on either side of the computed cut are labeled according to the source or sink to which they remain connected (Figure 5.23b). While graph cuts is just one of several known techniques for MRF energy minimization (Appendix B.5.4), it is still the one most commonly used for solving binary MRF problems.

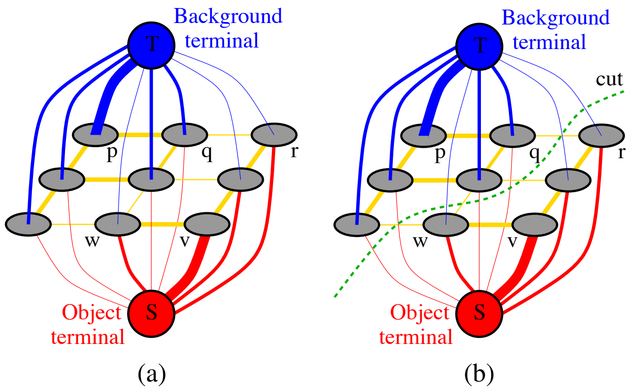


Figure 5.23 Graph cuts for region segmentation (Boykov and Jolly 2001) © 2001 IEEE: (a) the energy function is encoded as a maximum flow problem; (b) the minimum cut determines the region boundary.

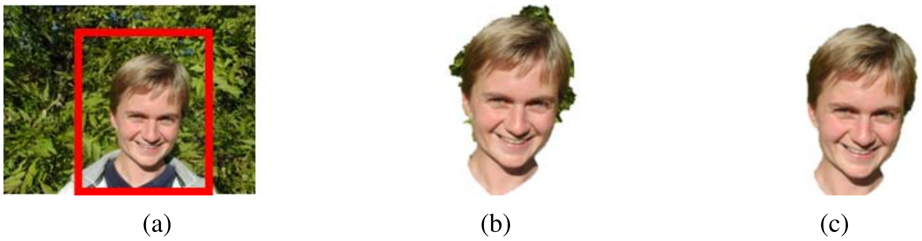


Figure 5.24 GrabCut image segmentation (Rother, Kolmogorov, and Blake 2004) © 2004 ACM: (a) the user draws a bounding box in red; (b) the algorithm guesses color distributions for the object and background and performs a binary segmentation; (c) the process is repeated with better region statistics.

The basic binary segmentation algorithm of Boykov and Jolly (2001) has been extended in a number of directions. The *GrabCut* system of Rother, Kolmogorov, and Blake (2004) iteratively re-estimates the region statistics, which are modeled as a mixtures of Gaussians in color space. This allows their system to operate given minimal user input, such as a single bounding box (Figure 5.24a)—the background color model is initialized from a strip of pixels around the box outline. (The foreground color model is initialized from the interior pixels, but quickly converges to a better estimate of the object.) The user can also place additional strokes to refine the segmentation as the solution progresses. In more recent work, Cui, Yang, Wen *et al.* (2008) use color and edge models derived from previous segmentations of similar objects to improve the local models used in GrabCut.

Another major extension to the original binary segmentation formulation is the addition of

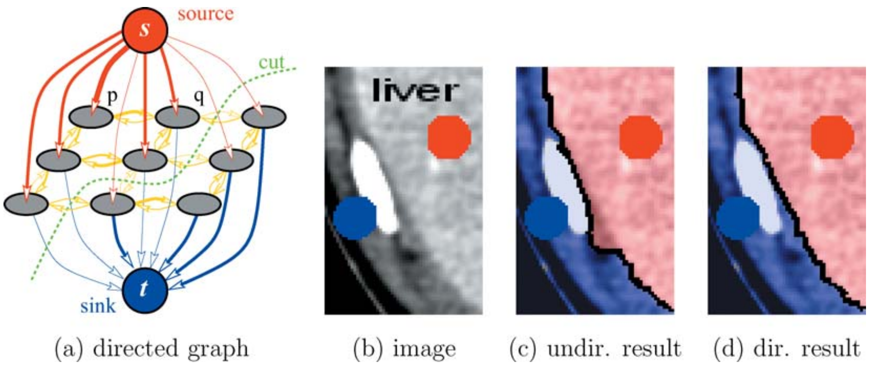


Figure 5.25 Segmentation with a directed graph cut (Boykov and Funka-Lea 2006) © 2006 Springer: (a) directed graph; (b) image with seed points; (c) the undirected graph incorrectly continues the boundary along the bright object; (d) the directed graph correctly segments the light gray region from its darker surround.

directed edges, which allows boundary regions to be oriented, e.g., to prefer light to dark transitions or *vice versa* (Kolmogorov and Boykov 2005). Figure 5.25 shows an example where the directed graph cut correctly segments the light gray liver from its dark gray surround. The same approach can be used to measure the *flux* exiting a region, i.e., the signed gradient projected normal to the region boundary. Combining oriented graphs with larger neighborhoods enables approximating continuous problems such as those traditionally solved using level sets in the globally optimal graph cut framework (Boykov and Kolmogorov 2003; Kolmogorov and Boykov 2005).

