A Review of Computer Vision Segmentation Algorithms

Dingding Liu, Bilge Soran, Gregg Petrie, and Linda Shapiro

Department of Computer Science Engineering

Department of Electrical Engineering

University of Washington

1 Introduction

The remote sensing and computer vision communities share a common goal of extracting useful information from raw imagery. Both communities have exploited several trends that support the increasingly timely, cost effective, accurate, and effective automated extraction of information from raw imagery that include increasingly powerful, affordable, and available computer hardware; increasingly sophisticated software tools, both commercial and open source, that have a proven track record; a growing community of computer knowledgeable users; and an increasing proliferation of sophisticated sensors, both active and passive, ranging from handheld digital cameras to commercial satellites with drastically improved spatial, radiometric, and temporal resolution.

While the computer vision (CV) community has many of the same goals as the remote sensing (RS) cummunity, its applications are not focused in characterising the earth's surface but include a much wider range of applications. Computer vision applications include industrial, biomedical, scientific, environment exploration, surveillance, document understanding, video analysis, graphics, games and entertainment. Computer vision systems have been designed that can control robots or autonomous vehicles, inspect machine parts, detect and recognize human faces, retrieve images from large databases according to content, reconstruct large objects or portions of cities from multiple photographs, track suspicious people or objects in videos, and more.

Remote sensing systems work on multi-spectral images that captures image data at specific frequencies across the electromagnetic spectrum. These images contain multiple bands, some from light frequencies visible to humans and some from frequencies beyond the visible light range, such as infrared. Computer vision systems have been developed for all types of images, but mainly work with single-band graytone images, three-band color images, and single-band depth images, sometimes registered to color images of the same scene. In spite of this difference, many of the same tools can be used in both computer vision and remote sensing.

Table 1 shows a simplified classification of computer vision tools, which are available in both open-source, such as Intel's OpenCV library in C++ [15] and NIH's ImageJ library in Java [3], and

Basic Tools	Example Applications
filters	noise suppression edge detection texture description
segmentation	object recognition image retrieval medical image analysis
interest operators	image matching motion analysis object recognition image retrieval
photogrammetric operations	3D reconstruction

Table 1: A Simple Classification of Computer Vision Tools

commercial packages such as Matlab with its Image Processing and Signal Processing toolkits [14]. Filter tools, which should be familiar to most RS workers, are used to enhance images or extract low-level features. Sharpening, brightening, noise removal, edge detection, and texture feature extraction are common filters. Interest operators, which have become popular in CV over the last decade, are operators that detect interesting points or small regions in images for purposes of image matching or object recognition. The best-known such operator is the SIFT operator [13], which detects interest points and describes them in a 128- dimensional vector that is invariant to translation, rotation, scale, and somewhat invariant to illumination. The points detected by SIFT stand out as interesting, because they are local minima and maxima of a difference of Gaussians operator applied at multiple scales. The SIFT operator can be used to find a starter set of matching points across pairs or sequences of images. SIFT can be used along with the RANSAC algorithm [11] for determining the best homography that maps the points in one image to matching points in another. This would be helpful to RS workers who need point correspondences to calculate 3D depth images from pairs of 2D images for registration of satellite imagery.

Segmentation operators partition an image into nonoverlapping regions, each of which is homogeneous in one or more features and maximal in terms of this homogeneity. Segmentation is important in both CV and RS, where it can be used to find areas that can be classified according to land use. Although segmentation algorithms include such old standards as split-and-merge [18] and region

growing [17], algorithms based on different forms of clustering have won out in recent years. Clustering algorithms can operate on gray-tone images, color images, or multi-spectral images, making them easily adaptible to the RS domain. In this paper, we will describe the most popular and useful clustering algorithms including K-Means clustering, Expectation-Maximization (EM) clustering, watershed segmentations, graph cuts, and mean-shift clustering. We will also discuss interactive clustering algorithms that allow some user input in order to segment out objects of interest that are not homogeneous in any feature but are important to the user.

Clustering algorithms can cluster any objects that can be represented by property vectors. In this paper, we will assume that the objects are pixels from an image and the property vectors contain numeric attributes of those pixels. In the simplest case, a gray tone image, each property vector contains only a single gray tone. For color images, property vectors usually represent the particular color space in use, such as RGB or HSV. For multispectral images, the property vector of a pixel would contain the value for that pixel in each of its bands. In addition to this "color" information, other attributes describing the location or texture of a pixel may also be added.

2 K-means Clustering

K-means clustering [23] is the simplest and most-used clustering algorithm. Given an image of N pixels, the goal is to partition the image into K clusters, where the value of K must be provided by the user. Clusters provide a grouping of the pixels that is dependent on their values in the image, but not necessarily on their locations in the image unless location is a specified property.

Let $\mathbf{X} = \{x_1, \ldots, x_N\}$ be a set of N image pixels, and let $V(x_i)$ denote the property vector associated with pixel x_i. The K-means algorithm has the following steps:

1. **Parameter Initialization:** The means of each of the K clusters are initialized to values of potential property vectors. In the classic K-means algorithm, the value of each element of the property vector is chosen randomly from the set of all possible values for that element.

For example, if the property vector is of the form (H,S,V) representing hue, saturation, and intensity value, the first element H woud be selected randomly from all possible hues.

2. **Hard Assignment of Pixels to Clusters:** Now that each of the K clusters C_k has a mean μ_k, each pixel x_i is assigned to the cluster with the closest mean, using a distance function that can compute the distance between 2 property vectors. At this point each pixel x_i is associated with a single cluster C_k.

3. **Parameter Recomputation:** The means of the clusters are now recomputed based on the property vector values of all the pixels in each cluster. Thus we have that μ_k is computed as the mean of $\{V(x_i) \mid x_i \in C_k\}$.

Steps 2 and 3 above are repeated until convergence, which occurs when no pixel moves from one cluster to another in a given iteration. Figure 1 illustrates a K-means segmentation of a color image into 4 clusters. Note that the roof of the building and the surface on which people are walking are approximately the same color in the image, so they are both assigned to the same cluster.

