I

IMAGE SEGMENTATION BY REGION BASED AND WATERSHED ALGORITHMS

INTRODUCTION

The *segmentation* of an image is defined as its partition into regions, in which the regions satisfy some specified criteria. A commonly used criterion is that *of homogeneity*, that is, the regions should be homogeneous with respect to some property, such as color or texture (1). An alternative criterion is that the regions should correspond to the constituent regions or objects of an image (2). An even stronger requirement could be that the regions have a strong correlation with regions or objects of the real world contained in the image (3). As an example, two possible segmentations (also referred to as *partitions*) of the image in Fig. 1(a) are shown in Fig. 1(c and d), in which the thick, white lines represent the boundaries of the regions. The segmentation in Fig. 1(c) was obtained with the help of user interaction, as shown in Fig. 1(b), in which the user marked the objects. The resulting regions therefore have a strong correlation with the real-world objects in the image. The segmentation in Fig. 1(d) was obtained by an automatic algorithm in which regions were defined based on a color homogeneity criterion. The resulting regions correspond less well to objects in the image. The algorithms used to obtain these segmentations, which are mentioned in the caption of Fig. 1, are described in the section entitled "watershed algorithms in practice."

In general, segmentation is an ill-defined problem, as it is impossible to define a single perfect segmentation for every image. The best segmentation is usually dependent on the application and the information to be obtained from the image. It is also often dependent on the scale at which the image is to be processed. For example, an aerial photograph of a landscape could be divided into regions that represent different land uses, with each forest covered by a single region. Alternatively, if the application is tree counting, then each tree of the forest should correspond to a region.

SEGMENTATION BASICS

After a discussion of the image properties generally used by image segmentation algorithms, the concept and representation of a segmentation partition is reviewed.

Edge and Region-Based Segmentation

Image segmentation algorithms in general are based on one or both properties of discontinuity and homogeneity.

Edge-based segmentation methods approach segmentation by finding the locations of discontinuities in an image, is done by applying an edge detection algorithm (see the entry on Edge Detection in Gray Scale, Color and Range Images). As the edge pixels produced by an edge detection algorithm hardly ever form continuous lines, it is difficult to convert from an edge image to an image partition. Algorithms such as *edge linking* are therefore applied to assemble edge pixels into continuous edges (2).

Region-based segmentation methods, which are described in the next section, build the regions of a partition directly. Whereas edge-based segmentation methods search for discontinuities to find the boundaries between regions, region-based methods use the criterion that regions should be homogeneous. The homogeneity criterion can be used either to add pixels to regions for which the criterion is satisfied (*region growing*) or to split regions that do not satisfy the homogeneity criterion (*region splitting*). A combination of both methods is also possible (*region splitting and merging*).

The *watershed transform*, which is described in future sections, combines aspects of both edge-based and region-based segmentation approaches. It grows regions of pixels around the local minima of an image, and it ensures that the boundaries of adjacent regions lie along the crest lines of the gradient image (4).

Partition Representation

The result of a segmentation algorithm is a *partition* of an image, which consists of several regions, as shown in Fig. 1(c and d). The partition must satisfy at a minimum the following requirements specified by Serra (5):

1. The union of all the partition regions must cover the full area of the image (alternatively stated that every pixel must belong to a region).
2. The regions must not overlap each other.

Other authors have specified more stringent requirements for a partition; for example Zucker (6) requires that the pixels of each region are connected and that a function exists which returns true when applied to a single region and false when applied to the union of two regions.

A good way of visualizing a partition is to superimpose the boundary lines of the partition regions onto the original image, as done in Fig. 1(c and d) (where the boundary lines have also been dilated for better visibility). A partition representation useful for additional processing is a *label image*, in which all pixels that belong to the same region receive the same numeric value. The label image that corresponds to the partition in Fig. 1(c) is shown in Fig. 2(a). For the label image, the above two requirements imply that every pixel of the image must be assigned a label and that no pixel may be assigned more than one label. The number of labels used depends on the application; this number can range from two labels for a segmentation into foreground and background regions (or interesting and uninteresting regions) to many labels. For a digital image, the largest possible number of labels is obtained if each pixel is assigned its own label (although such a segmentation is less useful in

Wiley Encyclopedia of Computer Science and Engineering, edited by Benjamin Wah.

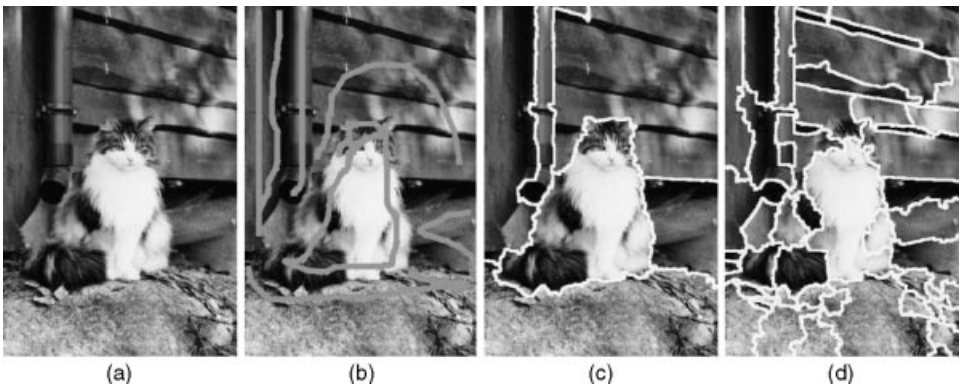

Figure 1. Example segmentations: (a) Original image. (b) Markers (in gray) drawn in by a user. (c) Marker-based watershed segmentation based on the markers in (b), see the section entitled "watershed with markers". (d) Hierarchical watershed segmentation using synchronous flooding by volume into 25 regions, see the section entitled "Hierarchical watersheds."

practice). A commonly used representation is to assign an additional "boundary" label to the pixels on the boundary between adjacent regions, as shown in Fig. 2(b). The choice of the representation with or without a boundary label depends on the application for which the segmentation is to be used. For example, it is useful to have the boundary if a requirement is to superimpose the edges of the partition regions onto the original image. However, if the aim is feature extraction from regions for object recognition, then the boundary is less useful as extracting features from it makes little sense, and it decreases the size of the regions in which features can be extracted. As an extreme example, if thin objects of the order of two pixels wide are to be segmented, then it may be a problem to use the representation with a boundary label, as a complete two-pixel-wide object could be labeled as a double boundary.

