MULTI-RESOLUTION REGION-PRESERVING SEGMENTATION FOR COLOR IMAGES OF NATURAL SCENE

GUO JUGUI

NATIONAL UNIVERSITY OF SINGAPORE
2004

Name: GUO JU GUI
Degree: Master of Science
Dept: Computer Science
Thesis Title: Multi-resolution region-preserving segmentation for color images of natural scene

Abstract

Image segmentation is one of the primary steps in image analysis for image labeling and retrieval. Recent Segmentation methods have shown a strong interest in graph based algorithm, and they have been quite successful in identifying significant regions and their boundaries. The cost functions used in these graph algorithms are usually based on low-level pixel-based image features such as position, intensity, and color. These methods tend to produce over-segmented results, especially for images of natural scenes whose regions contain complex but coherent mixture of colors. This thesis describes a multi-resolution segmentation algorithm which first constructs a region pyramid that preserves the color distributions of regions, and then applies a graph cut algorithm at the top level of the pyramid to identify main regions in the image, and finally refines the region boundaries with a top-down approach based on integer linear programming. This way, main image regions are identified while over-segmentation is minimized.

Keywords: Image segmentation
Graph Cut
Image pyramid

MULTI-RESOLUTION REGION-PRESERVING SEGMENTATION FOR COLOR IMAGES OF NATURAL SCENE

GUO JU GUI
(B. Sc. (Hon.) in Computer Science, NUS)

A THESIS SUBMITTED
FOR THE DEGREE OF MASTER OF SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF COMPUTING
NATIONAL UNIVERSITY OF SINGAPORE
2004

Acknowledgments

I would like to express my gratitude to my project supervisor, A/P Leow Wee Kheng, for providing his timely advice and guidance during the course of my honours and masters years. I would also like to express my thanks to A/P Leong Hon Wai for his enlightening discussions and advices.

I would like to thank my lab mates, Rui Xuan, Chen Ying, Indri, Saura and Henna for their help and support. Lastly, I would like to express my gratitude to Kenny, my housemates and my family for their continuous support.

Contents

List of Figures

List of Tables

Summary

Image segmentation is one of the primary steps in image analysis for image labeling and retrieval. Recent Segmentation methods have shown a strong interest in graph based algorithm, and they have been quite successful in identifying significant regions and their boundaries. The cost functions used in these graph algorithms are usually based on low-level pixel-based image features such as position, intensity, and color. These methods tend to produce over-segmented results, especially for images of natural scenes whose regions contain complex but coherent mixture of colors.

This thesis describes a multi-resolution segmentation algorithm which first constructs a region pyramid that preserves the color distributions of regions, and then applies a graph cut algorithm at a coarse level of the pyramid to identify main regions in the image. The coarse region boundaries found are refined using Dynamic Programming and Integer Linear Programming, and propagated down to the lowest level by a greedy method. Experimental results show that this approach can identify the main regions in many images and minimize over-segmentation.

Chapter 1

Introduction

1.1 Motivation

Image segmentation is one of the primary steps in image analysis such as image labeling and object identification. The resulting regions will represent certain semantic contents and may indicate the presence of objects or object parts, which can be verified later with an image analysis and recognition step.

Image segmentation is also an important tool for content-based image retrieval (CBIR). Each extracted region in the segmentation step contains a different region content which could be a combination of color, texture, brightness and spatial information. These information provide a natural link between the contents of the query images and those of the images in the database, which enables an accurate retrieval in response to the user's query. Recent CBIR system [8] could even allow the user to access the segmentation result of the query image and specify which aspects of the image are important to the query. Such interactions have greatly

assisted in query refinement and improved the performance of image retrieval.

1.2 Research Goal

This thesis addresses the image segmentation problem in the context of semantic labeling and image retrieval. In these application contexts, it is desirable to partition an image into semantically consistent regions. Especially in natural scene images, each region can contain a complex but coherent mixture of colors. Therefore, we can assume that a coherent color distribution provides a good indication of semantic consistency.

This thesis proposes a multi-resolution region preserving segmentation approach on color images. The resulting segmentation should have the following properties:

1. Each region is a closed connected component. This is essential to ensure the spacial consistency of each region.

2. Each region is of a significant size compared to the image size. Thus, only main regions are extracted.

3. Each region will have a coherent distribution of colors. This is a desirable property to bring about the semantic consistency of each region.

1.3 Overview of Proposed Algorithm

The proposed algorithm can be divided into three main steps (Figure. 1.1):

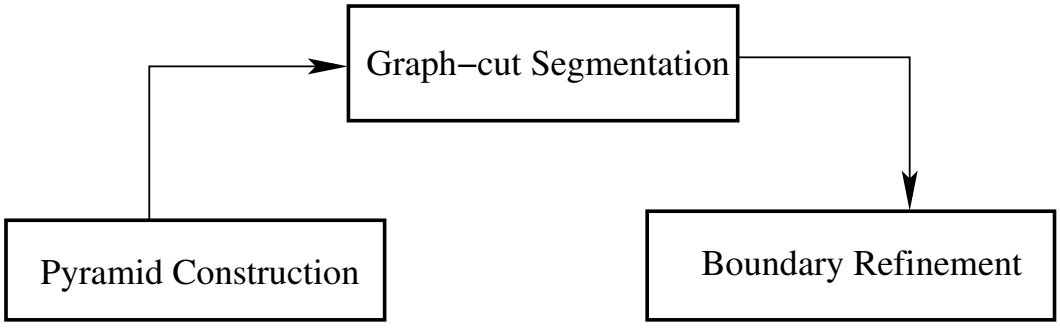


Figure 1.1: An overview of the segmentation algorithm.

1. Pyramid Construction: A region pyramid is constructed to capture the color distributions of image blocks at various resolutions. In a conventional image pyramid, each image block contains information of only a single mean color or texture. In the region pyramid introduced in this thesis, each block in the pyramid captures the color distributions of a region in the original image. Thus, we call the constructed pyramid a *region pyramid*. This step aims at preserving the information of color distributions of the image blocks at various levels of resolutions. That is, the number of image blocks is reduced at a lower resolution, but the color distributions are preserved in the image blocks.

2. Graph-Cut Image Segmentation: Perform segmentation based on graph-cut algorithm at a higher level in the region pyramid, which has a lower resolution, so that the main regions in the image can be identified.

3. Boundary Refinement: Refine the region boundaries obtained at step 2 top-down to the finest level to obtain the final segmentation result. This refinement process preserves the color distributions and the locations of the

regions obtained at step 2.

1.4 Thesis Overview

This section will give an overview of the thesis: Chapter 2 will introduce some background and related approaches on image segmentation. The proposed approach is discussed in detail in Chapters 3, 4 and 5. Chapter 6 will demonstrate some experimental results and illustrate the difference between the proposed method and some existing methods. Chapter 7 will suggest some future work, summarize the contributions and conclude this thesis.

Chapter 2

Related Work

Image segmentation is one of the most challenging problems in computer vision and has been studied from a wide variety of perspectives. But, no sufficiently rigorous and general solution to this problem is available. Techniques proposed include histogram thresholding, which is used for gray scale images; edge detection, region growing and splitting, clustering, and general optimization as well as graph-based optimization approaches which could be applied to both gray scale images and color images.

2.1 Traditional Approaches for Color Image Segmentation

The general segmentation methods for color images can be grouped into four main categories: Edge detection, region growing and splitting, clustering, and non-graph-based optimization methods.

Edge detection techniques [7, 19, 20] first perform filtering on the image to remove the noise in the image. Then, an edge detection algorithm such as LoG or Sobel filter is applied to generate an edge map. But the edge map just indicates the possible locations of the region boundaries. Further processing is needed to link the edges into closed boundaries and to remove unwanted line segments. The linking process could be carried out given a model, which is usually not available for real images.

Region growing and splitting aims to detect connected sets of pixels, that satisfy certain predefined homogeneity criteria, such as intensity consistency and color coherence. For region growing or merging techniques, input images are divided into a set of primitive regions, then an iterative process is carried out to repeatedly merge neighboring regions that are similar in features together into larger regions [1, 6, 11, 13]. Region splitting techniques work in the opposite way. The entire image is initially considered as one region. In the subsequent steps, regions are recursively split into more homogeneous regions.

The region-based algorithms are computationally more expensive than the edge detection techniques. But they can utilize several image properties directly and simultaneously to determine the region boundaries. Region merging has been the most popular approach in segmentation and is also used as a part of more comprehensive approaches.

Clustering methods perform grouping of pixels in the feature space, e.g., color space [9, 26]. The current histogram grouping algorithms have also taken into account local spatial features [21, 22]. They compute local color histograms of

each pixel and group the histograms into a fixed number of prototypical color-distribution models using Bayesian Theory. These methods typically require the features (e.g., color) to be quantized into a small number of intervals or bins so that the estimation of probability functions can be done. Therefore, they are more applicable to images with less complex distributions of colors.

Optimization techniques define a global function that measures the goodness of the segmentation result and seek to optimize the result. Examples of these techniques include Bayesian and Markov random fields methods [2, 5, 34, 36]. In Markov random field methods, the image is assumed to be a realization of a Markov or Gibbs random field function with a distribution that captures the spatial context of the scene. The commonly used statistical estimation principles like maximum a posteriori (MAP) estimation, maximization of the marginal probabilities (ICM) are used to minimize the difference between the given prior distribution of an image model and the segmented image. However, these methods require fairly accurate knowledge of the prior true image distribution and most of them are computationally expensive.

2.2 Graph-Theoretic Approach

The graph-theoretic approach is a newer optimization approach. The input image is represented as a graph, where the vertices of the graph are the pixels in the input image, and for every pair of neighboring pixels, an edge is formed between the corresponding pair of vertices. The cost of each edge is a function of the similarity between each adjacent pair of vertices. A partition of the vertices that minimizes

certain cost function will form a natural segmentation on the image [3, 30, 35].

Wu and Leahy [35] were the first to introduce the general approach to graph-cut algorithms and their algorithm has a polynomial time complexity. Their *minimum cut* algorithm formulates the cost function as the sum of the edge costs along the region boundary and aims to minimize this cost. Therefore, it is biased toward small regions which have shorter boundaries and, thus, smaller costs. Veksler [30] applied nested cut to find minimum-cost cycles around each pixel, if the cost of a cycle found is smaller than a threshold, the regions enclosed in the cycle will be grouped into those regions enclosed by a larger-cost cycle. This method requires the cost function to decrease rapidly with decreasing similarity to ease the decision of the threshold value[30]. Shi and Malik [29] and Belongie et al. [3] apply a normalized cost, instead of total cost, which is formulated as the sum of ratio of boundary cost over the total number of connections between each partition and the total area. Such a ratio will favour partitioning the image into regions of similar size [33].

Jermin and Ishikawa's method [14] finds globally optimal segmentation by determining the minimum mean (i.e., normalized) cost cycle in a directed graph. Wang and Siskind's *minimum mean cut* method [32] finds the minimum mean cost cycle in an undirected graph instead. They discovered that the use of mean cost in the graph algorithm leads to *spurious cuts* [32] (see detailed discussion in Chapter 4), which are globally optimal but not perceptually satisfactory. Their method is improved in [33] by incorporating region information and heuristics to speed up the segmentation process. The above algorithms, except [3, 14, 29], use

pixel intensity as the main feature. Thus, they are sensitive to salt-and-pepper noise and tend to over-segment the images [33].

The graph-theoretic approach has made the optimization approach achievable in polynomial time [32, 35]. Among the techniques discussed, MMC does not introduce explicit bias toward region size or length. Therefore, it will be adopted as part of our segmentation algorithm. Notice that MMC has only been applied to grayscale images and it uses only low-level image intensity in the segmentation process. The regions generated from MMC tend to be too fragmented for image labeling. We adapt this algorithm for segmentation at a lower resolution level to produce more semantically consistent regions.

2.3 Multi-Resolution Approach

Multi-resolution is a technique that constructs an image pyramid and applies the segmentation process at different levels of the pyramid. The initial segmentation is obtained at a coarser level, and a boundary refinement process is performed top-down to the finest level to obtain the final segmentation.

The general advantage of the multi-resolution scheme is that it provides a way to trade-off spatial resolution and robustness against noise. Repeatedly blurring and subsampling the image decreases the noise and improves the region boundary certainty, but at the expense of spatial resolution. Moreover, color variation in lower resolution images tend to be more obvious between regions. Therefore, it becomes possible to avoid inappropriate segmentations.

Examples of the multi-resolution approach include the hierarchical image seg-

mentation by Schroeter [27], which performs a clustering of texture at the coarsest level to determine the number of regions in the image. This is followed by an orientation-adaptive boundary refinement process. But this algorithm has only been applied to grayscale images. James Wang has proposed a multi-resolution approach for segmenting sharply focused object-of-interest from other foreground or background objects [31]. It employs the average intensity and wavelet coefficients in the high frequency bands to distinguish between the background and the object of interest. The method of Salembier [25] first groups pixels into many small regions based on similarity estimation of some generic features such as color homogeneity. These regions are characterized by the mean color values within the regions. Then these initial regions are grouped in various combinations into a hierarchical grouping. This hierarchical grouping can support different kinds of segmentation applications which require different details in the segmentation results. The multiscale segmentation method introduced by Sharon [28] performed an approximated normalized cut at a higher level of the image pyramid followed by a boundary sharpening step. The JSEG [11] algorithm first quantizes the colors in an image into several clusters, and the color of each pixel in the image is replaced by the corresponding cluster label. A criterion based on the distribution of the cluster labels is used to identify the initial possible boundaries and interiors of regions. Then a region growing method is used to segment the image based on the distribution of the cluster labels at different scale.

Existing multi-resolution image segmentation methods [4, 5, 11, 28, 31] characterize image regions by their mean or dominant colors and texture. However,

single mean or dominant color is not sufficient to characterize the complex mixture of colors present in the regions of natural scene images. And texture features tend to be ambiguous and not discriminative enough. Our method, on the other hand, characterizes regions by their color histograms, thus capturing the information of the color distribution of the regions more accurately than existing methods. Moreover, the region characteristics are preserved in the upper levels while the region pyramid is constructed.

2.4 Classification Approach

In the last year, a new kind of approach–the classification approach is introduced. The idea behind this approach is to train a classifier to classify good segmentation and poor segmentation results based on visual cues such as texture, brightness, contour energy and curvilinear continuity. An example of this approach is Ren's classification model [24] for segmentation which is implemented for gray-scale images. Good segmentation results are obtained from human labelled ground truth introduced in [18]. Poor segmentation results are obtained by randomly matching a human segmentation to a different image. The classifier linearly combines different features according to the training data and give scores to segmentations. Then the classifier is applied to search in the space of all segmentations to obtain an optimal segmentation.

Chapter 3

Pyramid Construction

A color histogram is a useful representation of color distribution. A simple color histogram essentially counts the number of pixels of each 'color'. The strength of a histogram representation is that it can capture the color distribution instead of a single color. In a complex color image, especially a natural scene image, each region contains a complex but coherent mixture of colors. Therefore, we can expect that the color histogram representation can capture the color region information more accurately.

The reason for adopting adaptive histogram instead of fixed binning histogram is that adaptive histograms can represent the distributions more efficiently than histograms with fixed binning [23]. Unlike fixed histograms, adaptive histograms adapt their binning schemes according to the color contents of the images. Therefore, different images will have different clusters of colors. They have been shown to yield the best overall performance in terms of good accuracy, small number of bins, and no empty bin compared to fixed-binning histograms [16]. Thus,

the use of adaptive histogram can reduce the overall memory requirement of the region pyramid.

3.1 Adaptive Color Histogram

An adaptive color histogram $\mathcal{H} = (n, C, H)$ is a 3-tuple consisting of a set C of n bins c_i, $i = 1 \ldots n$, and a set H of corresponding bin counts $h_i > 0$. The set of bins of H is also denoted as $C(H)$. Adaptive histogram is produced by adaptive binning, which determines the number of bins n and the bin counts.

3.2 Adaptive Binning

Adaptive binning is similar to k-means clustering or its variants. But the clustering algorithm is applied to the colors in an image instead of the colors in an entire color space. Therefore, adaptive binning produces different binnings for different images.

Adaptive binning groups pixels into clusters according to the distance measure d_{kp} between the centroid C_k of cluster k and pixel p with color C_p, which is defined as the CIE94 color-difference equation:

$$d_{kp} = \left[\left(\frac{\triangle L^*}{k_L S_L} \right)^2 + \left(\frac{\triangle C^*_{ab}}{k_c S_c} \right)^2 + \left(\frac{\triangle H^*_{ab}}{k_H S_H} \right)^2 \right]^{\frac{1}{2}} \tag{3.1}$$

where $\triangle L^*$, $\triangle ab^*$, and $\triangle H^*$ are the differences in light-ness, chroma, and hue between C_k and C_p, $S_L = 1 + 0.045\overline{C}^*_{ab}$, $S_H = 1 + 0.015\overline{C}^*_{ab}$, and $k_L = k_c = k_H = 1$ for reference conditions. The variable is the geometric mean between the chroma values of C_k and C_p. The CIE94 color-difference equation is used instead of the

simple Euclidean distance in CIELAB space because CIE94 is more perceptually uniform than Euclidean [16].

Adaptive binning groups a pixel p into its nearest cluster if it is near enough ($d_{kp} < R$). On the other hand, if the pixel p is far enough ($d_{kp} > D$) from its nearest neighbor, then a new cluster is created. Otherwise, it is left unclustered and will be considered again in the next iteration. The clustering process could be summarized as follows [16]:

Repeat

1. For each pixel p, find the nearest cluster k to pixel p.
   (a) If no cluster is found or distance $d_{kp} > D$, create a new cluster with p;
   (b) Else, if $d_{kp} < R$, add p to cluster k.
2. For each cluster i,
   (a) If cluster i has at least N_m pixels, update centroid c_i of cluster i.
   (b) Else, remove cluster i.

In the implementation in [**?**], this process repeats for 10 iterations, after that, the rest of the unclustered pixels are grouped into their nearest clusters.

3.3 Operations on Adaptive Color Histograms

1. Dissimilarity measure between histograms

   Since different adaptive histograms can contain different binnings, we cannot

use the traditional Euclidean distance measure. As illustrated in [16], the Earth Mover's Distance (EMD) for comparing histograms with different binnings is computationally expensive. Therefore, the weighted correlation is introduced and used instead [16]. The details of weighted correlation are explained as following.

- Bin Similarity

  The similarity $w(b, c)$ between bins b and c is given by a monotonic function inversely related to the distance $\|b - c\|$ between them. Bin similarity is symmetric $w(b, c) = w(c, b)$ and bounded: $0 \leq w(c_i, c_j) \leq 1$.

  The bins are taken to be spherical and $w(b, c)$ is defined in terms of the volume of intersection between them. In 3D, the volume of intersection $V_s(\alpha)$ between equal-sized spherical bins of radius R, separated by a distance αR, can be derived from elementary solid geometry as

$$V_s(\alpha) = V - \pi\alpha R^3 + \frac{\pi}{12}\alpha^3 R^3 \tag{3.2}$$

  where $V = 4\pi R^3/3$ is the volume of a sphere. The bin similarity is then defined as

$$w(c_i, c_j) = w(\alpha) = \frac{V_s(\alpha)}{V} = \begin{cases} 1 - \frac{3}{4}\alpha + \frac{1}{16}\alpha^3 & \text{if } 0 \leq \alpha \leq 2 \\ 0 & \text{otherwise} \end{cases}$$

  where R is the distance between bin centroids c_i and c_j. The equation approximates a Gaussian function of the form $\exp\{-\alpha^2 R^2/\sigma^2\}$ with an appropriate σ.