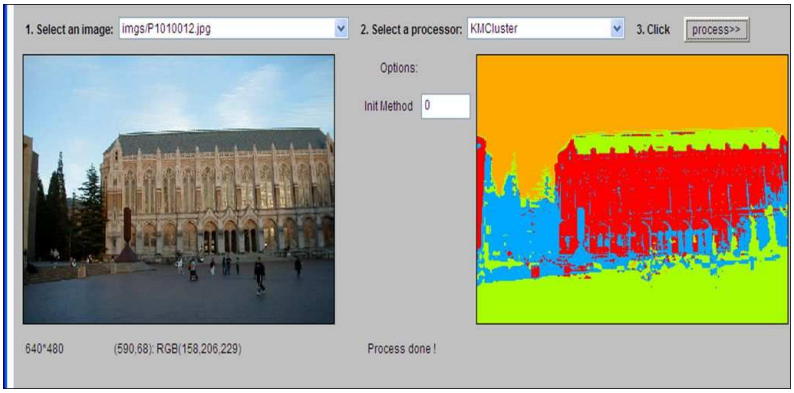

Figure 1: K-means segmentation of a building scene into 4 clusters.

3 Expectation-Maximization Clustering

The Expectation-Maximization (EM) algorithm [9] is related to K-means in that it also expects the user to select the number of clusters, and it has the same 3 steps: initialization, assignment of pixels to clusters, and parameter recomputation. The EM algorithm is more general than K-means in several ways. First, it is not just a clustering algorithm, but a general algorithm for a technique that finds maximum likelihood estimates in parametric models for incomplete data, which has many uses including both clustering and classification. Second, the clusters are represented by probability distributions, not just means. Gaussian distributions are most commonly used. Clusters are then represented by the mean μ and the covariance matrix Σ. Each cluster also has a weight w, and once the clusters (called components in general EM terminology) have been computed, each property vector can be expressed as a weighted *mixture*, or linear combination, of Gaussian components. Third, the assignment of pixels to clusters is a soft, or probabilistic, assignment instead of the hard assignment of each pixel to exactly one cluster in K-means. Each pixel will have a final probability of belonging to each of the final clusters. While it is common to assign it to the cluster which gets its highest probability, it is also possible to keep the entire vector of probabilities and use that in further analysis. The algorithm can be expressed as follows:

1. **Parameter Initialization:** The parameters for each cluster C_k, which are mean μ_k, the covariance matrix Σ_k and the weight $w_k = P(C_k)$, are initialized. The means can be initialized as random property vector values, as in K-means. The simplest way to initialize the covariance matrices is to set each of them to an $n \times n$ identity matrix for property vectors of length n. The weights are each initialized to $1/K$ so that all clusters are initially weighted equally.

2. **Soft Assignment of Pixels to Clusters (Expectation):** This step estimates the probability $P(C_k \mid x_i)$ for each pixel x_i and each cluster C_k. This conditional probability is computed according to the standard equation:

$$P(C_k \mid x_i) = P(x_i \mid C_k)P(C_k)/P(x_i) \tag{1}$$

where $P(x_i)$ is given by

$$P(x_i) = \sum_k P(x_i \mid C_k) P(C_k) \tag{2}$$

$P(C_k)$ is just the current weight w_k. If C_k is represented by a Gaussian distribution and $V(x_i)$ is the property vector of pixel x_i, $P(x_i \mid C_k)$ is given by

$$P(x_i|C_k) = \frac{1}{\sqrt{2\pi \mid \Sigma \mid}} e^{-\frac{1}{2} (V(x_i)-\mu_k)^T \Sigma^{-1} (V(x_i)-\mu_k)} \tag{3}$$

3. **Parameter Recomputation (Maximization):** The parameters μ_k, Σ_k, and w_k of each cluster C_k are now recomputed based on the property vector values $V(x_i)$ of all pixels x_i and the computed probabilities $P(C_k \mid x_i)$ from step 2. The formulas for updating the parameters are as follows:

$$u_k = \frac{\sum_i P(C_k \mid x_i) \cdot V(x_i)}{\sum_i P(C_k \mid x_i)} \tag{4}$$

$$\sigma_k = \frac{\sum_i P(C_k \mid x_i) \cdot (V(x_i) - \mu_k) \cdot (V(x_i) - \mu_k)^T}{\sum_i P(C_k \mid x_i)} \tag{5}$$

$$w_k = P(C_k) = \frac{\sum_i P(C_k \mid x_i)}{N} \tag{6}$$

EM clustering with K=4 was applied to the building image. The result is shown in Figure 2.

The EM algorithm was introduced to the computer vision community in a paper describing the Blobworld system [4], which uses color and texture features in the property vector for each pixel and the EM algorithm for segmentation as described above. Besides using the EM algorithm, the Blobworld paper also described a new set of texture features: polarity, anisotropy, and contrast [4]. Polarity is a meaure of the extent to which the gradient vectors in a neighborhood all point in the same direction, anisotropy is a function of the ratio of the eigenvalues of the second moment matrix, and contrast is a measure of homogeneity of the pixel values. The texture was computed at multiple

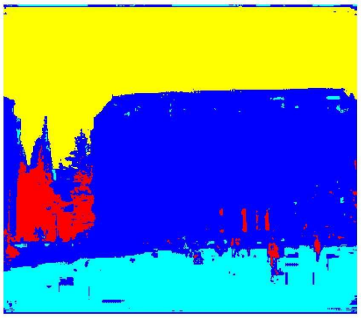

Figure 2: EM segmentation of the building scene using color features and K=4.

scales (neighborhood sizes) for each pixel, and the scale at which polarity changes became small (less than 2%) was selected as the best scale for that pixel. The algorithm was run with small values of K to produce a small number of regions called blobs that could be used in image retrieval. Figure 3 shows the result of clustering the building scene image using software from the Blobworld system. In order to use the Blobworld software, the building image was reduced to size 192×128.

Figure 3: Blobworld EM segmentation of the building scene using color and texture features.

4 Mean-Shift Clustering

Both K-means and EM segmentation require the user to provide the desired number of clusters. While this is reasonable for very constrained problems, it is often the case that the number of clusters depends very much on the given image, not on a predefined value. Clustering algorithms that can determine the number of clusters in the process of clustering are therefore more useful for segmentation of arbitrary images. The mean-shift clustering algorithm was specifically designed to choose the number of clusters in a theoretically sound fashion.