The decision as to which representation is required should be made when choosing the segmentation algorithm (versions of both the region growing and watershed algorithms exist to produce both representations), as converting directly between these two partition representations is not straightforward. To remove the boundary label would require replacing it by the label of one of the adjacent regions, but the choice between the regions is arbitrary. Similarly, adding a boundary label requires making a choice between which of a pair of adjacent regions will have a row of pixels removed. These arbitrary choices lead to a one-pixel uncertainty in the position of the boundary label or region labels.

An alternative representation of a partition is by the region adjacency graph (RAG), in which each region is represented by a vertex, and the vertices of those regions that are adjacent to each other (share a common boundary) are linked by edges [Fig. 2(c)]. The RAG can also have weights associated with the edges (forming a weighted graph). For the RAG, the edge weights usually encode some measure of the difference or similarity between the regions represented by the vertices linked by an edge. Features calculated from the regions can be associated with the vertices. The representation as a RAG allows operations from graph theory to be applied to the partition, and it often allows efficient implementation of segmentation algorithms. One of the most common operations on the RAG is to group vertices linked by edges that have weights indicating high region similarity. This operation corresponds to the merging of the regions associated with these grouped vertices in the partition, as shown in Fig. 2(d). Similarity criteria that have been used include color similarity (such as a distance in color space between the mean colors of the adjacent regions), texture similarity, low contrast along the common boundary of the adjacent regions, and the criteria based on watershed flooding described later in this article. *Hierarchies of partitions* can be built in which higher levels of the hierarchy correspond to the merging of a larger number of regions of the partition at the lowest level of the hierarchy [see Fig. 3(b)].

REGION-BASED SEGMENTATION

Region-based segmentation algorithms aim to build "homogeneous" regions, which are regions that satisfy a given

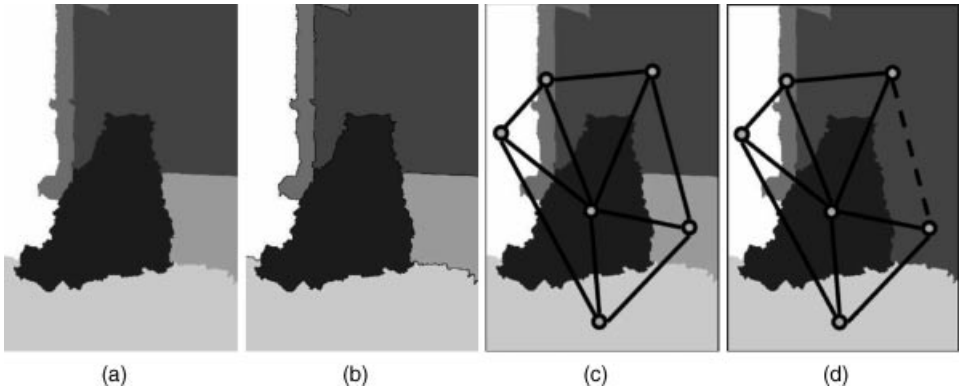

Figure 2. Partition representations: (a) Label image corresponding to the partition in Fig. 1 (c) (b) Label image in which the partition boundary lines are shown as an additional label (in black). (c) Region adjacency graph of partitions (a) and (b). (d) Two vertices of the RAG have been merged, as indicated by the broken line, which results in the merging of two regions of the partition (indicated by the assignment of the same region label).

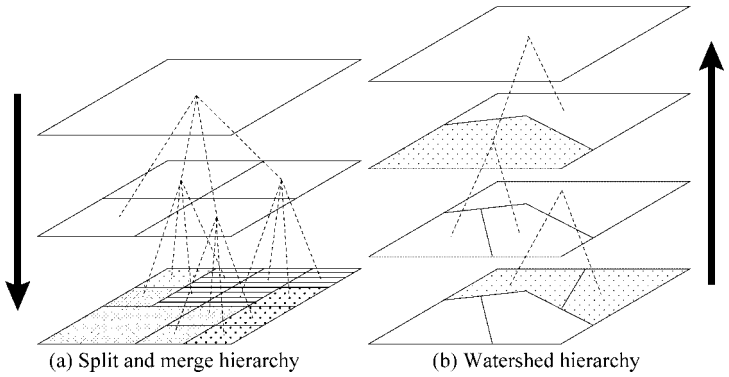

Figure 3. (a) Split and merge hierarchy—starting at the top, the subdivision into four regions continues until the resultant regions satisfy the homogeneity criterion. The merging step in the lowest level merges regions that together satisfy the homogeneity criterion, which leads to a reduction from 13 to 4 regions indicated by the fill patterns. (b) Watershed hierarchy—regions are merged pairwise until the whole image forms one region. A possible combination of regions from different levels to form an image partition is shown by the shading.

homogeneity criterion. This criterion can be based on image features such as gray scale, color, texture, shape, and so on. The regions can be built either by grouping pixels to form regions (*region growing* algorithm) or by starting with a single region and successively subdividing it (*split and merge* algorithm). The segmentations obtained by these two algorithms are (in almost every case) not the same.

The homogeneity criterion can be evaluated on any region or group of regions of a partition, in which the outcome is a binary decision on whether the region satisfies the criterion. Often, it is easier to specify it as a *similarity criterion*, which, given two regions, decides whether they are similar. A similarity criterion on two regions can also be implemented as a homogeneity criterion on the union of the two regions.

Region Growing

The region growing approach (6) grows a region of a partition starting from a seed (consisting of one or more pixels) by successively adding adjacent pixels to the region if they satisfy a similarity criterion. The growth stops when no adjacent pixels satisfy the similarity criterion. To create an image partition by region growing, one usually begins with a single pixel of the image as the seed and successively adds adjacent pixels to it until no adjacent pixels satisfy the similarity criterion. A new seed pixel is then chosen from the pixels that have no label, and the process is iterated until all pixels are labeled.

To use the region growing approach, one needs to specify:

1. an algorithm for generating the seeds (or the seeds themselves)
2. a similarity criterion

In general, different seed positions result in different partitions. A common practice is to take the first seed as the top left pixel of the image, with subsequent seeds being chosen as pixels adjacent to the already labeled regions. Alternatively, one can attempt to choose regions that are representative of the pixel characteristics being used. For example, for region growing based on gray values, pixels with gray values that correspond to peaks in the gray level histogram can be used as seeds. Background knowledge can also be used. For the application described in Ref. 2, it is known that the objects of interest all contain pixels with maximum gray value—these pixels are used as seeds.

The similarity criterion must be designed to compare a single pixel (the one being tested) with a group of pixels (those already part of the region). For the gray scale case, if the seed pixels all have the same gray value, then one could define a similarity criterion based on a threshold on the difference between the seed gray value and the gray value of the pixel being tested. If the seed contains pixels that have a range of gray values, then the mean gray value could be used. Instead of comparing pixels only with seed pixels, they can be compared with all pixels that are part of the region being grown. If a mean gray value is used as a region characteristic, then this mean should be updated with every new pixel added. For the similarity criterion presented in Ref. 7, a region is characterized by the mean and scatter of its gray levels, and the decision on whether a pixel is similar enough is based on a statistical significance test.

Adams and Bischof (8) propose an efficient region-growing algorithm that, given n seeds (groups of connected pixels) as input, assigns every pixel in the image to one of n partition regions that correspond to the seeds. This task is performed through the use of an ordered list of all unassigned pixels adjacent to the regions, which are sorted according to their similarity criterion calculated for every pixel with respect to its adjacent region. The following steps are iterated until all pixels have been assigned to a region: The pixel with the highest similarity measure is added to its adjacent region, the mean gray value of the region is updated, and the pixel list is updated to include the unassigned pixels now adjacent to the newly expanded region. It is also possible to assign pixels between two regions to a class of boundary pixels.

Region Splitting and Merging

Region splitting and merging (9) is a hierarchical approach to region-based image segmentation. The *splitting* component of the algorithm begins with the whole image considered as a single region. If the homogeneity criterion is not satisfied for this region, then it is split into four subregions. The homogeneity criterion is then tested on each subregion, and those that do not satisfy the criterion are split into four subregions. This process continues until all resultant

regions satisfy the homogeneity criterion. The resulting hierarchy can be represented conveniently by a *quadtree* (a tree in which each vertex has exactly four descendents), where the leaf vertices represent the homogeneous regions. This model is shown in Fig. 3(a) for a three-level hierarchy. In the second level, the lower left quadrant already satisfies the homogeneity criterion, so it is not subdivided in the third level. Using only this splitting procedure can obviously lead to adjacent regions that have very similar properties. The *merging* process is therefore applied to merge these adjacent similar regions—if the union of a pair of adjacent regions satisfies the homogeneity criterion, then these regions are merged. This process is demonstrated in the lowest level of the hierarchy in Fig. 3(a), in which the fill patterns show a possible merging of the 13 regions to obtain 4 regions. The full algorithm consists of iterating the splitting and merging steps until no more region splits or merges can be made.

Many variations on this algorithm exist, including specifications on the order of the splitting and merging steps (2); a version with overlapping regions; and a more efficient single-pass algorithm (3). A disadvantage of this algorithm is the assumption that the regions are composed of groups of rectangles, which leads to a block-like appearance of the resulting partition. The main disadvantage is that the algorithm is sensitive to image translations.

WATERSHED TRANSFORM

The watershed transform combines aspects of both region-based and edge-based approaches to image segmentation. The regions are built by pixel grouping (region-based), whereas the edges of the regions are located based on image discontinuities (edge-based). This section presents the basic algorithm as well as more efficient algorithms for its implementation. A framework for using the watershed transform in practical applications is presented in the next section.

Basic Definition

The idea for the watershed transform comes from *topography*. A drop of water falling somewhere on a landscape flows downhill until it reaches a river or other body of water. A body of water and the surrounding region from which the water drains into it is called a *catchment basin*. It is obvious that every local minimum (point surrounded only by points of higher altitude) on the landscape will be associated with a catchment basin. Catchment basins are separated topographically from adjacent catchment basins by lines of maximum altitude called *watershed lines* or simply *watersheds*. For a drop of water that falls onto a watershed line, it is not defined to which of the adjacent catchment basins it belongs. Those points on the landscape that do not belong to any catchment basin therefore form part of the watershed. These definitions allow one to classify every point of a topographic surface (landscape) as either belonging to a catchment basin associated with one of the local minima of the landscape or to the watershed, effectively dividing up (segmenting) the landscape.