- Weighted Correlation

  The weighted correlation between histograms $G = (m, B, P)$ and $H = (n, C, Q)$, denoted as $G \cdot H$, is defined as

$$G \cdot H = \sum_{i=1}^{m} \sum_{j=1}^{n} w(b_i, c_j) p_i q_j \tag{3.3}$$

  where $w(b_i, c_j)$ is the bin similarity between bin b_i and bin c_j. Weighted correlation is non-negative, $G \cdot H \geq 0$, and commutative, $G \cdot H = H \cdot G$, because the bin counts g_i and h_j are non-negative and the bin similarities $w(b_i, c_j)$ are non-negative and symmetric. The null histogram O is totally uncorrelated to any non-null histogram H: $H \cdot O = 0$

- Histogram dissimilarity

  The similarity $s(G, H)$ between histograms G and H is defined as the weighted correlation between their normalized forms $s(G, H) = \overline{G} \cdot \overline{H}$. The norm $\|H\|$ of histogram H is defined as $\|H\| = \sqrt{H \cdot H}$, so the normalized histogram of a histogram H is defined as $\overline{H} = H/\|H\|$. The dissimilarity $d(G, H)$ between them is defined as $d(G, H) = 1 - s(G, H)$, and is bounded between 0 and 1.

2. Mean of Histograms

   The mean of histograms is a mean histogram which is obtained by merging the normalized histograms [16]. Let histogram $G = X \bigcup Y$ and $H = X' \bigcup Z$ such that X and X' have the same set of bin centroids and X, Y and Z have disjoint sets of bin centroids. Then, the merged histogram $G \oplus H =$

$(X \bigcup Y) \oplus (X' \bigcup Z) = (X + X') \bigcup Y \bigcup Z$. That is, two histograms are merged by collecting all the bin centroids and adding the bin counts of the bins with identical centroids. So, the mean M of histograms H_i is

$$M = \biguplus_{i=1}^{n} \overline{H}_i = \overline{H}_1 \oplus \overline{H}_2 \oplus \ \ldots \oplus \overline{H}_n. \tag{3.4}$$

3.4 Pyramid Construction

3.4.1 Image Color Quantization

The colors in input image is first clustered to obtain a small number of color clusters using the adaptive binning algorithm (Section 3.2). Then a quantization step is performed on the image by replacing the color of each pixel with the color of its nearest cluster centroid. Such a quantization process can help to reduce the complexity of the color distribution in the image and extract a few representative colors which can differentiate neighboring regions in the image. It is shown in [16] that this adaptive color quantization method incurs only a very small error in the colors of the quantized image.

3.4.2 Pyramid Construction Algorithm

The region pyramid consists of L levels of maps, each containing a number of square blocks. The highest level of $l = 1$ contains a map with a single block that represents the entire image. The lowest level of $l = L$ contains a map with each block corresponding to a pixel in the original input image. The map at level l is derived from that at level $l + 1$ by combining 3×3 lower-level blocks into one

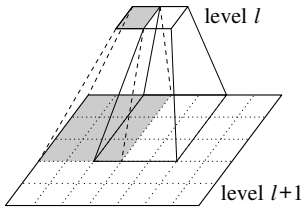


Figure 3.1: The region pyramid is constructed by combining 3×3 lower-level blocks into one higher-level block, with an overlap of one row or one column between neighboring blocks.

higher-level block, with an overlap of one row or one column between neighboring blocks in the lower-level (Figure 3.1). Therefore, the image coordinates (x_l, y_l) at level l is mapped to the coordinates at level $l + 1$ by the equations

$$(x_{l+1}, y_{l+1}) = (2\,x_l + 1, 2\,y_l + 1)\,. \tag{3.5}$$

The advantage of this coordinate mapping approach is that the center of a higher-level block maps exactly to the center of a lower-level block. On the other hand, the conventional method of combining 2×2 blocks into one block maps the center of a higher-level block to the intersecting boundaries of the 2×2 blocks.

Each block of the maps in the pyramid captures the distribution of colors within the corresponding region in the original image instead of a single mean color or dominant color of the region. Therefore, our method can capture region information more accurately than existing methods that represent each region by its mean or dominant color.

Let S_L denote either the width and the height of the input image, whichever

is smaller. Then,

$$L = \lfloor \log_2(S+1) \rfloor \tag{3.6}$$

$$S_l = \lfloor (S_{l+1} - 1)/2 \rfloor , \quad S_1 = 1 . \tag{3.7}$$

At the lowest level of $l = L$, each block corresponds to a pixel in the original image. Therefore, the histogram of such a block contains only one non-empty bin that represents the color of the corresponding pixel. On the other hand, the map at the highest level of $l = 1$ contains only one block that corresponds to the entire image. Its histogram will need to have enough color bins to capture the color distribution of the entire image accurately.

The region pyramid construction process is as follows:

Repeat for each block at (x_l, y_l) of each level $l = L-1, \ldots, 1$:

(a) Combine the histograms of the blocks at level $l+1$ into the histogram of block (x_l, y_l) as follows:

$$h_k(x_l, y_l) = \sum_{-1 \le i,j \le 1} w(i,j) h_k(x_{l+1} + i, y_{l+1} + j) \tag{3.8}$$

where h_k is the bin count of bin k of the color histogram of block (x_l, y_l) and $w(i,j)$ is a weighting factor used to prevent over counting of the bin counts of overlapping blocks and to give a higher weight to the centre block (Table 3.1).

The summation is performed over the 9 blocks at level $l+1$ that make up the corresponding block (x_l, y_l) at level l. The location of the center of the nine blocks is related to the location (x_l, y_l) by Eq. 3.5. If bin

Table 3.1: Weights for combining histograms.

0.25	0.5	0.25
0.5	2	0.5
0.25	0.5	0.25

k does not exist in the histogram of block (x_l, y_l), then it is an empty bin and its value is taken as 0.

(b) Remove empty bins and bins with very small bin counts. This is equivalent to setting the bin counts of these bins to 0. Removing these insignificant bins reduce the size of the histograms and, thus, the amount of memory required.

Figure 3.2 shows an exmaple of the region pyramid obtained. Instead of showing the histogram of each image block, the dominant color of the histogram for each block is shown.

3.5 Memory Requirement

In the current implementation of the region pyramid, the number of bins B_l of the histograms at level l is given as follows (Figure 3.3):

$$B_l = \min\left(B, 2^{L+1-l} - 1\right) \tag{3.9}$$

where B is the number of bins of the adaptive histogram for the entire input image derived using the adaptive clustering method given in section 3.2. The

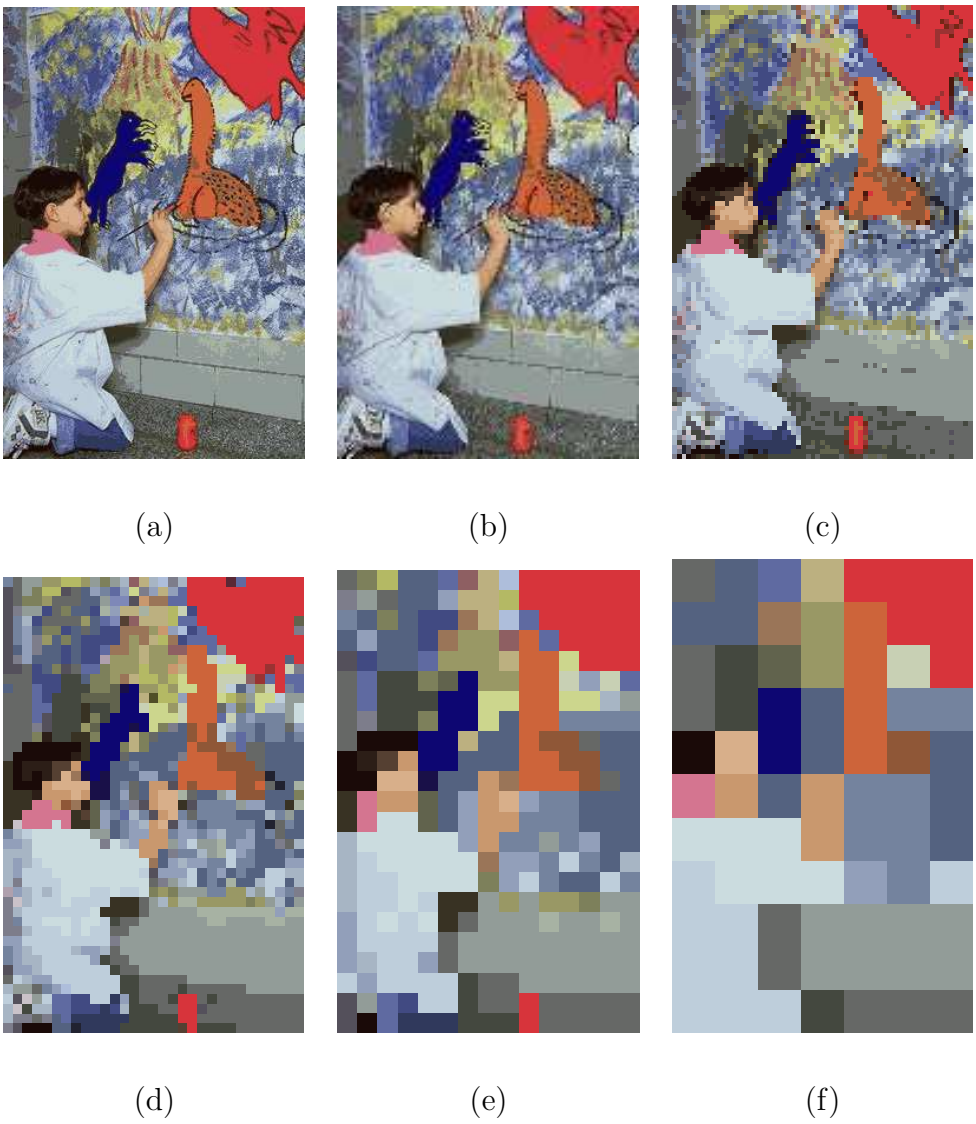


Figure 3.2: The region map of the Pyramid. (a) Level 8. (b) Level 7. (c) Level 6. (d) Level 5. (e) Level 4. (f) Level3. In this visualization of the region maps, each block is painted with the dominant color in its color histogram.

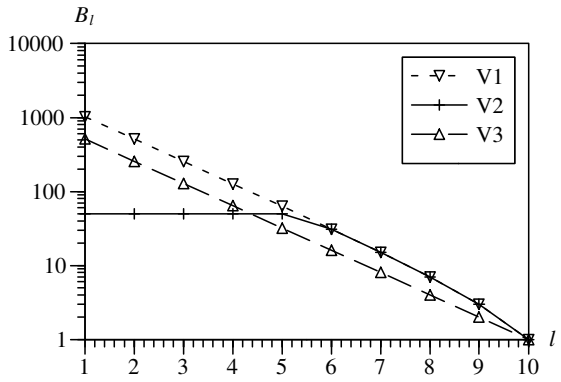


Figure 3.3: Number of bins of the histograms at level l (for $L = 10$). V1, V2, V3: region pyramids using variable number of histogram bins.

tests reported in [16] found that $B = 39$, averaged over 100 colorful Corel images of size 384×256.

To analyze the memory requirement, let us assume, for mathematical simplicity, that the input is a square image of width $S_L = 2^L - 1$ for some L. Then, the width of the image at level l is $S_l = 2^l - 1$, and the area (i.e., number of pixels) of the input image is $S_L^2 \approx 2^{2L}$. Let B_l denote the number of bins of the histogram at level l. Thus, the total amount of memory N used by the region pyramid is

$$N = \sum_{l=1}^{L} B_l S_l^2 \,. \tag{3.10}$$

A conventional image pyramid uses only one unit of memory space for each image pixel. That is, $B_l = 1$ for all l and the amount of memory N_0 required is

$$N_0 = \sum_{i=1}^{L} S_l^2 \approx \frac{1}{3} 2^{2(L+1)} \approx \frac{4}{3} S_L^2 \,. \tag{3.11}$$

Thus, a conventional image pyramid uses only 1/3 times more memory than that required for the input image.

Now, let us examine the memory requirement of our region-preserving region pyramid. If fixed-binning histograms of, say, 100 bins, are used to represent the color distributions of all the blocks in the region pyramid, then $N = 100N_0$. Replacing fixed-binning histograms with adaptive histograms can reduce the number of bins to, say 39, per histogram. This results in a total memory requirement of $N = 39N_0$. Both methods require lots of memory compared to a conventional image pyramid.

Obviously, the maps at the lower levels require far fewer bins than 39. Suppose we use the following memory scheme (V1)

$$B_l = 2^{L+1-l} - 1 \tag{3.12}$$

to estimate the number of bins required for the histograms at level l, then the total memory requirement N_1 becomes

$$N_1 = \sum_{l=1}^{L} (2^{L+1-l} - 1)(2^l - 1)^2 \approx \frac{2}{3} 2^{2(L+1)} = 2N_0 \,. \tag{3.13}$$

Compared to the cases of using a fixed number of bins, this method requires only two times as many memory space as that of a conventional image pyramid.

The memory requirement can be further reduced by using the following scheme (V3):

$$B_l = 2^{L-1} \,. \tag{3.14}$$

In this case, the memory requirement N_3 is

$$N_3 = \sum_{l=1}^{L} 2^{L-l} (2^l - 1)^2 \approx \frac{1}{2} 2^{2(L+1)} = \frac{3}{2} N_0 \,. \tag{3.15}$$

However, this scheme is not used in our current implementation because there

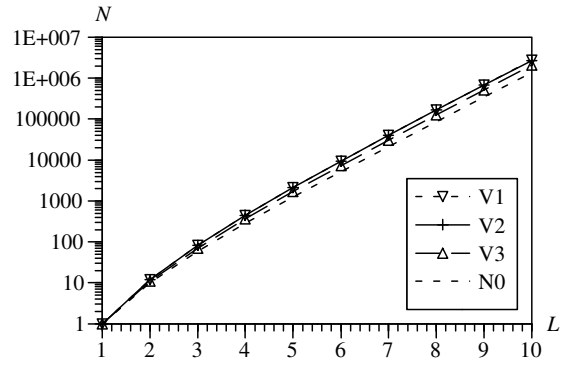


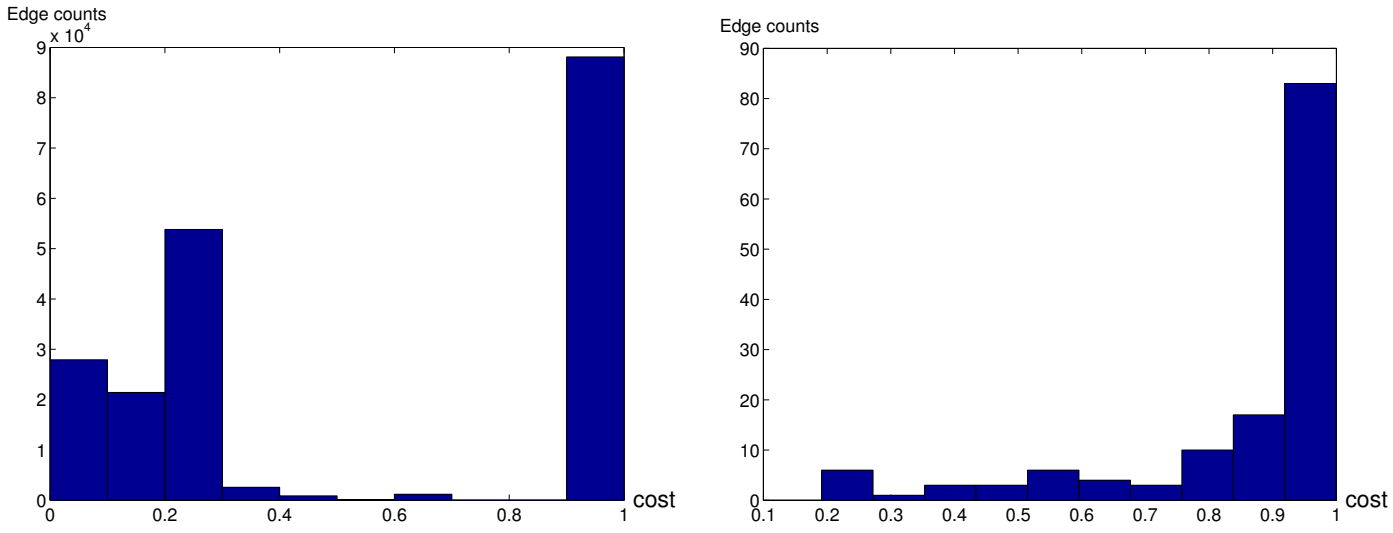
Figure 3.4: Memory requirement N of pyramids of height L. N0: conventional image pyramid, V1, V2, V3: region pyramids with variable number of histogram bins at different levels.

are too few bins in the low-level histograms to accurately represent the color distributions of the regions. Instead, the scheme given in Eq. 3.9 (V2) is used, which is similar to Eq. 3.12 except for the saturation at B. Numerical computation shows that the total memory requirement for this case, N_2, is less than N_1 which equals $2N_0$ (Figure 3.4).

3.5.1 Reduced Region Boundary Uncertainty

Another reason for constructing a region pyramid is that the region information at the higher level is more compact and the region boundary uncertainty is reduced with the trade off of a lower resolution. This can be shown in Figure 3.5.

Consider that there is an edge between each pair of neighbouring blocks, and the costs of these edges are measured by the similarity between the neighbouring blocks. Then the edges with smaller costs are likely to be region boundaries. From Figure 3.5 we can see that the percentage of possible boundary-edges reduced



(a) (b)

Figure 3.5: The x-axis represents the range of the edge costs (similarity between neighbouring blocks), the y-axis counts the number of edges with theirs costs falling into the different ranges shown in the x-axis. The percentage of possible boundary-edges dropped from (a) level 8 to (b) level 3.

significantly from level 8 (Figure 3.5(a)) to level 3 (Figure 3.5(b)), which means the region boundary uncertainty has been reduced significantly at level 3.

Chapter 4

Segmentation with Minimum Mean Cut

After constructing the region pyramid, segmentation is performed at level $l = 3$ or 4. These levels contain a sufficient number of blocks that correspond well with the main regions in the image. Furthermore, they contain far fewer blocks than the bottom-most level L. Thus, a comprehensive optimization algorithm can be applied at these levels to obtain globally optimal segmentation.

The recent graph-theoretic approach has provided us with such an optimization scheme. As discussed in Chapter 2, among the existing graph-cut algorithms, Minimum Mean Cut is an approach that does not introduce bias on boundary length or region size. Therefore, part of our algorithm will be based on Minimum Mean Cut. Let us review the Minimum Mean Cut algorithm below.

4.1 Introduction to Minimum Mean Cut

Here we will consider the recent Minimum Mean Cut (MMC) algorithm introduced in [32]. As stated earlier, MMC can extract significant contours without introducing bias on boundary length or size. It is based on minimizing the cost function

$$\overline{C}(A,B) = \frac{c(A,B|w(u,v))}{c(A,B|1)} \tag{4.1}$$

which finds the cut that groups the pixels in an image into groups A and B, and minimizes the average edge cost along the boundary. The average edge cost along the boundary becomes a measurement of a good segmentation, and its optimal solution which takes all possible boundaries as variables deduces an optimal partitioning of the pixels in the image.

In our application, each image block is regarded as a vertex of a graph G (Figure 4.1). A graph edge is connected between neighboring vertices, and it corresponds to the edge between the image blocks. This process constructs a grid graph from an input image. The edge cost is assigned as the similarity between the histograms of blocks u and v where $u \in A$ and $v \in B$, which is computed according to the histogram similarity discussed in Section 3.3. Therefore $c(A,B|w(u,v))$ computes the sum of the edge cost in between groups A and B, and it is normalized by $c(A,B|1)$, the boundary length, to obtain the mean cost $\overline{C}(A,B)$.

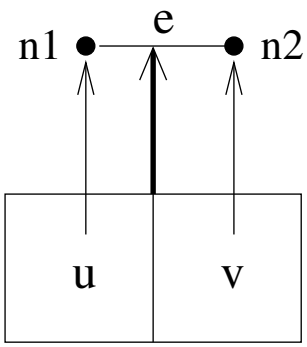


Figure 4.1: The construction of grid graph G from original image. (Blocks u and v in the original image correspond to vertex $n1$ and $n2$ in the graph. The graph edge e connects $n1$ and $n2$, and it corresponds to the edge between blocks u and v.)

4.1.1 Reducing Minimum Mean Cut to Minimum Mean Simple Cycle

The problem of finding a Minimum Mean Cut (MMC) can be reduced to the problem of finding a minimum mean simple cycle (MMSC) with the assumption that the grid graph $G = (V, E)$ is a connected-planar graph [32]. The reduction from Minimum Mean Cut to minimum mean simple cycle constructs a dual graph $\hat{G} = (\hat{V}, \hat{E})$. Figure 4.2 gives an example of the dual graph construction. The construction procedure adapted from [32] is given below:

1. For every grid (solid lines in Figure 4.2) in G, $\hat{G}$ contains a corresponding vertex located in the center of this grid. These vertices are called basic vertices and form a new grid system. In Figure 4.2, v_1 is one of the basic vertices.

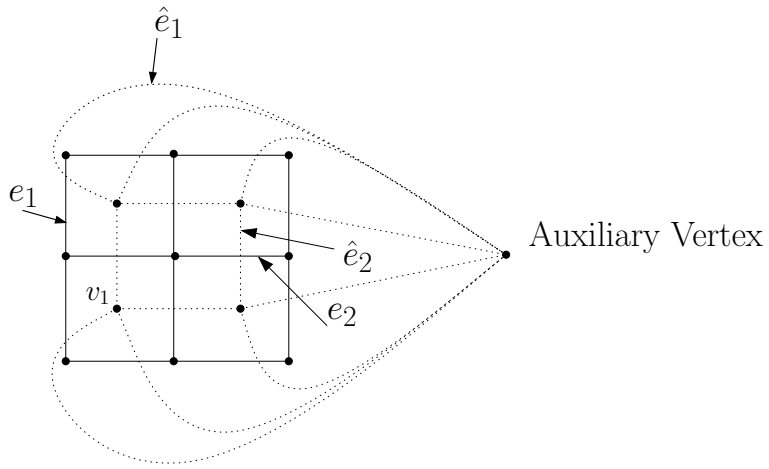


Figure 4.2: An example dual graph constructed from the original grid graph. Given the original grid graph G (solid lines), a dual graph $\hat{G}$ (dotted lines) can be constructed. See main text for the construction algorithm.

2. $\hat{G}$ contains a distinct vertex for all the border edges (e_1 and the other 7 solid edges that surrounds G in Figure 4.2) of G. These vertices are called *auxiliary* vertices.

3. Each non-border edge $e \in E$ is mapped to a corresponding edge $\hat{e} \in \hat{E}$ that goes across e and with the same cost as e. For example in Figure 4.2, e_2 is mapped to $\hat{e_2}$.

4. Each border edge $e \in E$ is mapped to a corresponding edge $\hat{e} \in \hat{E}$, with the same cost as well, and connects a border vertex to the auxiliary vertex for that border. For example in Figure 4.2, e_1 is mapped to $\hat{e_1}$.

For any simple cycle $\hat{c} = \hat{e_1}, \ldots, \hat{e_l}$ in $\hat{G}$, removing the edges $c = e_1, \ldots, e_l$ from E partitions G into two connected components and therefore corresponds to a cut

in G with boundary c. When $\hat{c}$ traverses an auxiliary vertex, c will become an open boundary; otherwise, c is a closed boundary.

4.1.2 Reducing Minimum Mean Simple Cycle to Negative Simple Cycle

The minimum mean cost cycle problem in directed graph has been addressed by Karp in 1978 which is solved by dynamic programming. We need to solve the minimum mean cost cycle in an undirected graph. The usual transformation of an undirected graph to a directed graph by transforming each undirected edge to two edges of opposite direction does not work because the minimum mean cycle will always fall on the cycle formed by the two edges transformed from the minimum cost edge.

The problem can be solved as follows [32]. The edge cost w of $\hat{G}$ can be transformed by $w' = w - b$, where b lies between the minimum and the maximum edge costs. Then, the negative simple cycle (a simple cycle with a negative total cost) of $\hat{G}$ that corresponds to the smallest b is the negative simple cycle that corresponds to the minimum mean simple cycle of $\hat{G}$.

4.1.3 Reducing Negative Simple Cycle to Minimum-Cost Perfect Matching

To determine whether the graph $\hat{G}$ has a negative simple cycle is equivalent to determining whether the graph G' constructed as follows (Figure 4.3) has a

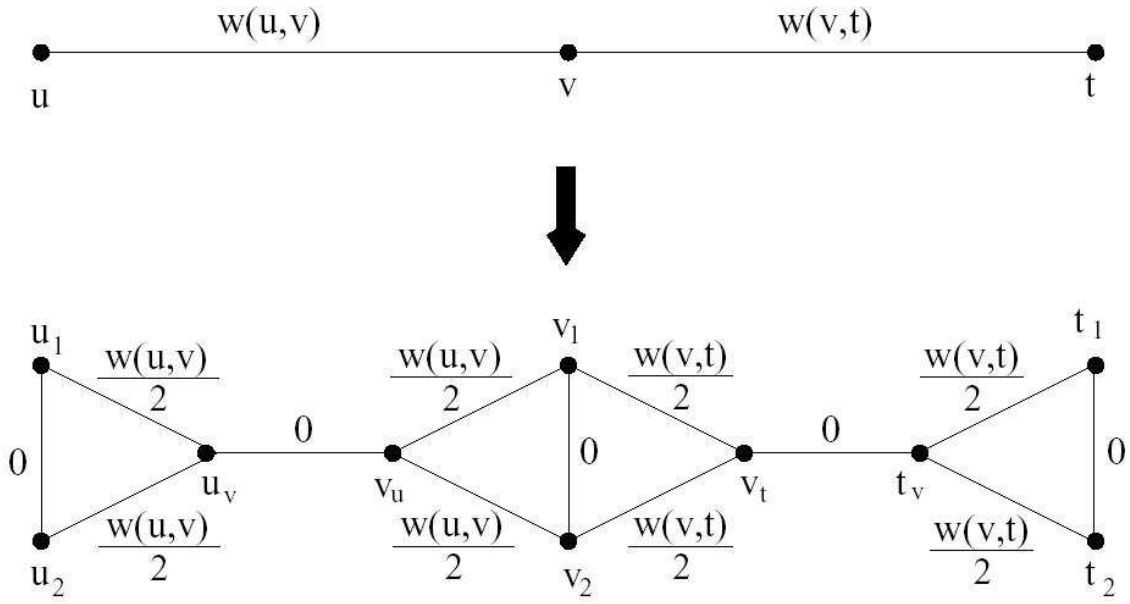


Figure 4.3: Graph constructed to reduce negative simple cycle to minimum-cost perfect matching.

negative-cost perfect matching [32]:

1. For each vertex u in $\hat{G}$, G' contains two corresponding vertices, u_1 and u_2, and one corresponding zero-cost edge (u_1, u_2).

2. For each edge (u, v) in $\hat{G}$, G' contains two corresponding vertices, u_v and v_u, and five corresponding edges with weights as follow: $w(u_1, u_v) = w(u_2, u_v) = w(v_1, v_u) = \frac{1}{2}w(u, v)$ and $w(u_v, v_u) = 0$.

The problem of finding the negative-cost perfect matching could be solved in polynomial time using the algorithm given in [12].

The three graph transformations above has shown that solving for MMC problem is equivalent to solving the corresponding Minimum-Cost Perfect Matching problem, and therefore it can also be solved in polynomial time.

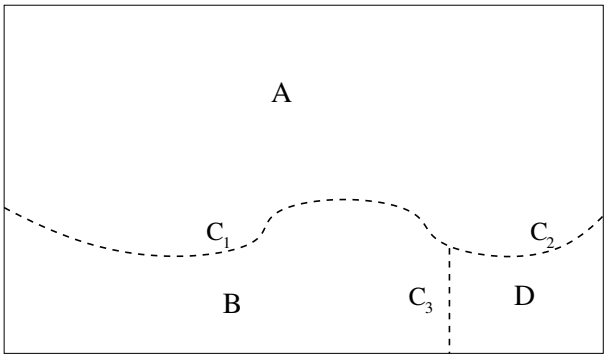


Figure 4.4: The spurious cut problem.

4.2 Interleaved Segmentation Algorithm

4.2.1 Shortcomings of MMC

In practice, MMC tends to produce many small regions which is undesirable for image labeling process. Furthermore, MMC has the spurious cut problem, which is illustrated in Figure 4.4. As discussed in [32], the desired cut boundary that corresponds to image edges is $c_1 \cup c_2$ with length $l_1 + l_2$. Although c_3 has a larger mean value than c_1 and c_2, MMC will produce the undesired cut boundary $c_1 \cup c_3$ when $w(c_1)/l_1 < w(c_2)/l_2$ and $l_3 \ll l_2 < l_1$, where $w(c_i)$ is the cost of cut boundary c_i.

The spurious cut problem arises because MMC is a single direction approach. All edges are considered as candidates of *inter-region edges* (i.e., parts of region boundaries), without making use of the fact that many of the edges should be considered as candidates of *intra-region edges* within regions.

4.2.2 Details of Interleaved Segmentation

The main idea of our segmentation algorithm is to determine whether the edge between two neighboring blocks is an inter-region edge or an intra-region edge. The labeling of both types of edges proceed at the same time. In contrast, the Minimum Mean Cut (MMC) algorithms described in [32, 33] focus only on identifying inter-region edges.

Let $e(b_i, b_j)$ denote the edge between two neighboring blocks b_i and b_j, and $s(b_i, b_j)$ denote the edge cost of $e(b_i, b_j)$, which is the similarity between the adaptive color histograms of the blocks. The similarity is measured using the weighted correlation method given in Section 3.3. If we sort all the edge costs in increasing order $s_1 \leq s_2 \leq \cdots \leq s_m$, then the edge with the smallest cost s_1 must correspond to an inter-region edge and that with the largest cost s_m must correspond to an intra-region edge. Since the similarity between two blocks across a region boundary is, in general, smaller than that within a region, there exists a *fuzzy* threshold Γ above which most edges are intra-region edges. This threshold is determined recursively during segmentation based on MMC.

The segmentation algorithm is as follows:

1. Set the initial estimate of threshold $\Gamma = s_m - k\sigma$, where k is a predefined value given in Table 4.1, and σ is the standard deviation of the edge costs $s_1, \ldots, s_m$.

2. Repeat

   (a) Mark all the unmarked edges with costs above Γ as intra-region edges.

Table 4.1: k value adjusted according to the σ value.

σ	< 0.14	[0.14, 0.19)	[0.19, 0.25)	≥ 0.25
k	0.02	0.04	0.08	0.1

(b) Use MMC to identify the minimum cost contour c among the unmarked edges. Mark the edges in c as inter-region edges.

(c) Decrement Γ by $k\sigma$.

until the costs of all unmarked edges are below Γ or no more contour is found.

3. Mark the remaining unmarked edges as intra-region edges, and compute the mean histogram of the blocks in each region.

4. Merge neighboring regions whose mean histograms have the same dominant color, i.e., the color with the largest bin count. The mean histogram for each region is calculated using the operation discussed in Section 3.3. Note that this merging criterion is not dependent on any threshold.

   Two similar regions that are connected at the corners are marked as different regions by the segmentation algorithm (Figure 4.5) because only simple cycles are considered when inter-region edges are marked. The regions to be considered in this merging process will include the neighbors located at horizontal, vertical, and diagonal directions.

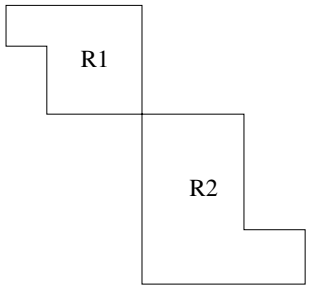
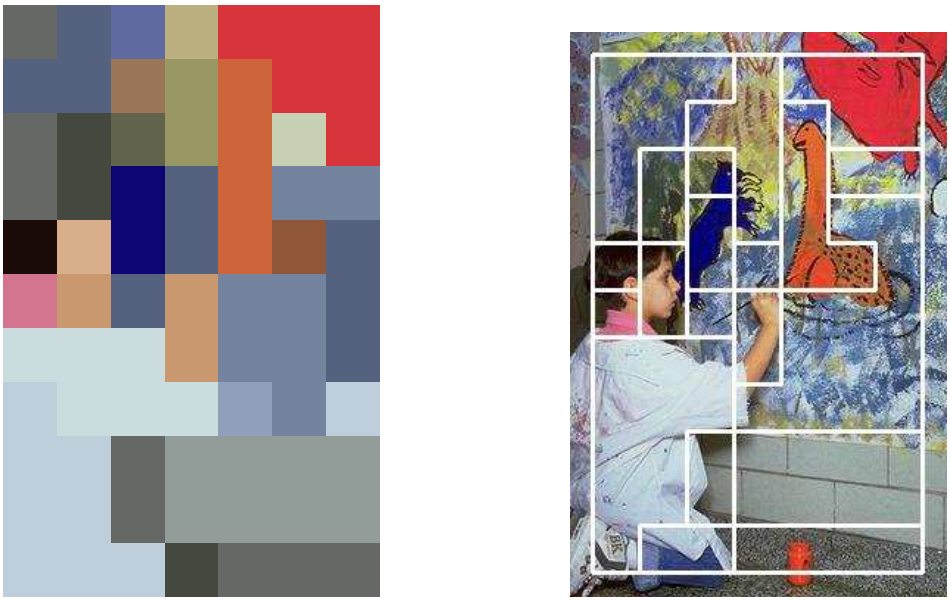


Figure 4.5: Two similar regions R1 and R2 connected at the corner. They were marked as different regions by our algorithm because only simple cycles are considered when marking the inter-region edges.

Compared to [32, 33], our method has a lower chance of producing spurious cuts because intra-region edges are identified before MMC is applied on the unmarked edges. Moreover, the process of marking intra-region edges has reduced the search space for the MMC operation, thus efficiency is improved. Figure 4.6 shows an example of the segmentation result obtained at level 3 of the region pyramid.



(a) (b)

Figure 4.6: Segmentation result at lower-resolution level. (a) Region map at level 3 shown with the dominant colors of the histograms of the image blocks. (b) Segment result at level 3 overlayed onto the input image.

Chapter 5

Boundary refinement

The boundary refinement process is a top down process that gradually locate the accurate boundary from a coarse level l to the finest level L. The proposed algorithm is divided into two steps. The first step is a global optimization approach which involves Dynamic Programming and Integer Linear Programming, This step is performed from level l to level $l+1$. The second step is a greedy approach, which is performed from level $l+1$ to level L.

5.1 Global Optimization Approach

Each inter-region edge at level l can be refined or expanded into a sequence of connected edges, we shall call this connected sequence of edges an *edge sequence*, that partitions a 3×5 area at level $l+1$ into two parts (Figure 5.1). Since the edge sequence is located in a fixed area, there is a fixed number of possible expansions of a level l edge into a level $l+1$ edge sequence. The level $l+1$ edge sequences must satisfy *connectivity constraints* that are consistent with the connectivity

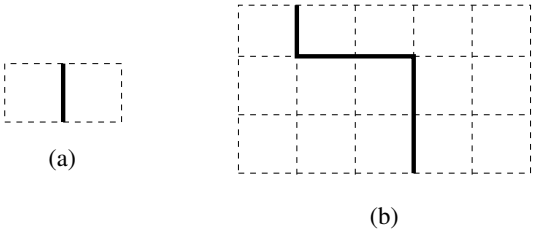


Figure 5.1: A region boundary edge at (a) level l is refined into (b) a sequence of connected edges called an edge sequence that partitions a 3×5 area at level $l+1$ into two parts.

of the level l edges. The detailed connectivity constraints will be discussed in Section 5.1.4.

Let c_{ij} denote the cost of the jth edge sequence of edge i (see Section 5.1.3 for details), and $v_{ij} \in \{0, 1\}$ denote whether the jth edge sequence is adopted for edge i. Since only one edge sequence can be chosen for each edge, $\sum_j v_{ij} = 1$.

Now, we can formulate the boundary refinement problem as a problem that minimizes the total cost C:

$$C = \sum_{i}^{m} \sum_{j}^{n} v_{ij} c_{ij} \tag{5.1}$$

where m is the number of edges and n is the number of edge sequences for each edge, subject to the constraint $\sum_j v_{ij} = 1$ and the connectivity constraints between the edges. This problem can, in general, be solved by Integer Linear Programming (ILP).

Note that the boundary refinement process affects only the detailed locations of the boundary edges. So, the main regions that are segmented at the top level are preserved. Therefore, our segmentation algorithm can identify the main regions

in an image, the accurate boundaries of the regions, and minimize the amount of over-segmentation.

5.1.1 Optimization by DP and ILP

In practice, the ILP problem discussed in the previous section is too expensive to solve directly. There are typically 100 edges to consider. Each edge is connected to one or more edges, giving an average of 120 junctions. On average, each junction can be expanded into 500 combinations of edge sequences. So, the ILP solver has to check through over 60,000 constraints.

A careful inspection on these junctions reveals that more than 80% of them are simple junctions (e.g., E and G in Figure 5.2) that connect only two edges. Less than 20% of them are complex junctions that connect three or four edges (e.g., T and X in Figure 5.2). So the main idea here is to use ILP to solve the connectivity constraints at complex junctions and minimize the boundary cost, and use Dynamic Programming (DP) to solve for the locally optimal expansions for the edge paths between complex junctions, or the paths that connect complex junctions (Figure 5.2) with the image border. The details of how DP works is described as follows:

- Consider a complex junction, e.g., T, and an edge incident at a complex junction, e.g., TE. TE can be expanded into n possible edge sequences.

- Given a possible edge sequence of TE, use DP to find the locally optimal expansion of the path TP starting from the given edge sequence of TE.

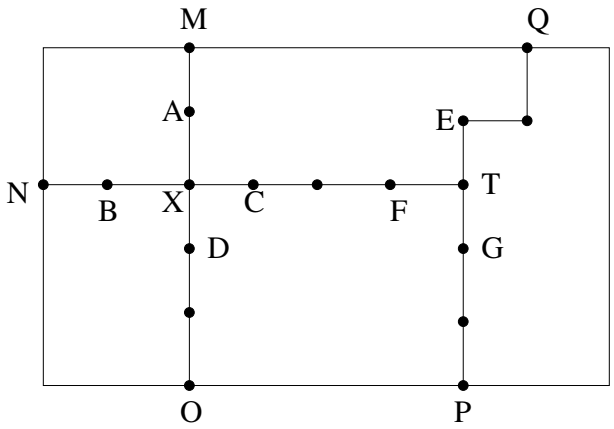


Figure 5.2: An example segmentation result. (T: 3-edge junction, X: 4-edge junction, A: 2-edge junction, M: point on image border.

- Therefore, for each possible edge sequence of TE, DP will return one locally optimal expansion of path TP. Each of these expansions of TP has a cost which equals to the mean cost of the expanded edge sequences of each edge in TP.
- Similarly, DP can be applied to path TQ as above.
- For path TX, both T and X are complex junctions. In this case, DP finds a locally optimal expansion of the path TX given a possible edge sequence of TF and a possible edge sequence of XC. So, for each pair of possible edge sequences of TF and XC, DP returns a locally optimal expansion of path TX and a path cost.

Then the ILP problem can be reformulate as:

$$\sum_{i=1}^{p}\sum_{j=1}^{q} v_{ij}c_{ij} \tag{5.2}$$

which is the same as Equation 5.1, with p representing the number of paths, and

q representing the number of possible expansions for each path. There are at most $n \times n$ possible expansions for the path in between two complex junctions and at most n possible expansions for a path that connects a complex intersection to the image border. Where n is the number of edge sequences of each edge, and c_{ij} is the cost of the jth expansion of path i. The new problem reduces the number of constraints to be checked by at least 35%. So, the optimization problem can be solved more efficiently.

5.1.2 Selection of Valid Edge Sequences

As observed in Figure 5.1, each inter-region edge at level l can be expanded into an edge sequence that partitions a 3×5 area at level $l+1$ into two parts. But how to decide what is a valid edge sequence? Suppose an inter-region edge exists between two blocks at level l (Figure 5.3). For simplicity of explanation, let us assume that the two blocks have two different colors $r1$ and $r2$. The 15 blocks at level $l+1$ covered by the two blocks can each be assigned either the color $r1$ or $r2$. So there is a total of 2^{15} possible assignments. And the valid edge sequences must correspond to one of these assignments. The following criteria are set up to identify the valid edge sequences. This analysis is based on vertical edges, and the same idea applies to horizontal edges.

1. Flipping the color of all the blocks does not change the edge sequence. Therefore the number of possible assignments can be reduced by half.

2. All of the blocks that receive the same color must be contiguous.

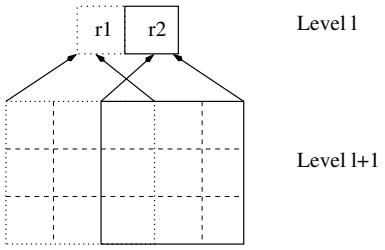


Figure 5.3: The two blocks at level l each covers a 3x3 area with 1 overlapping column.

3. The resulting edge sequences must have two end points, with one higher than the other. This is to ensure that the two end points of the edge sequence will correspond to the two end points of the vertical edge at the level l respectively.

4. The color distribution of the 9 blocks on the left (Figure 5.3), should not be equal to that of the 9 blocks on the right. Otherwise, the color distribution of the two blocks at level l will be the same, and it is very unlikely that there will be an edge in between the two blocks.

5. Each color should appear in at least 1/5 of the blocks for an edge to appear within the 15 blocks, which means there should be at least 3 blocks for each color. This is a reasonable heuristic assumption for an edge to appear in between the regions

With the above criteria, 240 edge sequences are identified as valid edge sequences for vertical edges and horizontal edges respectively. Selected edge sequences are shown in Appendix A.

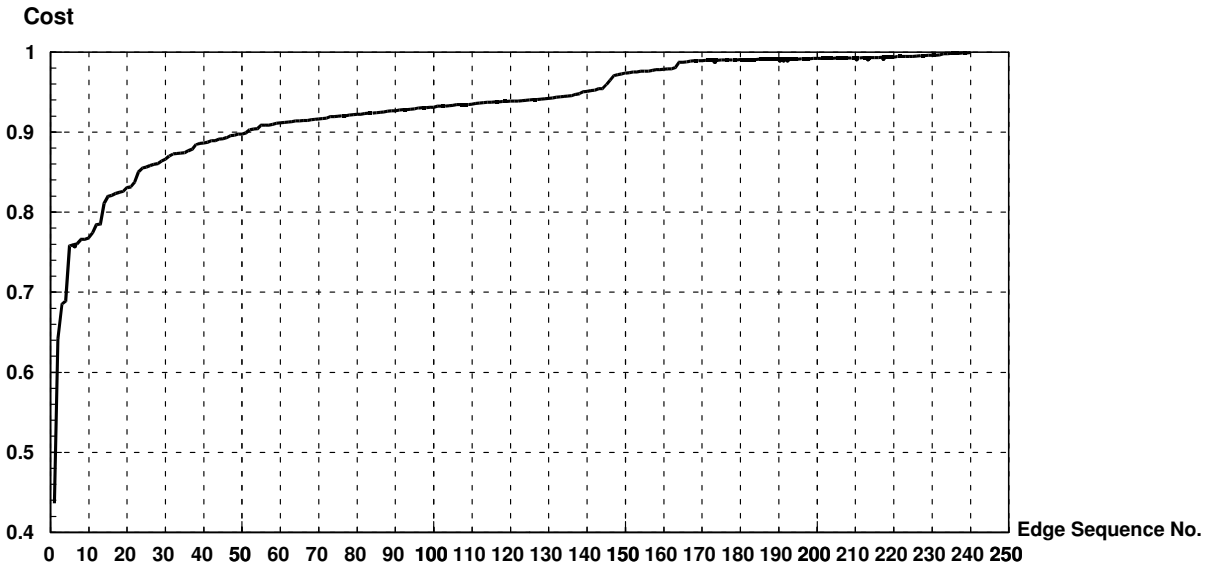


Figure 5.4: Trend of the edge sequence cost. The edge sequences are sorted in increasing order of their costs.

Only 1 out of 240 edge sequences will be adopted for each edge. Our objective is to obtain the optimal combinations out of all the valid combinations formed by these edge sequences. A question can be raised here: Are all the 240 edge sequences equally important? Are they all necessary? After sorting the edge sequences based on their edge sequence cost (details given in Section 5.1.3), we can get the answer. A sharp rise of the edge sequence cost is observed from Figure 5.4, and the cost grows rather smooth from around the 20th smallest value onward. Obviously those sequences with very large costs are not necessary to be considered, and we can set a threshold to reduce the number of edge sequences to consider. But what is the threshold and how many edge sequences shall we consider for each edge? Experiments have been carried out on a set of images, and some of the results are shown in Figure 5.5. The values shown in Figure 5.5

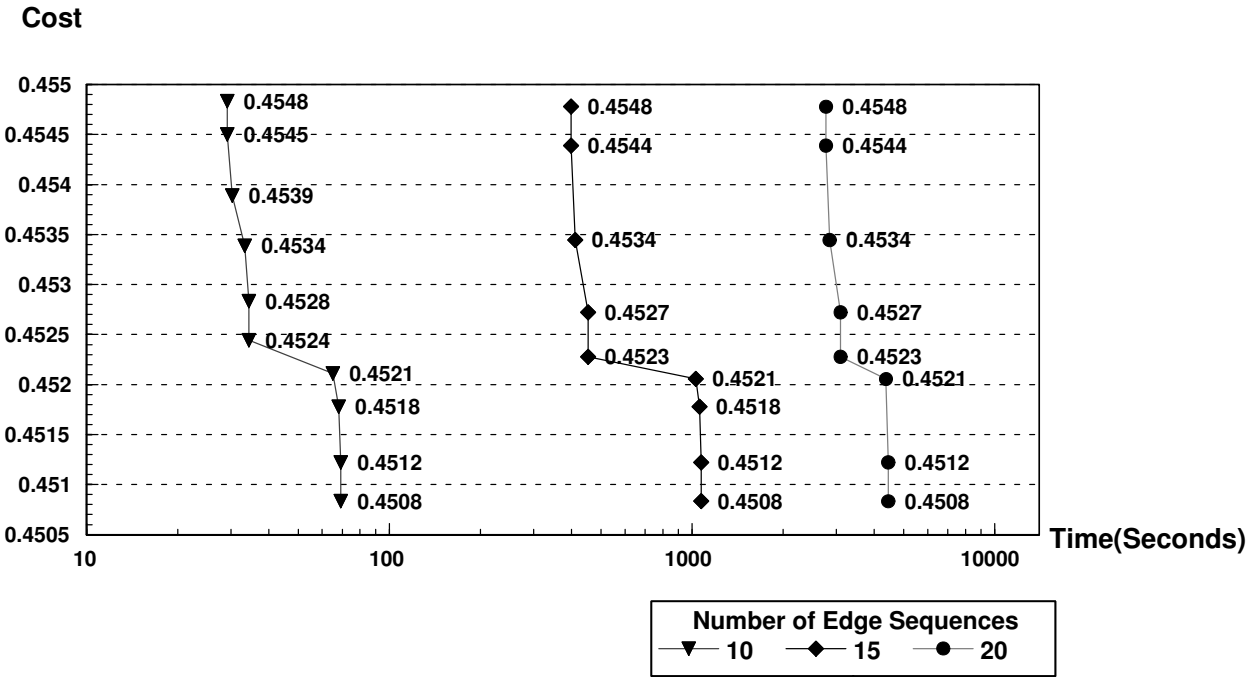


Figure 5.5: Feasible solutions obtained by ILP with 10, 15, and 20 edge sequences. Each line corresponds to the feasible solutions obtained in successive ILP iterations.

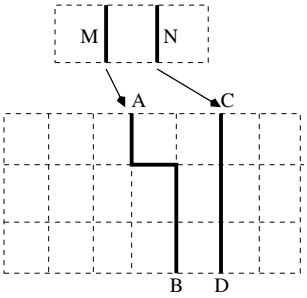


Figure 5.6: Expansion of edges M and N into segments AB and CD

are the costs of feasible solutions obtained using ILP at successive iterations. The number of edge sequences tested are 10, 15, and 20. We can see that optimal solution obtained using 10, 15, and 20 segments have the same minimum costs. This indicates that the optimal solution can be found within the 10 edge sequences with the lowest costs. Restricting to the best 10 edge sequences reduces the execution time significantly as shown in Figure 5.5.

5.1.3 Cost Function of Edge Sequences

The cost function for each edge sequence is addressed in detail in this section.

1. Total edge sequence cost normalized by the length for each edge sequence:

$$C = \frac{1}{n} \sum_{i=1}^{n} e_i \tag{5.3}$$

where e_i is the cost of edge i in the edge sequence, and n is the number of edges in the edge sequence. This is the simplest and most straightforward cost function. But, there is a problem with this cost function. Let us take a look at the situation in Figure 5.6. Suppose AB and CD are the actual edge sequences for edges M and N, with the edge sequence costs satisfying

the condition $C_{AB} < C_{CD}$. Let us further assume that AB is the only valid edge sequence of edge M, and AB and CD are both possible edge sequences of edge N. Then both edge M and edge N will adopt edge sequence AB, as AB has a lower cost than CD. This leads to a conflict that the expansions of two edges produces only one edge sequence. This happens because that the cost function does not incorporate spatial information between the edges.

2. Total cost normalized by the edge sequence length and block-parent association:

$$C = \frac{1}{nA} \sum_{i=1}^{n} e_i \tag{5.4}$$

A is called the block-parent association, and it has a larger value if the level $l+1$ blocks are assigned to the most similar parent blocks at level l.

Given an edge sequence that divides a 3×5 area into two parts P_1 and P_2, and the two regions R_1 and R_2 that the two neighbouring parent blocks belongs to, we can tell which region that each block B_i at level l belongs to as well. For example in Figure 5.7, with the given edge sequence CD, the blocks in P_1 belongs to region R_1, and the blocks in P_2 belongs to region R_2. The block-parent association A is defined as:

$$A = \frac{1}{15} \left[\sum_{B_i \in P_1} s(B_i, R_1) + \sum_{B_i \in P_2} s(B_i, R_2) \right] \tag{5.5}$$

where $s(B_i, R_j)$ denotes the similarity between the histogram of block B_i and the histogram of the parent region P_j at level l, $j = 1, 2$.

This cost function tries to minimize the average edge cost in the edge sequence, and maximize the similarity between the blocks and their parents

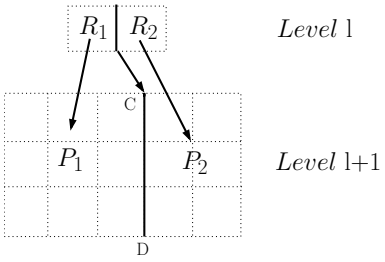


Figure 5.7: The association between child blocks and parent regions. Given an edge sequence CD, blocks in part P_1 at level $l+1$ belongs to region R_1, and blocks in part P_2 belongs to region R_2.

at the same time. In practice, since the similarity between the blocks and their parents lies in the range [0,1], the cost computed in Equation 5.4 will therefore fall in the range $(0, +\infty)$, which has no upper bound, and is, thus, less easy to assess the algorithm's performance. Recall that the difference d between two color histogram equals to $1 - s$ where s is the similarity (Section 3.3). Maximizing the similarity equals minimizing the difference. Therefore, the cost function can be rewritten as

$$C = \frac{1}{15n} \sum_{i=1}^{n} e_i \left[\sum_{r_j = P_1} d(B_j, P_1) + \sum_{r_j = P_2} d(B_j, P_2) \right] \tag{5.6}$$

Since both e_i and d_j are bounded within [0,1], so the cost function is also bounded within [0,1].

5.1.4 Connectivity Constraints

This section will address the connectivity constraints mentioned in Section 5.1. For the edges connected at a junction, their corresponding expanded edge sequences should also be connected. All the valid combinations of edge sequences

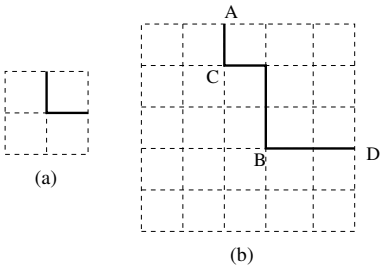


Figure 5.8: Example of a combination formed by edge sequences of 2 edges.

at a junction are collected in a set called the combination set $S(v)$. To determine the combination set, we need to consider combinations for the cases when 2, 3 or 4 edges are incident at a junction.

Combinations of Edge Sequences of 2-edge Joint

There are altogether 6 possible cases of 2 edges connected at a joint, and Figure 5.8(a) shows one of them. For any two connected edges, their edge sequence will fall into a 5×5 area as shown in Figure 5.8. The criteria for being a valid combination can be stated as follow:

1. The edge sequences of the two edges will split the 5×5 area into 2 regions.

2. For each level $l + 1$ edge in an edge sequence, the blocks on different sides of the edge must belong to different regions.

3. Each edge sequence contains two end points, one of them must be connected to the other edge sequence, and the other one should touch the border of the 5×5 area. There must be two end points that touch the border of the 5×5 area.

For example in Figure 5.8, the vertical edge at level l (Figure 5.8(a)) is expanded into edge sequence AB at level $l+1$ in Figure 5.8(b), and the horizontal edge is expanded into edge sequence CD. The bottom end point for edge sequence AB, i.e., point B, lies on edge sequence CD, and the left end point for edge sequence CD, i.e., point C, lies on edge sequence AB.

Combination of Edge Sequences of 3-edge Junction

There are altogether 4 possible cases of 3 edges connected at a junction, and Figure 5.9(a) shows one of them. The expansion of 3 connected edges also fall within the 5×5 area, and the criteria are as follow:

1. The edge sequences of the three edges will break the 5×5 area into 3 regions.

2. For each level $l+1$ edge in an edge sequence, the blocks on different sides of the edge must belong to different regions.

3. One out of the two end points of each edge sequence must be connected to one of the other two edge sequences, and the other end point should touch the border of the 5×5 area. In total, there must be three end points that touch the border of the 5×5 area.

4. Look at the example in Figure 5.9. Suppose the edge sequence of the vertical edge 1 is AB, which splits the upper 3×5 area into two parts $P1$ and $P2$, and the edge sequence of the horizontal edge 2 is CD. Then, its left end point C falls inside part $P2$. In this case, the relative positions of the edge sequences of edges 1 and 2 have been switched. Therefore, this is an invalid

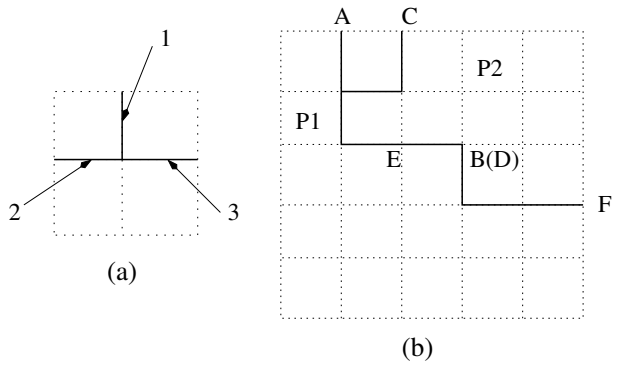


Figure 5.9: Example of a combination formed by edge sequences of 3 edges.

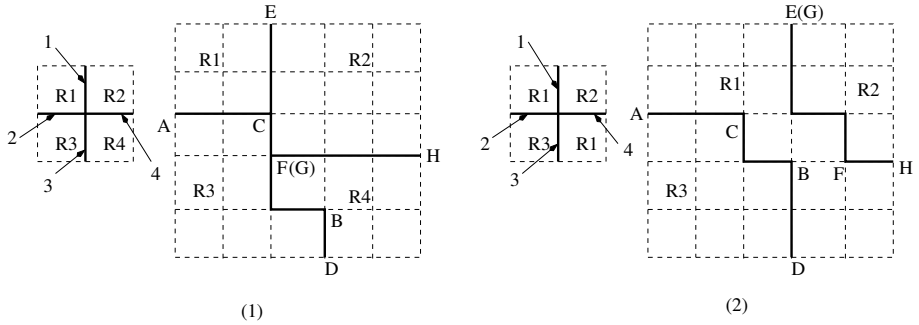


Figure 5.10: Two situations for the combinations formed by 4 edge sequences.

combination. If the edge sequence of edge 1 is CD, and the edge sequence of edge 2 is AB, then it will become a valid combination.

Combination of Edge Sequences of 4-edge Junction

This case is a bit more complex, as two situations need to be considered (Figure 5.10). There are four regions in the first case, and there are only three regions in the second case.

For the first case (Figure 5.10), the criteria are same as those of the 3-edge junctions except that the edge sequences split the 5×5 area into 4 regions, and there must be four end points that touch the border of the 5×5 area.

For the second case, there are altogether 2 such possible cases that the 4 edges connected at a junction partition the 4 blocks at level l into only 3 regions, and Figure 5.10(2) shows one them. The criteria for valid combinations are modified as follow (Figure 5.10(2)):

1. The edge sequences of 4-edge junction splits the 5×5 area into 3 regions.

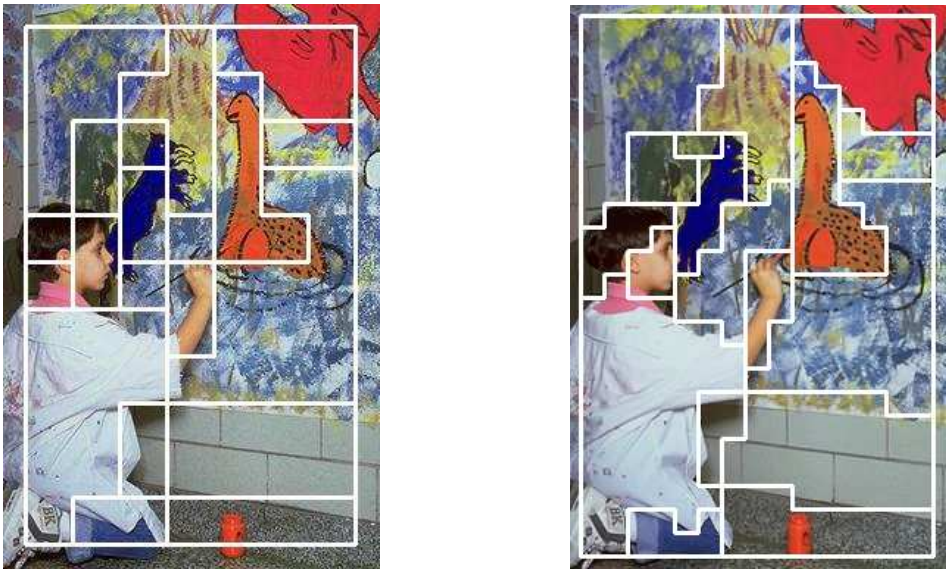
2. For each level $l+1$ edge in an edge sequence, the blocks on different sides of the edge must belong to different regions.

3. Suppose edges 1, 2, 3, 4 are expanded into segments EF, AB, CD, and GH. Then, the right end point for edge sequence AB must lie on edge sequence CD, and the top end point of edge sequence CD must lie on edge sequence AB, the left end point of edge sequence EF must lie on edge sequence GH, and the bottom end point of edge sequence GH must lie on edge sequence EF. In total, there must be four end points that touch the border of the 5×5 area.

Figure 5.11 shows an example of the refined region boundary when DP and ILP are applied from level 2 to level 3.

5.2 Greedy Local Optimization Approach

Greedy local optimization is used to refine region boundaries from level $l+1$ to the finest level L. This process does not directly work on the boundary edges. Instead, it tries to decide to which level $l+1$ region each level l block belongs, and this indirectly determines the boundary locations.



(a) (b)

Figure 5.11: Example of boundaries refined with DP and ILP. (a) Segmentation result at level 3. (b) Refined boundaries at Level 4 using DP and ILP.

The regions considered here are local regions along the boundary areas. At lower levels, the region size is larger and the region gets more complex and captures more color information. The histogram of the blocks in the region center can be quite different from those at the boundary area. Considering the full region will lose the local information along the boundary area. Therefore, *local region histograms* along the region boundary are used. Suppose block (x, y) located beside a boundary at level l belongs to region R, then the *local region histogram* $R(x, y)$ is computed by merging the normalized histograms of the blocks in region R and located within an $n \times n$ window centered at (x, y):

$$R(x, y) = \biguplus_{i,j \in N(x,y)} H(i, j) \tag{5.7}$$

where $H(i, j)$ is the histogram of block (i, j), $N(x, y)$ is the $n \times n$ neighborhood

Table 5.1: The window size for computing local region histogram at each level.

level	2	3	4	5	6	7	8	9
size	5	5	5	3	3	3	3	3

around block (x, y), and the windows size n is given in Table 5.1 for different levels.

The boundary refinement problem can then be formulated as minimizing the following cost function:

$$C = \sum_i s(B_i, R_i) \tag{5.8}$$

where B_i is the histogram of level $l+1$ block i, R_i is the local region histogram along the region boundary at level l, $s(B_i, R_i)$ computes the similarity between the histogram of blocks B_i at level $l+1$ and the region histogram R_i at level l. The function seeks the optimal region assignment for each block along the region boundary.

For each edge found at level l, there are two parent regions R_1 and R_2 located on the two sides of the edge. So R_1 and R_2 are both possible region assignments for the 3×5 corresponding blocks at level $l+1$ (Figure 5.3). The similarity $s(B_i, R_1)$ and $s(B_i, R_2)$ can then be computed for all the 15 blocks. Therefore, by going through all the edges found at level l, we can obtain all the possible region assignments and find the optimal assignment for each block, which is in fact the optimal solution to Equation 5.8.

To obtain a smoother boundary without affecting the boundary location, we

apply a Gaussian filter to the blocks at the last two levels of the image pyramid before the region assignment process, with a window size of 7 and $\sigma = 2$.

The connectivity constraint is not enforced for this region assignment process, and some ambiguous blocks exist at the boundary area. The resulting assignment will cause some of the regions to be disconnected. A heuristic approach is recursively applied to refine the assignments so as to preserve the region connectivity.

1. For each region i at level l, locate the block B_i at level $l+1$ that is most similar to region i at level l and mark each B_i as refined.

2. Repeat

   (a) For all the blocks that are the neighbors of a refined block B_i, and have been assigned to region i, mark them as refined and apply this refinement recursively.

   (b) Choose an un-refined block U_i and its refined neighbor B_i which has the greatest similarity. Then assign U_i to the region that B_i belongs to. Note that the similarity between all pairs of neighboring blocks have already been computed before hand during the pyramid construction process.

until all the blocks have been refined.

Note that most of the blocks are assigned correctly to their corresponding regions and can be rapidly refined at step 2(a). Therefore, the number of blocks that need to be refined at step 2(b) is much fewer than the total number of boundary blocks. So, this refinement process is quite efficient. Although this method

(a) (b)

Figure 5.12: Final segmentation result using greedy refinement. (a) Refined boundary obtained at level 3. (b) Final segmentation result.

is not necessarily optimal, it is efficient and the results produced are accurate enough. Figure 5.12 shows the final segmentation result obtained after the greedy refinement.

Chapter 6

Experimental Results

The proposed algorithm has been implemented in C. The minimum cost perfect matching is based on the blossom4 implementation [10], and the Integer Linear Programming part is solved using the ILOG OPLStudio solver (www.ilog.com).

6.1 Experimental Set Up

The experiments were carried out on a benchmark dataset of 100 images [18]. Each image is of size 321x481 or 481x321. Five segmentation algorithm were compared:

1. JSEG [11]: a multi-scale method based on region growing and merging. The implementation of JSEG was downloaded for the JSEG project website http://vision.ece.ucsb.edu/segmentation/jseg/.

2. Ncut [29]: a graph-theoretic method using normalized cut. Ncut is used for comparison instead of MMC because the MMC algorithm given in [32]

has only been implemented for gray-scale images. Moreover running MMC on 321x481 images takes a long time. The implementation of Ncut was downloaded from ftp://ftp.ecn.purdue.edu/qobi/.

3. Blobworld [8]: An algorithm designed for image retrieval systems using color and texture features. The implementation of BlobWorld was downloaded from http://elib.cs.berkeley.edu/src/blobworld/.

4. RP-ILP: Our region-preserving segmentation algorithm using interleaved MMC for segmentation at level 2, and DP, ILP and greedy algorithm for boundary refinement.

5. RP-G: Our region-preserving segmentation algorithm using interleaved MMC for segmentation at level 2 and only greedy algorithm for boundary refinement.

For RP-ILP and RP-G, the adaptive threshold k used for interleaved MMC were set according to the guidelines given in Table 4.1. For 27 of the images, RP-ILP cannot generate any feasible solution within the 10 lowest-cost edge sequences. This is because these images are over-segmented at level 3. So for the 27 images, k has been adjusted to higher values for RP-ILP to generate feasible solutions, and Figure 6.5 shows the result of one of these images. This shows that ILP has a strict requirement on the segmentation result obtained at the higher level, while the greedy algorithm is more tolerant of the initial segmentation.

For Ncut, it could not produce any result for 12 of the images, so only the segmentation results for the other 88 images are included for comparison.

6.2 Quantitative Evaluation

It is difficult to define a good measure of the quality of segmentation results. Therefore, image segmentation has for a long time been evaluated only through visual inspection. A recent benchmark introduced by Marin [18, 17] provides an attempt for a quantitative assessment of segmentation result. The evaluation is based on a comparison between the boundary maps produced by a the segmentation algorithm and the ground-truth boundary maps provided by human subjects. An F-measure is computed as the harmonic mean of precision rate P and recall rate R:

$$F = PR/(\alpha R + (1 - \alpha)P) \tag{6.1}$$

where precision is the probability that a detected boundary pixel is a true boundary pixel, and recall is the probability that a true boundary pixel is detected. α is used to adjust between the importance of the precision rate and recall rate, and it is set to 0.5 in [18, 17]. A higher F-measure value indicates that the segmentation result better approximates the ground-truth boundaries.

The benchmark program requires the segmentation result to be in a boundary map format. BlobWorld's output consists of a lot of discarded regions (Figure 6.1) which is hard to convert to a boundary map. Therefore, BlobWorld's results were not compared using the benchmark program, and were considered only for the qualitative evaluation.

Figure 6.2 shows a distribution of the F-measures of the 100 images, and Table 6.1 shows the overall statistics of the F-measures. The F-measure is computed for each human segmentation by comparing it to the other segmentations of the

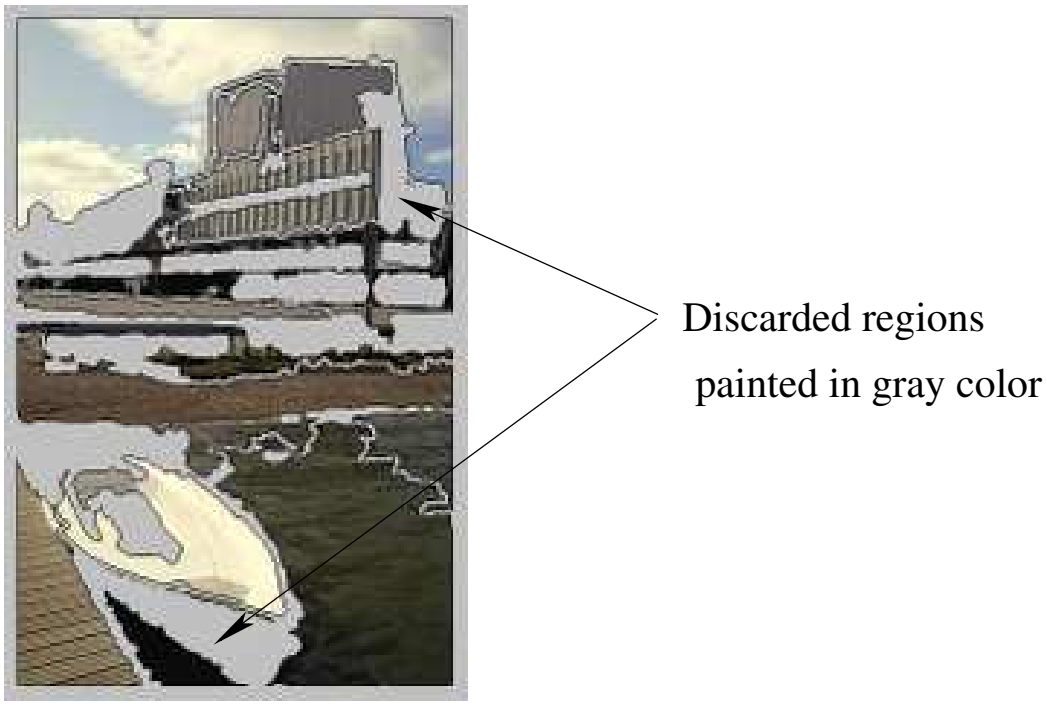
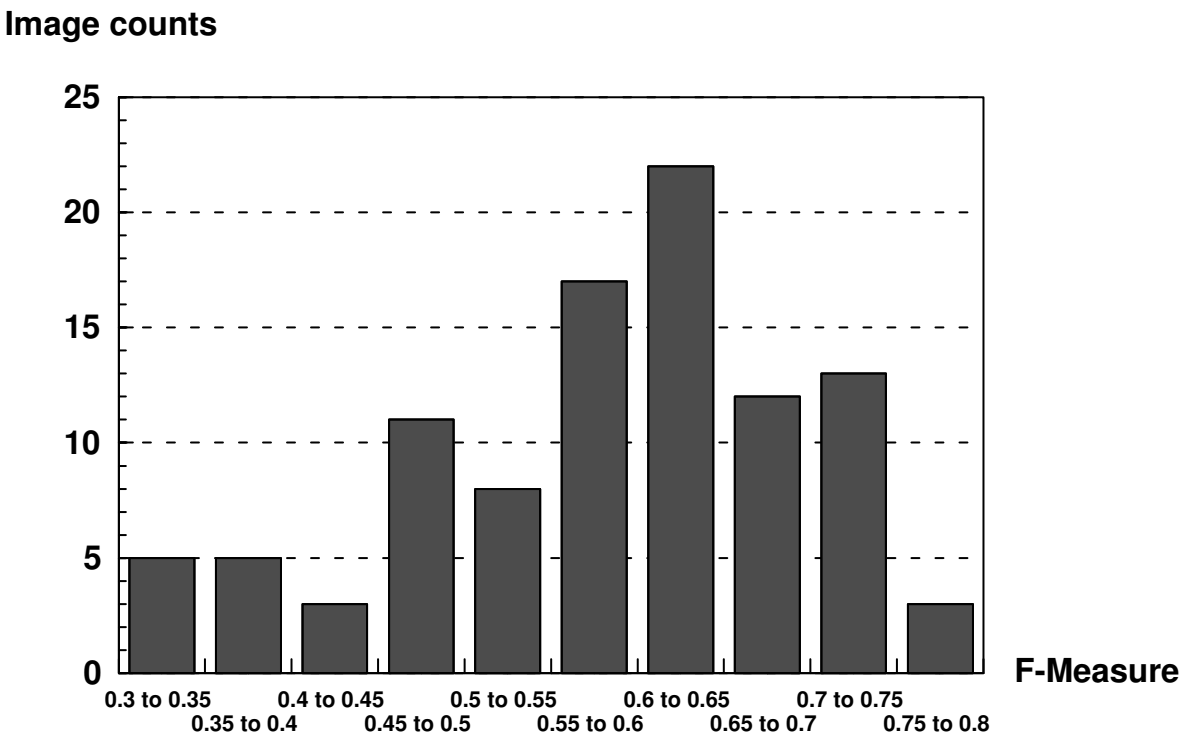


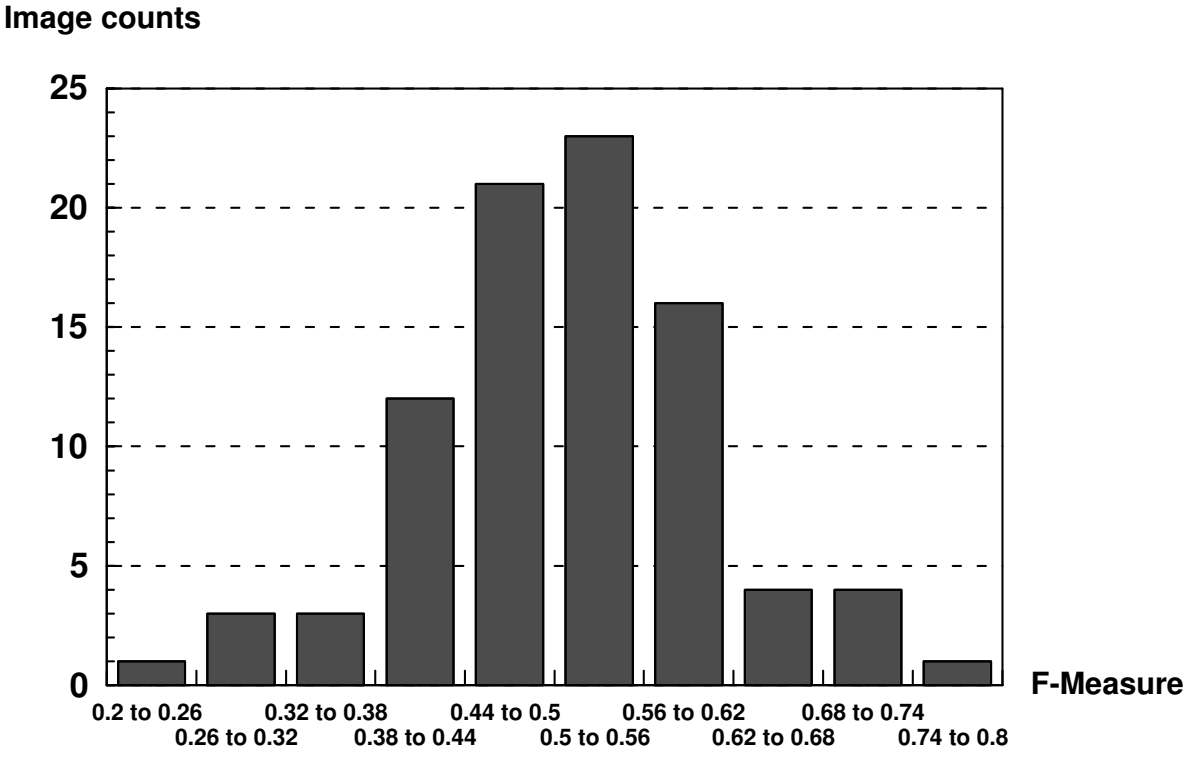
Figure 6.1: Sample BlobWorld segment result with discarded regions.

Table 6.1: Statistics on F-Measure.

F-measure	JSEG	Ncut	RP-ILP	RP-G	Human
Mean	0.5847	0.5049	0.5253	0.5435	0.7841
Std Dev	0.1118	0.0981	0.1128	0.0924	0.1021
Min	0.3180	0.2054	0.1834	0.3292	0.5017
Max	0.7963	0.7452	0.7436	0.7752	0.9577

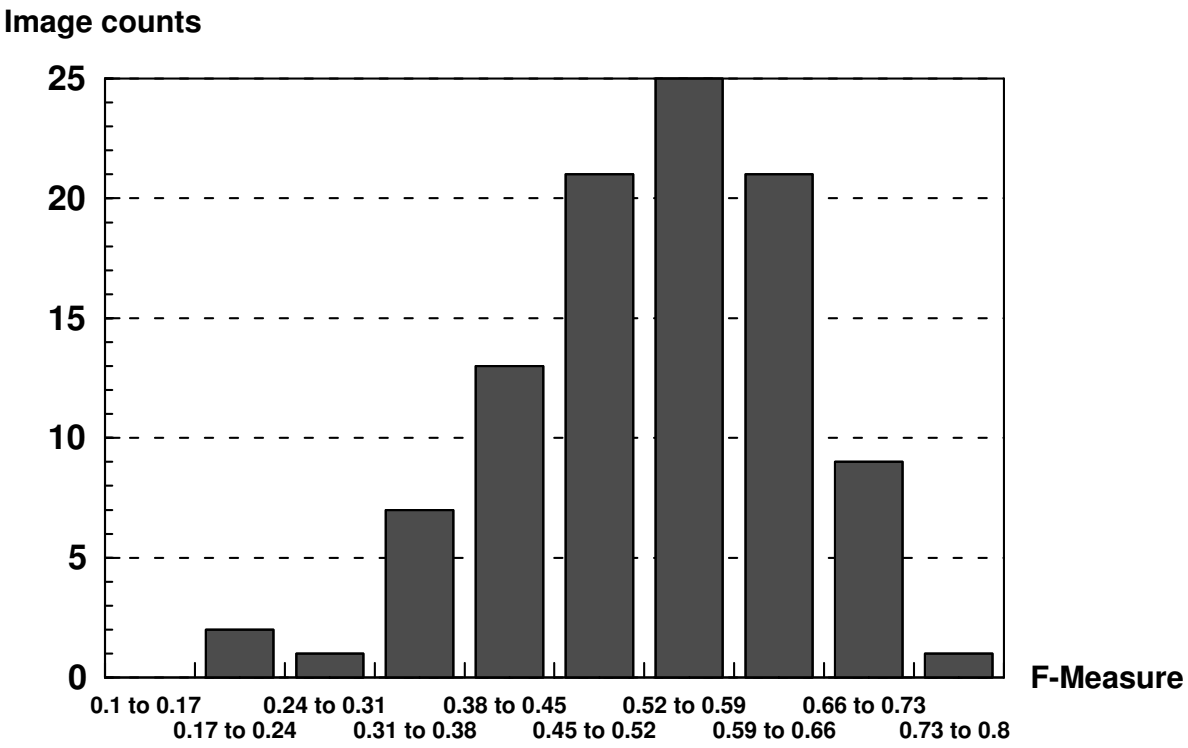


(a)

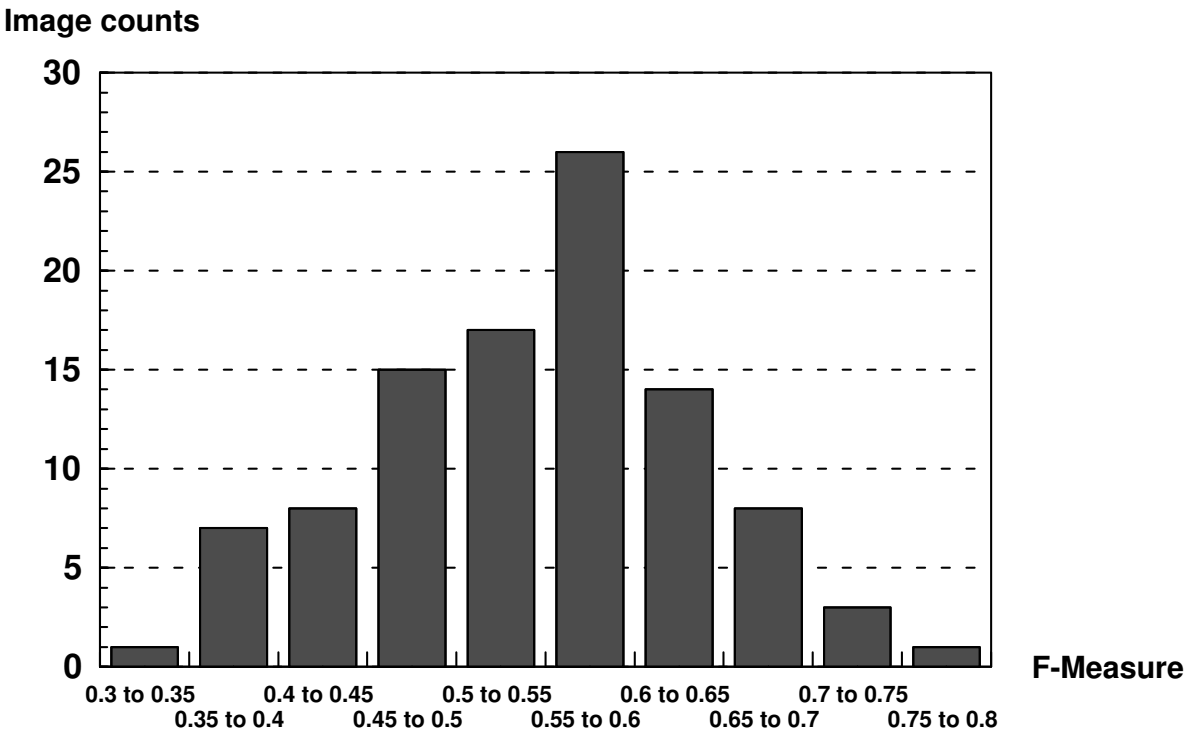


(b)

Figure 6.2: F-measure values for the test images. (a) JSEG, (b) Ncut.



(c)



(d)

Figure 6.2: F-measure values for the test images (continued). (c) RP-ILP, (d), RP-G.

same image, and the maximum value is taken as the F-measure for human, which is shown in the last column of Table 6.1. The F-measure for human provides an upper bound for the machine segmentation algorithms. A higher value of the upper bound also indicates the easier an image can be segmented, and vice versa. We can see that the region-preserving methods have a higher mean values than Ncut but lower than JSEG. Figure 6.3 shows an example of the segmentation result and the corresponding scores. The ground-truth boundary maps obtained from human subjects, tend to have many weak boundaries, i.e., boundaries that appear in only some human-segmented results. These weak boundaries usually do not correspond to major region boundaries. For two segmentation results with the same precision, the one with more weak boundaries will have a higher recall rate, and thus a higher F-measure. This can be seen in Figure 6.3 which shows that JSEG detected more weak boundaries than RP-ILP and RP-G, and thus obtained a higher F-measure.

The weak boundaries are not of concern for image labeling and retrieval which are only interested in main regions. Therefore, we also compared the precision of the segmentation results. Table 6.2 shows the statistics on the precision of the algorithms. From Table 6.2, we can see that RP-ILP and RP-G have better precision than Ncut and JSEG, which means the boundaries detected are more likely to be true region boundaries.

The average processing time of algorithms are shown in Figure 6.3. The time was taken on a Pentium PC with 1.6GHz processor and 256MB memory.

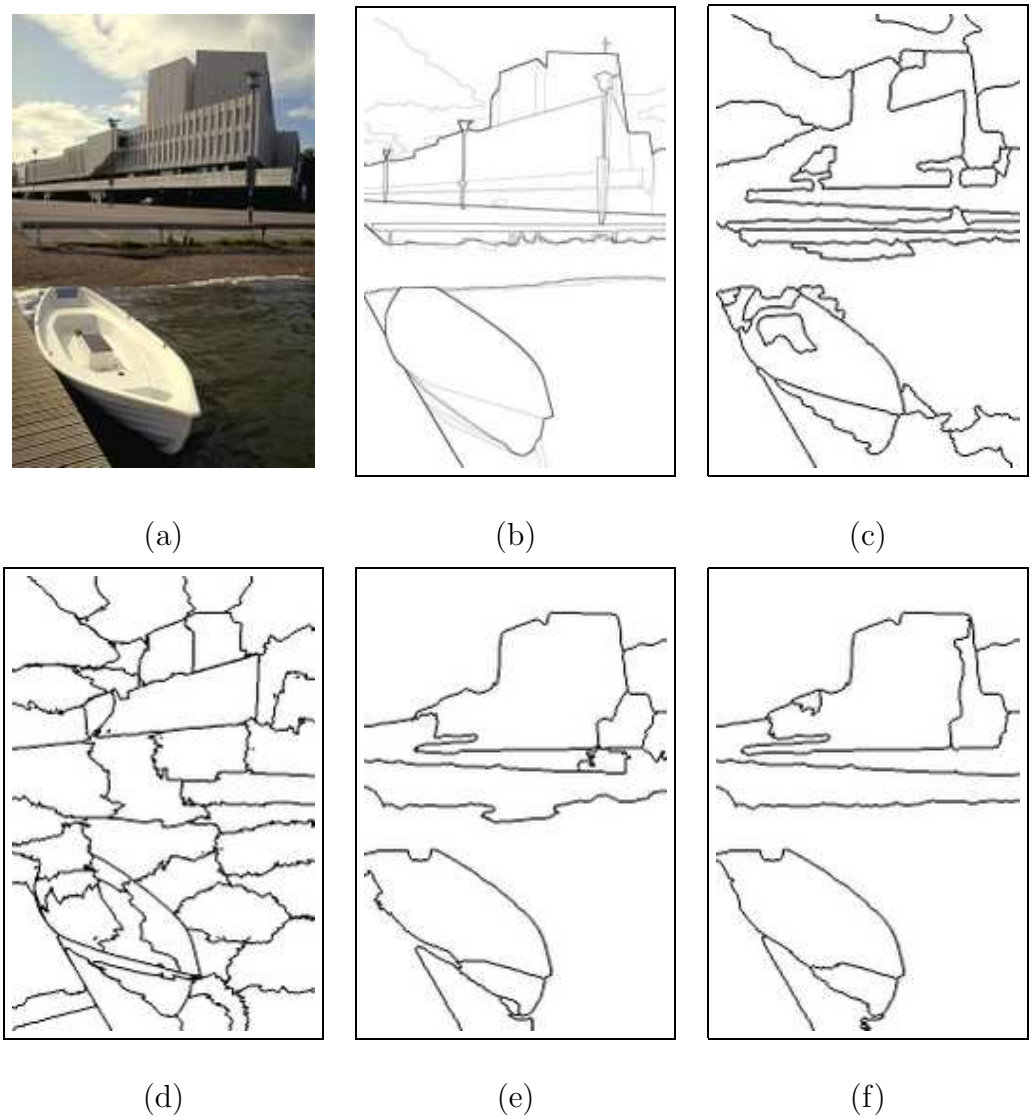


Figure 6.3: Test result 1. (a) Input image. (b) human-segmented result. Dark boundaries are strong boundaries and light boundaries are weak boundaries. F=0.811, P=0.912. (c) JSEG Result: F=0.707, P=0.712. (d) Ncut Result: F=0.453, P=0.461. (e) RP-ILP Result: F=0.691, P=0.822. (f) RP+G Result: F=0.634, P=0.753.

Table 6.2: Statistics on the Precision Measure.

Precision Rate	JSEG	Ncut	RP-ILP	RP-G	Human
Mean	0.5584	0.4812	0.6208	0.5995	0.8922
Std Dev	0.1455	0.1314	0.141	0.1453	0.0807
Min	0.2235	0.1360	0.3099	0.2582	0.5860
Max	0.8589	0.7870	0.9843	0.9868	0.9982

Table 6.3: Average processing time of algorithms measured in seconds on images of size 321x481.

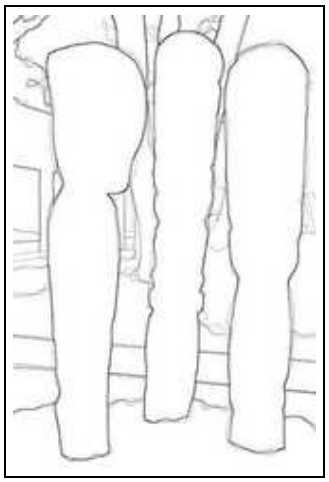
BlobWorld	JSEG	Ncut	RP-ILP	RP-G
3000	15	2400	350	50

6.3 Qualitative Evaluation

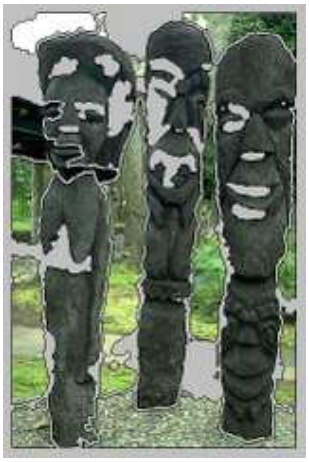
This section demonstrates some segmentation results performed on the benchmark dataset for qualitative evaluation. The results are shown from Figure 6.4 to Figure 6.12. From the given examples, we can see that region-preserving approach produce less over-segmented results compared to JSEG and Ncut. BlobWorld can also identify the main regions in the images. But, the region boundaries extracted tend to be less accurate.

There are also cases that the segmentation results are not satisfactory. Figure 6.16 shows an example. We can see that all machine segmentation results are very different from the human segmentation. The reason is that the main regions in the image are not distinguishable using only color information. We can expect the result to be improved if other features such as texture or morphological information are incorporated. This is suggested by the BlobWorld result, which is able to obtain some reasonable regions based on consistent textures.

(I)



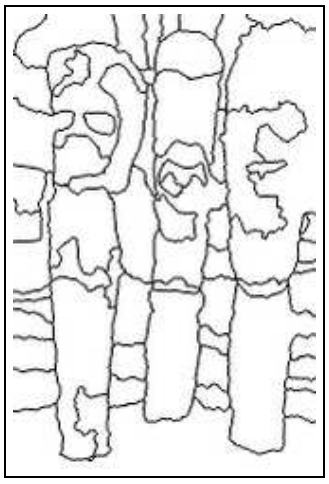
(H)



(B1)

(B2)

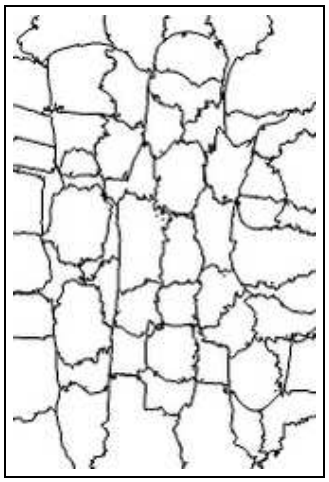
(J1)



(J2)

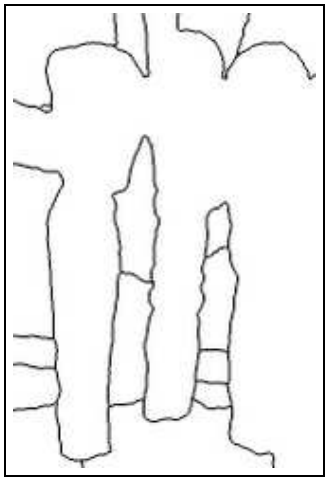
Figure 6.4: Test result 2. (I) Input image. (H) Human. (B) BlobWorld. (J) JSEG.

(N1)



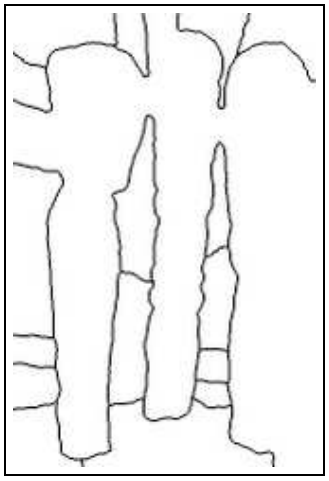
(N2)

(P1)



(P2)

(G1)



(G2)

Figure 6.4: Test result 2 (continued). (N) Ncut. (P) RP-ILP. (G) RP-G.

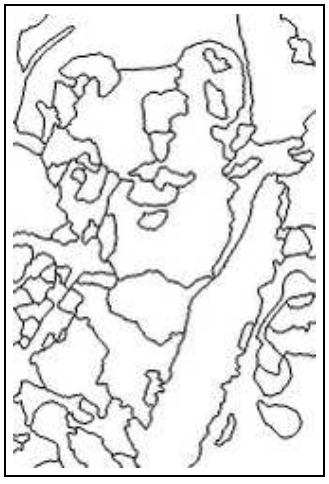
(I)

(H)

(B1)

(B2)

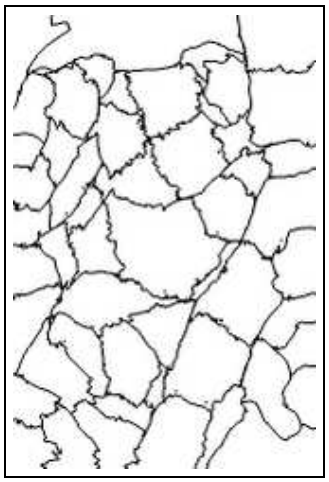
(J1)



(J2)

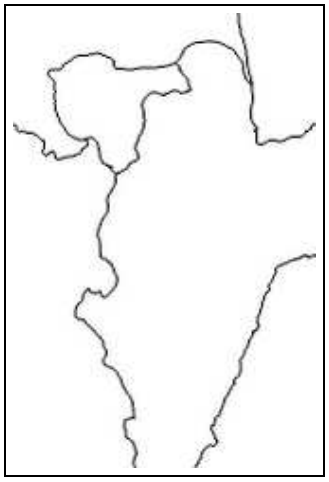
Figure 6.5: Test result 3.(I) Input image. (H) Human. (B) BlobWorld. (J) JSEG.

(N1)



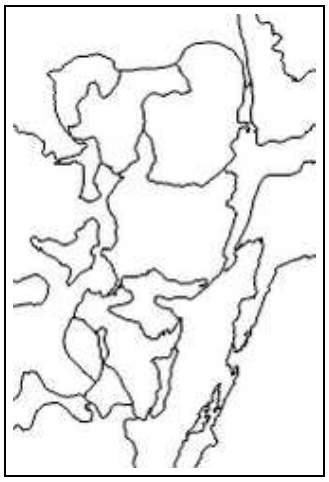
(N2)

(P1)



(P2)

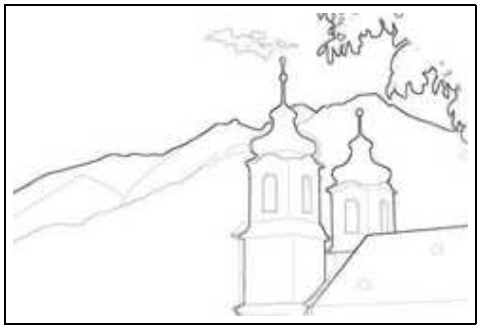
(G1)



(G2)

Figure 6.5: Test result 3 (continued). (N) Ncut. (P) RP-ILP. (G) RP-G.

(I)

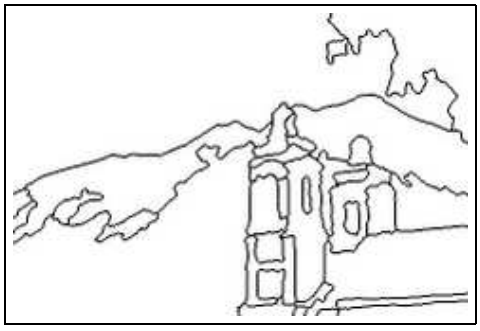


(H)

(B1)

(B2)

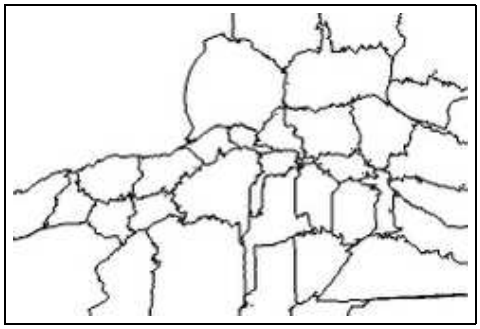
(J1)



(J2)

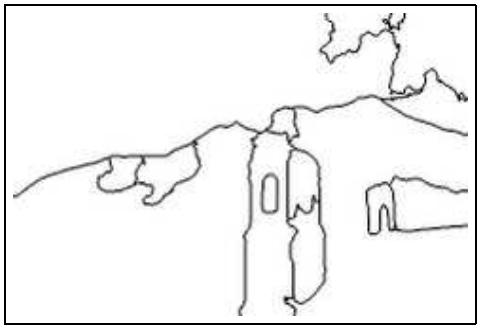
Figure 6.6: Test result 4.(I) Input image. (H) Human. (B) BlobWorld. (J) JSEG.

(N1)



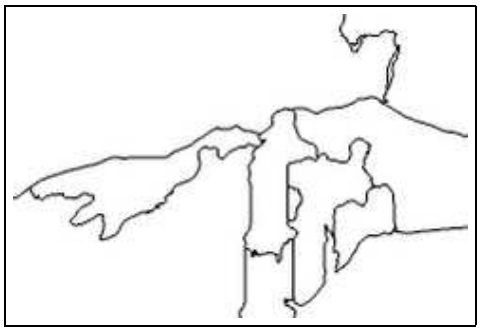
(N2)

(P1)



(P2)

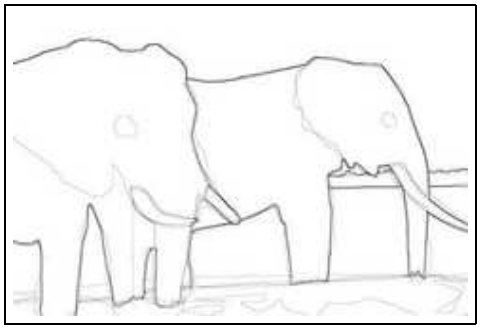
(G1)



(G2)

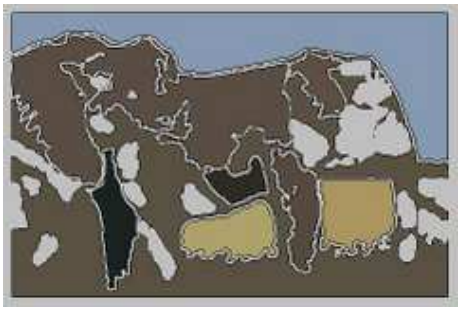
Figure 6.6: Test result 4 (continued). (N) Ncut. (P) RP-ILP. (G) RP-G.

(I)



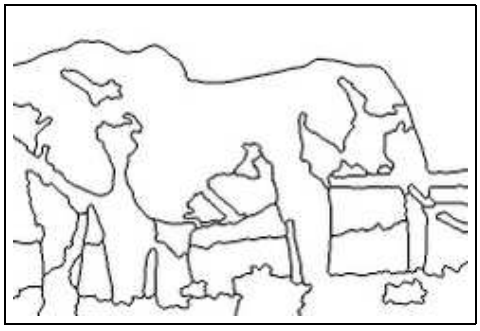
(H)

(B1)



(B2)

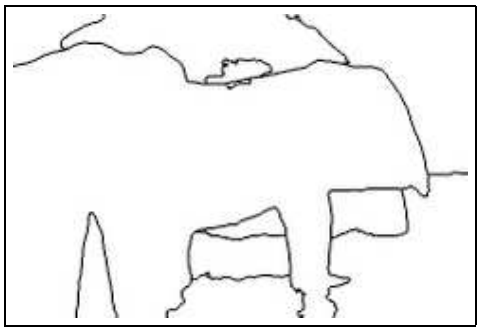
(J1)



(J2)

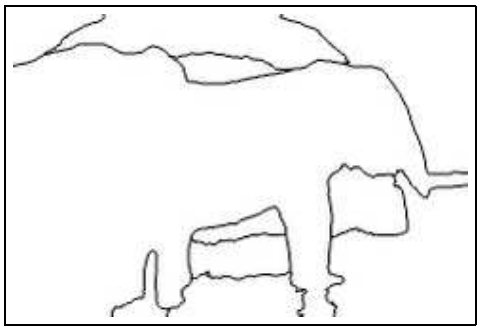
Figure 6.7: Test result 5.(I) Input image. (H) Human. (B) BlobWorld. (J) JSEG.

(P1)



(P2)

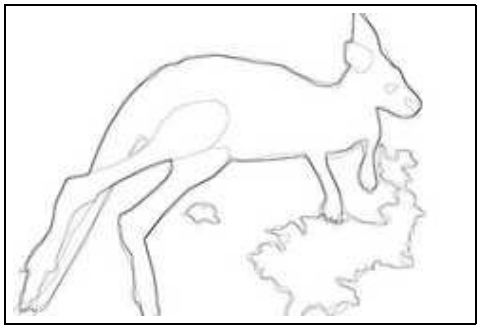
(G1)



(G2)

Figure 6.7: Test result 5 (continued). (P) RP-ILP. (G) RP-G. Ncut cannot generate result for this image.

(I)

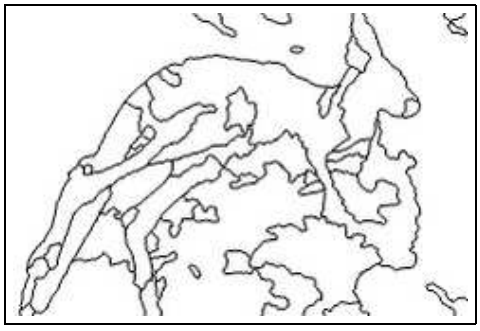


(H)

(B1)

(B2)

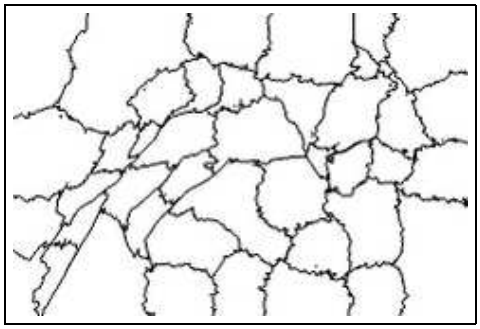
(J1)



(J2)

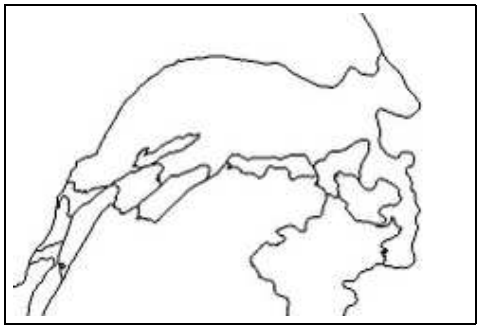
Figure 6.8: Test result 6.(I) Input image. (H) Human. (B) BlobWorld. (J) JSEG.

(N1)



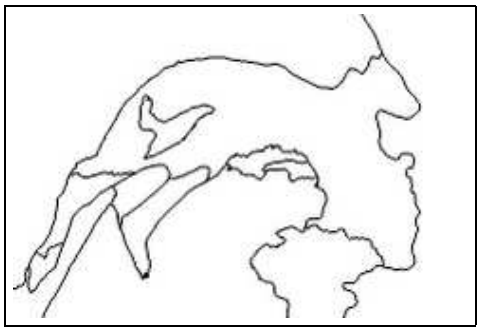
(N2)

(P1)



(P2)

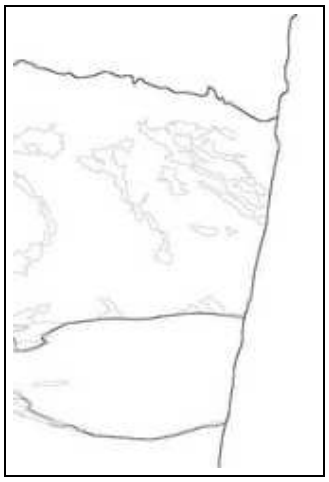
(G1)



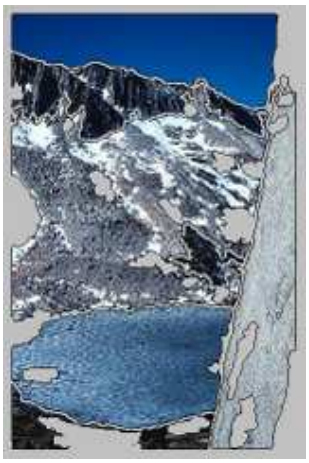
(G2)

Figure 6.8: Test result 6 (continued). (N) Ncut. (P) RP-ILP. (G) RP-G.

(I)



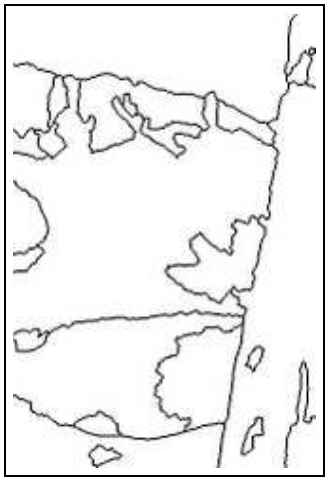
(H)



(B1)

(B2)

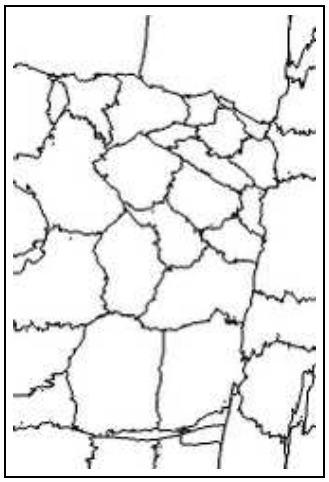
(J1)



(J2)

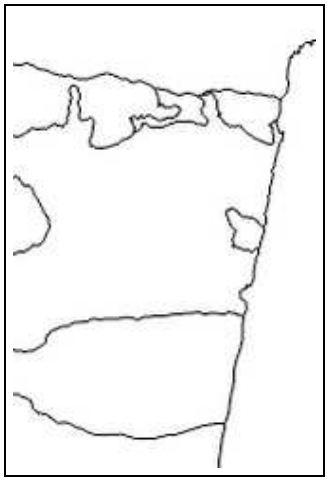
Figure 6.9: Test result 7.(I) Input image. (H) Human. (B) BlobWorld. (J) JSEG.

(N1)



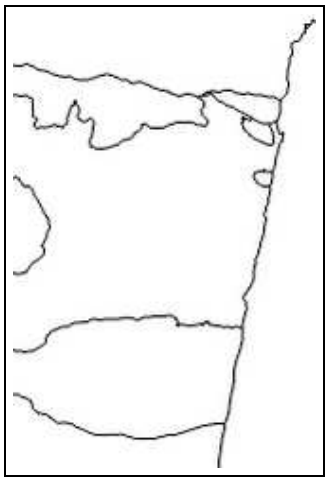
(N2)

(P1)



(P2)

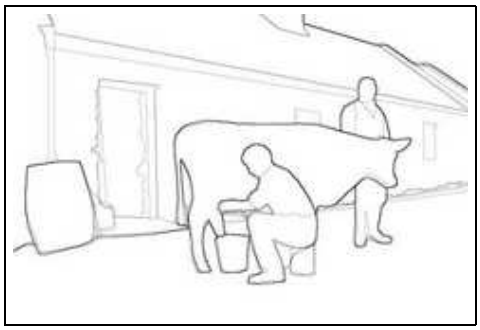
(G1)



(G2)

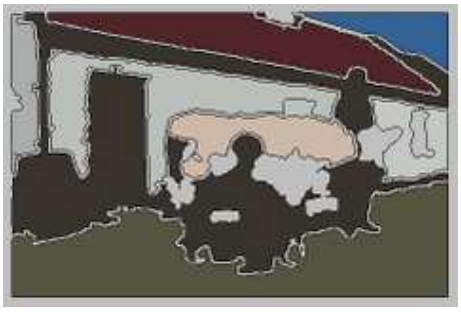
Figure 6.9: Test result 7 (continued). (N) Ncut. (P) RP-ILP. (G) RP-G.

(I)



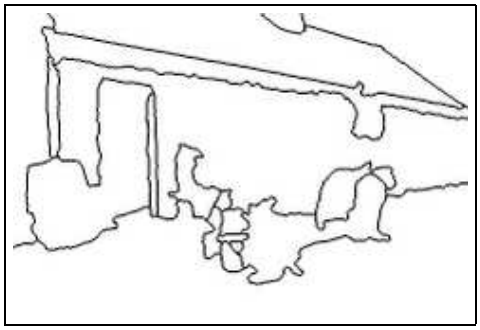
(H)

(B1)



(B2)

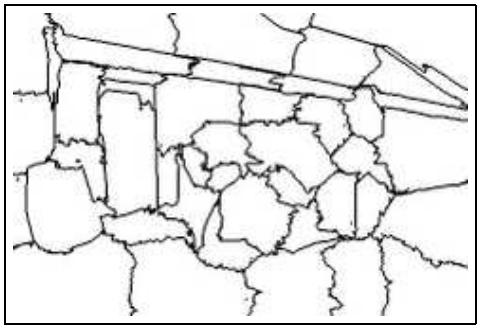
(J1)



(J2)

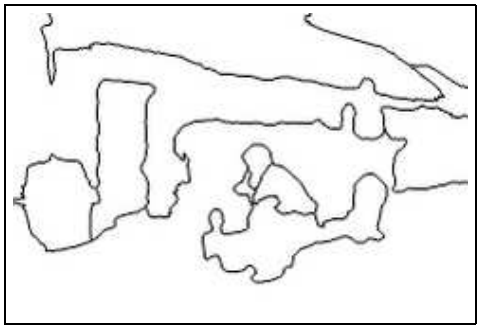
Figure 6.10: Test result 8.(I) Input image. (H) Human. (B) BlobWorld. (J) JSEG.

(N1)



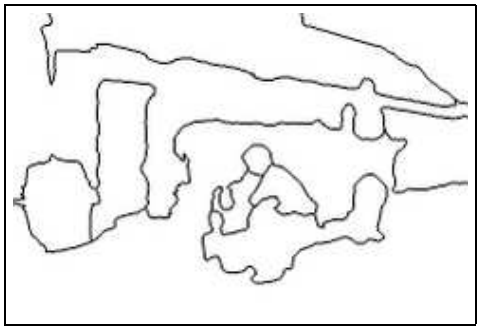
(N2)

(P1)



(P2)

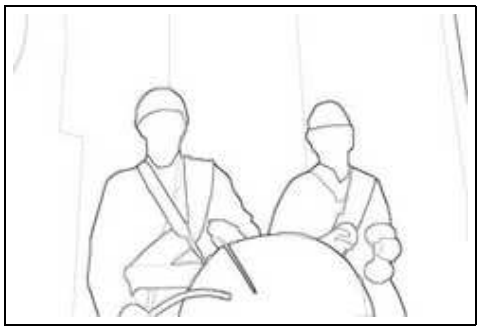
(G1)



(G2)

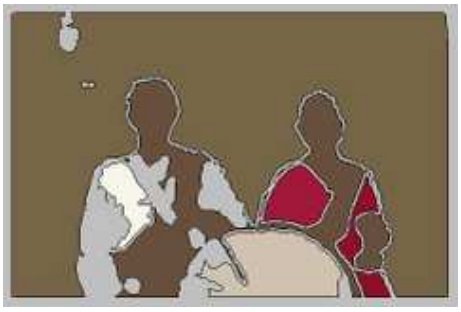
Figure 6.10: Test result 8 (continued). (N) Ncut. (P) RP-ILP. (G) RP-G.

(I)



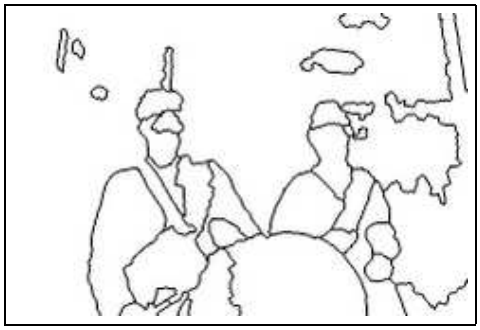
(H)

(B1)



(B2)

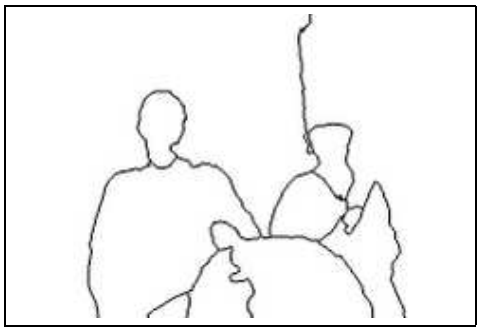
(J1)



(J2)

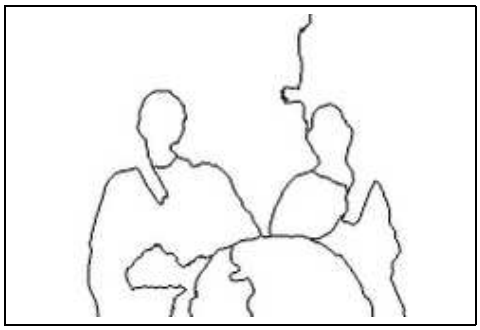
Figure 6.11: Test result 9.(I) Input image. (H) Human. (B) BlobWorld. (J) JSEG.

(P1)



(P2)

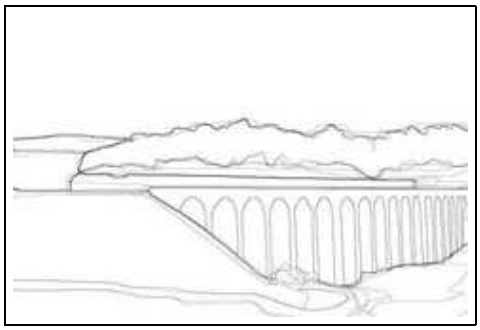
(G1)



(G2)

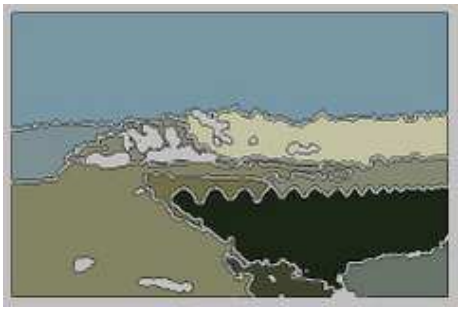
Figure 6.11: Test result 9 (continued). (P) RP-ILP. (G) RP-G. Ncut cannot generate result for this image.

(I)



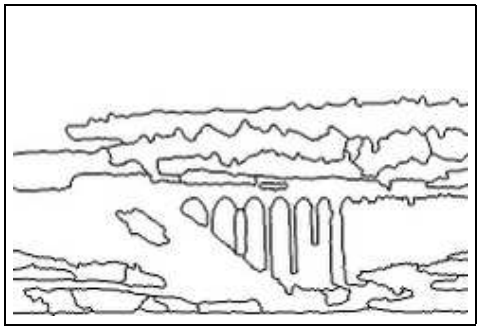
(H)

(B1)



(B2)

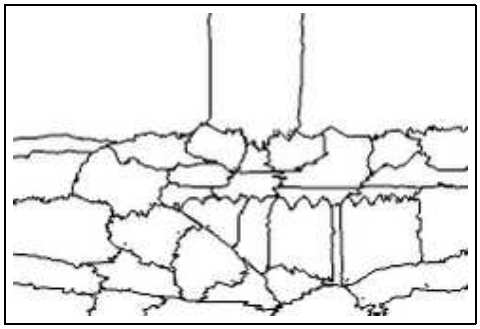
(J1)



(J2)

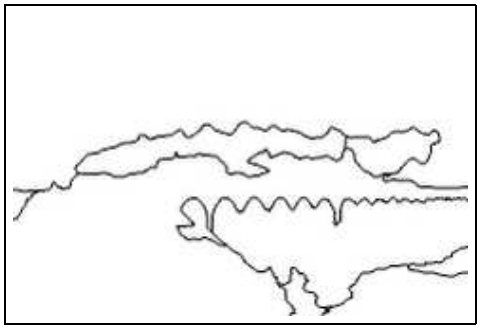
Figure 6.12: Test result 10.(I) Input image. (H) Human. (B) BlobWorld. (J) JSEG.

(N1)



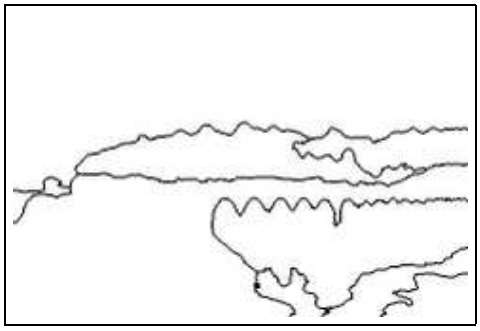
(N2)

(P1)



(P2)

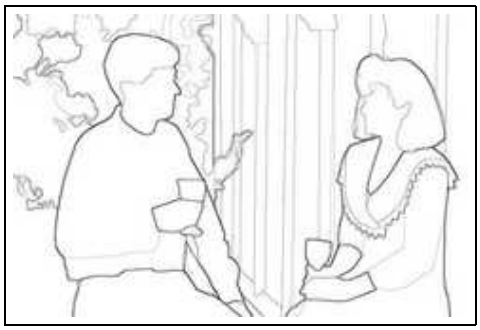
(G1)



(G2)

Figure 6.12: Test result 10 (continued). (P) RP-ILP. (G) RP-G.

(I)

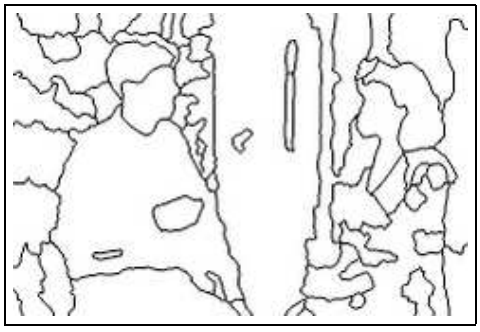


(H)

(B1)

(B2)

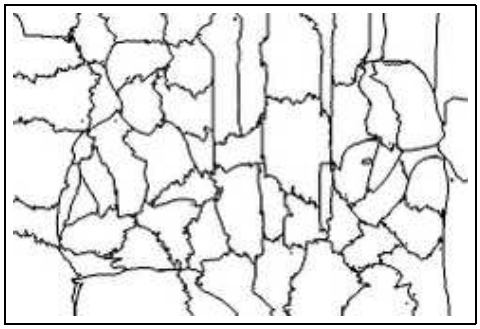
(J1)



(J2)

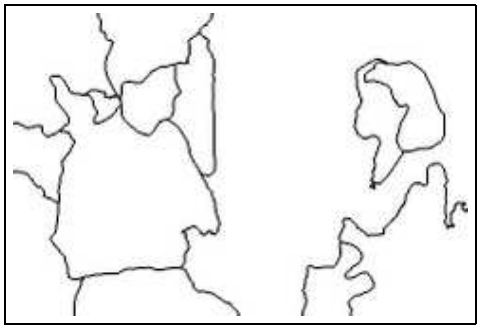
Figure 6.13: Test result 11.(I) Input image. (H) Human. (B) BlobWorld. (J) JSEG.

(N1)



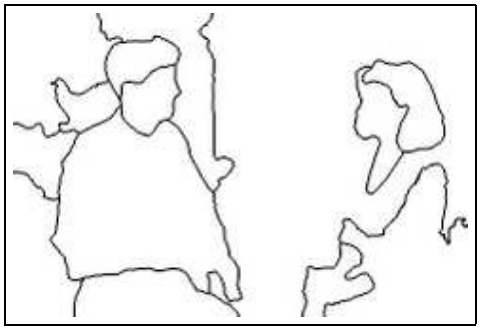
(N2)

(P1)



(P2)

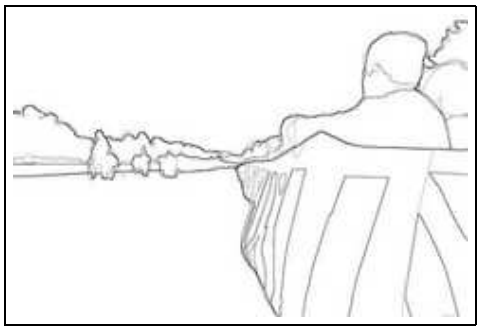
(G1)



(G2)

Figure 6.13: Test result 11 (continued). (N) Ncut. (P) RP-ILP. (G) RP-G.

(I)

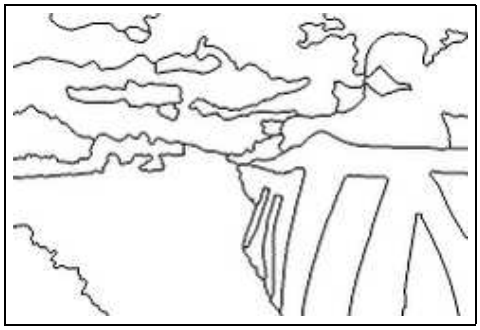


(H)

(B1)

(B2)

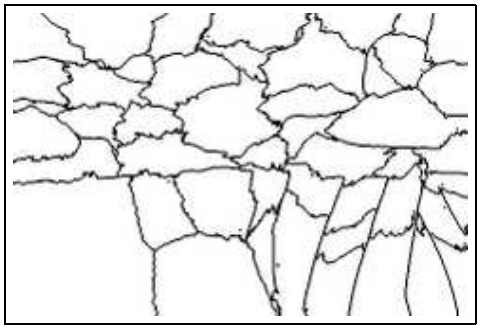
(J1)



(J2)

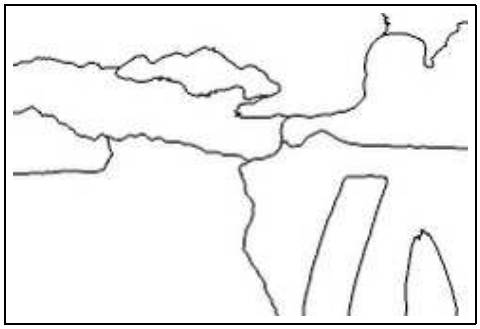
Figure 6.14: Test result 12.(I) Input image. (H) Human. (B) BlobWorld. (J) JSEG.

(N1)



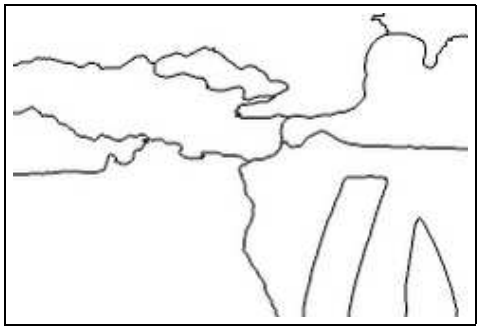
(N2)

(P1)



(P2)

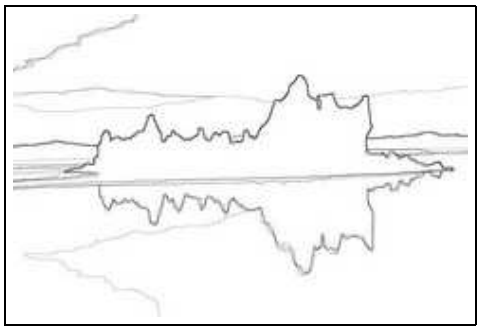
(G1)



(G2)

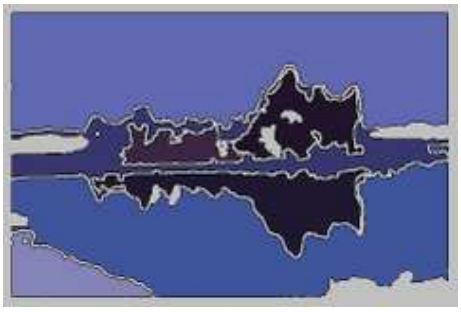
Figure 6.14: Test result 12 (continued). (N) Ncut. (P) RP-ILP. (G) RP-G.

(I)



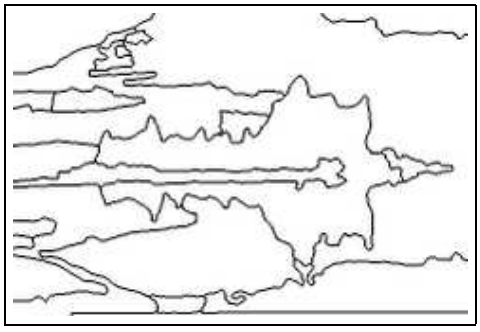
(H)

(B1)



(B2)

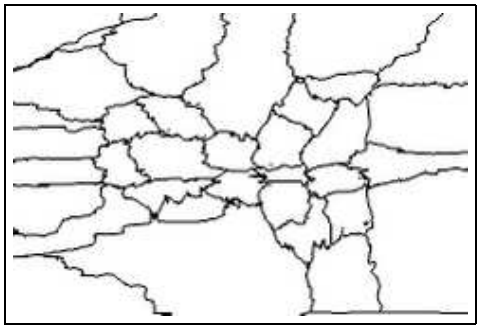
(J1)



(J2)

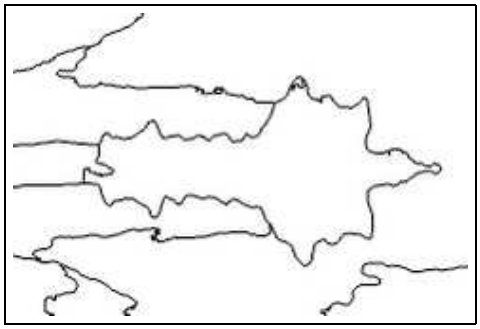
Figure 6.15: Test result 13.(I) Input image. (H) Human. (B) BlobWorld. (J) JSEG.

(N1)



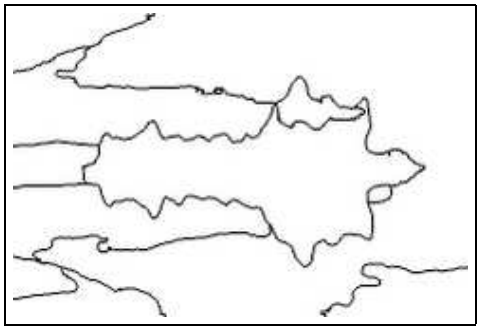
(N2)

(P1)



(P2)

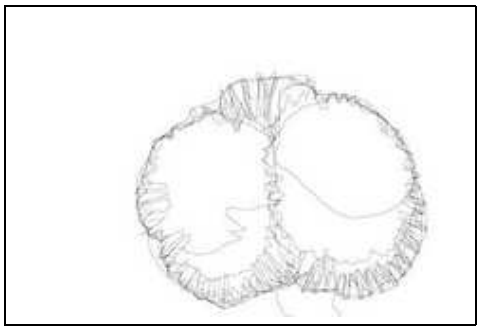
(G1)



(G2)

Figure 6.15: Test result 13 (continued). (N) Ncut. (P) RP-ILP. (G) RP-G.

(I)

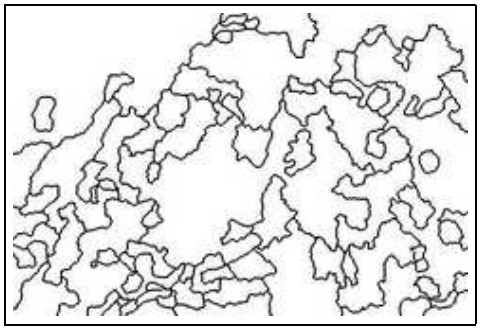


(H)

(B1)

(B2)

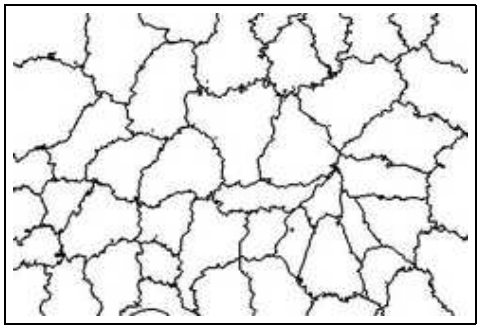
(J1)



(J2)

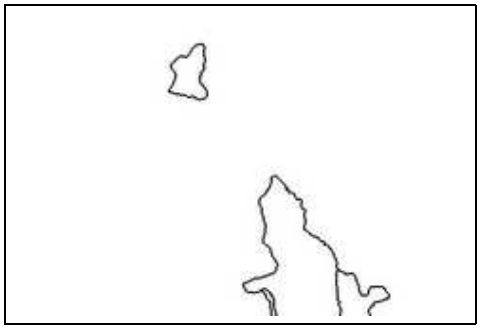
Figure 6.16: Test result 14.(I) Input image. (H) Human. (B) BlobWorld. (J) JSEG.

(N1)



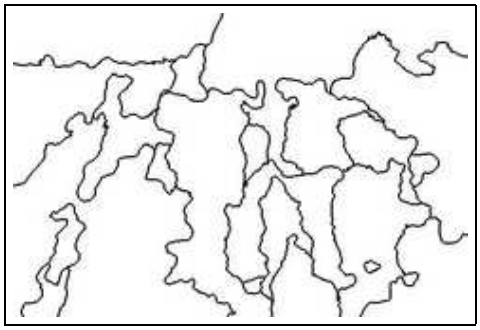
(N2)

(P1)



(P2)

(G1)



(G2)

Figure 6.16: Test result 14 (continued). (N) Ncut. (P) RP-ILP. (G) RP-G.

Chapter 7

Conclusion and Future Work

7.1 Future Work

The segmentation algorithm proposed in this thesis has made use of color histogram and region continuity features. This approach will become limited for images whose regions differ in texture or other feature instead of color distribution. So, including texture information will help to identify regions with similar colors but different textures, and thus enable this algorithm to be applicable to more images.

The graph-cut algorithm applied at the higher level tries to minimize the mean similarity between regions without directly maximizing the similarity within regions. As the problem of looking for *minimum ratio cycle*, can be solved in polynomial time [14]. This problem assumes each edge has two cost, and looks for the cycle that minimizes the ratio between the sum of the first edge cost and the sum of the second edge cost. In this thesis, the first edge cost is the similarity

between the color histograms of neighbouring blocks, and the second edge cost is just the length of each edge, i.e., the unit length 1. If the second edge cost can provide further intra-region informations, the thesis can be extended to identify regions of certain given features.

Currently, ILP is not propagated down to lower level because ILP problem is NP-hard, and the problem size grows exponentially down the image pyramid. As a general linear programming problem can be solved in polynomial time [15], if we can approximate the current ILP approach to a general linear programming problem, then the problem can be solved more efficiently.

7.2 Contribution

The main contribution of this thesis is the construction of region pyramid that preserves color distribution information. With the use of adaptive color histograms, the region pyramid requires less than twice the amount of memory in a conventional image pyramid that captures only mean or dominant color. It also enables a comprehensive segmentation to be performed at a lower resolution level to capture the main regions in the image.

Segmentation is done by interleaving adaptive thresholding and Minimum Mean Cut to provide a better control over the segmentation result as compared to graph cut algorithms alone. The segmentation done at a lower-resolution level, instead of at the finest level as what graph cut algorithms usually do, has greatly reduced over-segmentation. This is because segmentation at lower-resolution level is based on color distribution variation between the main regions whereas segmen-

tation at finest level is based on color or texture variation of the pixels or small groups of pixels.

Another contribution is the formulation of boundary refinement process by combining two approaches: (1) global optimization using Dynamic Programming and Integer Linear Programming at higher level and (2) greedy local optimization at lower levels. The global optimization finds the globally optimal refinement of boundaries at higher level with lower resolution. Greedy local optimization refines the boundaries efficiently down to the finest level. This approach combines the strength of the global optimization and local optimization without incurring much processing time. It is much faster than global optimization such as graph cut applied on the finest level and it is more accurate than using greedy algorithms alone.

7.3 Conclusion

This thesis has presented a multi-resolution region-preserving approach for image segmentation. Given an image, it constructs a pyramid of region maps at various resolutions. Each block of the map corresponds to a part of a region and it captures the region characteristics in an adaptive color histogram instead of a single color. Segmentation is performed at the top level using adaptive thresholding and Minimum Mean Cut. The coarse region boundaries found are refined using Dynamic Programming and Integer Linear Programming, and propagated down to the lowest level by a greedy method. Experimental results show that this approach can identify the main regions in many images and minimize over-

segmentation. Compared to several existing algorithms, the region boundary that it identifies are more likely to be the true boundaries. The algorithm runs quite efficiently and thus can be used for segmentation of a large number of images for semantic labeling and image retrieval.

Bibliography

[1] R. Adams and L.Bischof. Seeded region growing. *IEEE Trans. PAMI*, 16(6):641–647, 1994.

[2] P. Andrey and P. Tarroux. Unsupervised segmentation of Markov random field modeled textured images using selectionist relaxation. *IEEE Trans.PAMI*, 20(3):252–262, 1998.

[3] S. Belongie, C. Fowlkes, F. Chung, and J. Malik. Spectral partitioning with indefinite kernels using the Nystrom extension. In *Proc. ECCV*, 2002.

[4] A. Bhalerao and R. Wilson. Unsupervised image segmentation combining region and boundary estimation. *Image and Vision Computing*, 19:353–386, 2001.

[5] C. Bouman and M. Shapiro. A multiscale random field model for Bayesian image segmentation. *Image Proc.*, 3(2):162–177, 1994.

[6] P. J. Bsel and R. C. Jain. Segmentation through variable-order surface fitting. *IEEE Trans. PAMI*, 10(2):167–192, 1988.

[7] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

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

[9] M. Celenk. A color clustering technique for image segmentation. *Computer Vision, Graphics, and Image Proc.*, 52:145–170, 1990.

[10] W. Cook and A. Rohe. Computing minimum-weight perfect matchings. *INFORMS Journal on Computing*, 11:138–148, 1999.

[11] Y. Deng, B. S. Manjunath, and H. Shin. Color image segmentation. In *Proc. IEEE CVPR*, 1999.

[12] J. Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards*, pages 125–130, 1965.

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

[14] I. Jermyn and H. Ishikawa. Globally optimal regions and boundaries as minimum ratio weight cycles. *IEEE Trans. PAMI*, 23(10):1075–1088, 2001.

[15] L. G. Khachiyan. A polynomial time algorithm for linear programming. *Dokl. Akad. Nauk SSSR*, 244:1093–1096, 1979.

[16] W. K. Leow and R. Li. Adaptive binning and dissimilarity measure for image retrieval and classification. In *Proc. IEEE CVPR*, pages II–234–II–239, 2001.

[17] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on pattern Analysis and Machine Intelligence*, 26(5):530–549, 2004.

[18] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[19] W.A. Perkins. Area segmentation of images using edge points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(1):8–15, 1980.

[20] J.M. Prager. Extracting and labeling boundary segments in natural scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(1):16–27, 1980.

[21] J. Puzicha and S. Belongie. Model-based halftoning for color image segmentation. In *Proc. ICPR*, 2000.

[22] J. Puzicha, T. Hofmann, and J. M. Buhmann. Histogram clustering for unsupervised image segmentation. In *Proc. CVPR*, pages 602–608, 1999.

[23] J. Puzicha, Y. Rubner, C. Tomasi, and J. Buhmann. Empirical evaluation of dissimilarity measures for color and texture. In *Proceedings the IEEE International Conference on Computer Vision(ICCV-1999)*, pages 1165–1173, 1999.

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

[25] P. Salembier and F. Marques. Region-based representations of image and video: segmentation tools for multimedia services. *IEEE Transactions on Circuits and Systems for Video Technology*, 9:1147–1169, 1999.

[26] R. Schettini. A segmentation algorithm for color images. *Pattern Recognition Letters*, 14:499–506, 1993.

[27] P. Schroeter and J. Bigun. Hierarchical image segmentation by multi-dimensional clustering and orientation adaptive boundary refinement. *Pattern Recognition*, 28(5):695–709, 1995.

[28] E. Sharon, A. Brandt, and R. Basri. Fast multiscale image segmentation. In *Proc. CVPR*, pages 70–77, 2000.

[29] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. PAMI*, 22(8):888–905, 2000.

[30] O. Veksler. Image segmentation by nested cuts. In *Proc. CVPR*, pages 339–344, 2000.

[31] J. Z. Wang, J. L., R. M. Gray, and G. Wiederhold. Unsupervised multiresolution segmentation for images with low depth of field. *IEEE Trans. PAMI*, 23(1):85–90, 2001.

[32] S. Wang and J. M. Siskind. Image segmentation with minimum mean cut. In *Proc. ICCV*, pages 517–524, 2001.

[33] S. Wang and J. M. Siskind. Image segmentation with ratio cut. *IEEE Trans. PAMI*, 25(6):675–690, 2003.

[34] R. A. Weisenseel, W. C. Karl, D. A. Castanon, and R. C. Brewer. MRF-based algorithms for segmentation of SAR images. In *Proc. ICIP*, pages 770–774, 1998.

[35] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Trans. PAMI*, 15(11), 1993.

[36] S. C. Zhu and A. Yuille. Region competition: Unifying snakes, region growing, and Bayes/MDL for multiband image segmentation. *IEEE Trans.PAMI*, 18(9):884–900, 1996.

Appendix A

Example of Valid Edge Sequences