The idea of mean-shift clustering is very simple and based on a histogram of the image. We will explain the algorithm for a 1D graytone histogram and then discuss its use on multi-band images. As before, let $\mathbf{X} = \{x_1, \ldots, x_N\}$ be a set of N image pixels, and assume that the property vectors $V(x_i)$ have only one dimension: graytone. Let $H(X)$ be the image histogram. $H(X)$ is an array of bins $H_0, H_1, \ldots, H_{NG}$, where 0 is the mininum graytone value and NG is the maximum graytone value. A window of the histogram is a contiguous set of bins of some fixed length ws centered about a particular bin b_i. In general, the length ws of W will be an odd number.

The idea of mean-shift is to start with windows centered on a random bin and shift the center of the bin according to the data within it until the window converges at a mode of the histogram. This mode defines a cluster, and the process is repeated until all bins are associated with one of the computed modes. The shifting procedure is given by the following steps and guaranteed to converge.

1. Initialize with a random seed that selects a bin and select the window W centered on that bin.

2. Calculate the center of gravity or weighted mean wm of the histogram values in window W.

$$wm = \sum_{b_i \in W} b_i \; f(b_i)$$

   where $f(b_i)$ is the count in bin b_i normalized by dividing by the sum of counts in all the bins of window W.

3. Translate the search window W to the weighted mean wm

4. Repeat step 2 until convergence.

The shift procedure is illustrated in Figure 4 for 1D property vectors, such as gray tone. In the figure a window of length 9 of the histogram is shown with bins 1 through 9, centered on bin 5. The first 5 bins have count 0, while the other 4 have nonzero values totaling to 19. The new mean is calculated as

$$\frac{1(6) + 1(7) + 7(8) + 10(9)}{19} = 8$$

Thus the mean would be shifted from bin 5 to bin 8, and the procedure continued till convergence.

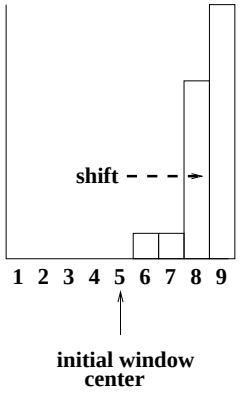

Figure 4: Example of mean shift for 1D histogram.

To use the mean-shift procedure for graytone image segmentation, the shift procedure is run for each gray tone value and its convergence point is stored. While theoretically, the distinct convergence points should all be clusters, in practice, convergence points that are within a small distance ϵ of one another are grouped to form the cluster centers, and each pixel is then assigned to the group of its convergence point. Small regions can be eliminated, as they are in most clustering procedures.

The general mean-shift procedure [5] [6] works with multi-dimensional property vectors and has three parameters. For the spatial domain, ie. the 2D image points, the parameter σ_s specifies the size of the spatial neighborhood in which shifting is performed. For the range domain, ie. the property vectors, the parameter σ_r normalizes the range of the data. A third parameter, minimum region size,

filters out small regions. The algorithm then generalizes to use a multi-dimensional spatial region S of normalized property vectors instead of a one-dimensional window of gray tones. General mean-shift filtering, as defined in [5], is given by the following procedure in which $\{\mathbf{x}_j\}_{j=1\ldots n}$ is the set of pixels and $\{\mathbf{z}_j\}_{j=1\ldots n}$ is the resulting convergence point of each pixel.

For each $j = 1, \ldots, n$

1. Initialize $k = 1$ and let $\mathbf{y}_k = \mathbf{x}_j$.
2. Compute $\mathbf{y}_{k+1} = \frac{1}{n_k} \sum_{\mathbf{x}_i \in S(\mathbf{y}_k)}$, $k \leftarrow k + 1$ till convergence
3. Assign $\mathbf{z}_j = (\mathbf{x}_j^s, \mathbf{y}_{conv}^r)$.

where the last assignment specifies that the filtered data at the spatial location of $\mathbf{x}_j$ will be assigned to the range components of the point of convergence $\mathbf{y}_{conv}$. The window $S(\mathbf{y}_k)$ is centered on $\mathbf{y}_k$ and has n_k points. Figure 5 shows two examples of mean-shift segmentations of the building image with different parameters. The segmentation on the left has a spatial neighborhood parameter of 50 and data range parameter of 5, while the segmentation on the right has a spatial neighborhood parameter of 5 and data range parameter of 2.5. Thus the left image is a very rough segmentation, while the right image has much more detail. In both cases, regions less than 20 pixels were eliminated.

5 Watershed Segmentation

A watershed in geography is a ridge that divides areas drained by different river systems [14]. It has catchment basins that are geographical areas that drain into a river or other body of water. If a graytone image is viewed as a topological surface in which the graytone at each pixel represents the height of the surface, then the resulting basins can define a segmentation of the image. Figure 6 shows a 1D signal with two basins (Basin 1 and Basin 2) and the watershed that separates them.

Figure 5: Two mean-shift clustering results with different parameters. Left: $\sigma_s = 50,\ \sigma_r = 5.0$ Right $\sigma_s = 5,\ \sigma_r = 2.5$. Both: minimum region size = 20.

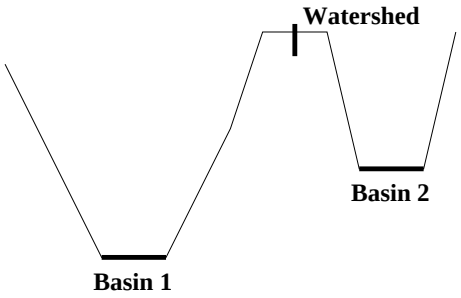

Figure 6: Example of watershed basins for a 1D signal.

However, the original graytone image will have too many different heights to produce a practical segmentation. Instead, the watershed concept can be used with transformations of the graytone image. In particular, the gradient magnitude is commonly used instead of the original image, and preprocessing is invoked to smooth the gradient image, in order to further control the size and number of regions. Furthermore, one popular variant (described below) uses foreground and background markers to aid the segmentation.

The procedure for marker-controlled watershed segmentation using gradients comes from [24] and was intended for images of multiple dark blobs. It uses operations of mathematical morphology to place foreground markers in the blobs and background markers for the areas without blobs. It can be applied to more general scenes with changes in the parameters, but it may then require some user

interaction to obtain desired results. The main steps of the procedure are as follows.

1. Convert the color image to graytone.
2. Compute the gradient magnitude image from the graytone image.
3. Mark the foreground objects (either interactively or automatically)
4. Mark the background (either interactively or automatically)
5. Modify the gradient magnitude image so that its regional minima occur at foreground and background marker pixels.
6. Apply the watershed transform to the modified gradient image.

The automatic marking of the foreground objects is done via mathematical morphology. The first step, called opening-by-reconstruction, is done by a grayscale erosion of the original grayscale image followed by a grayscale reconstruction. Grayscale erosion replaces the value of each pixel by the minimum value in its neighborhood as defined by the shape of the structuring element. Grayscale reconstruction uses grayscale dilation, which replaces each pixel by the maximum value in its neighborhood iteratively until there are no more changes to the image. The second step, called closing-by-reconstruction, is done by a grayscale dilation of the result of the first step followed by a a grayscale complement operation. The third step is the computation of regional maxima of the result of the second step plus some noise removal, which yields the foreground markers. The automatic marking of the background first thresholds the original grayscale image and then thins the background by computing a skeleton. The gradient magnitude image is then modified as explained in step 5 above.

We ran three different versions of watershed segmentation. In the first version, we did not place any markers and obtained what is called an oversegmentation into small regions. In the second version, we used the automatic marking algorithm with structuring element of size 20 and obtained a rough segmentation that did not separate the building from the plaza below it. In the third version,

we manually added a marker on the building and obtained a rough segmentation with the building separated from the plaza. Figure 7 shows the three different results.

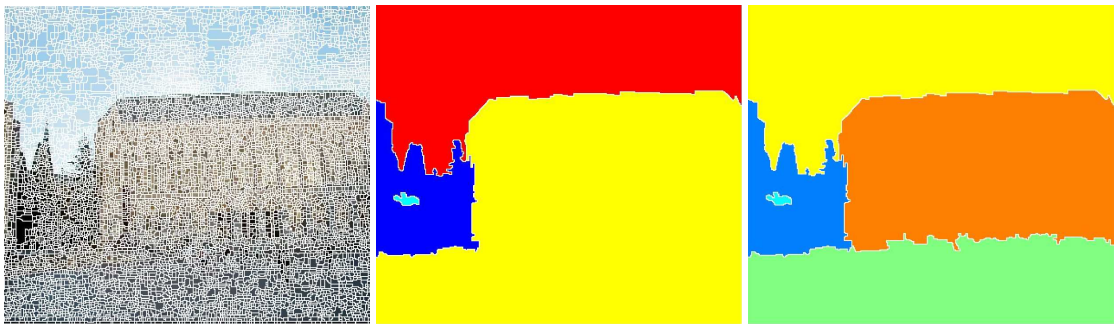

Figure 7: Three watershed clustering results. Left: watershed segmentation without markers; Middle: watershed segmentation with automatic markers; Right: Watershed segmentation with automatic markers plus a manual marker added for the building.

6 Normalized Graph Cuts

Spectral clustering is a kind of clustering that uses eigenvectors of the data in the clustering procedure. Normalized graph cuts is a kind of spectral clustering developed particularly for image segmentation [29]. In this approach, the pixels of the image form the nodes of a graph whose weighted edges represent similarity (in gray tone, color, or other attributes) between pixels, and the algorithm *cuts* the graph into two subgraphs.

Let $G = (V, E)$ be a graph whose nodes are points in measurement space and whose edges each have a weight $w(i, j)$ representing the similarity between nodes i and j. The goal in segmentation is to find a partition of the vertices into disjoint sets $V_1, V_2, \ldots, V_m$ so that the similarity within the sets is high and across different sets is low. The graph can be partitioned into two disjoint graphs with node sets A and B by removing any edges that connect nodes in A with nodes in B. The degree of dissimilarity between the two sets A and B can be computed as the sum of the weights of the edges that have been removed; this total weight is called a *cut*.

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v) \tag{7}$$

One way of formulating the segmentation problem is to look for the *minimum cut* in the graph, and to do so recursively until the regions are uniform enough. The minimum cut criterion, however, favors cutting small sets of isolated nodes, which is not useful in finding large uniform color/texture regions. The *normalized cut* (Ncut) is defined in terms of $cut(A, B)$ and the *association* between A and the full vertex set V defined by:

$$asso(A, V) = \sum_{u \in A, t \in V} w(u, t) \tag{8}$$

The definition of normalized cut is then

$$Ncut(A, B) = \frac{cut(A, B)}{asso(A, V)} + \frac{cut(A, B)}{asso(B, V)} \tag{9}$$

With this definition, the cut that partitions out small isolated point sets will not have small $Ncut$ values, and the partitions that do produce small $Ncut$ values are more likely to be useful in image segmentation. The procedure for finding the minimum normalized cut is as follows:

1. Let $G = (V, E)$ be the weighted graph, and let N be the size of its nodeset V. Define the vector d with $d(i)$ given by

$$d(i) = \sum_{j} w(i, j) \tag{10}$$

   so that $d(i)$ represents the total connection from node i to all other nodes. Let D be an $N \times N$ diagnonal matrix with d on its diagonal. Let W be an $N \times N$ symmetrical matrix with $W(i, j) = w(i, j)$.

2. Let x be a vector whose components are defined by

$$x_i = \begin{cases} 1 & \text{if node } i \text{ is in } A \\ -1 & \text{otherwise} \end{cases} \tag{11}$$

and let y be the continuous approximation to x defined by

$$y = (1 + x) - \frac{\sum_{x_i>0} d_i}{\sum_{x_i<0} d_i}(1 - x). \tag{12}$$

Solve the system of equations

$$(D - W)y = \lambda Dy \tag{13}$$

for the eigenvectors y and eigenvalues λ.

3. Use the eigenvector with the second smallest eigenvalue to bipartition the graph to find the splitting point such that $Ncut$ is minimized.[1]

4. Decide if the current partition should be subdivided further by checking the stability of the cut and making sure that $Ncut$ is below a pre-specified threshold value.

5. Recursively repartition the segmented parts if necessary.

The edge weights $w(i, j)$ were defined in [29] by

$$w(i,j) = e^{\frac{-\|F(i)-F(j)\|_2^2}{\sigma_I^2}} * \begin{cases} e^{\frac{-\|X(i)-X(j)\|_2^2}{\sigma_X^2}} & \text{if } \|X(i) - X(j)\|_2 < r \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

where $X(i)$ is the spatial location of node i, $F(i)$ is the feature value based on intensity, color, or other information, and σ_I^2 and σ_X^2 are the variances of the image features and spatial locations,

[1]The second smallest eigenvector of the generalized eigensystem (13) is the real-valued solution to the normalized cut problem.

respectively. Note that the weight $w(i, j)$ is set to 0 for any pair of nodes i and j that are more than a prespecified number r of pixels apart.

Figure 8 illustrates the operation of the normalized graph-cut algorithm on the building image.

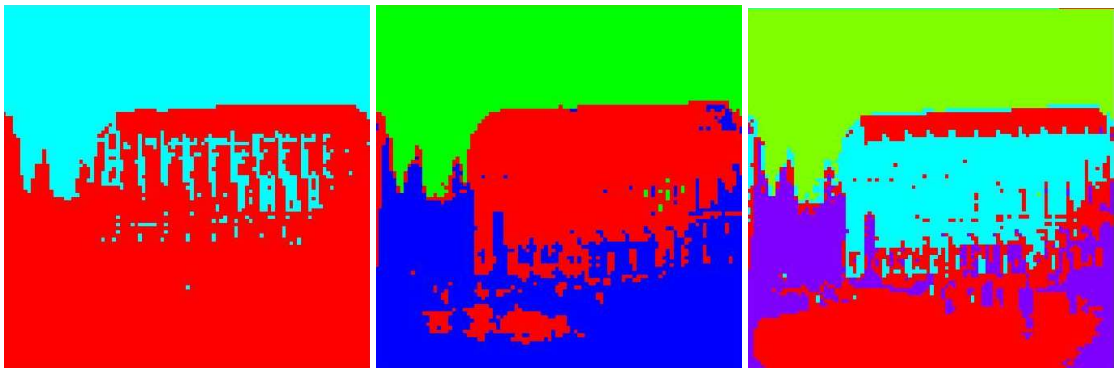

Figure 8: Three results from normalized graph cuts. Left: first cut; Middle: second cut; Right: third cut.

7 Interactive Image Segmentation

Interactive image segmentation algorithms provide flexibility and precision to object segmentation by incorporating user inputs. The ease of use and speed are two important aspects. Two approaches will be introduced in this section: (1) an effective image segmentation approach with a novel automatic boundary refinement procedure, and (2) fast image segmentation by discriminative clustering.

7.1 Interactive Image Segmentation with Automatic Boundary Refinement

Interactive image segmentation algorithms produce better object boundaries with more user inputs. To reduce the amount of user inputs, an effective automatic boundary refinement algorithm is proposed [22]. The over-segmentation is first performed to segment the image into many small regions. Then for the non-boundary part of the object, the over-segmented regions are merged using the mean color of each region in CIELab space as the feature. For the regions near the initial boundary,

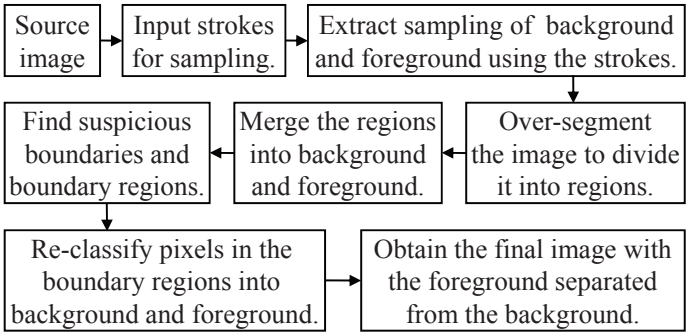

Figure 9: Work flow of the method in [22].

low-contrast object boundaries are detected by adaptively thresholding the boundary regions. These low-contrast object boundaries are treated as possibly erroneous boundary regions and relabeled by incorporating local information from pixels and global information from those regions.

Figure 9 shows the work flow of our approach. The strokes are first input to extract a sampling of foreground and background of the source image. After over-segmenting the source image to generate many regions, the resulting regions are merged into background and foreground using the maximal-similarity-based region merging (MSRM) rule [25], producing the initial image segmentation. Next, suspicious low-contrast object boundaries are detected. Pixels in those boundary regions are re-classified to determine to which class the boundary region belongs, and the region is re-labeled if necessary. After all suspicious boundary regions are processed, the final segmented image is obtained. Details of the approach are as follow, with a detailed example shown in Figure 10.

7.1.1 Merge over-segmented regions

As in the approaches of [25] and [19], the image is first over-segmented. Since the mean-shift algorithm [7] preserves the boundaries well, it is used for the initial over-segmentation. After the user marks the foreground and background regions with short strokes, the background regions are merged using the MSRM rule [25]. However, instead of using computationally expensive color histograms, the mean color of each region is used. Then the initial labeling of each region, either foreground (marked as 1) or background (marked as 0), is generated. The size of the smallest over-segmented regions can be adjust by a parameter s, which is set as 10 in the experiments to generate

results in Figure 10 and Figure 11.

7.1.2 Detect possibly erroneous low-contrast object boundary regions

To find candidate regions for more careful analysis, only certain regions at the boundary between the foreground and the background need to be considered. Those regions have similar colors but different labels as described in the section 7.1.1. The boundary regions U_{bd} are defined as the regions that have at least one neighboring region with a different initial labeling. For example, if a foreground region A_i has a neighbor B_j that is marked as background, then A_i and B_j are boundary regions. Such regions are required to share a boundary that is at least 4 pixels long.

For each boundary region, the mean color μ_{bd} is calculated, and the neighbor of opposite label with the most similar mean color is found. All the boundary regions U_{bd} are sorted according to their minimal color differences $d_{bd}^{i,j}$. Then the regions with $d_{bd}^{i,j} < d_{Thresh}$ are selected as the possibly erroneous low-contrast object boundary regions to be refined. The threshold d_{Thresh} is simply the median color difference over all boundary region pairs such that one region is foreground and the other one is background.

7.1.3 Refine possibly erroneous boundary regions

After all possibly erroneous low-contrast object boundary regions are detected, they are analyzed and reclassified. The assumption is that the initial segmentation using the mean-shift algorithm includes the correct region boundary. Using the local and global information of the pixels inside each region, each pixel is classified to be foreground or background. Then the number of foreground and background pixels are counted inside each region. If one region has more foreground pixels, it is classified as a foreground region, otherwise as background.