Even more recent developments in graph cut-based segmentation techniques include the addition of connectivity priors to force the foreground to be in a single piece (Vicente, Kolmogorov, and Rother 2008) and shape priors to use knowledge about an object's shape during the segmentation process (Lempitsky and Boykov 2007; Lempitsky, Blake, and Rother 2008).

While optimizing the binary MRF energy (5.50) requires the use of combinatorial optimization techniques, such as maximum flow, an approximate solution can be obtained by converting the binary energy terms into quadratic energy terms defined over a continuous $[0, 1]$ random field, which then becomes a classical membrane-based regularization problem (3.100–3.102). The resulting quadratic energy function can then be solved using standard linear system solvers (3.102–3.103), although if speed is an issue, you should use multigrid or one of its variants (Appendix A.5). Once the continuous solution has been computed, it can be thresholded at 0.5 to yield a binary segmentation.

The $[0, 1]$ continuous optimization problem can also be interpreted as computing the probability at each pixel that a *random walker* starting at that pixel ends up at one of the labeled seed pixels, which is also equivalent to computing the potential in a resistive grid where the resistors are equal to the edge weights (Grady 2006; Sinop and Grady 2007). K-way segmentations can also be computed by iterating through the seed labels, using a binary problem with one label set to 1 and all the others set to 0 to compute the relative membership probabilities for each pixel. In follow-on work, Grady and Ali (2008) use a precomputation of the eigenvectors of the linear system to make the solution with a novel set of seeds faster, which is related to the Laplacian matting problem presented in Section 10.4.3 (Levin, Acha, and Lischinski 2008). Couprie, Grady, Najman *et al.* (2009) relate the random walker to watersheds and other segmentation techniques. Singaraju, Grady, and Vidal (2008) add directed-edge constraints in order to support flux, which makes the energy piecewise quadratic and hence not solvable as a single linear system. The random walker algorithm can also be used to solve the Mumford–Shah segmentation problem (Grady and Alvino 2008) and to compute fast multigrid solutions (Grady 2008). A nice review of these techniques is given by Singaraju, Grady, Sinop *et al.* (2010).

An even faster way to compute a continuous $[0, 1]$ approximate segmentation is to compute *weighted geodesic distances* between the 0 and 1 seed regions (Bai and Sapiro 2009), which can also be used to estimate soft alpha mattes (Section 10.4.3). A related approach by Criminisi, Sharp, and Blake (2008) can be used to find fast approximate solutions to general binary Markov random field optimization problems.

5.5.1 *Application*: Medical image segmentation

One of the most promising applications of image segmentation is in the medical imaging domain, where it can be used to segment anatomical tissues for later quantitative analysis. Figure 5.25 shows a binary graph cut with directed edges being used to segment the liver tissue (light gray) from its surrounding bone (white) and muscle (dark gray) tissue. Figure 5.26 shows the segmentation of bones in a $256 \times 256 \times 119$ computed X-ray tomography (CT) volume. Without the powerful optimization techniques available in today's image segmentation algorithms, such processing used to require much more laborious manual tracing of individual X-ray slices.

The fields of medical image segmentation (McInerney and Terzopoulos 1996) and medical image registration (Kybic and Unser 2003) (Section 8.3.1) are rich research fields with their own specialized conferences, such as *Medical Imaging Computing and Computer Assisted Intervention (MICCAI)*,[11] and journals, such as *Medical Image Analysis* and *IEEE Transactions on Medical Imaging*. These can be great sources of references and ideas for research in this area.

[11] http://www.miccai.org/.

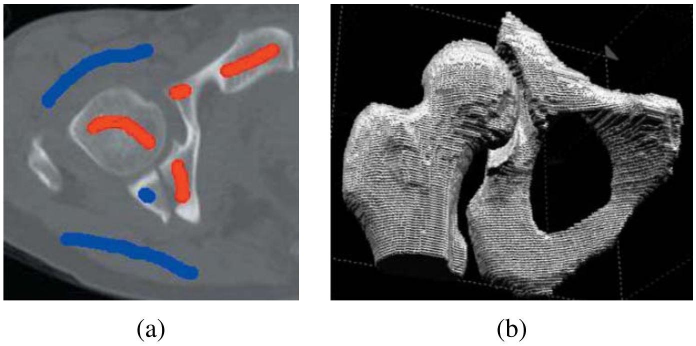


Figure 5.26 3D volumetric medical image segmentation using graph cuts (Boykov and Funka-Lea 2006) © 2006 Springer: (a) computed tomography (CT) slice with some seeds; (b) recovered 3D volumetric bone model (on a $256 \times 256 \times 119$ voxel grid).