A gray-scale digital image can also be viewed as a topographic surface, in which the altitude at each pixel is given by its gray value. A simulated drop of water falling onto this surface will flow in the direction of steepest gradient to a gray-value minimum. By this approach, one can assign each pixel of the image to a catchment basin associated with an image minimum or to the watershed. The resulting catchment basins then form the regions of an image partition, which are separated by pixels that belong to the watershed. The result of a watershed transform of a gray-scale image is usually represented by a label image in which every pixel that belongs to a given catchment basin is assigned the same label, and the pixels that belong to the watershed are assigned a unique, distinct label [see Fig. 2(b)].

Algorithms

The above watershed definition is intended for use in a continuous Euclidean space. Using an algorithm based on water-drop flow simulation to segment a digital image is not the best approach, as such images often contain plateaus (areas of constant gray value) in which the direction of steepest descent cannot be determined. Determining the catchment basins by simulated immersion overcomes this problem (10). This process can be visualized as follows: A hole is made in the topographic surface at each local minimum, and the surface is gradually immersed in water. A lake will form in each catchment basin, and all catchment basins will be filled to the same level [Fig. 4(a)]. When two lakes that form in separate catchment basins are about to merge, a dam wall is built to prevent the lakes from merging [Fig. 4(b)]. Once the topographic surface is submerged to its highest level, the dam walls give the positions of the watershed lines, and the areas delimited by the dam walls

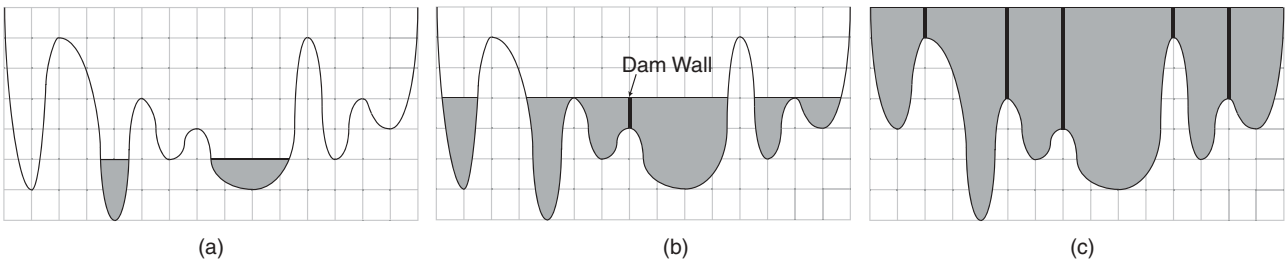

Figure 4. Three different stages of watershed construction by flooding on a function of one variable. The final watershed lines are given by the thick vertical "dam walls" in (c).

correspond to the catchment basins (regions), with one catchment basin produced by each minimum [Fig. 4(c)]. For simplicity, the example in Fig. 4 is shown on a function of one variable (as is done for all examples in this article). The watershed algorithm is however applicable to functions of two variables (gray-level images), three variables (volume images or sequences of gray-level images), four variables (sequences of volume images), and in feature spaces of higher dimensions.

The watershed algorithm can be directly implemented by making use of morphological geodesic operators on successive thresholds of an image (4). A more efficient algorithm that uses queues of pixels to access those pixels which must be modified at each step (10) is used in most practical implementations. An alternative definition and algorithm for the digital case using topographical distance is presented in Ref. 11. It is based more closely on the definition for the continuous case in the section entitled "Basic Definition," but it includes a strategy for dealing with plateaux. An overview of algorithms and parallelization strategies for both the immersion simulation and topographical distance approaches can be found in Ref. 1. Note that, as often occurs when concepts from continuous space are transferred to digital space, the immersion simulation and topographic distance approaches can produce different results for some gray-level configurations. Another consequence of the watershed definition in digital space is the occasional occurrence of *thick watersheds*, which means that the watershed pixels do not form one-pixel thick lines but extended areas. These thick watersheds occur for both approaches, but they tend to be less pronounced for the immersion simulation approach (1). Watershed algorithms exist to create partitions both with and without a boundary label.

WATERSHED ALGORITHMS IN PRACTICE

The image to which the watershed is applied should have each region of the partition to be computed marked by a gray-scale minimum, and the boundaries of the regions marked by gray-scale maxima. As the image to be segmented usually does not satisfy these requirements, it is preprocessed to produce an image that better satisfies them. For example, for a gray-scale image in which the edges of the regions are characterized by abrupt differences in gray values, the modulus of its gradient would satisfy the requirements. The problem with the watershed segmentation, however, is that the input image usually has an excess of local minima, which leads to an over-segmentation of the image, as one partition region is created for each local minimum.

Three approaches are used to reduce over-segmentation, which correspond to the three paths through the tree in Fig. 5. The first part of all three paths is *preprocessing and filtering*, in which the aim of this step is to reduce the number of local minima and enhance the object boundaries. The result of this step is referred to as the *segmentation image*. The watershed can optionally be applied directly to this image (path 1 in Fig. 5). Starting with the segmentation image, two approaches exist to reduce over-segmentation even more. Following path 2 allows one to specify the positions at which the flooding should start by creating a *marker image*, which allows knowledge that one already has about a scene to be used. In path 3, a hierarchy of partitions is created, in which the lowest level has the most regions, and regions are merged as one rises through the levels of the hierarchy. One can then choose the level (or combination of levels) of the hierarchy that corresponds to the required partition. These methods are described in the next subsections.

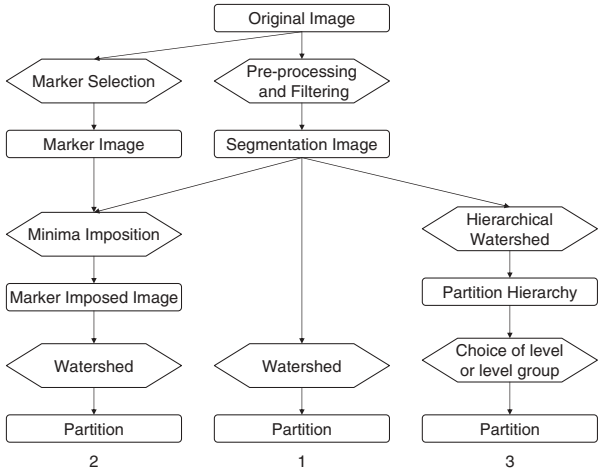

Figure 5. The three approaches to applying the watershed discussed in the text. The numbers below indicate the endpoints of the three possible paths traversing the tree.

Filtering and Preprocessing

The aim of the filtering and preprocessing step is to create a *segmentation image* that satisfies the requirements for a good watershed segmentation—the number of spurious local minima should be small, and the boundaries of the regions should be marked by maxima. This step has been divided into a *filtering* part, which has the aim of filtering noise and hence reducing the number of spurious local minima, and a *preprocessing* part, which aims to enhance the region boundaries. These parts may be applied in any order or interleaved. For example, an image may either be filtered before calculating its gradient, or the gradient may be filtered.

As is clear from Fig. 5, the output from this step can either be passed directly to the watershed algorithm, or used as input to the watershed with markers or the hierarchical watershed. Obviously, if the original image is already perfectly suited to being used with the watershed algorithm, then this step can be ignored.

Filtering. Every local minimum produces a region in the partition created by the watershed algorithm. However, as most local minima are often caused by noise or artefacts and hence not significant, a filter that aims to reduce the noise or other small spurious details is usually applied. Such a filter could also reduce the variation in textured areas of an image. Any arsenal of image filters can be applied to this task. Note that for a marker-based watershed (see

Watershed with Markers), this filtering of minima is not required, as the minimum imposition algorithm removes all minima except for those specified in the marker image.

A desirable property of filters that are used prior to image segmentation is that the position of the image contours should be preserved as well as possible. Linear filters are therefore less suitable as they tend to blur the contours. Nonlinear filters, such as the median filter or some morphological filters, better satisfy this contour-preserving property. Another advantage of morphological filters is that they may be used to remove specific image structures while preserving others (4). In practice, one needs to find the best compromise between the creation of smooth regions and the preservation of well-localized contours.

The *levelings* (12) are morphological filters specifically created for simplifying an image prior to segmentation. They take as input the original image and a simplified version of it. The latter is often created by an alternating sequential filter (4), which consists of a sequence of alternating opening and closing operators using structuring elements of successively increasing size, but it does not preserve the contour positions. The leveling is constructed by retaining those contours of the original image that are also present in the simplified version, while flattening the regions between these contours. The *h-minima* operator, although it does not satisfy the properties of a morphological filter, is useful for suppressing minima if one knows that only minima which are deeper than a specified depth should be taken into account (4). This operator takes a gray-scale image and a numeric depth as arguments and produces an image in which all local minima that have a depth less than the specified value have been removed.

Preprocessing. The aim of preprocessing is to create well-defined region boundaries marked by gray-scale maxima. To do this, one needs a model of what constitutes a boundary. If the regions to be found correspond to areas where the gray-level varies little, and the region boundaries correspond to large changes in gray-level, then the modulus of a gradient of the initial image, such as the morphological gradient (4), should be calculated. If the regions are characterized by different textures, then an operator that enhances the boundary between different textures should be used. For color images in which regions are characterized by having a fairly constant color, an operator that creates a gray-scale gradient from a color image can be used. Gradients that can be used on color images are the saturation-weighted hue and brightness gradient (13), which gives a higher weight to color differences in more colorful areas of an image (areas with higher saturation) and a higher weight to brightness differences in less colorful (lower saturation) areas. The quasi-invariant gradients (14) can be used to ignore edges caused by shadows or shading or by specular reflection. A classic preprocessing approach is used for round objects (such as coffee beans or cells) that, even though they can be separated from the background by a threshold, still touch each other in the resulting binary image. To separate the objects, one applies a distance transform to the thresholded image (4). This transform creates a gray-scale distance image that encodes the shortest distance of each foreground pixel from the background. The center of each object is then a maximum, and applying the watershed to the inverse of this distance image results in the objects being separated.

Watershed with Markers

The watershed with markers can also be visualized by using the immersion simulation. The difference is that instead of punching holes at each local minimum of the topographic surface, holes are only punched at positions specified by a set of markers. As only one region is created per marker, having fewer markers will reduce the number of regions in the final segmentation. An example is shown in Fig. 6, in which the positions of three markers are indicated by the arrows. In Fig. 6(a), it can be observed that the flooding only starts in the marked catchment basins. In Fig. 6(b), a dam wall is constructed as usual where two dams meet. For catchment basins that do not contain markers, the water simply flows in, as can be observed for the catchment basin on the right. Fig. 6(c) shows the final partition into three regions. In practice, it is not necessary that markers be placed at local minima. Markers may also cover a larger area.

To use markers in practice, one creates a *marker image*, which is a binary image in which each connected component corresponds to a marker, or, using the immersion visualization, a "hole" through which water will flow. A marker can range in size from a single pixel to a large, connected component. These markers are imposed on the image on which the watershed is to be calculated through *minima imposition* or *swamping* (4). The minima imposition operator takes both the marker image and segmentation image as inputs, and it produces as output an image in which minima are present only at the positions indicated in

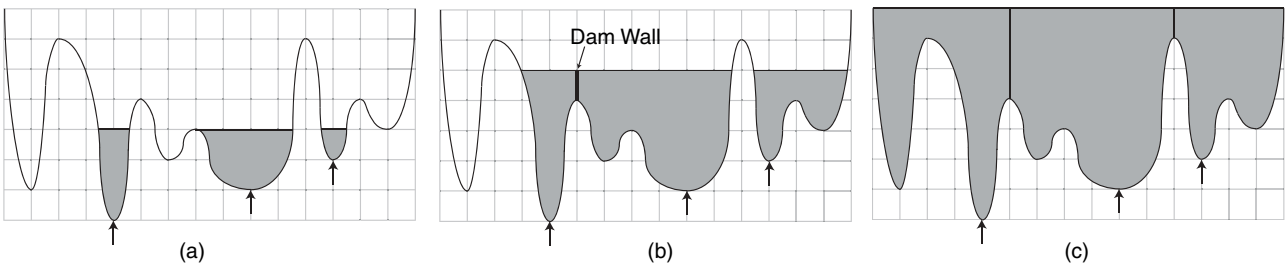

Figure 6. A flooding sequence for a marker controlled watershed segmentation. The positions of the three markers are indicated by the arrows.

the marker image and have been eliminated elsewhere. Applying the watershed algorithm to this *marker-imposed image* produces the desired results, with one region produced for each connected component in the marker image. An efficient algorithm that combines the minima imposition and watershed computation into a single step through the use of an array of priority queues is presented in Ref. 15 .

The key component in successfully using a watershed with markers is creating the marker image. One marker is required for each region in the final segmentation. If many objects need to be separated from the background, then in addition to one marker per object, a marker for the background is required. The simplest way of obtaining markers is to request that a user draw in the markers. Figure. 1(b) shows markers drawn in by the user which lead to the partition in Fig. 1(c). Although manual marker specification tends to produce the best segmentations, it is also very time consuming and labor intensive. Hence, it is not suited to applications in which an extremely large number of images are to be segmented. It is used, for example, in medical applications, in which medical personnel are in general prepared to indicate regions if the resulting automatic segmentation leads to time savings. In other words, the resulting segmentation should be almost identical to the best manual segmentation, but it should be obtained by very few user interactions (i.e., marker drawing).

Alternatively, the markers are extracted automatically, for which one needs a model of the characteristics of a marker. Obviously, these characteristics are different for different applications. Possibilities include those discussed for creating region growing seeds in the section entitled "Region growing," extended minima [the regional minima obtained after applying an h-minima filter (4)], flat zones (zones with constant gray value), quasi-flat zones (zones with near constant gray value), areas with a specific color or texture, and so on. These regions can be filtered even more by an area opening, which removes connected components that have an area smaller than the provided number of pixels (4), or by the more general attribute opening (16), which also allows filtering based on criteria such as the area of the convex hull of a connected component or its maximum Feret diameter.

The determination of an adequate preprocessing and filtering of the input image and of a marker image has been referred to as the "intelligent part" of the morphological approach to segmentation (15). An example that demonstrates the determination of these images for the segmentation of the gray-scale jelly beans image [Fig. 7(a)] is now presented, in which the aim is to create a region that corresponds to each jelly bean. To create an image of the boundaries, one could calculate the morphological gradient of this image. However, as the the bright specular reflection spot on each jelly bean also causes a strong gradient, they must first be removed. This removal is done by applying a morphological opening by a 5 x 5 square structuring element to the original image [Fig. 7(b)] and calculating the morphological gradient of the opened image using a 3 x 3 structuring element [Fig. 7(c)]. Now a marker is needed for each jelly bean. Here we use the fact the each jelly bean seems to be marked by a specular reflection. An extended maxima operator of height 10 applied to the original image produces the marker image in Fig. 7(d). Because of the unevenness of the background, a marker is also produced for the background. A watershed is now applied to the gradient image on which the white areas in the marker image have been imposed as minima. The resulting watershed lines, which are overlaid in white on the original image, are shown in Fig. 7(e). Although this segmentation delimits most jelly beans, for some the algorithm has failed. This failed segmentation would have to be corrected by improving the steps in the intelligent part of the process.

Hierarchical Watersheds

Whereas the watershed algorithms described above produce a single partition as output, the hierarchical watershed algorithm produces a hierarchy of partitions. The partition at the bottom of the hierarchy is the output of the classic watershed algorithm, and it has the largest number of regions. The regions are progressively merged to form larger regions as one moves to higher levels of the hierarchy, so that the highest possible level of the hierarchy consists of a single region that covers the whole image, as shown schematically in Fig. 3(b). A partition at the desired scale or containing the desired number of regions is obtained by selecting the corresponding level of the hierarchy. Alternatively, a combination of partition regions at different levels can be chosen if one requires segmentation at a finer scale for some parts of the image and at a coarser scale for other parts. The shaded regions in Fig. 3(b) show a partition that resulted from a combination of partitions at different levels of the hierarchy. Such a combination of partitions at different levels has been used in an interactive segmentation application (17), in which the user may split a region into subregions (effectively moving to a lower level of the hierarchy for the chosen region) or may merge regions.

Two methods for creating hierarchies based on flooding are presented. The hierarchies produced through *synchronous flooding* tend to have a large number of levels, as in general only two adjacent regions are merged as one moves to the next higher level of the hierarchy. The *waterfall algorithm*, in contrast, has a region-merging rule that results in a single region within only a few levels of the hierarchy (typically less than 10).

Figure 7. The watershed segmentation process for the jelly beans image.

The approaches described here are bottom-up approaches, in which one starts with a large number of regions and successively merges them. Top-down watershed approaches, which are related to the idea behind the split-and-merge algorithm, have also been developed (18). The image is first strongly filtered to produce a segmentation into a small number of regions (coarse segmentation). This partition is then refined by iteratively resegmenting the regions of the coarse segmentation using less strong filtering.

Hierarchies from Synchronous Flooding. The partition at the lowest level of the hierarchy is the output of the classic watershed algorithm, in which each local minimum in the segmentation image produces a region. A version of the algorithm that does not label the boundary pixels is usually used. To build the hierarchy, the regions are merged pair wise, with each merge producing a new level of the hierarchy with one region less. The order of merging of the regions is determined by a new flooding of the segmentation image.

In the classic watershed algorithm, lakes in adjacent catchment basins are prevented from merging by building dam walls. The final watershed is given by the dam walls when the flooding has reached its highest level. When building a hierarchy of region merges based on the flooding, dam walls are not built. As before, water rises through holes punched at all the local minima in the image topography. When two lakes meet, they are allowed to merge. Every merging of a pair of lakes produces the next highest level of the hierarchy. The resulting hierarchy can then be used as outlined above to produce a partition into the required number of regions.

In the classic flooding approach, referred to as *uniform flooding*, the global water level is constant. However, the partition hierarchy constructed from uniform flooding is less useful in practice, because the order of the region merging is determined only by the height of the lowest point (pass-point) in the topography separating adjacent lakes and not by the characteristics of the lakes themselves. To overcome this problem, size-oriented synchronous flooding was introduced (17). Water floods each catchment basin so that a chosen size measurement is the same for each lake. This measurement can be the *depth, volume,* or *surface area* of the lakes. When a lake fills a catchment basin so that it cannot grow any more, it merges with the lake in the catchment basin adjacent to the pass-point. An example of synchronous flooding by depth is shown in Fig. 8. The measurement chosen to characterize a lake determines the order of region merging. For depth, the deepest lakes, which correspond to the most contrasted regions, will survive to the highest levels of the hierarchy. For surface area, the largest regions will survive to the highest levels. The volume provides a balance between contrast and size. A segmentation of the image in Fig. 1(a) into 25 regions by such a hierarchical approach using equal volume measures is shown in Fig. 1(d). The hierarchy was built on the saturation-weighted hue and brightness gradient (13) of the original color image. Additional methods to influence the hierarchy are described in Ref. 17 —for example, analogous to the watershed with markers, markers can be placed in regions that should survive to the higher levels of the hierarchy. To achieve this, the associated catchment basins are simply not flooded.

A more efficient way of keeping a record of the merging of regions is to use the RAG. Each minimum (and its associated region in the watershed partition) is represented by a vertex in the graph. When the lakes associated with adjacent minima merge, the vertices that represent these minima in the graph are connected by a weighted edge, for which the weight is set to the size measurement of the lake that is full. A simple example of the construction of the graph is shown in Fig. 8 at the bottom of each subfigure. The black circles (vertices) represent the catchment basins. Each time a lake fills its catchment basin, its merging with the neighboring lake is indicated by drawing in an edge, for which the weight is set to the depth of the full lake. It is possible to show that the graph resulting from this algorithm is a minimum-spanning tree (17). To produce a partition that contains N regions, one simply cuts the $N - 1$ edges with the highest weights. For example, cutting the edges with weights 4 and 5 in Fig. 8(e) produces the partition into three regions in Fig. 8(c). This method is equivalent to choosing the level in the partition hierarchy that contains N regions.

Waterfall Hierarchies. The waterfall algorithm is an alternative method for creating a partition hierarchy, which is introduced in Ref. 19. The partition produced by the

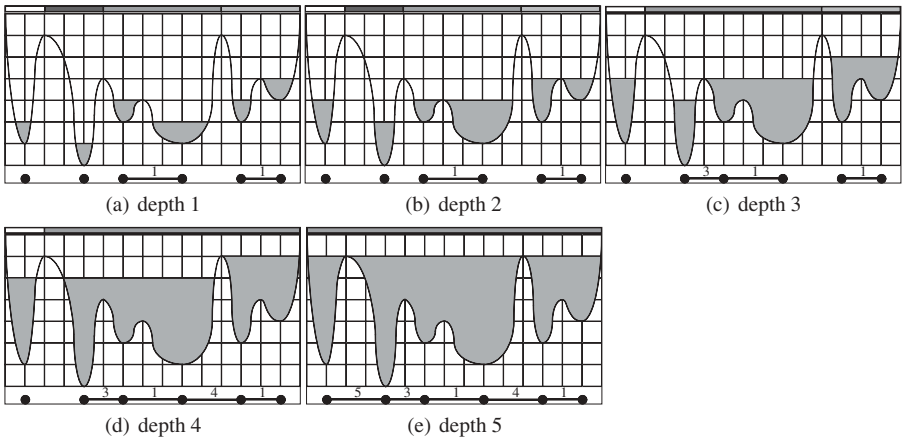

Figure 8. Five different stages in the construction of a partition hierarchy by synchronous flooding using depth on a function of one variable. The partition is shown above each function; different shades of gray represent different regions. The construction of a graph that describes the flooding process is shown below each function, in which the black circles (vertices) represent the catchment basins, and the numbered lines between them represent the weighted links.

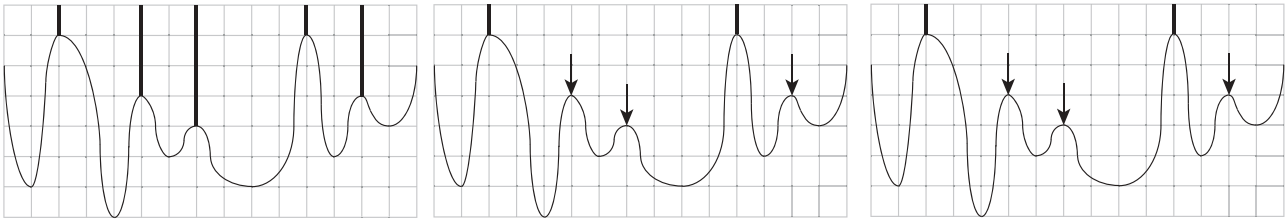

Figure 9. Three levels of a waterfall hierarchy: (a) Level 0 is a standard watershed segmentation, where every minimum produces a region with boundaries indicated by the thick vertical lines. (b) At level 1, the boundaries at the peaks marked by the arrows are removed as they are surrounded by higher peaks, which lead to the segmentation into three regions. (c) At level 2, the remaining boundaries are removed so that only one region remains.

classic watershed algorithm again forms the lowest level of the waterfall partition hierarchy. In the original version, the mutual boundary between adjacent pairs of catchment basins is characterized by the height of the pass-point (lowest point on the boundary). To create the next level of the hierarchy, all boundaries that are surrounded by boundaries that have higher pass-points are removed. This algorithm is demonstrated in Fig. 9, in which the arrows in Fig. 9(b) indicate the peaks that are removed because they are surrounded by higher peaks, which leads to the segmentation into three regions with boundaries indicated by the thick vertical lines. This removal step can be iterated to create additional levels of the hierarchy. This merging criterion results in many pairs of regions being merged as one moves to each subsequent hierarchy level.

The original waterfall algorithm was implemented using morphological reconstruction operators. An efficient graph-based algorithm that operates on the minimum-spanning tree is presented in Ref. 20. This implementation has the advantage that the characterization of lakes by their pass-point can be replaced by one of the characteristics used with synchronous flooding.

SUMMARY

Region-based segmentation algorithms construct regions of a segmentation partition based on a specified homogeneity criterion. Algorithms include region growing and split-and-merge. The watershed approach includes aspects from edge-based segmentation approaches, in that the region boundaries can be made to lie on lines that correspond to discontinuities in the image. In practice, one has a large array of possibilities to influence the result of a watershed transform, including the choice of preprocessing and filtering to create the segmentation image, the choice of markers for the marker-based watershed, and the use of hierarchical watershed approaches.

BIBLIOGRAPHY

1. J. B. T. M. Roerdink and A. Meijster, The watershed transform: definitions, algorithms and parallelization strategies, *Fundament. Informat.*, **41**: 187–228, 2001.
2. R. C. Gonzalez and R. E. Woods, *Digital Image Processing,* 2nd ed., Englewood Cliffs, N. J., Prentice-Hall, 2002.
3. M. Sonka, V. Hlávač, and R. Boyle, *Image Processing, Analysis and Machine Vision*, 2nd ed., Belomont, CA: Brooks/Cole Publishing, 1999.
4. P. Soille, *Morphological Image Analysis*, 2nd ed., New York: Springer, 2002.
5. J. Serra, *Image Analysis and Mathematical Morphology Volume 2: Theoretical Advances*, London: Academic Press, 1988.
6. S. W. Zucker, Region growing: childhood and adolescence, *Comp. Graph. Image Process.*, **5**: 382–399, 1976.
7. R. M. Haralick and L. G. Shapiro, Image segmentation techniques, *Comp. Vis. Graph. Image Process.*, **29** (1): 100–132, 1985.
8. R. Adams and L. Bischof, Seeded region growing, *IEEE Trans. Patt. Anal. Mach. Intell.*, **16** (6): 641–647, 1994.
9. S. L. Horowitz and T. Pavlidis, Picture segmentation by a directed split-and-merge procedure, *Proc. Second International Joint Conference on Pattern Recognition*, 1974, pp. 424–433.
10. L. Vincent and P. Soille, Watersheds in digital spaces: an efficient algorithm based on immersion simulations, *IEEE Trans. Patt. Anal. Mach. Intell.*, **13** (6): 583–598, 1991.
11. F. Meyer, Topographic distance and watershed lines, *Signal Process.*, **38**: 113–125, 1994.
12. F. Meyer, Levelings, image simplification filters for segmentation, *J. Mathemat. Imag. Vis.*, **20**: 59–72, 2004.
13. J. Angulo and J. Serra, Modelling and segmentation of color images in polar representations, *Image Vision Comput.*, **25** (4): 475–495, 2007.
14. J. van de Weijer, T. Gevers, and J. Geusebroek, Edge and corner detection by photometric quasi-invariants, *IEEE Trans. Patt. Anal. Mach. Intell.*, **27** (4): 625–630, 2005.
15. S. Beucher and F. Meyer, The morphological approach to segmentation: the watershed transformation, in E. Dougherty, (ed.), *Mathematical Morphology in Image Processing*, New York: Marcel Dekker, 1993, pp. 433–481.
16. E. J. Breen and R. Jones, Attribute openings, thinnings, and granulometries, *Comp. Vision Image Understan.*, **64** (3): 377–389, 1996.
17. F. Meyer, An overview of morphological segmentation, *Internat. J. Patt. Recog. Artif. Intell.*, **15** (7): 1089–1118, 2001.
18. P. Salembier, Morphological multiscale segmentation for image coding, *Signal Process.*, **38** (3): 359–386, 1994.
19. S. Beucher, Watershed, hierarchical segmentation and waterfall algorithm, *Mathematical Morphology and its Applications to Image Processing, Proc. ISMM'94*, 1994, pp. 69–76.

20. B. Marcotegui and S. Beucher, Fast implementation of waterfall based on graphs, *Mathematical Morphology and its Applications to Image Processing, Proc. ISMM'05*, 2005, pp. 177–186.

ALLAN HANBURY
Vienna University of Technology
Vienna, Austria