This problem can be formulated as a binary labeling problem of all the pixels belonging to the suspicious low-contrast object boundary regions, with different energy terms than previous work [19]. The algorithm not only considers the pixels' similarities to their neighbors, but also their similarities

to the regions marked by the user in terms of color means and standard deviations.

Suppose all the pixels in the suspicious low-contrast object boundary regions form a graph $G = \langle \upsilon, \varepsilon \rangle$, where υ is the set of all pixels and ε is the set of all arcs connecting the four adjacent pixels. The algorithm assigns a unique label x_i for each pixel $i \in \upsilon$, where $x_i = 1$ if i belongs to foreground and $x_i = 0$ if i belongs to background. The energy function to minimize is

$$E(X) = \sum_{i \in \upsilon} E_1(x_i) + \lambda \sum_{(i,j) \in \varepsilon} E_2(x_i, x_j), \tag{15}$$

where E_1 is likelihood energy, E_2 is prior energy, and $\lambda = 1$ in our experiments. E_1 encodes the color similarity of a pixel to the marked foreground or background, taking into account the color standard deviation (std) within the regions, which can be regarded as the simplest texture information. They are also used to weight the color difference and std difference.

For each pixel i, suppose $C(i)$ is its color, μ_n^F is the mean color of marked foreground regions, μ_m^B is the mean color of marked background regions, and $\sigma(i)$ is the std of the region which it belongs to. σ_n^F is the foreground region std, and σ_m^B is the background region std. The following distances are computed:

$$\begin{cases} dm_i^F &= min_n \parallel C(i) - \mu_n^F \parallel \\ dm_i^B &= min_m \parallel C(i) - \mu_m^B \parallel \\ d\sigma_i^F &= min_n \parallel \sigma(i) - \sigma_n^F \parallel \\ d\sigma_i^B &= min_m \parallel \sigma(i) - \sigma_m^B \parallel \end{cases}. \tag{16}$$

Then, $E_1(x_i)$ is defined as follows:

$$E_1(x_i = 1) = \frac{z^F}{z^F + z^B} \tag{17}$$

$$E_1(x_i = 0) = \frac{z^B}{z^F + z^B}, \tag{18}$$

where $z^X = \frac{1}{\sigma(i)\sigma_n^X} dm_i^X + \sigma(i)\sigma_n^X d\sigma_i^X$ for $X \in \{F, B\}$.

(a) (b) (c) (d) (e)

Figure 10: Comparisons between the method in [25] and the proposed algorithm. (a) The source images with user input strokes; blue strokes are used to mark the background regions, and red strokes are used to mark the foreground regions. (b) Results obtained by the method in [25]. (c) Enlarged boundary region of the results obtained by method in [25]. (d) Results obtained by the proposed algorithm. (e) Enlarged corrected boundary region by the proposed algorithm.

Because the marked boundary regions have been eliminated from Section 7.1.2, the Likelihood energy is different from that of equation (2) in [19]. The std of regions is also included, and the intuition is that if the region has high color std, d_{σ_i} is more important in the comparison, otherwise d_{m_i} is more important.

As in the methodology of [19], E_2 is the energy due to the gradient along the object boundary. It also acts like a smoothing term, enforcing that similar neighboring pixels have the same label.

$$E_2(x_i, x_j) = \frac{|x_i - x_j|}{(\|C(i) - C(j)\|^2 + 1) \times scale}. \tag{19}$$

The difference is that the energy between pixels belonging to different regions is scaled down, so that the cut through the region boundary is facilitated. In the experiment, $scale = 1$ if pixel i and j belong to the same region, $scale = 2$ otherwise.

There are many methods to minimize the energy function. The max-flow library [2] is one of the most efficient. Note that the assumption that the initial over-segmentation includes all correct boundary segments is crucial for good results.

Figure 11 shows the results of applying this methodology to the building image.

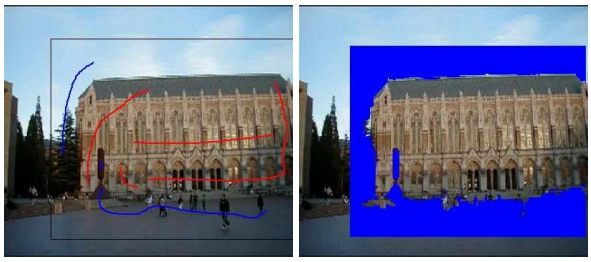

Figure 11: (a) The image with user input strokes; blue strokes are used to mark the background regions, and red strokes are used to mark the foreground regions. (b) Result obtained by the proposed algorithm.

7.2 Fast Interactive Image Segmentation by Discriminative Clustering

Unlike all the methods using complex global optimization of energy functions[1, 16, 8], the interactive image segmentation algorithm proposed in [21] begins with an initial over-segmentation using the mean shift algorithm and follows this by discriminative clustering and local neighborhood classification. It is straightforward to implement, much faster than graph cuts, and can be further speeded up by an image pyramid and boundary refinement procedure [20].

The algorithm in [21] contains the following steps and is illustrated in Figure 12.

7.2.1 Pre-segmentation by the mean-shift algorithm

After the user input strokes to specify the foreground and the background, the selected image is first over-segmented using the mean-shift algorithm [7]. There are three reasons for choosing the mean-shift algorithm for the initial over-segmentation. First, it is observed to preserve the boundaries better than other methods [10, 27]. Second, its speed has been improved significantly in recent years [12, 26]. Third, there are fewer parameters to tune, and our algorithm is not sensitive to the change of parameters as long as they are within a reasonable range.

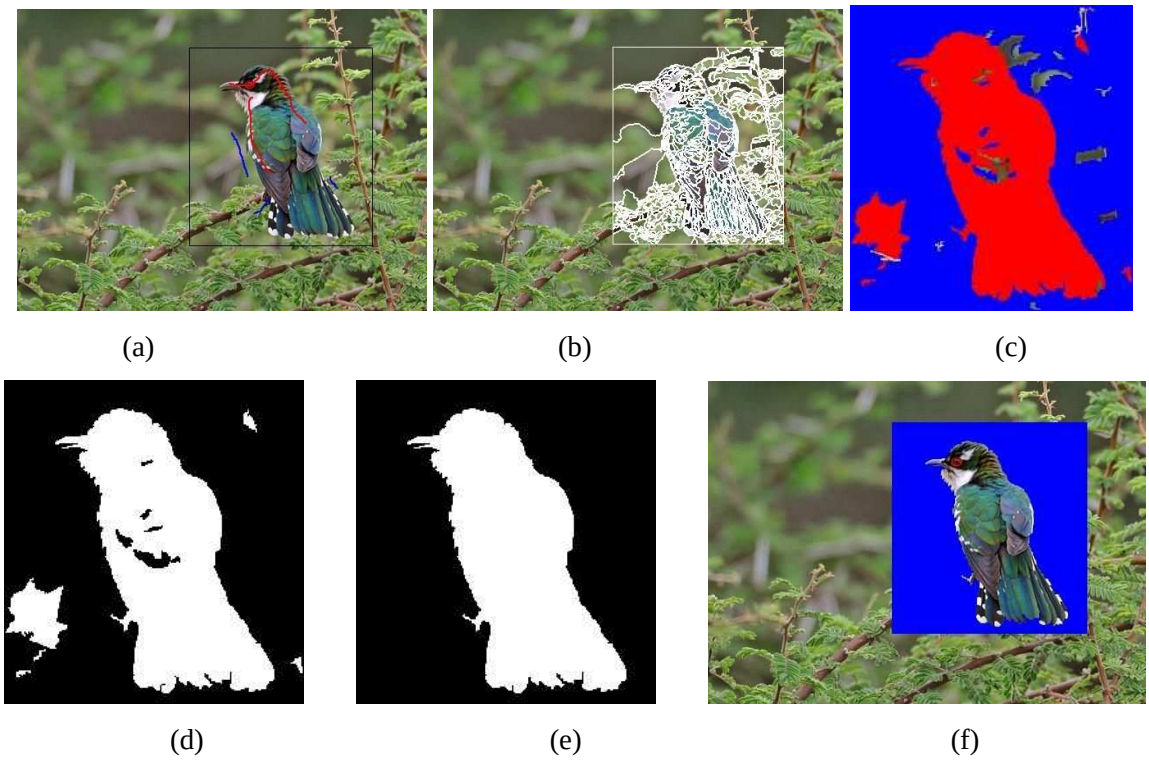

Figure 12: Steps of the proposed algorithm: (a) Original image with user input. (b) Over-segmentation by the mean-shift algorithm. (c) Result after discriminative clustering, with red indicating foreground, blue indicating background, and original color indicating ambiguous regions. (d) Result after merging ambiguous regions from (c). (e) Result of local neighborhood pruning. (f) Final result on color image with bounding box shown.

7.2.2 Merge regions by discriminative clustering

Region merging is the critical step that makes our algorithm an order of magnitude faster than other methods after the pre-segmentation. After the user input is given, the algorithm calculates the mean color of marked foreground and background regions. Suppose there are N_f foreground regions M_f marked by the user, and N_b marked background regions M_b. For each region $A_i \in M_f$ and $B_j \in M_b$, their mean colors μ_{A_i} and μ_{B_j} in CIELab color-space are computed. The minimal difference between the mean colors of the marked foreground and background regions is also computed to be taken as the threshold d_{thresh} for later clustering usage. That is, $d_{thresh} = min(\mu_{A_i} - \mu_{B_j})$. If d_{thresh} itself is smaller than a threshold λ, then it is doubled. The intuition is that if some marked foreground and background segments have very similar colors, more ambiguous regions need be

further processed (Section 7.2.3). In the implementation to output Figure 12, λ is set to 2.

Two three-dimensional kd-trees are constructed, one for each type of marked region. In the foreground kd-tree, each node stores the mean color of one marked foreground region, and in the background kd-tree, each node stores the mean color of one marked background region. For each unmarked region R_u, the algorithm takes its mean color as a query point in the 3D space and does a nearest neighbor search in those two kd-trees. This quickly returns the marked regions A_{nn} and B_{nn} that have the most similar mean color to it, and the color differences d_{uf} and d_{ub}.

Let $dif_d = d_{uf} - d_{ub}$. If $dif_d > d_{thresh}$, R_u is marked as background. If $dif_d < -d_{thresh}$, R_u is marked as foreground. The output of this step on the bird image is shown in Figure 12 (c), where the foreground regions are marked red, background regions are blue, and the ambiguous regions are just marked with their original colors. Some regions with dark green remain ambiguous.

7.2.3 Local neighborhood region classification and pruning

After initial cluster merging, there may still be unmarked regions that are not classified as either foreground or background. Whereas in the first step the unmarked regions are classified only by their mean colors without considering the spatial information, the local neighborhood information is utilized to better prune the segmentation result. There are two steps:

1. If there are N_u remaining unmarked regions from the first processing step, each of them is assigned the label of the most similar of its neighboring regions in terms of their mean color.

   If the most similar neighboring region is also an unmarked region, they are merged together to become a new unmarked region and the process repeats.

   If there is a tie again in terms of the most similarly labeled neighboring region, the label of the region that has the most similar color variance is used.

   For the bird image, this step generates the image shown in Figure 12 (d), with white indicating foreground and black indicating background.

2. After all the regions have been marked as either foreground or background, a connected components algorithm [28] is applied to find isolated foreground or background regions that are surrounded by regions of the opposite labeling. The labels of isolated regions are changed to the opposite label only when the following conditions are satisfied:

   (a) The region was not marked by the user.

   (b) The region is not the biggest region with that label.

   (c) The region is smaller than its surrounding regions.

For the bird image, this step generates the image Figure 12 (e). This mask is used to cut out the object as shown in Figure 12 (f).

With different user inputs, this algorithm can help the user segment out different objects in the building image. Figure 13 shows two different sets of user inputs and the corresponding results.

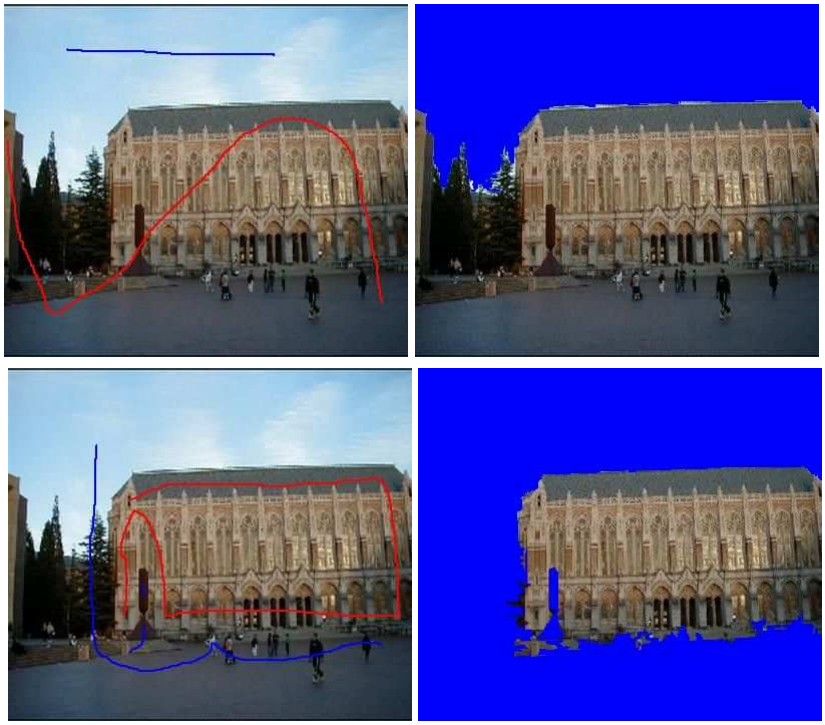

Figure 13: (a) Original image with user input. (b) Segmentation results.

The first user marked the building, trees, and some of the pedestrian area as foreground and only the sky as background. The resulting segmentation retained the building, the trees, and all of the pedestrian area. The second user marked the building as foreground, but she marked the trees, pedestrian area, and sky as background. In this case the building was segmented from the image. Not only the trees on the left of the building, but a tree in front of the building was removed.

8 Conclusions

In this paper, we have described the most common segmentation algorithms currently used by the computer vision community for possible use in remote sensing. The K-means algorithm is the simplest and most commonly used algorithm, and the EM clustering algorithm is a probabilistic version of K-means. They both require the user to preselect the number of clusters K. The mean-shift algorithm has become popular because it does not require the number of clusters to be selected, but it has its own parameters that also control the number of and sizes of the regions. Watershed segmentation is based on the idea of catchment basins in geography. The watershed segmentation algorithm has mainly been used for fairly simple foreground-background segmentation, such as images of dark blobs (ie. cell nuclei) on a light background. It does not seem useful for remote sensing applications. Normalized graph cuts is a very powerful algorithm that is now used heavily in computer vision applications. It finds the best place to cut the graph the represents image pixels (or small regions) connected by edges representing their similarity and iterates until a threshold is reached on the normalized cut value. It has been tried on multiple different forms of data (gray tone images, color images, videos) and should be consided for remote sensing applications. Finally, interactive segmentation can be used when automatic methods are not accurate enough for the desired application. We discuss interactive segmentation methods from our own recent research and show results both on general images and on the building image used to show results of all the methods throughout this paper.

References

[1] Y. Boykov and M.P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in ND images. In *International Conference on Computer Vision*, volume 1, pages 105–112. Citeseer, 2001.

[2] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1124–1137, 2004.

[3] W. Burger and M. J. Burge. *Digital Image Processing: An Algorithmic Introcution using Java*. Springer, 2008.

[4] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1026–1038, 2002.

[5] D. Comaniciu and P. Meer. Mean shift analysis and applications. In *IEEE International Conference on Computer Vision*, pages 1197–1203, 1999.

[6] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5), 2002.

[7] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 603–619, 2002.

[8] A. Criminisi, T. Sharp, C. Rother, and P. P'erez. Geodesic image and video editing. *ACM Transactions on Graphics (TOG)*, 29(5):134, 2010.

[9] A. P. Dempster, N. M. Laird, and D. B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39(1):1–38, 1977.

[10] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[11] M. Fischler and R. Bolles. RANdom SAmpling consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 1981.

[12] D. Freedman, P. Kisilev, and I. Haifa. Fast Mean Shift by Compact Density Representation. In *Proc. CVPR*, pages 1818–1825, 2009.

[13] D. g. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 2004.

[14] R. C. Gonzalez, R. E. Woods, and S. L. Eddins. *Digital Image Processing using MATLAB*. Gatesmark, LLC, 2009.

[15] G. Gradski and A. Kaehler. *Learning OpenCV: Computer Vision withthe OpenCV Library*. O'Reilly Media, 2008.

[16] L. Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1768–1783, 2006.

[17] R. M. Haralick and L. G. Shapiro. Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, 29(1):100–132, 1985.

[18] S. L. Horowitz and T. Pavilidis. Picture segmentation by a tree traversal algorithm. *Journal of the ACM*, 23(2):368–388, 1976.

[19] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *ACM Trans. Graph.*, 23(3):303–308, 2004.

[20] D. Liu. *Intelligent Interactive Image Segmentation for Camera Phones*. PhD thesis, UNIVERSITY OF WASHINGTON, 2011.

[21] D. Liu, K. Pulli, L.G. Shapiro, and Y. Xiong. Fast Interactive Image Segmentation by Discriminative Clustering. In *Proceedings of the 2010 ACM Multimedia Workshop on Mobile Cloud Media Computing*, pages 47–52. ACM, 2010.

[22] D. Liu, Y. Xiong, L. Shapiro, and K. Pulli. Robust interactive image segmentation with automatic boundary refinement. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 225–228. IEEE, 2010.

[23] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Mathematical Statistics and Probability*, pages 281–297. University of California Press, 1967.

[24] F. Meyer. Topographic distance and watershed lines. *Signal Processing*, 38(1):113 – 125, 1994.

[25] J. Ning, L. Zhang, D. Zhang, and C. Wu. Interactive image segmentation by maximal similarity based region merging. *Pattern Recognition*, 43(2):445–456, 2010.

[26] S. Paris and F. Durand. A topological approach to hierarchical segmentation using mean shift. In *Proc. CVPR*, pages 1–8, 2007.

[27] X. Ren and J. Malik. Learning a classification model for segmentation. In *Proc. ICCV*, pages 10–17, 2003.

[28] L. Shapiro and G. Stockman. *Computer Vision*. Prentice Hall, 2002.

[29] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.