5.6 Additional reading

The topic of image segmentation is closely related to clustering techniques, which are treated in a number of monographs and review articles (Jain and Dubes 1988; Kaufman and Rousseeuw 1990; Jain, Duin, and Mao 2000; Jain, Topchy, Law *et al.* 2004). Some early segmentation techniques include those describerd by Brice and Fennema (1970); Pavlidis (1977); Riseman and Arbib (1977); Ohlander, Price, and Reddy (1978); Rosenfeld and Davis (1979); Haralick and Shapiro (1985), while examples of newer techniques are developed by Leclerc (1989); Mumford and Shah (1989); Shi and Malik (2000); Felzenszwalb and Huttenlocher (2004b).

Arbeláez, Maire, Fowlkes *et al.* (2010) provide a good review of automatic segmentation techniques and also compare their performance on the Berkeley Segmentation Dataset and Benchmark (Martin, Fowlkes, Tal *et al.* 2001).[12] Additional comparison papers and databases include those by Unnikrishnan, Pantofaru, and Hebert (2007); Alpert, Galun, Basri *et al.* (2007); Estrada and Jepson (2009).

The topic of active contours has a long history, beginning with the seminal work on snakes and other energy-minimizing variational methods (Kass, Witkin, and Terzopoulos 1988; Cootes, Cooper, Taylor *et al.* 1995; Blake and Isard 1998), continuing through techniques such as intelligent scissors (Mortensen and Barrett 1995, 1999; Pérez, Blake, and Gangnet 2001), and culminating in level sets (Malladi, Sethian, and Vemuri 1995; Caselles, Kimmel, and Sapiro 1997; Sethian 1999; Paragios and Deriche 2000; Sapiro 2001; Osher and Paragios 2003; Paragios, Faugeras, Chan *et al.* 2005; Cremers, Rousson, and Deriche 2007; Rousson and Paragios 2008; Paragios and Sgallari 2009), which are currently the most widely

[12] http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/.

used active contour methods.

Techniques for segmenting images based on local pixel similarities combined with aggregation or splitting methods include watersheds (Vincent and Soille 1991; Beare 2006; Arbeláez, Maire, Fowlkes *et al.* 2010), region splitting (Ohlander, Price, and Reddy 1978), region merging (Brice and Fennema 1970; Pavlidis and Liow 1990; Jain, Topchy, Law *et al.* 2004), as well as graph-based and probabilistic multi-scale approaches (Felzenszwalb and Huttenlocher 2004b; Alpert, Galun, Basri *et al.* 2007).

Mean-shift algorithms, which find modes (peaks) in a density function representation of the pixels, are presented by Comaniciu and Meer (2002); Paris and Durand (2007). Parametric mixtures of Gaussians can also be used to represent and segment such pixel densities (Bishop 2006; Ma, Derksen, Hong *et al.* 2007).

The seminal work on spectral (eigenvalue) methods for image segmentation is the *normalized cut* algorithm of Shi and Malik (2000). Related work includes that by Weiss (1999); Meilă and Shi (2000, 2001); Malik, Belongie, Leung *et al.* (2001); Ng, Jordan, and Weiss (2001); Yu and Shi (2003); Cour, Bénézit, and Shi (2005); Sharon, Galun, Sharon *et al.* (2006); Tolliver and Miller (2006); Wang and Oliensis (2010).

Continuous-energy-based (variational) approaches to interactive segmentation include Leclerc (1989); Mumford and Shah (1989); Chan and Vese (1992); Zhu and Yuille (1996); Tabb and Ahuja (1997). Discrete variants of such problems are usually optimized using binary graph cuts or other combinatorial energy minimization methods (Boykov and Jolly 2001; Boykov and Kolmogorov 2003; Rother, Kolmogorov, and Blake 2004; Kolmogorov and Boykov 2005; Cui, Yang, Wen *et al.* 2008; Vicente, Kolmogorov, and Rother 2008; Lempitsky and Boykov 2007; Lempitsky, Blake, and Rother 2008), although continuous optimization techniques followed by thresholding can also be used (Grady 2006; Grady and Ali 2008; Singaraju, Grady, and Vidal 2008; Criminisi, Sharp, and Blake 2008; Grady 2008; Bai and Sapiro 2009; Couprie, Grady, Najman *et al.* 2009). Boykov and Funka-Lea (2006) present a good survey of various energy-based techniques for binary object segmentation.

5.7 Exercises

Ex 5.1: Snake evolution Prove that, in the absence of external forces, a snake will always shrink to a small circle and eventually a single point, regardless of whether first- or second-order smoothness (or some combination) is used.

(Hint: If you can show that the evolution of the $x(s)$ and $y(s)$ components are independent, you can analyze the 1D case more easily.)

Ex 5.2: Snake tracker Implement a snake-based contour tracker: