

國立臺灣大學電機資訊學院電信工程學研究所



碩士論文

Graduate Institute of Communication Engineering
College of Electrical Engineering and Computer Science
National Taiwan University
Master Thesis

無參數需求之高精準度與高強健性影像切割演算法
High Accuracy and High Robust Natural Image Segmentation
Algorithm without Parameter Adjusting

盧奕帆

I-FAN LU

指導教授：丁建均 博士

Advisor: Jian-Jiun Ding, Ph.D.

中華民國 104 年 6 月

June, 2015

誌謝



首先要感謝我的指導教授，丁建均老師。這兩年的研究，老師給予了許多建議與指導，每次的討論，都令我滿載而歸。除了研究，老師也會關心生活上的大小事，既窩心又感動，謝謝老師！

此外，十分感謝影像處理實驗室的大家。無論是同屆的景文、麒璋、品萱、士昌、浩軒、能謙，還是學長姐、學弟妹們，信慧、泊泓、振維、胡豪、盈柔、思維、威昇、家蓉、雅馨、巧薇、宣亦、柏任、又誠、立昂、宏毅，研究路上共同修課、承接計畫研究、參加所上活動等等，充滿了數不盡的美好回憶。實驗室歡樂而溫馨的氣氛，正是我嚮往的環境。

大學的好同窗兼實驗好夥伴，研究所還能一起修課的好同學，子謙，謝謝你在碩士班二年級專利課程上的協助與共同完成期末專題。記得課堂後總是聊天討論未來求職的大小事，往後都在台北工作，有空一定要聚聚。

兩年的研究所生涯，因為有你而美好繽紛，奕含，我摯愛的未婚妻，謝謝你總是不斷鼓勵我、為我加油打氣。研究上許多的想法與點子，常常都是在與你討論後激盪出來的。能夠與你相遇重逢，真是太幸運太奢侈了，未來我們要互相扶持照顧，一起走過每一個春夏秋冬。

最後，感謝我的爸爸、媽媽和妹妹。從小我就受到家人極大的關愛，求學路上更是無條件的支持。近年來爸爸生病，我好捨不得，謝謝爸爸無怨無悔地為這個家付出。媽媽永遠是我最好的媽媽，謝謝你辛苦扶養陪伴我成長。妹妹開始踏入社會，加油，有什麼問題哥哥借你肩膀。

短短一頁實在無法將我所有的感謝表達出來，謝謝所有幫助過我的貴人、同學、老師、朋友、家人、奕含、奕含的家人，謝謝你們，我才得以完成碩士論文與學位。

盧奕帆 謹誌

民國 104 年 7 月 7 日

國立臺灣大學明達館 531 實驗室

中文摘要



在電腦視覺和影像處理的領域中，影像分割一直是重要的基礎工作。雖然本主題已經被研究了許多年，然而，要如何在全自動、無需調整參數的前提下，仍然可以將大部分的自然影像精準的分割，仍然是一項具有挑戰性的任務。此外，近年來關於超像素(superpixel) 的研究有很大的進展，這種新技術使傳統的影像分割演算法具有更高的效率和更好的性能。在這篇論文中，我們將提出一種運用超像素等多項技術的「全自動」影像分割方法，使用者無需輸入參數或是調整參數即可得到高精確度的影像分割結果。

我們的演算法採用了基於熵率的超像素(ERSs)、邊緣偵測、顯著影像偵測以及計算紋理特徵等技術。將原始影像轉為基於熵率的超像素表示，使得演算法效率大幅提高。透過計算超像素周邊以及內部的梯度資訊，傳統的邊緣偵測得以修正，進一步防止超像素過度合併。利用顯著影像偵測與計算紋理特徵，超像素合併的門檻值可根據影像作自適應調整，避免影像過度分割。模擬結果顯示，對於任意自然影像的分割處理，我們方法表現的分割結果相當符合人類感知，且不需要額外的參數調整，而這也超越了其他現有已知最先進的方法的模擬結果。

關鍵字：影像分割、基於熵率的超像素、邊緣偵測、顯著影像偵測、電腦視覺。

ABSTRACT



In computer vision and image processing, image segmentation is always an important fundamental work. Though this topic has been researched for many years, it is still a challenging task to well segment most of the natural images automatically without adjusting any parameter. Recently, the researches of superpixels have great improvement. This new technique makes the traditional segmentation algorithms more efficient and has better performances. In this thesis, an automatic image segmentation algorithm based on superpixels and many other techniques is proposed. It can accurately segment almost all of the natural images without parameter adjustment.

In our algorithm, the techniques of entropy rate superpixels (ERSs), edge detection, saliency detection, and computing texture feature are adopted. With the aid of ERSs, the proposed algorithm can be implemented very efficiently. To prevent over-merge of superpixels, modified edge detection which computes the gradient information of the contours and the interiors of superpixels is used. Saliency detection and the texture features of an image are also used to prevent over-segmentation. Moreover, an adaptive threshold is also used for superpixel merging. These techniques make the segmentation result more consistent with human perception without adjusting any parameter. Simulations show that our proposed method can well segment most of natural images and outperform state-of-the-art methods.

Index terms: Image segmentation; ERS superpixels; edge detection; saliency detection; computer vision

CONTENTS



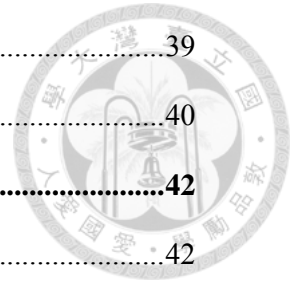
口試委員會審定書	
誌謝	i
中文摘要	ii
ABSTRACT	iii
CONTENTS	iv
LIST OF FIGURES	vii
LIST OF TABLES	ix
Chapter 1 Introduction.....	1
1.1 Motivation	1
1.2 Main Contribution	2
1.3 Organization	2
Chapter 2 Reviews of Segmentation Algorithms.....	4
2.1 Mean Shift	4
2.1.1 About Mean Shift	4
2.1.2 The Algorithm	5
2.1.3 Simulation Results	5
2.2 Watershed Approach	6
2.2.1 About Watershed Approach.....	6
2.2.2 The Algorithm	7
2.2.3 Simulation Results	8
2.3 Normalized Cut.....	9
2.3.1 About Normalized Cut	9



2.3.2	The Grouping Algorithm.....	11
2.3.3	Simulation Results	13
2.3.4	Multiscale Normalized Cut (MNCut)	14
2.4	Efficient Graph-based Segmentation.....	15
2.4.1	About Efficient Graph-based Segmentation	15
2.4.2	The Algorithm	17
2.4.3	Simulation Results	19

Chapter 3 Reviews of Superpixels and Superpixel-based Segmentation

	Algorithms	20
3.1	Introduction of Superpixel.....	20
3.2	Simple Linear Iterative Clustering Superpixels.....	21
3.2.1	About Simple Linear Iterative Clustering Superpixels	21
3.2.2	The Algorithm	24
3.2.3	Simulation Results and Discussion	25
3.3	Entropy Rate Superpixel Segmentation.....	26
3.3.1	About Entropy Rate Superpixel Segmentation	26
3.3.2	Preliminaries of this Method.....	27
3.3.3	Problem Formulation and Algorithm.....	28
3.3.4	Conclusion and Simulation Results	30
3.4	Segmentation by Aggregating Superpixels (SAS).....	32
3.4.1	About Segmentation by Aggregating Superpixels	32
3.4.2	The Algorithm	36
3.4.3	Simulation Results and Discussion	38
3.5	Learning Full Pairwise Affinities for Spectral Segmentation (MLSS).....	39
3.5.1	About MLSS (Multi-Layer Spectral Segmentation)	39



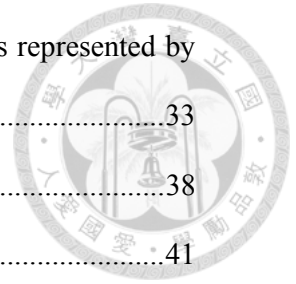
3.5.2	The Algorithm	39
3.5.3	Simulation Results	40
Chapter 4	Proposed Algorithm	42
4.1	Introduction	42
4.2	Supapixel Generation and Saliency Detection	45
4.2.1	Supapixel Generation	45
4.2.2	Saliency Detection	46
4.3	Modified Edge Detection and Texture Features	48
4.3.1	Modified Edge Detection	48
4.3.2	Texture Features	51
4.4	Proposed Segmentation Algorithm	53
4.5	Analysis of Our Algorithm	58
Chapter 5	Simulations	65
5.1	Compared with the state-of-the-art methods	65
5.1.1	Database	65
5.1.2	Visual Comparison	68
5.2	Compared with the MLSS method and the SAS method	77
Chapter 6	Conclusion and Future Work.....	84
6.1	Conclusion	84
6.2	Future Work	85
REFERENCE		86
PUBLICATION.....		90

LIST OF FIGURES



Fig. 1.1 Natural image segmentation based on our method.	3
Fig. 2.1 Segmentation examples based on the mean shift approach.	6
Fig. 2.2 Segmentation examples based on the watershed approach.	8
Fig. 2.3 A case where minimum cut gives a bad partition.	10
Fig. 2.4 (a) shows an image of a zebra. The remaining images show the major components of the partition. The texture features used correspond to convolutions with DOOG filters at six orientations and five scales.	13
Fig. 2.5 Segmentation using the nearest neighbor graph can capture spatially non-local regions ($\sigma = 0.8, k = 300$).	19
Fig. 3.1 Reducing the superpixel search regions. (a) standard k-means searches the entire image. (b) SLIC searches a limit region.	22
Fig. 3.2 The superpixel result of SLIC superpixels and other state-of-the-art superpixel methods.	25
Fig. 3.3 The role of entropy rate in obtaining compact and homogeneous clustering.	29
Fig. 3.4 The role of the balancing function is to obtain clusters of similar sizes.	30
Fig. 3.5 Superpixel segmentation examples. The images contain 100 superpixels. The ground truth segments are color-coded and blended on the images. The superpixels (boundaries shown in white) respect object boundaries and tend to divide an image into similar-sized regions.	31
Fig. 3.6 The proposed bipartite graph model with K over-segmentations of an image. A black dot denotes a pixel while a red square denotes a superpixel. For the graph, each pixel is connected to the superpixels containing it and each superpixel is connected to itself and its nearest neighbor in the feature space	

among its spatially adjacent superpixels . Each superpixel is represented by the average LAB color of the pixels within it.	33
Fig. 3.7 Segmentation examples on Berkeley Segmentation Database.....	38
Fig. 3.8 Segmentation examples using MLSS.....	41
Fig. 4.1 The overall flow chart of our framework.....	43
Fig. 4.2 The overview of our framework.....	44
Fig. 4.3 An example of the processed result generated by ERS superpixel method.	45
Fig. 4.4 Diagram of the saliency detection model.....	47
Fig. 4.5 A sobel edge detection mask	49
Fig. 4.6 A 5x5 standard Laplacian of Gaussian edge detection mask	50
Fig. 4.7 Comparison of Gabor and Log-Gabor on both linear and logarithmic spatial frequency scales.....	52
Fig. 4.8 Segment image with ERS superpixel method.....	53
Fig. 4.9 The segmentation result from Stage 1	55
Fig. 4.10 The saliency map of an image.....	56
Fig. 4.11 The segmentation result from Stage 2	57
Fig. 4.12 An example of merging two superpixels with high complexity levels.	60
Fig. 4.13 Due to two superpixels with low complexity levels, the proposed algorithm would not merge them.	60
Fig. 5.1 Our image database.....	66
Fig. 5.2 The visual comparison between our proposed method and the other state-of-the- art methods.	74
Fig. 5.3 Compare our segmentation method with the MLSS method.....	80
Fig. 5.4 Compare our segmentation method with the SAS method.....	83
Fig. 6.1 (a) Original image. (b) Specular-free image.	85



LIST OF TABLES



Table 4.1 The revised edge rate in **Stage 2**.....63

Chapter 1 Introduction



1.1 Motivation

Image segmentation is to partition a single image into non-overlapping regions, and then extract the objects of interest under the complex background environment. It has been found, with a great potential in applications, to be not only a fundamental low-level vision problem but also the key problem in the research of image analysis, pattern recognition, computer vision, medical image processing, and even image understanding.

In natural images, many problems make segmentation difficult, such as determining faint object boundaries and separating highly cluttered backgrounds. While human can parse a natural image into coherent regions easily, it is found rather difficult for automatic vision systems. Despite a variety of segmentation techniques have been proposed, it remains challenging for any single method to do segmentation successfully due to the broad diversity and ambiguity of visual patterns in a natural image.

In the last two decades, the researches of superpixel segmentation have great improvement. The major advantage of using superpixels is computational efficiency. A superpixel representation greatly reduces the number of image primitives compared to the pixel representation. On the other hand, the saliency detection is a new topic of image processing. The task of saliency detection is to identify the most important and informative part of a scene.

Therefore, in this thesis, we aim at utilizing the techniques of superpixel, saliency detection and edge detection to construct a high accuracy and high robust natural image segmentation algorithm without parameter adjusting.

1.2 Main Contribution

Our main contributions are summarized as follows.

1. We review the researches in image segmentation, recent works of superpixels and superpixel-based segmentation algorithms.
2. We propose a segmentation framework which combines superpixel technique, saliency detection, texture feature oriented clustering and modified edge detection which computes the gradient information of the contours and the interiors of superpixels
3. Experimental results show that our method has higher accuracy in natural image segmentation compared to the existing methods.

1.3 Organization

This thesis is organized as follows. An overview of recent image segmentation approaches is presented in Chapter 2. In Chapter 3, the fundamental knowledge of superpixel is illustrated and some superpixel-based segmentation algorithms are introduced. The proposed image segmentation framework as well as algorithms will be presented in Chapter 4. Experimental results for natural image segmentation and discussion are given in Chapter 5. Finally, the thesis would be concluded in Chapter 6.



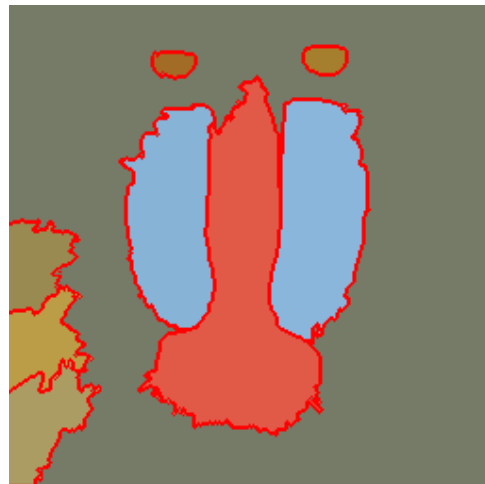
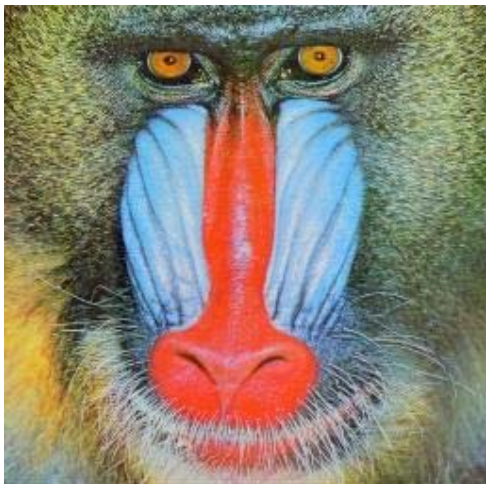


Fig. 1.1 Natural image segmentation based on our method.

Chapter 2 **Reviews of Segmentation Algorithms**

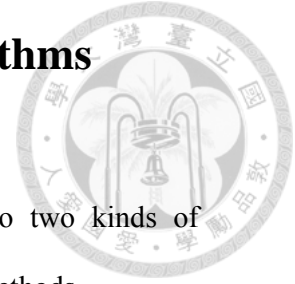


Image segmentation algorithms can usually be categorized into two kinds of methods; one is graph-based methods and the other is gradient ascent methods.

Starting from a rough initial clustering of pixels, gradient ascent methods iteratively refine the clusters until some convergence criterion is met to form segmentation result. In Section 2.1-2.2, we review two classic image segmentation algorithms based on gradient ascent methods.

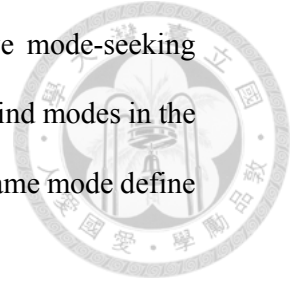
The Graph-based approaches treat each pixel as a node in a graph. Edge weights between two nodes are proportional to the similarity between neighboring pixels. The segmentation result is created by minimizing a cost function defined over the graph. In Section 2.3-2.4, we introduce segmentation image algorithms which are the graph-based approaches.

2.1 Mean Shift

2.1.1 About Mean Shift

Mean shift, a simple iterative procedure that shifts each data point to the average of data points in its neighborhood.

The generalization of mean shift makes some k-means like clustering algorithms its special cases. It is shown that mean shift is a mode-seeking process on a surface constructed with a “shadow” kernel. For Gaussian kernels, mean shift is a gradient mapping. Convergence is studied for mean shift iterations. Cluster analysis is treated as a deterministic problem of finding a fixed point of mean shift that characterizes the data.



For the application of the superpixel, Mean shift [4], an iterative mode-seeking procedure for locating local maxima of a density function, is applied to find modes in the color or intensity feature space of an image. Pixels that converge to the same mode define the superpixels.

2.1.2 The Algorithm

Let x_i and z_i , $i=1, \dots, n$, be the d -dimensional input and filtered image pixels in the joint spatial-range domain and L_i the label of the i^{th} pixel in the segmented image.

1. Run the mean shift filtering procedure for the image and store all the information about the d -dimensional convergence point in z_i , i.e., $z_i = y_{i,c}$.

2. Delineate in the joint domain the clusters $\{C_p\}_{p=1 \dots m}$ by grouping together all z_i which are closer than h_s in the spatial domain and h_r in the range domain, i.e., concatenate the basins of attraction of the corresponding convergence points.

3. For each $i=1, \dots, n$, assign $L_i = \{p \mid z_i \in C_p\}$.

4. Optional: Eliminate spatial regions containing less than M pixels.

2.1.3 Simulation Results

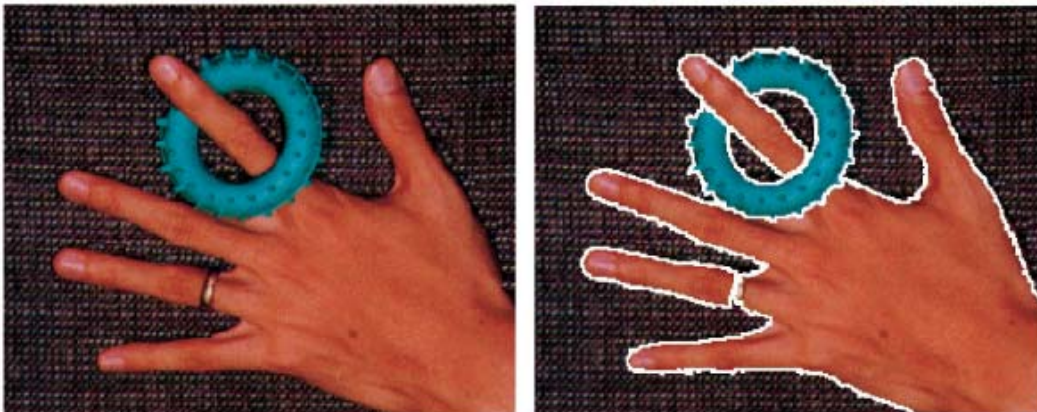




Fig. 2.1 Segmentation examples based on the mean shift approach.

2.2 Watershed Approach

2.2.1 About Watershed Approach

The watershed approach [6] performs a gradient ascent starting from local minima to produce watersheds, lines that separate catchment basins. The resulting superpixels are often highly irregular in size and shape, and do not exhibit good boundary adherence.

The watershed transform is one of the classic methods for image segmentation, and has been studied for three decades. The basic idea of watershed segmentation is to consider the regions to be extracted as catchment basins in topology. The watershed lines, S , are the boundaries of catchment basins. Starting from the minima of lowest altitude, the water will progressively fill up all the different catchment basins. At some point the water one basin will start to merge with water from neighboring regions. This merging

can be prevented by constructing dams at high altitude.

Another thought that is close to watershed is Toboggan contrast enhancement, proposed by Fairfield in 1990. An image is first applied the toboggan contrast enhancement and then a contrast segmentation. The concept is later used to implement watershed segmentation. Conceptually, we can view the original watershed segmentation as an approach that starts from low altitude to high altitude, and the toboggan approach as an approach that starts from high altitude to low altitude. In the toboggan approach, the algorithm tries to find a downstream path from each pixel. Pixels that slide into the same local minimum can then be grouped together into a catchment basin.

The segmentation produced by a naive application of the watershed algorithm is often inadequate: the image is usually over-segmented into a large number of minuscule regions. As a result, several post-processing steps have been proposed to produce results that match human eye. The most common method is to use markers for identifying relevant minima. However, finding markers can be problematic and produce inadequate results.

2.2.2 The Algorithm

1. A set of markers, pixels where the flooding shall start, are chosen. Each is given a different label.
2. The neighboring pixels of each marked area are inserted into a priority queue with a priority level corresponding to the gray level of the pixel.
3. The pixel with the highest priority level is extracted from the priority queue. If the neighbors of the extracted pixel that have already been labeled all have the same label, then the pixel is labeled with their label. All non-marked neighbors that are not yet in the priority queue are put into the priority queue.



4. Redo step 3 until the priority queue is empty. The non-labeled pixels are the watershed lines.



2.2.3 Simulation Results

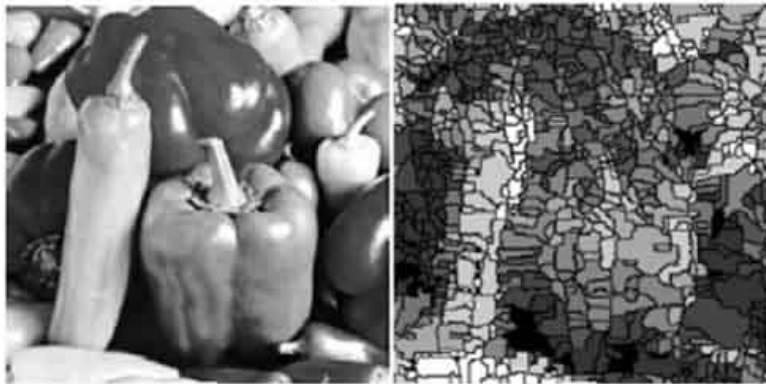


Fig. 2.2 Segmentation examples based on the watershed approach.



2.3 Normalized Cut

2.3.1 About Normalized Cut

The Normalized cuts algorithm [1] recursively partitions a graph of all pixels in the image using contour and texture clues, globally minimizing a cost function defined on the edges at the partition boundaries. It produces very regular, visually pleasing superpixels. However, the boundary adherence of this algorithm is relatively poor and it is the slowest among the methods (particularly for large images), although there are some method attempting to speed up the algorithm, the problem of complexity is still exist.

This approach avoids trivial partitions of the affinity weighed graph with the use of the normalized cut criterion. Based on the spectral graph theory, given a graph $G=(V,E)$, it can be partitioned into two disjoint sets, A,B , $A \cup B = V$, $A \cap B = \phi$, by simply removing edges connecting the two parts. The degree of dissimilarity between these two pieces called as *cut* is as follows

$$cut(A,B) = \sum_{u \in A, v \in B} w(u,v) \quad (2.1)$$

where $w(u,v)$ is the similarity between node u and v . The traditional optimal bipartitioning of a graph is the one that minimizes this cut value. Finding the minimum cut is a well-studied problem and there exist efficient algorithms for solving it. However, the minimum cut criteria favors cutting small sets of isolated nodes in the graph, and gives bad partition in some cases such as Fig. 2.3.

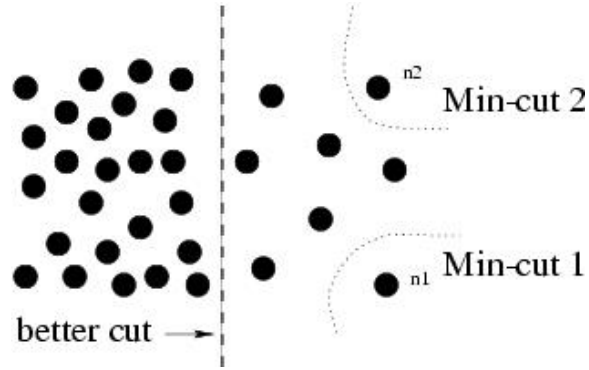


Fig. 2.3 A case where minimum cut gives a bad partition.

This criterion has some unnatural bias for partitioning out small sets of points. To avoid it, a new measure of disassociation between two groups is proposed. Instead of looking at the value of total edge weight connecting the two partitions, the measure computes the cut cost as a fraction of the total edge connections to all the nodes in the graph. This disassociation measure is called as the *normalized cut* (Ncut).

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}, \quad (2.2)$$

where $assoc(A, V) = \sum_{u \in A, t \in V} w(u, t)$ is the total connection from node A to all nodes in the graph and $assoc(B, V)$ is similarly defined.

To obtain the optimal bipartitioning of a graph is equal to minimize the criterion mentioned above. Let $\mathbf{d}(i) = \sum_j w(i, j)$ be the total connection from node i to all other nodes. Let \mathbf{D} be an $N \times N$ diagonal matrix with \mathbf{d} on its diagonal, $D_{ii} = \sum_j w(i, j)$, \mathbf{W} be an $N \times N$ symmetric affinity matrix whose entries encode the similarity between pixels, $\mathbf{W}(i, j) = w(i, j)$. Shi and Malik proved that is equivalent to the discrete optimization problem.

$$\min_x Ncut(x) = \min_y \frac{\mathbf{y}^T (\mathbf{D} - \mathbf{W}) \mathbf{y}}{\mathbf{y}^T \mathbf{D} \mathbf{y}} \quad (2.3)$$

If y is relaxed to take real values, the above criterion can be approximated by solving the generalized eigenvalue system.

$$(D-W)y = \lambda Dy, \quad (2.4)$$

Amazingly, the second smallest eigenvector y gives the solution of the normalized cut problem.

2.3.2 The Grouping Algorithm

Given an image or image sequence \mathbf{I} , set up a weighted graph $G = (V, E)$, the node \mathbf{V} are pixels of the image \mathbf{I} and the edge \mathbf{E} is the weight on connecting two nodes to be a measure of the similarity between the two nodes. Let \mathbf{N} be the number of nodes (pixels), i.e., $|\mathbf{V}|$.

Step 1

Construct an $N \times N$ symmetric similarity matrix \mathbf{W} as:

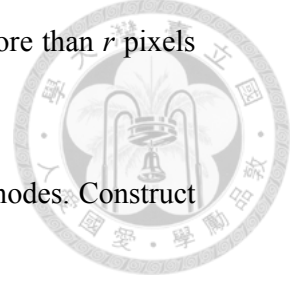
$$w(i, j) = \exp \frac{-\|F(i) - F(j)\|_2^2}{\sigma_f^2} * \begin{cases} \exp \frac{-\|X(i) - X(j)\|_2^2}{\sigma_x^2} & \text{if } \|X(i) - X(j)\|_2 < r \\ 0 & \text{otherwise} \end{cases} \quad (2.5)$$

where $\mathbf{X}(i)$ is the spatial location of node i , i.e., the coordinates in the original image \mathbf{I} , and $\mathbf{F}(i)$ is a feature vector defined as:

- $\mathbf{F}(i) = 1$ for segmenting point sets,
- $\mathbf{F}(i) = \mathbf{I}(i)$, the intensity value, for segmenting brightness (gray scale) images,
- $\mathbf{F}(i) = [v, u \cdot s \cdot \sin(h), v \cdot s \cdot \cos(h)](i)$, where h, s, v are the HSV values, for color segmentation,
- $\mathbf{F}(i) = [|\mathbf{I} * f_1|, \dots, |\mathbf{I} * f_n|](i)$, where the f_i are DOOG filters at various scales and orientations, for texture segmentation.

Note that the weight $w(i, j) = 0$ for any pair of nodes i and j that are more than r pixels apart.

Let $d(i) = \sum_j w(i, j)$ be the total connection from node i to all other nodes. Construct an $N \times N$ diagonal matrix \mathbf{D} with \mathbf{d} on its diagonal.



Step 2

Solve a generalized eigensystem,

$$(D - W)y = \lambda Dy \quad (2.6)$$

and get an eigenvector with the second smallest eigenvalue.

Step 3

Use the eigenvector to bipartition the graph. In the ideal case, the eigenvector should only take on two discrete values, and the signs tell us exactly how to partition the graph

$$(A = \{V_i | y_i > 0\}, B = \{V_i | y_i \leq 0\}).$$

However, y is relaxed to take real values, therefore, we need to choose a splitting point.

There are several ways such as

- Take 0
- Take median
- Search a splitting point which results in that $Ncut(A, B)$ is minimized.

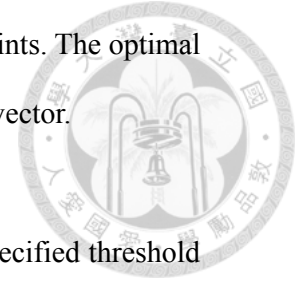
The splitting point which minimizes $Ncut$ value also minimizes

$$\frac{y^T (D - W)y}{y^T Dy} \quad (2.7)$$

where $y = (1+x) - b(1-x)$, $b = k / (1-k)$, $k = \frac{\sum_{x_i > 0} d_i}{\sum_i d_i}$

where x is an N dimensional indicator, $x_i = 1$ if node i is in A and -1 , otherwise.

To find the minimal Ncut, we need to try different values of splitting points. The optimal splitting point is generally around the mean value of the obtained eigenvector.



Step 4

Repeat bipartition recursively. Stop if Ncut value is larger than a pre-specified threshold value (Large Ncut value means that there is no clear partition point any more).

Furthermore, stop if the total number of nodes in the partition (Area) is smaller than a pre-specified threshold value (this is another criteria added newly to the paper's algorithm.)

2.3.3 Simulation Results

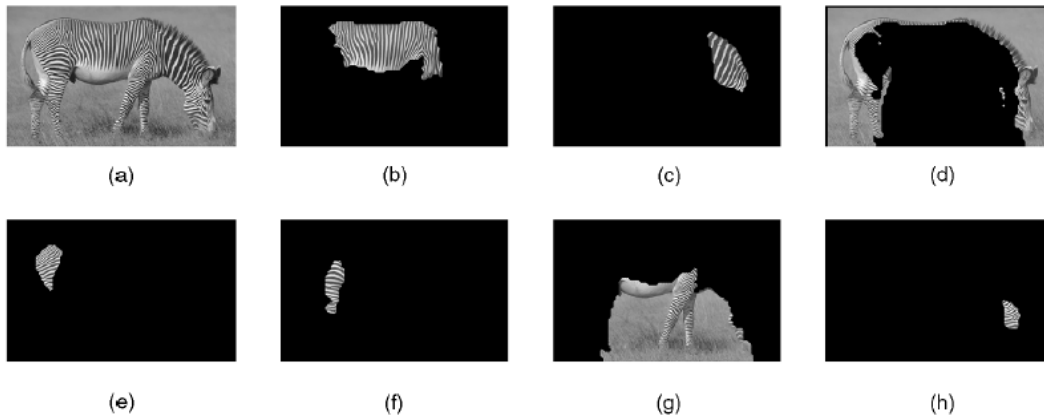
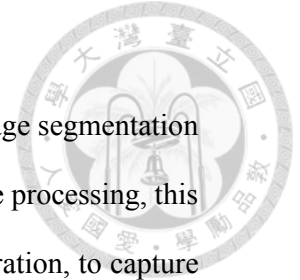


Fig. 2.4 (a) shows an image of a zebra. The remaining images show the major components of the partition. The texture features used correspond to convolutions with DOOG filters at six orientations and five scales.

2.3.4 Multiscale Normalized Cut (MNCut)

Cour *et al.* proposed MNCut [9], which is a multiscale spectral image segmentation algorithm, to segment large images. In contrast to most multiscale image processing, this algorithm works on multiple scales of the image in parallel, without iteration, to capture both coarse and fine level details. The algorithm is computationally efficient, allowing to segment large images. It use the Normalized Cut graph partitioning framework of image segmentation and construct a graph encoding pairwise pixel affinity, and partition the graph for image segmentation. They demonstrate that large image graphs can be compressed into multiple scales capturing image structure at increasingly large neighborhood. They also show that the decomposition of the image segmentation graph into different scales can be determined by ecological statistics on the image grouping cues. Images that previously could not be processed because of their size have been accurately segmented thanks to this method.





2.4 Efficient Graph-based Segmentation

2.4.1 About Efficient Graph-based Segmentation

Felzenszwalb and Huttenlocher [2] propose an alternative graph-based approach that has been applied to generate superpixels. It performs an agglomerative clustering of pixels as nodes on a graph such that each superpixel is the minimum spanning tree of the constituent pixels. This algorithm adheres well to image boundaries in practice, but produces superpixels with very irregular sizes and shapes. It is fast in practice. However, it does not offer an explicit control over the amount of superpixels or their compactness.

Their goal is to develop computational approaches to image segmentation that are broadly useful, much in the way that other low-level techniques such as edge detection are used in a wide range of computer vision tasks. In order to achieve such broad utility, it is important that a segmentation method have the following properties:

1. Capture perceptually important groupings or regions, which often reflect global aspects of the image. Two central issues are to provide precise characterizations of what is perceptually important, and to be able to specify what a given segmentation technique does. There should be precise definitions of the properties of a resulting segmentation, in order to better understand the method as well as to facilitate the comparison of different approaches.

2. Be highly efficient, running in time nearly linear in the number of image pixels. In order to be of practical use, the segmentation methods should run at speeds similar to edge detection or other low-level visual processing techniques, meaning nearly linear time and with low constant factors. For example, a segmentation technique that runs at several frames per second can be used in video processing applications.

For the first property, they want the segmentation to be neither too coarse nor too fine. In other words, it means the ability to preserve detail in low-variability image regions while ignoring detail in high-variability regions. This is achieved by an adaptive threshold which considers not only the differences between other regions but also the internal local variation.

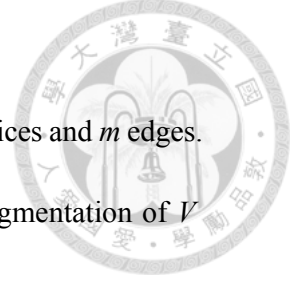
For the second property, they introduce a simple but effective modification of Kruskal's algorithm. As in Kruskal's algorithm,

- it begins with the completely disconnect graph,
- edges are added one at a time in increasing order of their weight,
- it maintains a forest of MSTs for its current components.

Combining the two properties mentioned above, the segmentation process is as follows. Given a graph in which pixels are nodes and edges weights measure the dissimilarity between nodes, each node is initially placed in its own component. Define the internal difference of a component $Int(C)$ as the largest weight in the minimum spanning tree of C . Considering edges in non-decreasing order by weight, each step of the algorithm merges components C_1 and C_2 connected by the current edge if the weight is less than:

$$\min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2)) \quad (2.8)$$

where $\tau(C) = k/|C|$. k is a scale parameter that can be used to set a preference for component size. The merging criteria in Eq. 2.8 allows efficient graph-based clustering to be sensitive to edges in areas of low variability, and less sensitive to them in areas of high variability. This is intuitively the property we would like to see in a clustering algorithm.



2.4.2 The Algorithm

Given a graph $G=(V, E)$ be fully undirected connected, with n vertices and m edges.

Each vertex is a pixel, x , represented in the feature space. The final segmentation of V

will be $S=(C_1, \dots, C_r)$ where C_i is a cluster. The algorithm is:

Step 1

Sort $E=(e_1, \dots, e_m)$ such that $|e_t| \leq |e_{t'}| \forall t < t'$.

Step 2

Let $S^0 = (\{x_1\}, \dots, \{x_n\})$, in other words each initial cluster contains exactly one vertex.

Step 3

For $t = 1, \dots, m$

(a) Let x_i and x_j be the vertices connected by e_t .

(b) Let $C_{x_i}^{t-1}$ be the connected component containing point x_i on iteration $t-1$,

and $l_i = \max_{mst} C_{x_i}^{t-1}$ be the longest edge in the minimum spanning tree of $C_{x_i}^{t-1}$.

Likewise for l_j .

(c) Merge $C_{x_i}^{t-1}$ and $C_{x_j}^{t-1}$ if

$$|e_t| < \min \left\{ l_i + \frac{k}{|C_{x_i}^{t-1}|}, l_j + \frac{k}{|C_{x_j}^{t-1}|} \right\} \quad (2.9)$$

where k is a constant.

Step 4

Return $S = S^m$.

There are two parameters used to adjust the result of segmentation, σ and k . In the author's implementation, the input image is first smoothed using a Gaussian filter to remove some artifact. The parameter σ is the standard difference of the Gaussian filter, usually set as 0.8. The parameter k is used to compute the threshold function τ . The function $\tau(C) = k/|C|$ where $|C|$ is the number of elements in C . Thus k effectively sets a scale of observation, in that a larger k causes a preference for larger components.

On the other hand, the author provides two approaches to define the initial graph $G = (V, E)$. One is the Grid Graphs approach, where each image pixel p_i has a corresponding vertex $v_i \in V$. The edge set E is constructed by connecting pairs of pixels that are neighbors in an 8-connected sense (any other local neighborhood could be used). This yields a graph where $m = 8 \cdot (n)$, so the running time of the segmentation algorithm is $O(n \log n)$ for n image pixels. The other is the Nearest Neighbor Graphs approach, where the edges are defined by connecting pairs of feature points that are nearby in the feature space, rather than using neighboring pixels in the image grid. Each point is connected to a fixed number of nearest neighbors or all the neighbors within some fixed distance d . In any event, it is desirable to avoid considering all $O(n^2)$ pairs of feature points.



2.4.3 Simulation Results

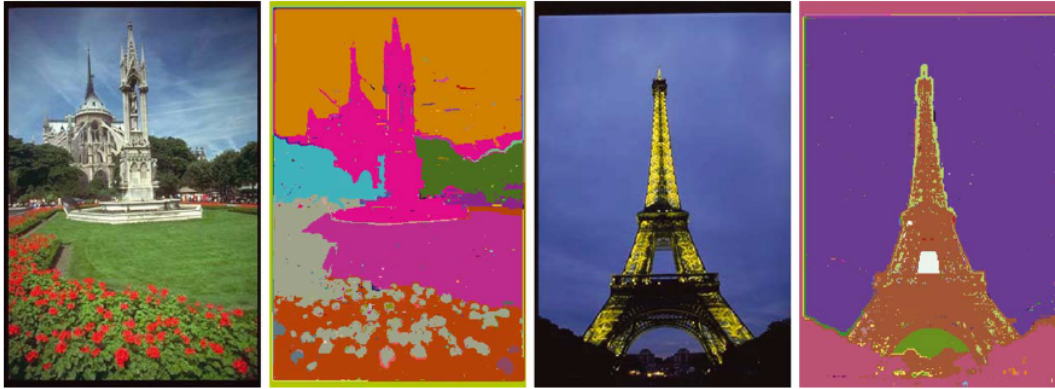


Fig. 2.5 Segmentation using the nearest neighbor graph can capture spatially non-local regions ($\sigma = 0.8$, $k = 300$).

Chapter 3 Reviews of Superpixels and Superpixel-based Segmentation Algorithms



We first introduce the concept and possible applications of superpixels in Section 3.1. Then, two kinds of superpixel methods which perform well nowadays are reviewed in Section 3.2 and Section 3.3. Finally, we introduce superpixel-based segmentation algorithms in Section 3.4.

3.1 Introduction of Superpixel

A superpixel is commonly defined as a perceptually uniform region in the image. The major advantage of using superpixels is computational efficiency—a superpixel representation greatly reduces the number of image primitives compared to the pixel representation. It has been proved useful for applications such as depth estimation, image segmentation, body model estimation and object localization.

Generally, superpixel contains the following properties:

- Every superpixel should overlap with only one object.
- The set of superpixel boundaries should be a superset of object boundaries.
- The mapping from pixels to superpixels should not reduce the achievable performance of the intended application.
- The above properties should be obtained with as few superpixels as possible.



3.2 Simple Linear Iterative Clustering Superpixels

3.2.1 About Simple Linear Iterative Clustering Superpixels

Computer vision applications have come to rely increasingly on superpixels in recent years. Achanta *et al.* believes that an ideal approach of generating superpixels should have the following properties:

1. Superpixels should adhere well to image boundaries.
2. When used to reduce computational complexity as a pre-processing step, superpixels should be fast to compute, memory efficient, and simple to use.
3. When used to for segmentation purposes, superpixels should both increase the speed and improve the quality of the results.

They propose a new method for generating superpixels, *Simple linear iterative clustering* (SLIC) [15], which is faster than existing methods, more memory efficient, exhibits state-of-the-art boundary adherence, and improve the performance of segmentation algorithms.

It is an adaptation of *k*-means for superpixels generation, with two important distinctions:

1. The number of distance calculations in the optimization is dramatically reduced by limiting the search space to a region proportional to superpixel size. This reduces the complexity to be linear in the number of pixels N and independent of the number of superpixels K .
2. A weighted distance measure combines color and spatial proximity while simultaneously providing control over the size and compactness of the superpixels.

SLIC is simple to use and understand. By default, the only parameter of the algorithm is K , the desired number of approximately equally sized superpixels. For a color image in the CIELAB color space, the clustering procedure begins with an

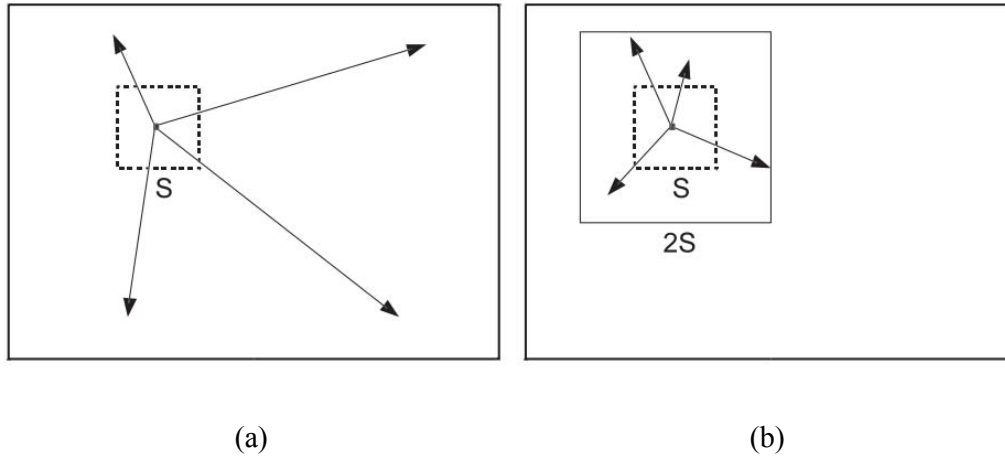


Fig. 3.1 Reducing the superpixel search regions. (a) standard k-means searches the entire image. (b) SLIC searches a limit region.

initialization step where K initial cluster centers $C_k = [l_k \ a_k \ b_k \ x_k \ y_k]^T$ with $k = [1, K]$ are sampled on a regular grid spaced S pixels apart. To produce roughly equally sized superpixels, the grid interval is $S = \sqrt{N/K}$, so the approximate size of each superpixel is therefore N/K pixels for an image with N pixels. The centers are moved to seed locations corresponding to the lowest gradient position in a 3×3 neighborhood. This is done to avoid centering a superpixel on an edge and to reduce the chance of seeding a superpixel with a noisy pixel.

Next, in the assignment step, each pixel i is associated with the nearest cluster center whose search region overlaps its location, as depicted in Fig. 3.1. This is the key to speeding up our algorithm because limiting the size of the search region significantly reduces the number of distance calculations, and results in a significant speed advantage over conventional k -means clustering where each pixel must be compared with all cluster centers. Since the expected spatial extent of a superpixel is a region of approximate size $S \times S$, the search for similar pixels is done in a region $2S \times 2S$ around the superpixel center.

There is a problem that how to define the distance measure D . While the maximum possible distance between two colors in the CIELAB space is limited, the spatial distance in the xy plane depends on the image size. It is not possible to simply use the Euclidean distance in this 5D space without normalizing the spatial distances. In order to cluster pixels in this 5D space, therefore a new distance measure that considers superpixel size is introduced. Using it enforce color similarity as well as pixel proximity in this 5D space such that the expected cluster sizes and their spatial extent are approximately equal. The measure is defined by combining the color proximity and spatial proximity normalized by their respective maximum distances within a cluster, N_c and N_s , as follows

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2} \quad (3.1)$$

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \quad (3.2)$$

$$D' = \sqrt{\left(\frac{d_c}{N_c}\right)^2 + \left(\frac{d_s}{N_s}\right)^2} = \sqrt{\left(\frac{d_c}{m}\right)^2 + \left(\frac{d_s}{S}\right)^2} \quad (3.3)$$

where d_c and d_s are distances in color and spatial space, respectively. The parameter m is a constant to represent the respective maximum color distance N_c , and the maximum spatial distance expected within a given cluster should correspond to the sampling interval, therefore $N_s = S$.

In practice, the distance measure D is defined as

$$D = \sqrt{d_c^2 + \left(\frac{d_s}{S}\right)^2} m^2. \quad (3.4)$$

By defining D in this manner, m can be adjusted to weigh the relative importance between color similarity and spatial proximity. When m is large, spatial proximity is more important and the resulting superpixels are more compact (i.e., they have a lower area to perimeter ratio). When m is small, the resulting superpixels adhere more tightly to image

boundaries, but have less regular size and shape. When using the CIELAB color space, m can be in the range [1,40].

Once each pixel has been associated to the nearest cluster center, an update step adjusts the cluster centers to be the mean $[l\ a\ b\ x\ y]^T$ vector of all the pixels belonging to the cluster. The L_2 norm is used to compute a residual error E between the new cluster center locations and previous cluster center locations. The assignment and update steps can be repeated iteratively until the error converges, but in most of time that 10 iterations suffices for most images, and report all results in this paper using this criteria.

Finally, a post-processing step enforces connectivity by reassigning disjoint pixels to nearby largest superpixels.

3.2.2 The Algorithm

Given an image with N pixels, define the number of superpixels K and compactness control parameter m , the algorithm is as follows:

Step 1

Initialize K cluster centers $C_k = [l_k\ a_k\ b_k\ x_k\ y_k]^T$ by sampling pixels at regular grid step S , $S = \sqrt{N/K}$.

Move cluster centers to the lowest gradient position in a 3×3 neighborhood.

Set label of pixel i , $l(i) = -1$ for each pixel.

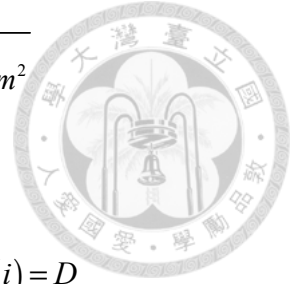
Set distance between nearest cluster center and pixel i , $d(i) = \infty$ for each pixel.

Step 2

For each cluster center C_k **do**

For each pixel i in a $2S \times 2S$ region around C_k **do**

Compute the distance D between C_k and i , $D = \sqrt{d_c^2 + \left(\frac{d_s}{S}\right)^2} m^2$



If $D < d(i)$ **then**

Set distance between nearest cluster center and pixel i , $d(i) = D$

Set label of pixel i , $l(i) = k$

End if

End for

End for

Step 3

Compute new cluster centers using the mean value of pixels in each cluster.

Compute residual error E , the L_2 distance between previous centers and recomputed centers.

If $E \leq \text{threshold}$ **then** return to Step 2.

Step 4

Enforce connectivity by reassigning disjoint pixels.

3.2.3 Simulation Results and Discussion

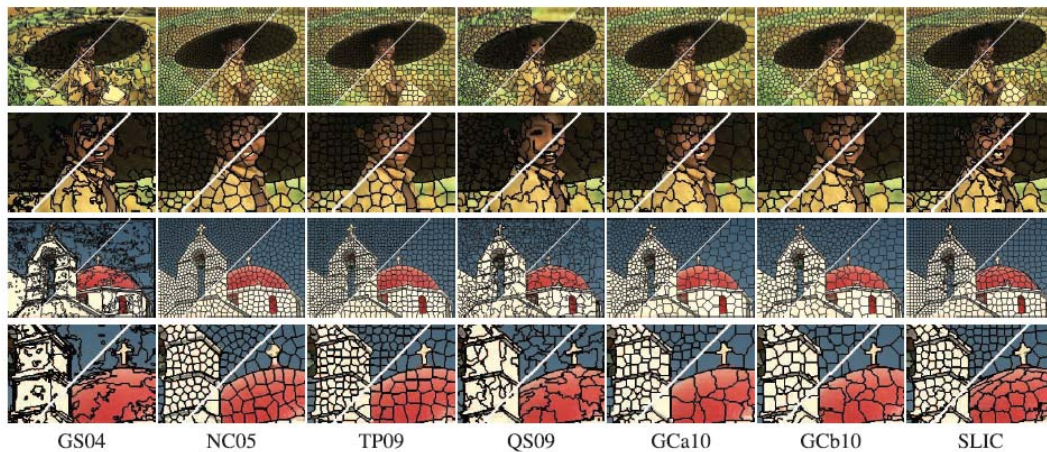


Fig. 3.2 The superpixel result of SLIC superpixels and other state-of-the-art superpixel methods.



3.3 Entropy Rate Superpixel Segmentation

3.3.1 About Entropy Rate Superpixel Segmentation

Liu *et al.* propose solving the superpixel segmentation problem by maximizing an objective function on the graph topology [16]. The objective function consists of an entropy rate and a balancing term for obtaining superpixels with similar sizes. They also present a greedy optimization scheme to reduce the complexity of the algorithm to $O(N \log N)$. Entropy Rate Superpixel (ERS) offers control over the amount of superpixels and also their compactness.

This method presents a new clustering objective function which consists of two terms:

- (1) the entropy rate of a random walk on a graph.
- (2) a balancing term on the cluster distribution.

The entropy rate favors compact and homogeneous clusters—encouraging division of images on perceptual boundaries and favoring superpixels overlapping with only a single object; whereas the balancing term encourages clusters with similar sizes—reducing the number of unbalanced superpixels.

The clustering formulation leads to an efficient algorithm with a provable bound on the optimality of the solution. The objective function is a monotonically increasing submodular function. Submodularity is the discrete analogue of convexity in continuous domains. Knowing whether a function is submodular is better to understand the underlying optimization problem. In general, maximization of submodular functions leads to NP-hard problems, for which the global optimum is difficult to obtain. Nevertheless, by using a greedy algorithm and exploiting the matroid structure present in the formulation, it can obtain a bound of $1/2$ on the optimality of the solution.

3.3.2 Preliminaries of this Method



(1) Graph representation:

$G = (V, E)$ denotes an undirected graph where V is the vertex set and E is the edge set.

The vertices and edges are denoted by v_i and $e_{i,j}$ respectively. The similarity between vertices is given by the weight function w . In an undirected graph, the edge weights are symmetric, that is $w_{i,j} = w_{j,i}$.

(2) Graph partition:

A graph partition S refers to a division of the vertex set V into disjoint subsets

$S = \{S_1, S_2, \dots, S_K\}$ such that $S_i \cap S_j = \emptyset$ for $i \neq j$.

(3) Entropy:

The uncertainty of a random variable is measured by entropy H . Entropy of a discrete random variable X with a probability mass function p_X is defined by

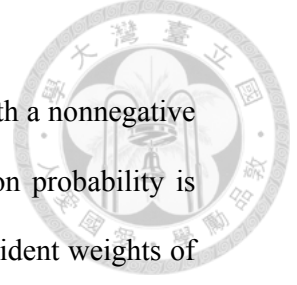
$$H(X) = -\sum_{x \in X} p_X(x) \log p_X(x) \quad (3.5)$$

(4) Entropy rate:

The entropy rate quantifies the uncertainty of a stochastic process $X = \{X_t \mid t \in T\}$ where T is some index set. For a discrete random process, the entropy rate is defined as an asymptotic measure:

$$H(X) = \lim_{t \rightarrow \infty} H(X_t \mid X_{t-1}, X_{t-2}, \dots, X_1), \quad (3.6)$$

which measures the remaining uncertainty of the random process after observing the past trajectory.



(5) Random walks on graphs:

Let $X = \{X_t | t \in T, X_t \in V\}$ be a random walk on the graph $G = (V, E)$ with a nonnegative similarity measure w . A random walk model described in the transition probability is defined as $p_{i,j} = \Pr(X_{t+1} = v_j | X_t = v_i) = w_{i,j}/w_i$, where w_i is the sum of incident weights of the vertex v_i , and the stationary distribution is given by

$$\mu = (\mu_1, \mu_2, \dots, \mu_{|V|})^T = \left(\frac{w_1}{w_T}, \frac{w_2}{w_T}, \dots, \frac{w_{|V|}}{w_T} \right)^T \quad (3.7)$$

where w_T is the normalization constant. For a disconnected graph, the stationary distribution is not unique. However, μ in (4) is always a stationary distribution.

3.3.3 Problem Formulation and Algorithm

This method considers clustering as a graph partitioning problem. To partition the image into K superpixels, the method searches for a graph topology that has K connected subgraphs and maximizes the proposed objective function.

(1) Graph Construction

The algorithm maps an image to a graph $G = (V, E)$ with vertices denoting the pixels and the edge weights denoting the pairwise similarities given in the form of a similarity matrix. The goal is to select a subset of edges $A \subseteq E$ such that the resulting graph, $G = (V, A)$, contains exactly K connected subgraphs.

(2) Entropy Rate

The algorithm uses the entropy rate of the random walk on the constructed graph as a criterion to obtain compact and homogeneous clusters.

Consequently, the entropy rate of the random walk on $G = (V, A)$ can be written as a set function :

$$H(A) = - \sum_i \mu_i \sum_j p_{i,j}(A) \log(p_{i,j}(A)) \quad (3.8)$$

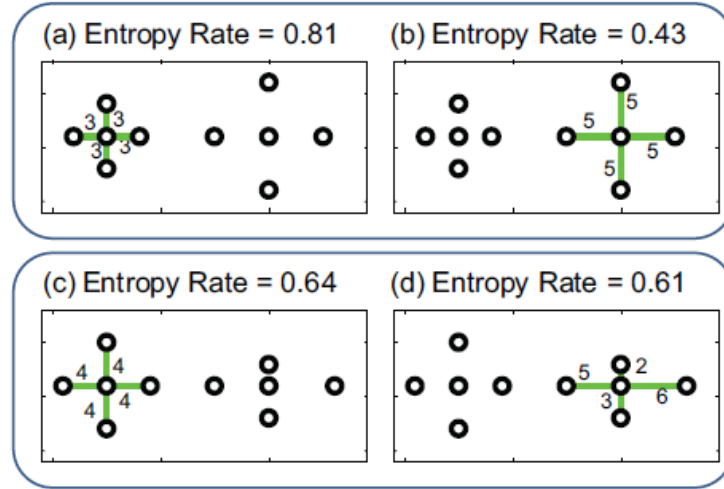


Fig. 3.3 The role of entropy rate in obtaining compact and homogeneous clustering.

(3) Balancing Function

The balancing function encourages clusters with similar sizes. Let A be the selected edge set, N_A is the number of connected components in the graph, and Z_A be the distribution of the cluster membership. For instance, let the graph partitioning for the edge set A be $S_A = \{S_1, S_2, \dots, S_{N_A}\}$. Then the distribution of Z_A is equal to

$$p_{Z_A}(i) = \frac{|S_i|}{|V|}, i = \{1, \dots, N_A\} \quad (3.9)$$

and the balancing term is given by

$$B(A) \equiv H(Z_A) - N_A = -\sum_i p_{Z_A}(i) \log(p_{Z_A}(i)) - N_A \quad (3.10)$$

The entropy $H(Z_A)$ favors clusters with similar sizes; whereas N_A favors fewer number of clusters.

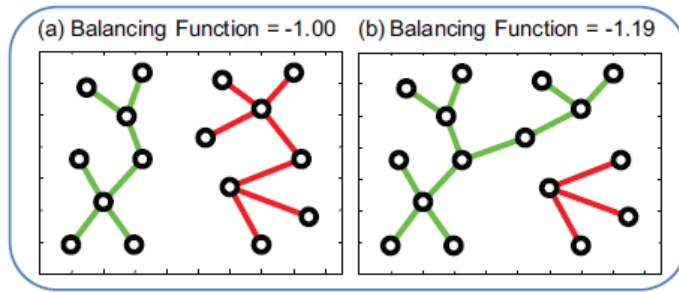


Fig. 3.4 The role of the balancing function is to obtain clusters of similar sizes.

(4) Objective Function

The objective function combines the entropy rate and the balancing function and therefore favors compact, homogeneous, and balanced clusters. The clustering is achieved via optimizing the objective function with respect to the edge set:

$$\begin{aligned} & \max_A && H(A) + \lambda B(A) \\ & \text{subject to} && A \subseteq E \text{ and } N_A \geq K, \end{aligned} \tag{3.11}$$

where $\lambda \geq 0$ is the weight of the balancing term. Linear combination with nonnegative coefficients preserves submodularity and monotonicity, therefore the objective function is also submodular and monotonically increasing. The additional constraint on the number of connected subgraphs enforces exactly K clusters since the objective function is monotonically increasing.

3.3.4 Conclusion and Simulation Results

This paper formulated the superpixel segmentation problem as an optimization problem on graph topology. They proposed a novel objective function based on the entropy rate of a random walk on the graph and derived an efficient algorithm with a bound on the optimality of the solution.

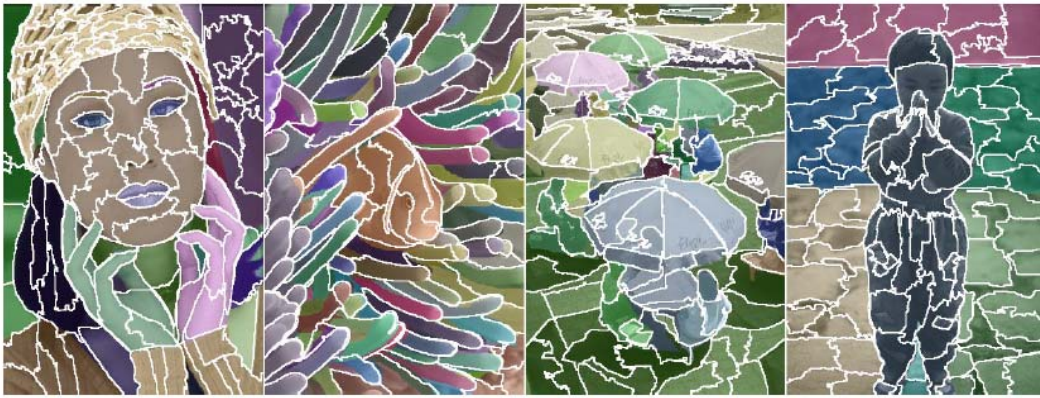
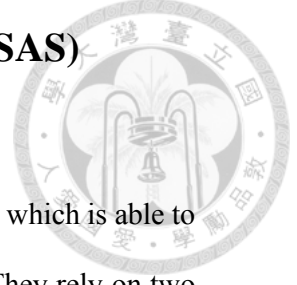


Fig. 3.5 Superpixel segmentation examples. The images contain 100 superpixels. The ground truth segments are color-coded and blended on the images. The superpixels (boundaries shown in white) respect object boundaries and tend to divide an image into similar-sized regions.

3.4 Segmentation by Aggregating Superpixels (SAS)



3.4.1 About Segmentation by Aggregating Superpixels

Li *et al.* propose a novel graph-based segmentation framework [17] which is able to efficiently integrate cues from multi-layer superpixels simultaneously. They rely on two simple observations:

1. Pixels within one superpixels tend to belong to one coherent region. (superpixel cues).
2. Neighboring pixels which are close in feature space tend to belong to one coherent region. (smoothness cues).

They show that both cues can be effectively encoded using one bipartite graph, and further develop an efficient algorithm for unbalanced bipartite graph partitioning.

A bipartite graph is constructed over pixels and superpixels collectively, as shown in Fig. 3.6. To enforce superpixel cues, a pixel is connected to a superpixel if the pixel is included in that superpixel. To enforce smoothness cues, it can be done simply connecting neighboring pixels weighted by similarity, but this would end up with redundancy because the smoothness regarding neighboring pixels within superpixels were incorporated when enforcing superpixel cues. It may also incur complex graph partitioning due to denser connections on the graph. To compensate for the smoothness on the neighboring pixels across superpixels, the neighboring superpixels which are close in feature space are connected.

Formally, denote S a set of (multi-layer) superpixels over an image I , and let $G = \{X, Y, B\}$ be a bipartite graph with node set $X \cup Y$, where $X := I \cup S = \{x_i\}_{i=1}^{N_x}$ and $Y := S = \{y_j\}_{j=1}^{N_y}$ with $N_x = |I| + |S|$ and $N_y = |S|$, the number of nodes in X and

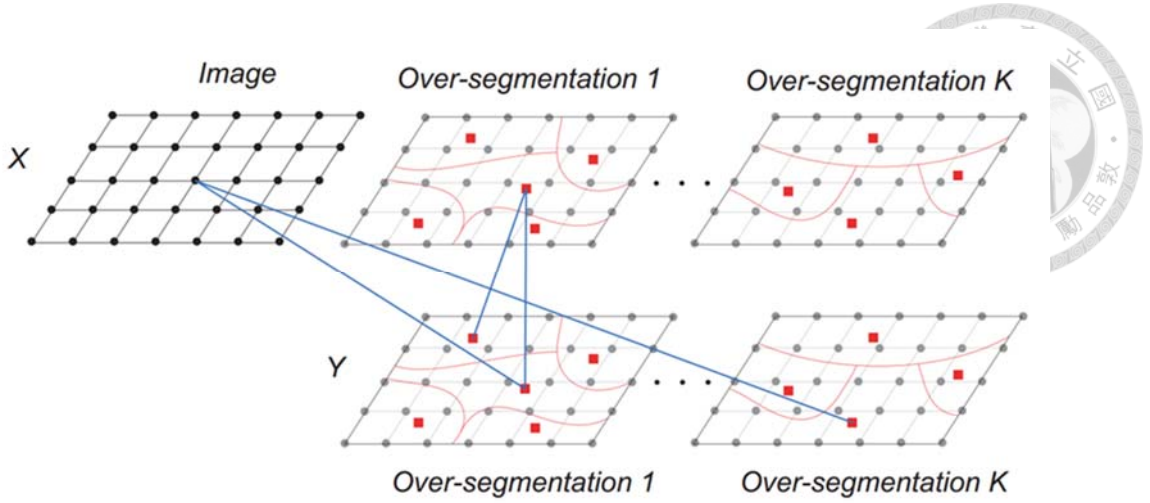


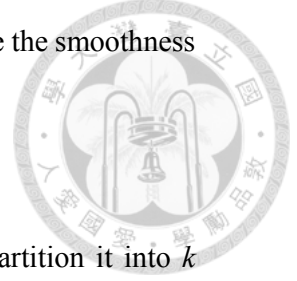
Fig. 3.6 The proposed bipartite graph model with K over-segmentations of an image. A black dot denotes a pixel while a red square denotes a superpixel. For the graph, each pixel is connected to the superpixels containing it and each superpixel is connected to itself and its nearest neighbor in the feature space among its spatially adjacent superpixels. Each superpixel is represented by the average LAB color of the pixels within it.

Y , respectively. The across-affinity matrix $B = (b_{ij})_{N_x \times N_y}$ between X and Y is constructed as follows:

$$\begin{aligned}
 b_{ij} &= \alpha, & \text{if } x_i \in y_j, x_i \in I, y_j \in S; \\
 b_{ij} &= e^{-\beta d_{ij}}, & \text{if } x_i \sim y_j, x_i \in S, y_j \in S; \\
 b_{ij} &= 0, & \text{otherwise,}
 \end{aligned} \tag{3.12}$$

where d_{ij} denotes the distance between the features of superpixels x_i and y_j , \sim denotes a certain neighborhood between superpixels, $\alpha > 0$ and $\beta > 0$ are free parameters controlling the balance between the superpixel cues and the smoothness cues, respectively. By this construction, a pixel and the superpixels containing it are likely to be grouped together due to the connections between them. Two neighboring (defined by \sim) superpixels are more likely to be grouped together if they are closer in feature space.

Particularly, superpixels are included in both parts of the graph to enforce the smoothness over superpixels.



Given the above bipartite graph $G = \{X, Y, B\}$, the task is to partition it into k groups, where k is manually specified. Each group includes a subset of pixels and/or superpixels, and the pixels from one group form a segment. The spectral clustering is used since it has been successfully applied to a variety of applications including image segmentation. Spectral clustering captures essential cluster structure of a graph using the spectrum of graph Laplacian. Mathematically, it solves the generalized eigen-problem:

$$Lf = \gamma Df \quad (3.13)$$

where $L = D - W$ is the graph Laplacian and $D = \text{diag}(W\mathbf{1})$ is the degree matrix with W denoting the affinity matrix of the graph and $\mathbf{1}$ a vector of ones of appropriate size. For a k -way partition, the bottom k eigenvectors are computed. Clustering is performed by applying k -means [20] or certain discretization technique [21].

To solve (3.13), one can simply treat the bipartite graph constructed above as a general sparse graph and apply the Lanczos method [29] to $\bar{W} = D^{-1/2}WD^{-1/2}$, the normalized affinity matrix, which takes $O(k(N_x + N_y)^{3/2})$ running time empirically.

Alternatively, it was shown, in the literature of document clustering, that the bottom k eigenvectors of (3.13) can be derived from the top k left and right singular vectors of the normalized across-affinity matrix $\bar{B} = D_x^{-1/2}BD_y^{-1/2}$, where $D_x = \text{diag}(B\mathbf{1})$ and $D_y = \text{diag}(B^T\mathbf{1})$ are the degree matrices of X and Y , respectively. This partial SVD, done typically by an iterative process alternating between matrix-vector multiplications $\bar{B}v$ and $\bar{B}^T u$, is essentially equivalent to the Lanczos method on W . It does not bring

substantial benefit on solving (3.13) and is still subject to a complexity of $O(k(N_X + N_Y)^{3/2})$.

However, the above methods do not exploit the structure that the bipartite graph can be unbalanced, i.e., $N_X \neq N_Y$. In this case, $N_X = N_Y + |I|$. This unbalance can be translated into the efficiency of partial SVDs without losing accuracy. One way is by using the dual property of SVD that a left singular vector can be derived from its right counterpart, and vice versa.

An essential equivalence between the eigen-problem (3.13) on the original bipartite graph and the one on a much smaller graph over superpixels is revealed as follows.

$$L_Y \mathbf{v} = \lambda D_Y \mathbf{v}, \quad (3.14)$$

where $L_Y = D_Y - W_Y$, $D_Y = \text{diag}(B^T \mathbf{1})$, and $W_Y = B^T D_X^{-1} B$. L_Y is exactly the Laplacian of the graph $G_Y = \{Y, W_Y\}$ because $D_Y = \text{diag}(B^T \mathbf{1}) = \text{diag}(W_Y \mathbf{1})$. It should be noted that this analysis can be applied to spectral clustering on any bipartite graph.

The result states that the bottom k eigenvectors of (3.13) can be obtained from the bottom k eigenvectors of (3.14), which can be computed efficiently given the much smaller scale of (3.14). Formally, we have the following Theorem 1.

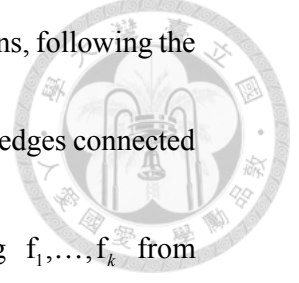
Theorem 1. Let $\{(\lambda_i, \mathbf{v}_i)\}_{i=1}^k$ be the bottom k eigenpairs of the eigen-problem (3.14) on

the superpixel graph $G_Y = \{Y, B^T D_X^{-1} B\}$, $0 = \lambda_1 \leq \dots \leq \lambda_k < 1$. Then $\{(\gamma_i, \mathbf{f}_i)\}_{i=1}^k$ are the

bottom k eigenpairs of the eigen-problem (1.13) on the bipartite graph $G = \{X, Y, B\}$,

where $0 \leq \gamma_i < 1$, $\gamma_i(2 - \gamma_i) = \lambda_i$, and $\mathbf{f}_i = (\mathbf{u}_i^T, \mathbf{v}_i^T)^T$ with $\mathbf{u}_i = \frac{1}{1 - \gamma_i} P \mathbf{v}_i$. Here

$P = D_X^{-1} B$ is the transition probability matrix from X to Y .



By Theorem 1, computing f_i from v_i needs $2N_X d_X + 2N_X$ operations, following the execution order $\frac{1}{1-\gamma_i} (D_X^{-1} (Bv_i))$, where d_X is the average number of edges connected to each node in X . So it takes $2k(1+d_X)N_X$ operations for computing f_1, \dots, f_k from v_1, \dots, v_k , plus a cost of $O(kN_Y^{3/2})$ for deriving v_1, \dots, v_k with the Lanczos method. The complexity is a linear time (w.r.t. N_X) with a small constant.

3.4.2 The Algorithm

The approach to bipartite graph partitioning is summarized in Algorithm 1, which is called as Transfer Cuts (Tcut) since it transfers the eigen-operation from the original graph to a smaller one. In step 5, one may employ the discretization technique which is tailored to Ncut, or apply k -means to the rows of the matrix $F = (f_1, \dots, f_k)$ after each row is being normalized to unit length, which is justified by stability analysis.

Algorithm 1 Transfer cuts (Tcut)

Input: A bipartite graph $G = \{X, Y, B\}$ and a number k .

Output: A k -way partition of G .

Step 1

Form $D_X = \text{diag}(B\mathbf{1})$, $D_Y = \text{diag}(B^T\mathbf{1})$, $W_Y = B^T D_X^{-1} B$, and $L_Y = D_Y - W_Y$.

Step 2

Compute the bottom k eigenpairs $\{(\lambda_i, v_i)\}_{i=1}^k$ of $L_Y v = \lambda D_Y v$.

Step 3

Obtain γ_i such that $0 \leq \gamma_i < 1$ and $\gamma_i(2 - \gamma_i) = \lambda_i$, $i = 1, \dots, k$.



Step 4

Compute $f_i = (u_i^T, v_i^T)^T$, with $u_i = \frac{1}{1-\gamma_i} D_X^{-1} B v_i$, $i = 1, \dots, k$.

Step 5

Derive k groups of $X \cup Y$ from f_1, \dots, f_k .

The segmentation procedures are listed in Algorithm 2, which is called as **Segmentation by Aggregating Superpixels (SAS)**. The main cost of SAS is in collecting superpixels (step 1) and bipartite graph partitioning (step 3). The cost of step 3 is linear in the number of pixels in the image with a small constant, and is negligible compared to that of step 1.

Algorithm 2 Segmentation by Aggregating Superpixels (SAS)

Input: An image I and a number of segments k .

Output: A k -way partition of I .

Step 1

Collect a bag of superpixels S for I .

Step 2

Construct a bipartite graph $G = \{X, Y, B\}$ with $X = I \cup S$, $Y = S$

Step 3

Apply Tcut in Algorithm 1 to derive k group of G .

Step 4

Treat pixels from the same group as a segment.

3.4.3 Simulation Results and Discussion

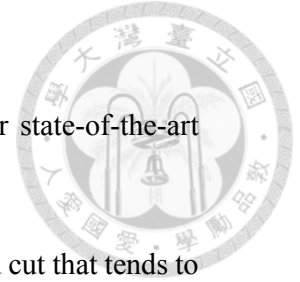


Fig. 3.7 shows the segmentation result of SAS method and other state-of-the-art superpixel methods.

It can be seen that the algorithm still has the property of normalized cut that tends to partition the image into regions with regular sizes. The performance of this method is better than most of the existing segmentation algorithms, it is the main target for comparison in this thesis.

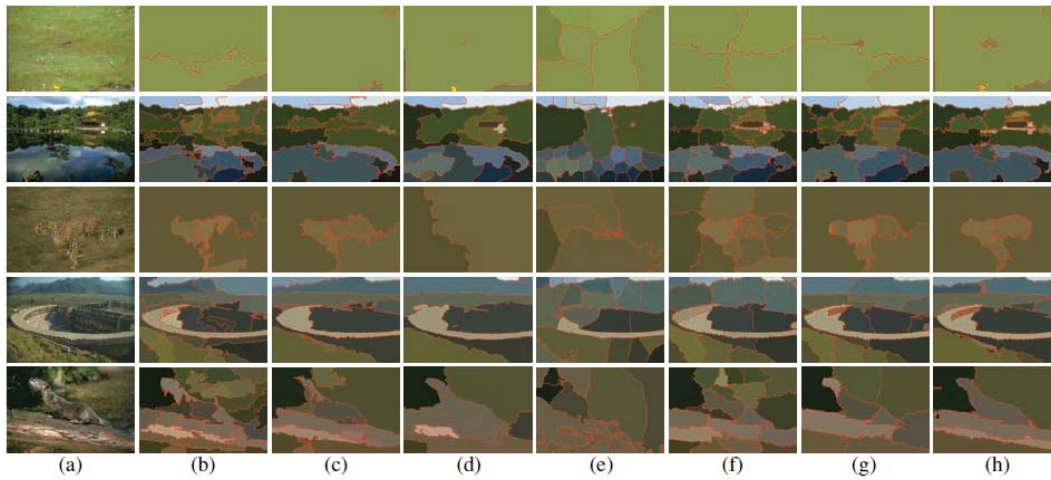


Fig. 3.7 Segmentation examples on Berkeley Segmentation Database.

(a) Input images. (b) Mean Shift. (c) FH. (d) TBES. (e) Ncut. (f) MNcut. (g) MLSS.
(h) SAS.

3.5 Learning Full Pairwise Affinities for Spectral Segmentation (MLSS)



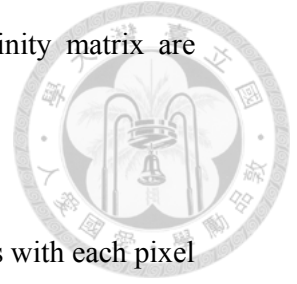
3.5.1 About MLSS (Multi-Layer Spectral Segmentation)

In recent years, spectral segmentation has become a major trend in image segmentation. It typically starts from local information encoded in a graph-based representation of a given image, and partitions that image according to the global information embedded in the spectrum of the graph affinity matrix. Kim *et al.* design and use a full range affinity model in the spectral segmentation framework [18] so that they can obtain high-quality segmentation results efficiently by using the proposed affinity model.

3.5.2 The Algorithm

They introduce a new affinity model for spectral segmentation and then use the relevance scores between all pairs of pixels, estimated by semisupervised learning (SSL) [22], as affinities is proposed.

1. A multi-layer graph whose nodes consist of the over-segmented regions by MShift as well as the pixels is first constructed. Then, the affinities between each node and other nodes are estimated by applying the SSL strategy to this graph through assuming the current node as labeled data and the others as unlabeled data.
2. Spectral analysis of our full affinity matrix is done efficiently. In general, performing spectral analysis of a full large matrix requires a prohibitively expensive computation. However, since our full affinity matrix is expressed as the inverse of a sparse matrix, its eigen-decomposition is very efficient using the basics of matrix computation.



3. Two types of spectral segmentation algorithms based on our affinity matrix are constructed:

(1) K-way segmentation [1], [7], [19]:

They borrow the basic idea of Normalized Cuts [1], which associates with each pixel a length K descriptor formed from entries of the K eigenvectors and uses a clustering algorithm such as K-means to create a hard partition of the image.

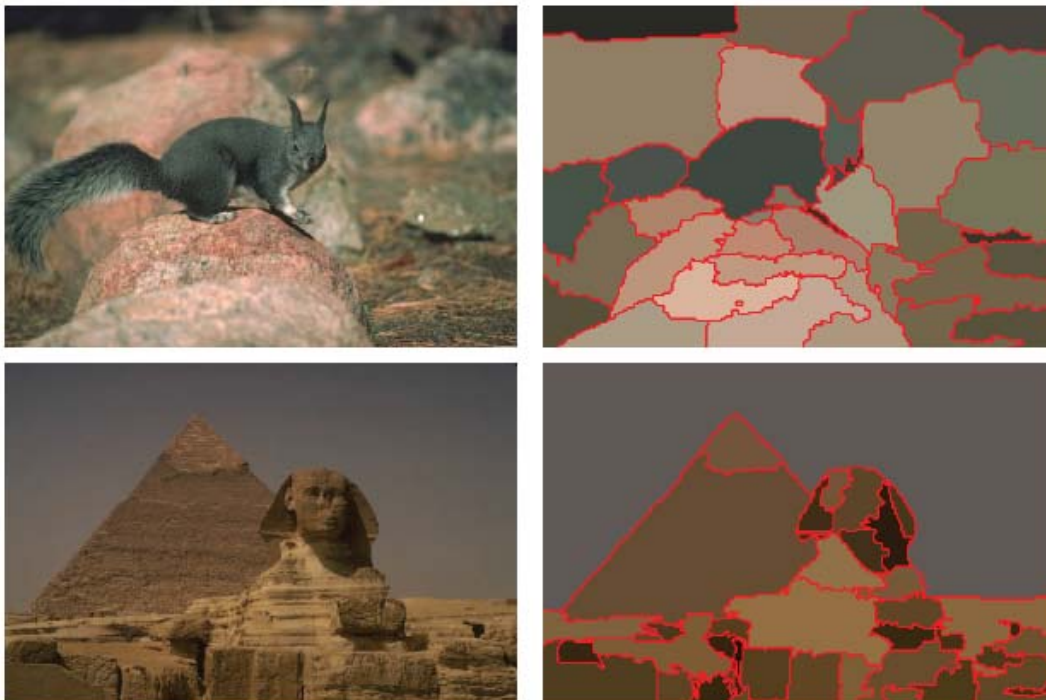
The algorithm FNCut clusters all pixels and regions simultaneously into the K visually coherent groups in a single multilayer framework of Normalized Cuts.

(2) Hierarchical segmentation [8]:

They transform the contour signals, produced by combining contour information in different eigenvectors, into a hierarchy regions.

The algorithm fPb-OWTUCM produces a hierarchy of regions from the contour signals in the eigenvectors using a sequence of two transformations: the Oriented Watershed Transform (OWT) [10] and the Ultrametric Contour Map (UCM) [23].

3.5.3 Simulation Results





(a)

(b)

Fig. 3.8 Segmentation examples using MLSS.

(a) Input images. (b) MLSS.

Chapter 4 Proposed Algorithm



We have discussed the recent trend of image segmentation and techniques of superpixels in the chapter 2 and chapter 3. In this chapter, the proposed segmentation algorithm will be presented.

This chapter is organized as follows. In Section 4.1, the framework of the proposed natural image segmentation is introduced. In Section 4.2, we describe the generation of superpixels and review the saliency detection algorithm used in our method. In Section 4.3, we propose modified edge detection and show how to get texture features from an image. In Section 4.4, the proposed segmentation algorithm which is consisted of two main steps, clustering superpixels according to their texture features and saliency-oriented region merging, will be described in details. In Section 4.5, the new techniques of our proposed algorithm will be compared with the state-of-the-art segmentation methods.

4.1 Introduction

In superpixel-based segmentation algorithms, the main problem we want to prevent is over-merge, which cluster the superpixels improperly. For example, parts of foreground and background are clustered into the same region. In order to deal with this problem, we propose modified edge detection which computes the gradient information of the contours and the interiors of superpixels. By using the modified edge detection, our experimental results has higher accuracy compared with other segmentation methods. The flow chart is illustrated in Fig. 4.1 and the overview of our framework is shown in Fig. 4.2.

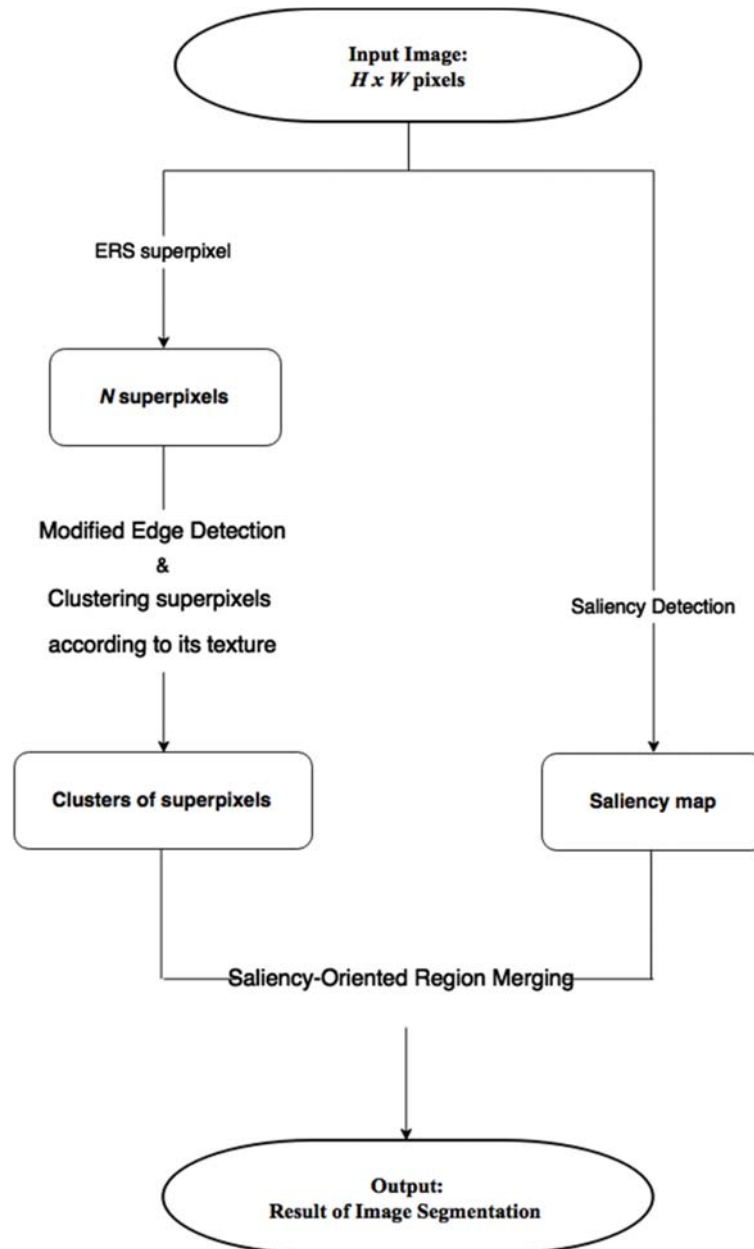


Fig. 4.1 The overall flow chart of our framework.

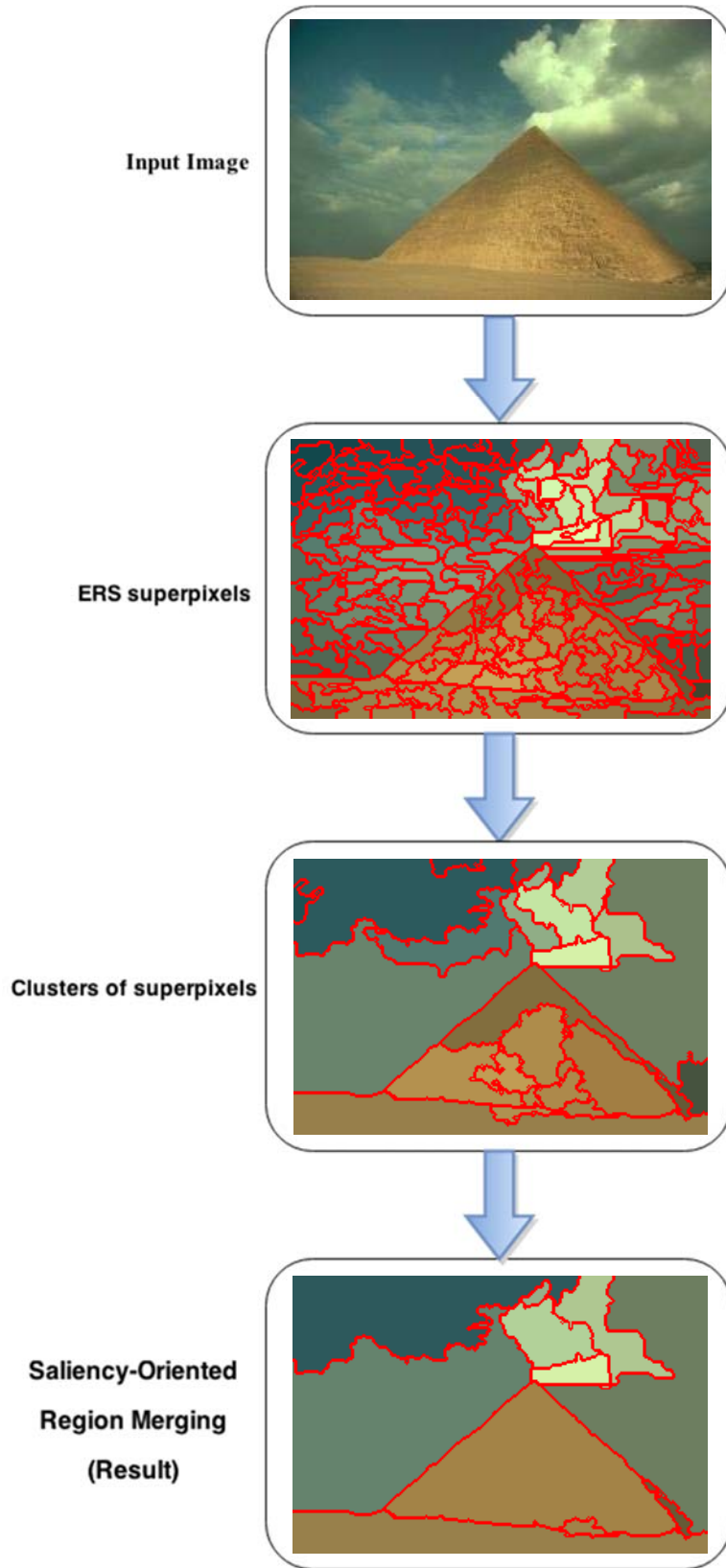


Fig. 4.2 The overview of our framework.

4.2 Superpixel Generation and Saliency Detection



4.2.1 Superpixel Generation

In our case, we look for superpixel methods having the following properties:

- Produce superpixels which have similar sizes:

If superpixels have similar sizes, they would likely to be in more regular arrangement. For instance, every superpixel tends to have similar numbers of neighbors, which is close to the original arrangement of pixels.

- Short processing time:

Although we do not have to choose the fastest superpixel method, we still hope to restrict the processing time of generating superpixels under several seconds.

In our algorithm, we use the technique of our proposed modified edge detection, so the generated superpixels should adhere well to image boundaries. We found that the Entropy Rate Superpixels (ERS) [16] meet our requirements and this approach is sufficiently efficient as a pre-processing step.

ERS is simple to use and understand. By default, the only parameter of the algorithm is N , the desired number of approximately equally sized superpixels. Here we set the number of superpixels N to $N=200$, because it could balance the complexity of the proposed algorithm and the accuracy of the segmentation results.

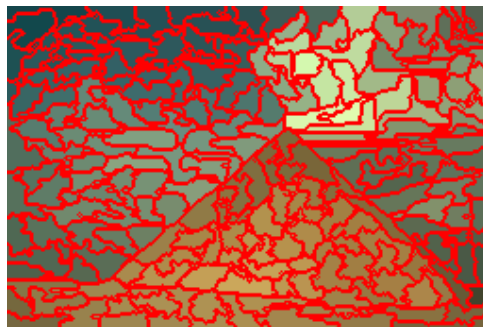


Fig. 4.3 An example of the processed result generated by ERS superpixel method.

4.2.2 Saliency Detection

Saliency regions are those parts of an image to which a human pays more attention, and the task of saliency detection is to identify the most important and informative part of a scene. Salient object detection algorithms usually generate bounding boxes, binary foreground and background segmentation, or saliency maps.

Since the saliency map can indicate the saliency likelihood of each pixel, in our segmentation framework, we use saliency-oriented region merging algorithm according to saliency maps of images. We apply the saliency detection method [24] (Saliency Detection via Graph-Based Manifold Ranking) proposed by Yang *et al*, and the following is brief introduction of this method.

There are two types of computational models for saliency detection: top-down saliency models that use high-level features based on knowledge from the neurosciences, biology, computer vision, and other fields, the selection of which depends on the specific task; and bottom-up saliency models that use low-level stimulus such as intensity, color contrast, orientation, and motion.

Most existing bottom-up methods measure the foreground saliency of a pixel or region based on its contrast within a local context or the entire image, whereas a few methods focus on segmenting out background regions and thereby salient objects. Instead of considering the contrast between the salient objects and their surrounding regions, Yang *et al* consider both foreground and background cues in a different way. They rank the similarity of the image elements (pixels or regions) with foreground cues or background cues via graph-based manifold ranking. The saliency of the image elements is defined based on their relevances to the given seeds or queries. They represent the image as a close-loop graph with superpixels as nodes. These nodes are ranked based on the similarity to background and foreground queries, based on affinity matrices. Saliency

detection is carried out in a two-stage scheme to extract background regions and foreground salient objects efficiently. In short, they model saliency detection as a manifold ranking problem and propose a two-stage scheme for graph labelling.

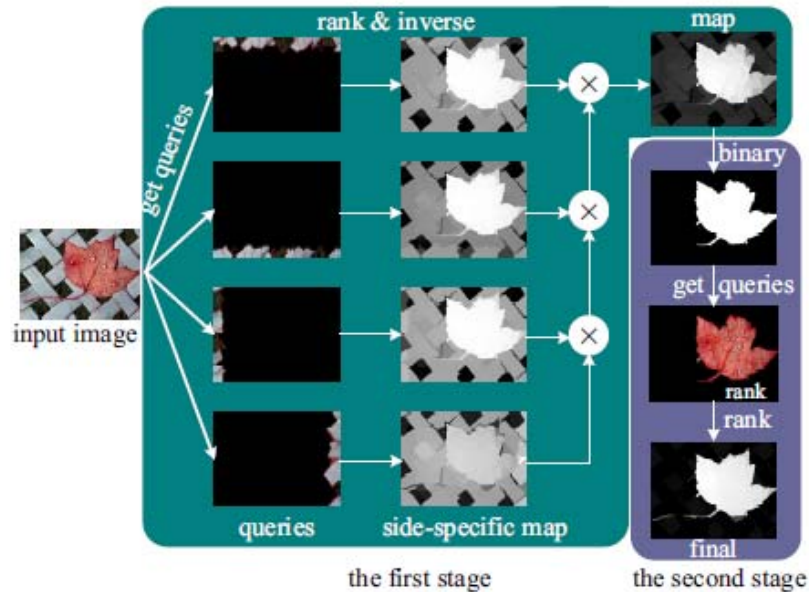


Fig. 4.4 Diagram of the saliency detection model.

Figure 4.4 shows the main steps of the saliency detection algorithm. In the first stage, Yang *et al* exploit the boundary prior [25] by using the nodes on each side of image as labelled background queries. From each labelled result, they compute the saliency of nodes based on their relevances (i.e, ranking) to those queries as background labels. The four labelled maps are then integrated to generate a saliency map. In the second stage, they apply binary segmentation on the resulted saliency map from the first stage, and take the labelled foreground nodes as salient queries. The saliency of each node is computed based on its relevance to foreground queries for the final map.

4.3 Modified Edge Detection and Texture Features

4.3.1 Modified Edge Detection

Edge detection is a very important field [26] in image processing and image segmentation. Edges in digital images are areas with strong intensity contrasts and a jump in intensity from one pixel to the next can create major variation in the picture quality. For those reasons edges form the outline of an object and also indicate the boundary between overlapping objects. As discontinuities in intensity values of an image form the edges of objects, so it is essential to detect accurate discontinuities in intensity levels for accurate edge detection. Such discontinuities are detected by using first- and second-order derivatives.

With the help of first-order and second-order derivatives such discontinuities are detected. The first-order derivative of choice in image processing is the gradient. An image gradient is a directional change in the intensity or color in an image. Image gradients may be used to extract information from images. The gradient of a 2-D function, $f(x,y)$, is defined as the vector

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (4.1)$$

The magnitude of this vector is

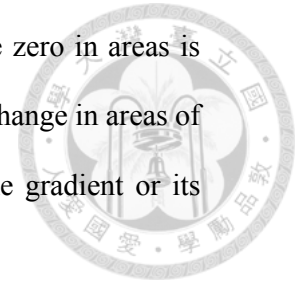
$$\nabla f = \text{mag}(\nabla f) = [g_x^2 + g_y^2]^{\frac{1}{2}} = \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{\frac{1}{2}} \quad (4.2)$$

This quantity is approximated sometimes by omitting the square root operation,

$$\nabla f \approx g_x^2 + g_y^2 \quad (4.3)$$

Or by using absolute values,

$$\nabla f \approx |g_x^2| + |g_y^2| \quad (4.4)$$



These approximations still behave as derivatives; that is, they are zero in areas of constant intensity and their values are related to the degree of intensity change in areas of variable intensity. It is common practice to refer the magnitude of the gradient or its approximations simply as “gradients”.

Several edge detector operators [27] are there for generating gradient images like sobel, prewitt, laplacian and Laplacian of Gaussian (LOG). These edge detectors work better under different conditions. In our modified edge detection algorithm, we adopt sobel and Laplacian of Gaussian (LOG) operators to construct our gradient maps.

(1) Sobel edge detector

The sobel edge detector computes the gradient by using the discrete differences between rows and columns of a 3X3 neighborhood. The sobel operator is based on convolving the image with a small, separable, and integer valued filter. In below a sobel edge detection mask is given which is used to compute the gradient in the x (vertical) and y (horizontal) directions.

-1	-2	-1	-1	0	1
0	0	0	-2	0	2
1	2	1	-1	0	1

G_x G_y

Fig. 4.5 A sobel edge detection mask

(2) Laplacian of Gaussian edge detector

The Laplacian of Gaussian edge detector finds edges by looking for zero crossings after filtering $f(x,y)$ with a Laplacian of Gaussian filter. In this method, the Gaussian filtering is combined with Laplacian to break down the image where the intensity varies to detect the edges effectively. It finds the correct place of edges and testing wider area around the pixel. In below a 5x5 standard Laplacian of Gaussian edge detection mask is given.



0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

Fig. 4.6 A 5x5 standard Laplacian of Gaussian edge detection mask

With the gradient map of an image, each pixel in an image has its gradient value.

Here we propose the modified edge detection:

(1) Define the complexity level CL of a superpixel:

$$CL = \frac{\text{Number of pixels with high gradient value in a superpixel}}{\text{Number of pixels in a superpixel}} \quad (4.5)$$

where “a pixel with high gradient value in a superpixel” means that the gradient value of this pixel is higher than the mean gradient value of the superpixel where this pixel locates.

Note: $CL(i)$ is the complexity level of the superpixel i .

(2) Define the edge E and the edge rate ER :

For the superpixel i , the superpixel j , and the border $B(i,j)$ between i and j ,

$$\text{edge } E(i, j) = \frac{\text{Number of pixels with high gradient value in } B(i, j)}{\text{Number of pixels in } B(i, j)} \quad (4.6)$$

$$\text{edge rate } ER(i, j) = \frac{E(i, j)}{CL(i) + CL(j)} \quad (4.7)$$

where “a pixel with high gradient value in the border” means that the gradient value of this pixel is higher than the mean gradient value of the border where this pixel locates.

We consider edge rate ER as our modified edge detection between adjacent superpixels, and it will be used in our proposed clustering algorithm. (Section 4.4)

4.3.2 Texture Features

We choose Log-Gabor Filter [28] to extract the texture feature of each superpixel. The Log-Gabor function is an alternative to the Gabor function proposed by Field. Gabor function represents a minimum in terms of the spread of uncertainty in space and spatial frequency. However, its mathematical property is only pure in Cartesian coordinates where the channels are the same size respectively in frequency and space. The relative spread and overlap of neighboring units would be altered if changed to polar distribution. Log-Gabor function can restore some destructive effects of such polar mapping with frequency response

$$G(f) = \exp \left(- \frac{\left(\log \left(\frac{f}{f_0} \right) \right)^2}{2 \left(\log \left(\frac{\sigma}{f_0} \right) \right)^2} \right) \quad (4.8)$$

, which is a Gaussian on log frequency axis.

Thus, a very important aspect of Log-Gabor function is that its frequency response is symmetric on a log axis, which is the standard method for representing the spatial-frequency response of visual neurons. Though not the best-fitting function, Log-Gabor function is a better model for suiting the visual system than Gabor function.

Another advantage of Log-Gabor is that its bandwidth increases with frequency, meaning that the bandwidths are constant in octaves. As displayed in Fig. 4.7, Log-Gabor function spreads the information equally in each channel while Gabor function overrepresents the low frequencies.

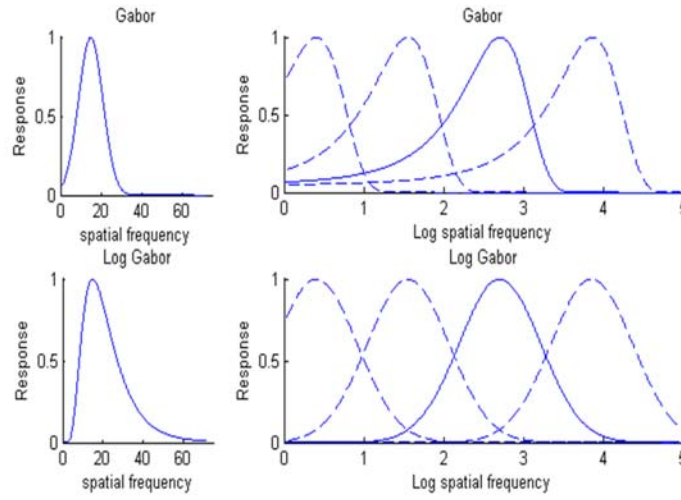


Fig. 4.7 Comparison of Gabor and Log-Gabor on both linear and logarithmic spatial frequency scales.

For the consideration in the algorithm, we choose Log-Gabor functions with scale = 2 and orientation = 4 as our texture feature (eight dimensions in total). For each superpixel i , we compute its mean texture $T(i)$. Here we define the texture distance $dT(i, j)$ from the superpixel i to the superpixel j as

$$dT(i, j) = |T(i) - T(j)| \quad (4.9)$$

We consider the texture distance dT as our indicative similarity between adjacent superpixels, and it will be used in our proposed clustering algorithm. (Section 4.4)



4.4 Proposed Segmentation Algorithm

In this section, the main steps of our proposed segmentation algorithm are described in **Stage 1** and **Stage 2**.

Stage 1

Clustering superpixels according to their texture features

Input: Image **I**.

Output: labels **L1**.

Step 1:

Segment **I** with selected superpixel method and store the result in **S**. Here we adopt ERS superpixel method to generate superpixels. We set the number of superpixels N to $N=200$.

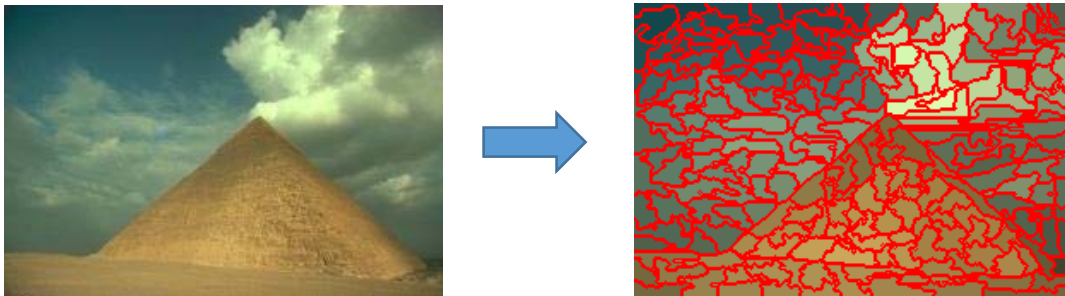


Fig. 4.8 Segment image with ERS superpixel method.

Step 2:

Store color values of **I** in CIELAB color space instead of RGB color space, each pixel has a color vector lab (3-dimensional: the L space, the A space and the B space).

Step 3:

Classify **I** into normal natural image or medical image according to the variance of the A space and the B space in **I**.

Step 4:

Apply the Log-Gabor filter to **I** and store the response, which can measure texture distance dT between any two superpixels which are adjacent.



Step 5:

Apply Sobel filter and LOG filter to **I** and store the responses, which are our gradient maps **G1**(Sobel) and **G2**(LOG).

Step 6:

(1)

Use **G1** to compute the complexity level **CL** of each superpixel, and then use **G1** in normal case or **G2** in shadow-detected case to measure the edge **E** between any two superpixels which are adjacent.

(2)

For any two superpixels(the superpixel **i** and the superpixel **j**) which are adjacent and the border **B(i,j)** between **i** and **j**, we use our proposed modified edge detection in Eq. (4.7) to measure edge rate between **i** and **j**

$$ER(i, j) = \frac{E(i, j)}{CL(i) + CL(j)}$$

where **E(i,j)** denotes the edge **E** between **i** and **j**; **CL(i)** denotes the complexity level **CL** of the superpixel **i**.

(We have defined the complexity level **CL** and the edge **E** in Eq. (4.5) and Eq. (4.6))

Step 7:

Compute the number of the segments **N1** in **I**. (Before clustering)

for each superpixel **i** in **S** **do**

for the superpixel **j** which is an adjacent superpixel of **i** **do**

if $ER(i, j) \leq 50\%$ **then**

 measure the texture distance $dT(i, j)$ in Eq. (4.9) between **i** and **j**

$$dT(i, j) = \|T(i) - T(j)\|$$

 where **T(i)** denotes the mean texture of **i**.

end if

end for

 Cluster **i** and one of its adjacent superpixels, which has the minimum texture distance to **i**.

end for

Compute the number of the segments **N2** in **I**. (After clustering)

If $N1 - N2 > 0$ **then** return to Step 7.

Step 8:

Each cluster of superpixels from step 6 will be labeled, and outputs labels **L1**.

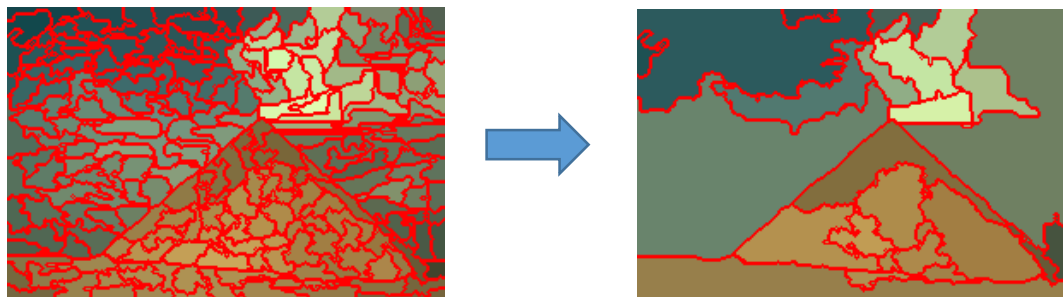


Fig. 4.9 The segmentation result from **Stage 1**.



Stage 2

Saliency-oriented region merging

Input: Image **I**, labels **L1**.

Output: labels **L2**.

Step 1:

Use **I** to construct the saliency map **SM**.

(Saliency Detection via Graph-Based Manifold Ranking in Section 4.2.2)



Fig. 4.10 The saliency map of an image.

Step 2:

According to the saliency map, compute the mean saliency value in each region which has the same label, and that mean value will represent the saliency value of the region.

Step 3:

Cluster regions with high saliency values by using the “revised edge rate”.

(“High” saliency value means that the value is higher than the mean value of **SM**.)

Step 4:

Cluster regions with low saliency values by using the “revised edge rate”.

(“Low” saliency value means that the value is lower than the mean value of **SM**.)

(The revised edge rate will be described in the next section.)

Step 5:

Combine the Step3 and the Step4, and then output labels **L2**, which is our final natural image segmentation result.

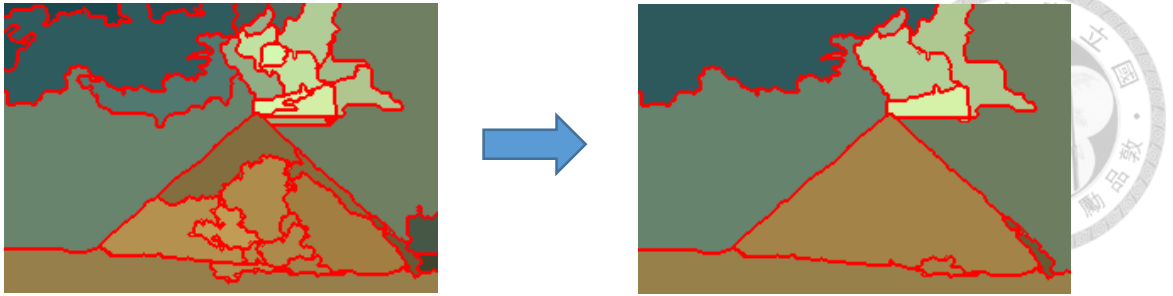


Fig. 4.11 The segmentation result from **Stage 2**.

4.5 Analysis of Our Algorithm



Compared to state-of-the-art methods [1], [2], [4], [17], and [18], we summarize the advantages and the uniqueness of our algorithm.

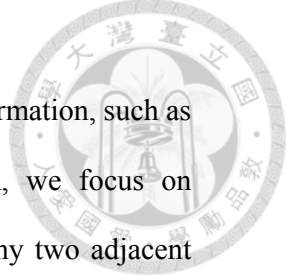
A. Superpixel-based segmentation algorithms:

Because there are significantly fewer superpixels than pixels, the computational complexity of the proposed algorithm is much lower than that of algorithms not based on superpixels. In addition, the superpixel method [16] we employ in our framework can preserve the object boundary effectively and prevent local noise from interfering with the segmentation result. Therefore, the proposed algorithm provides better segmentation results than algorithms not based on superpixels.

B. Image Classification

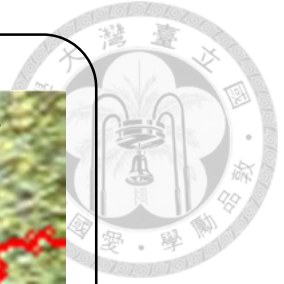
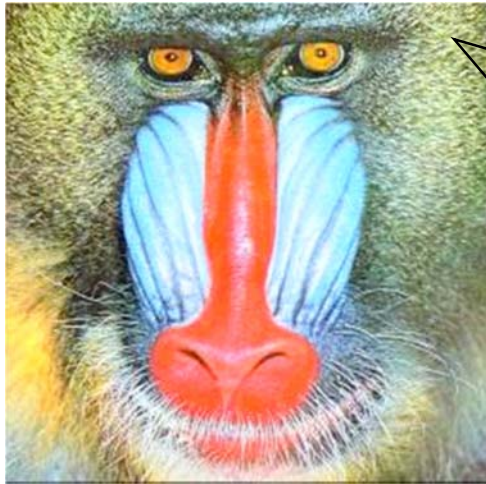
Users may need to segment medical images such as MRI(magnetic resonance imaging)s, CT(computed tomography) scans, and PET(positron emission tomography) scans. For this purpose, we classify input images before executing our main algorithm. In our framework, each image will first be stored in CIELAB color space; we then measure the variance of the A space and B space in the images. If the variance of the A space and B space in the image is lower than 5, which indicates the image is almost in black-and-white, it would be considered a medical image. Medical images typically do not contain the colors found in normal, natural images, owing to processing that occurs after MRIs, CT scans, and PET scans are obtained. When our classification system classifies an image as a medical image, the main segmentation algorithm focuses on the image's light variations.

C. Modified Edge Detection:



Other superpixel-based algorithms consider all superpixel information, such as color and lightness. In our modified edge detection algorithm, we focus on information related the “border” between superpixels. First, for any two adjacent superpixels, we compute the gradient values on the border between them, and find the edge E in Eq. (4.6). We then define the edge rate ER in Eq. (4.7) as the threshold required by the algorithm that merges two adjacent superpixels. When the edge rate is lower than 50%, we will merge two adjacent superpixels if the texture distance between them is small (the texture distance should be smaller than 0.25). When the edge rate is higher than 50%, we will not merge the superpixels, even if the texture distance between them is small.

We also define the complexity level CL of each superpixel in Eq. (4.5); it decides the edge rate ER . When the sum of the complexity levels of two adjacent superpixels is relatively high, the edge rate of the superpixels will be relatively low, which allows them to merge easily. For example, the superpixels generated from grass or animal fur have a high complexity level; therefore, the edge rate between them will be low, which allows them to merge easily. We can state that our modified edge detection algorithm has more perceptual meaning than other region-merging methods, because we consider the borders between superpixels and the complexity in superpixels simultaneously.



$$\text{edge rate } ER(i, j) = \frac{E(i, j)}{CL(i) + CL(j)} = \frac{0.74}{0.85 + 0.71} = 0.47 < 0.5$$

Fig. 4.12 An example of merging two superpixels with high complexity levels.



$$\text{edge rate } ER(i, j) = \frac{E(i, j)}{CL(i) + CL(j)} = \frac{0.42}{0.23 + 0.38} = 0.69 > 0.5$$

Fig. 4.13 Due to two superpixels with low complexity levels, the proposed algorithm would not merge them.

D. Clustering superpixels according to their texture features:

Spectral clustering algorithms have proved to be more effective at finding clusters than some traditional algorithms such as k-means, and they have been utilized in many segmentation algorithms. However, spectral clustering algorithms may break a large homogeneous region into smaller segments. To solve this problem, in our framework, we cluster superpixels according to their texture features instead of using spectral clustering algorithms, because large homogeneous regions will have nearly the same texture features. Furthermore, our proposed texture-based clustering algorithm can prevent overmerging between the foreground and background, because foreground texture features and background texture features typically have significant differences.

E. Recording merge times

After clustering superpixels in **Stage 1**, we record the merge times of each region; the recording method steps are as follows.

Step 1:

Record the central coordinate of each superpixel generated by the ERS superpixel method during the first step of **Stage 1**.

Step 2:

After clustering superpixels in **Stage 1**, the image is segmented into regions; we then count the central coordinates in each region. These counting results determine the regions' merge times.

If the merge times of a region is less than two times and the average saliency value of the region is small, we would merge this region to its adjacent region, because it might be noise from the background of the image.

F. Shadow Detection

Shading and highlights caused by uneven illumination can trigger image measurement problems. These phenomena may cause an object and its shadow to be segmented into different regions.

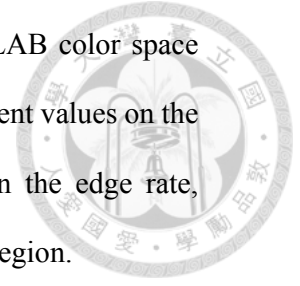


We found that the main differences between an object and the shadows in a portion of the object are as follows:

1. When we observe the object and its shadow in the CIELAB color space, they have a large variance in the L space but little variance in the A space and B space. In other words, their illumination levels vary significantly but their color is nearly the same.
2. In general, the texture features in the object and its shadow should be very close, because the shadow represents a portion of the object. In our framework, we adopt Log-Gabor functions to measure texture distance in Eq. (4.9), and the results demonstrate that the texture distance between the object and its shadow is close to zero. That is to say, their texture features are very similar.

If any two adjacent superpixels meet the aforementioned conditions (large differences in lightness, nearly the same color, and similar texture features), we use a LOG filter to measure the edge E between them in Eq. (4.6). Because LOG filters are based on second-order derivatives, the gradient values on the border between adjacent superpixels will be small, because the difference in illumination between the object and its shadow is analogous to a step. Because of the small gradient values on the border, the edge rate ER in Eq. (4.7) will also be small, which allows the adjacent superpixels to be clustered more easily.

Briefly speaking, our shadow detection algorithm uses CIELAB color space and texture features to detect shadows, and then measures the gradient values on the border using a LOG filter; this causes a significant reduction in the edge rate, allowing the object and its shadow to be segmented into the same region.



G. Saliency-oriented region merging

To make our segmentation result more perceptual, we propose saliency-oriented region merging to cluster the superpixel regions into specific areas. In each superpixel region, we average the pixel saliency values to obtain the region saliency, which ensures that each superpixel region shares the same saliency value. We then merge the regions that have high and low saliency values, because the regions with high and low saliency values usually represent the foreground and background in an image; moreover, the regions in the foreground and background should be labeled into the same area respectively.

Unlike **Stage 1**, the proposed saliency-oriented region merging algorithm uses the “revised edge rate” to cluster regions. For any two adjacent regions, the revised edge rate can be obtained from the following table.

Table 4.1 The revised edge rate in **Stage 2**.

saliency values ≥ 225	Revised ER = ER / 1.9
$200 \leq$ saliency values < 225	Revised ER = ER / 1.5
$50 \leq$ saliency values < 200	Revised ER = ER
$10 \leq$ saliency values < 50	Revised ER = ER / 1.5
saliency values < 10	Revised ER = ER / 1.9

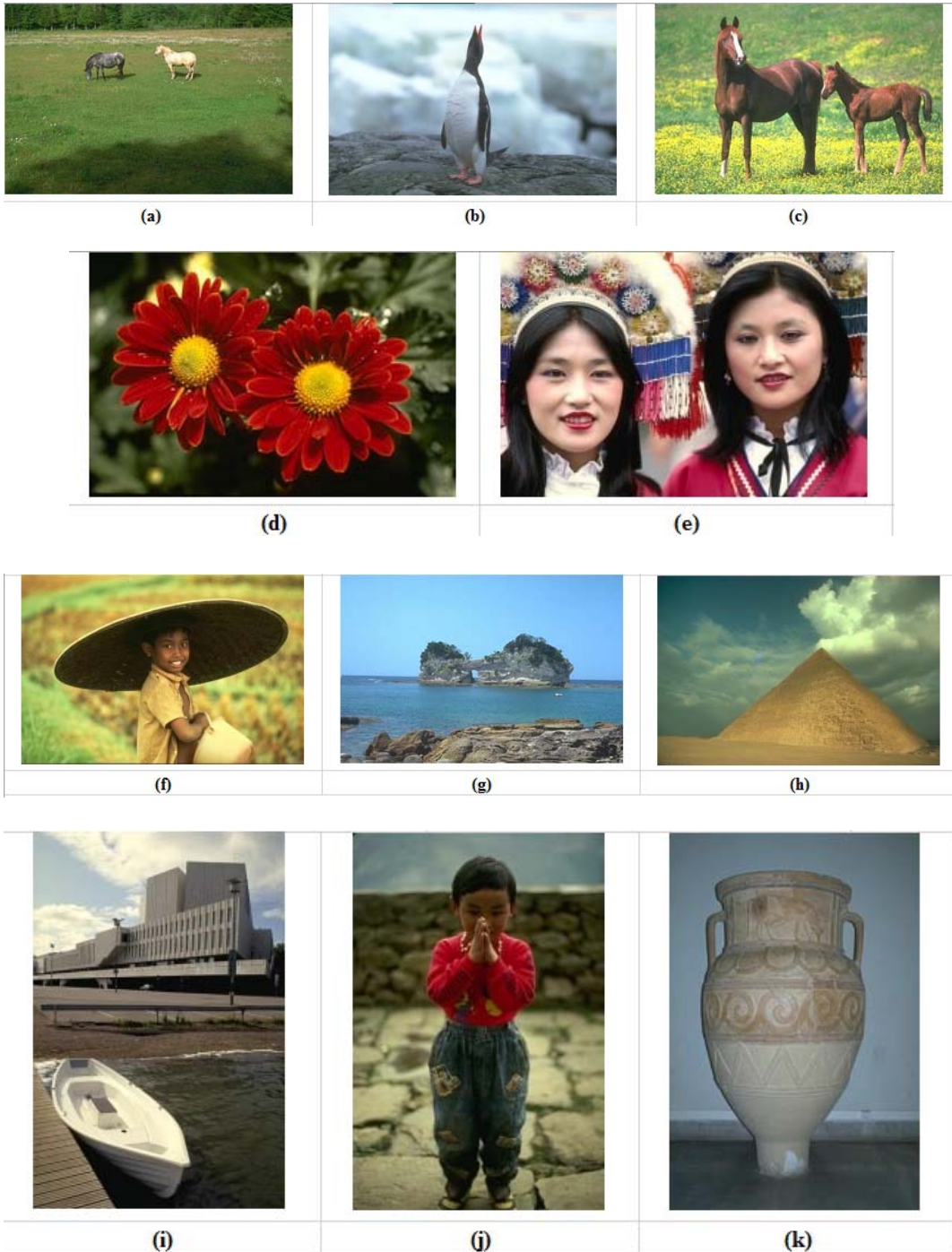
According to the Table 4.1, when the saliency values for any two adjacent regions are relatively high or relatively low, the revised edge rate between them will be lower. For example, if we find two regions with very high saliency values, we should merge them together because these two regions might be the foreground, and the lower revised edge rate would allow them be merged more easily. This is the reason we adopt the revised edge rate instead of the edge rate in **Stage 2**.

Chapter 5 Simulations



5.1 Compared with the state-of-the-art methods

5.1.1 Database



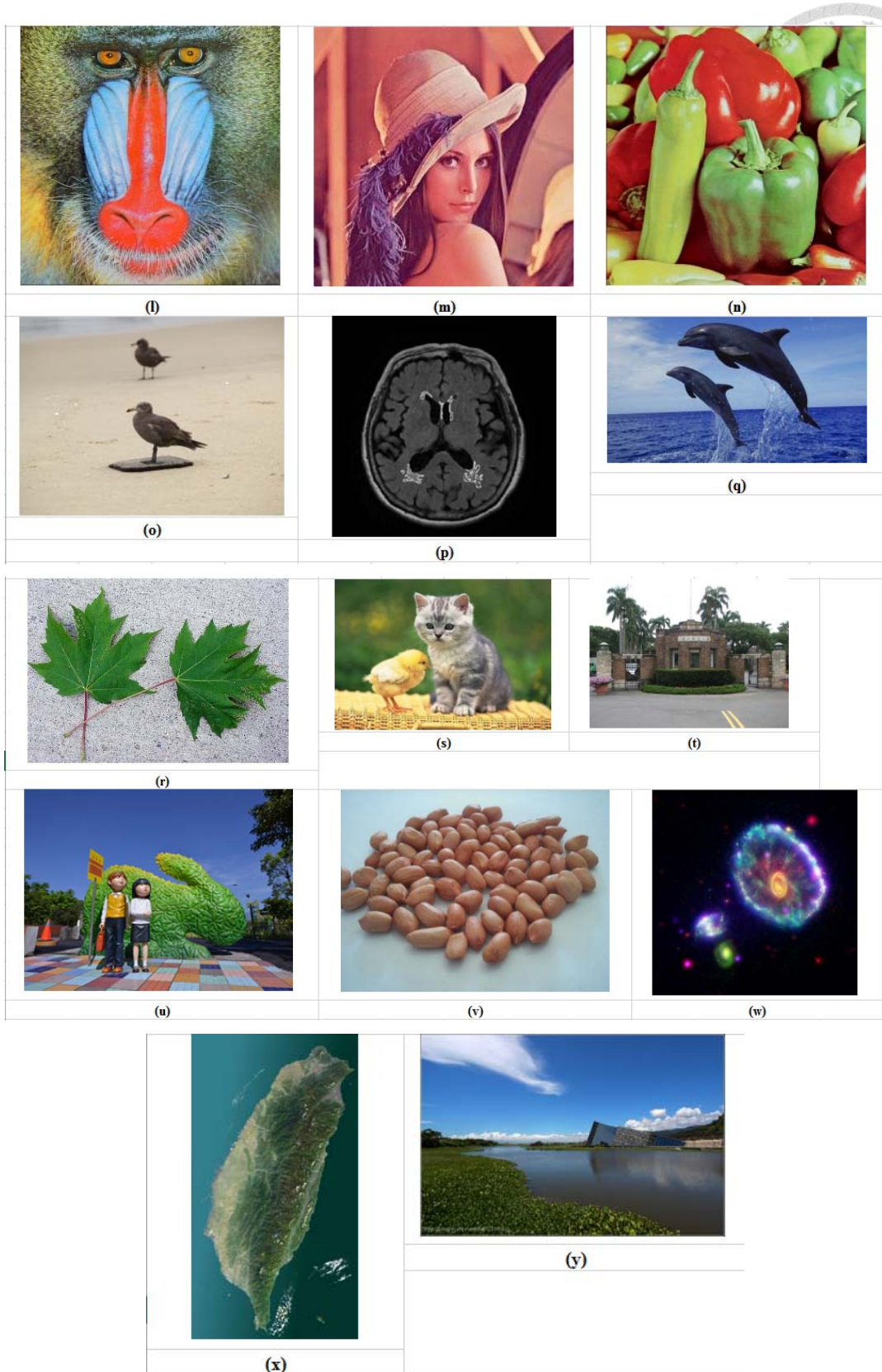
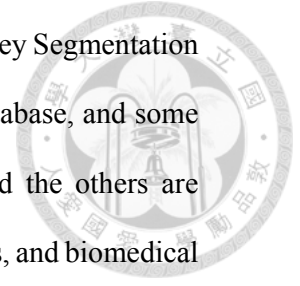


Fig. 5.1 Our image database

Among the test images used, some images are taken from the Berkeley Segmentation Database and the Microsoft Research Cambridge object recognition database, and some images are from classic images in the field of image processing, and the others are collected online and include natural scenery images, astronomical images, and biomedical images.



The sources of the test images are as follows:

- (a)~(k): From the Berkeley Segmentation Database.
- (l)~(n): Common images for image processing.
- (o): From the Microsoft Research Cambridge object recognition database.
- (p): From <http://web.shs.kyushu-u.ac.jp/~arimura/research-1/pg57.html>
- (q): From <http://hk.apple.nextmedia.com/realtime/international/20120731/51026523>
- (r): From http://www.pddl.purdue.edu/ppdl/weeklypics/Weekly_Picture6-25-01-1.html
- (s): From http://www.zhifeifw.com/fjgc/zbfy_list.php?id=7462
- (t): From <http://utome.asia/pin/12997/>
- (u): From <http://okgo.tw/butyview.html?id=2897>
- (v): From http://www.indonetwork.co.id/ud_barokah_co/1196034
- (w): From http://aeaa.nmns.edu.tw/x_games/ap0817.html
- (x): From <http://www.hopemap.net/HopeDetail.php?hid=116>
- (y): From <http://blog.xuite.net/rita5031/blog>

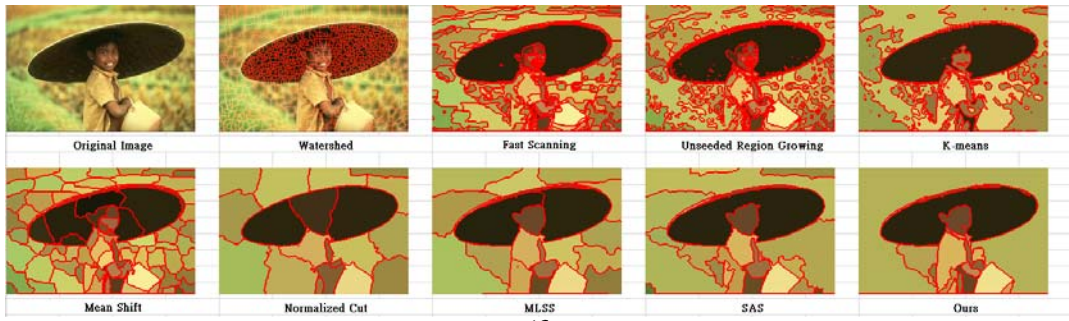
5.1.2 Visual Comparison

We use Watershed [6], Fast Scanning, Unseeded Region Growing [11], K-means [20], Mean Shift [4], Normalized Cut [1], MLSS [18], and SAS [17] for the visual comparison shown in Fig. 5.2.

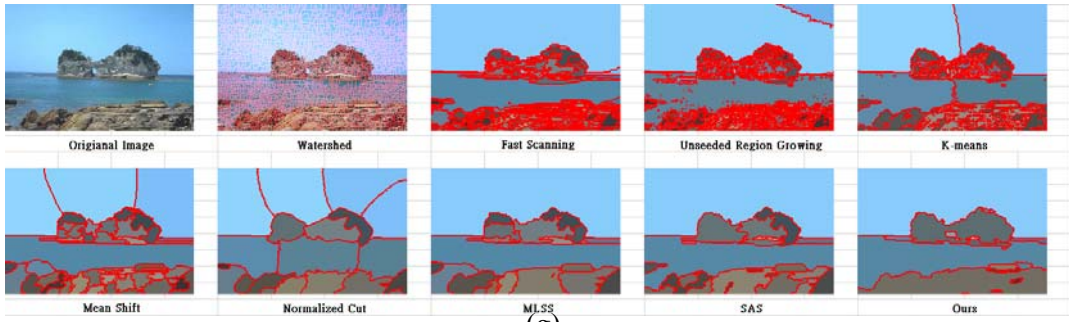




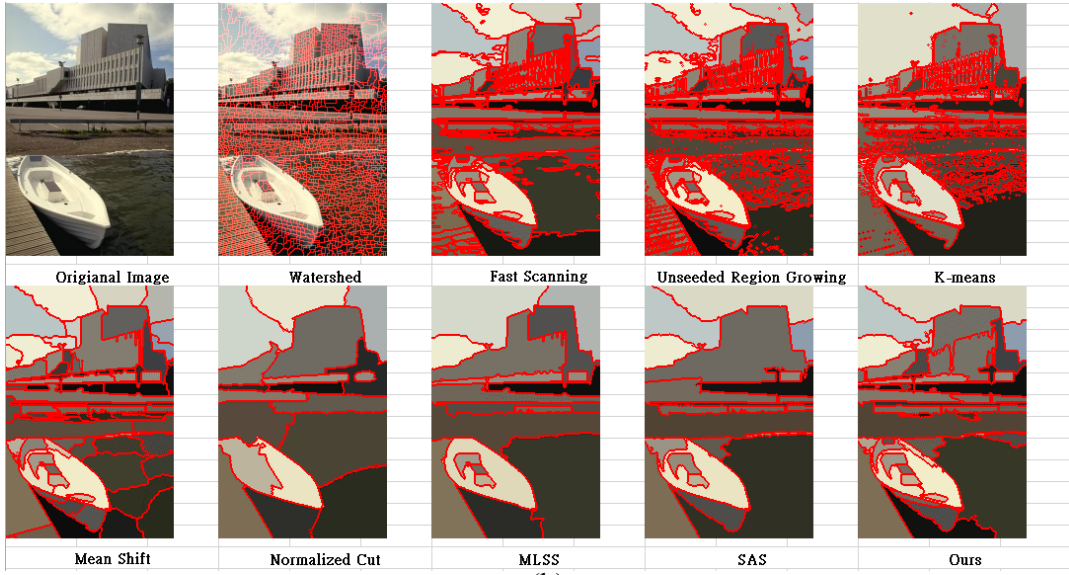
(e)



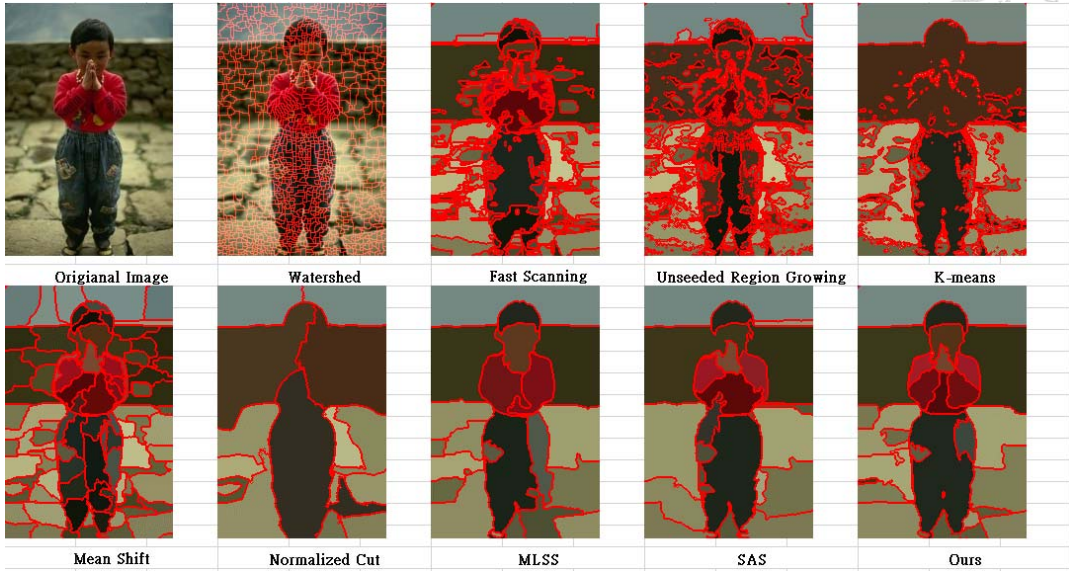
(f)



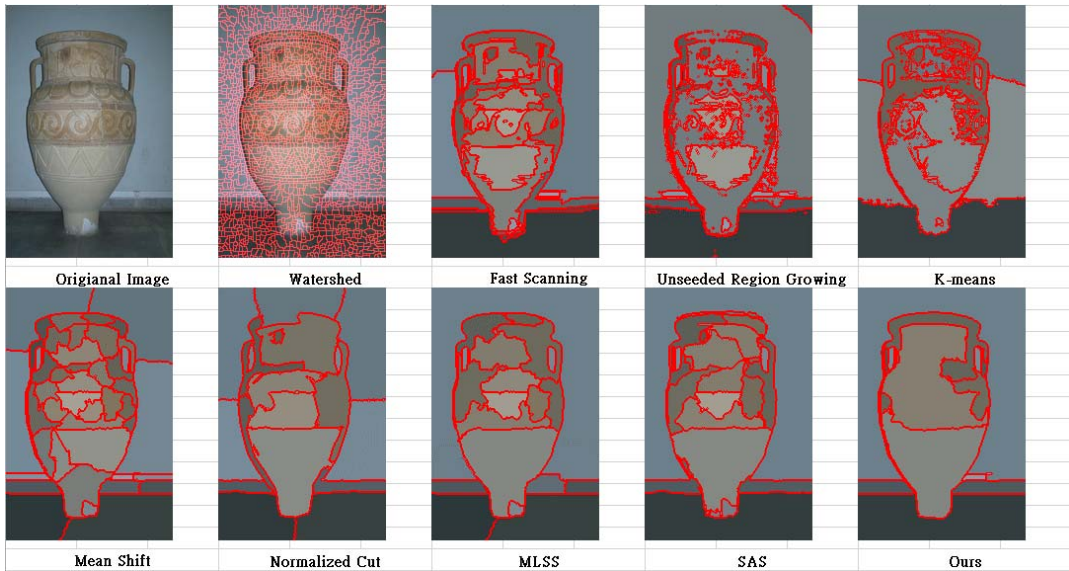
(g)



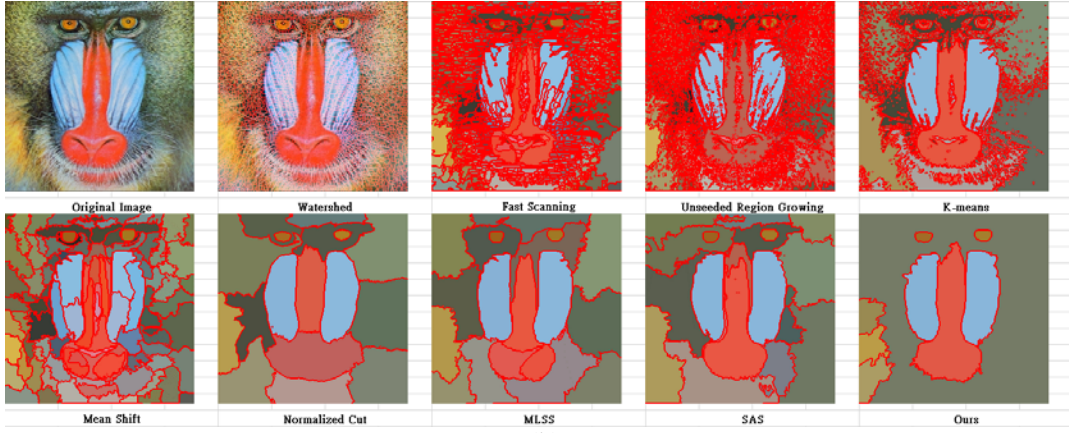
(h)



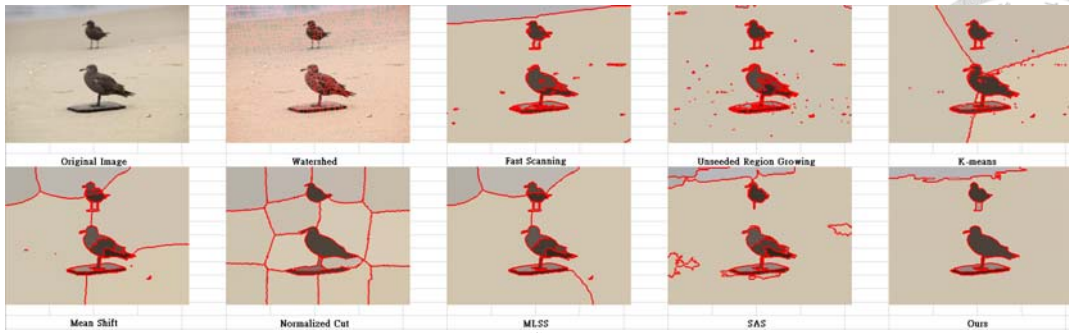
(i)



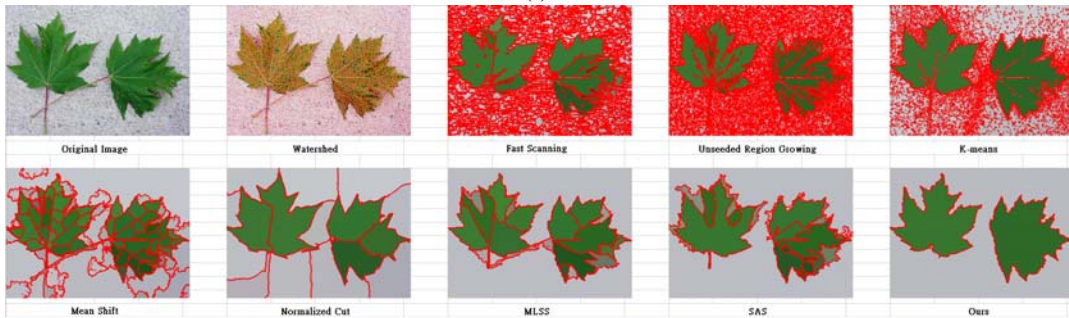
(j)



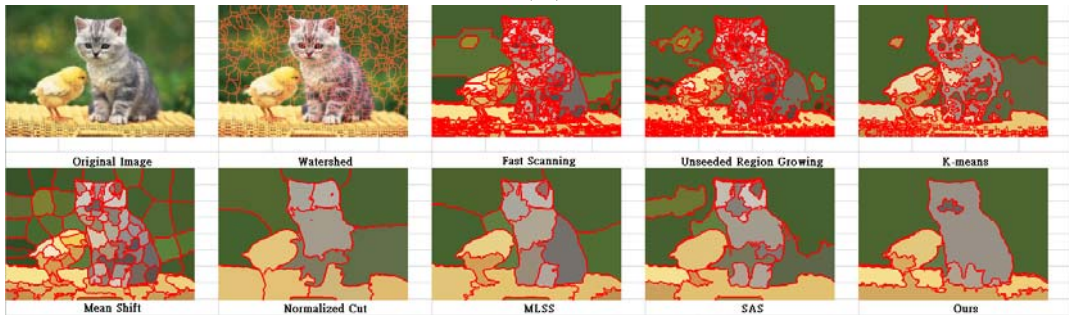
(k)



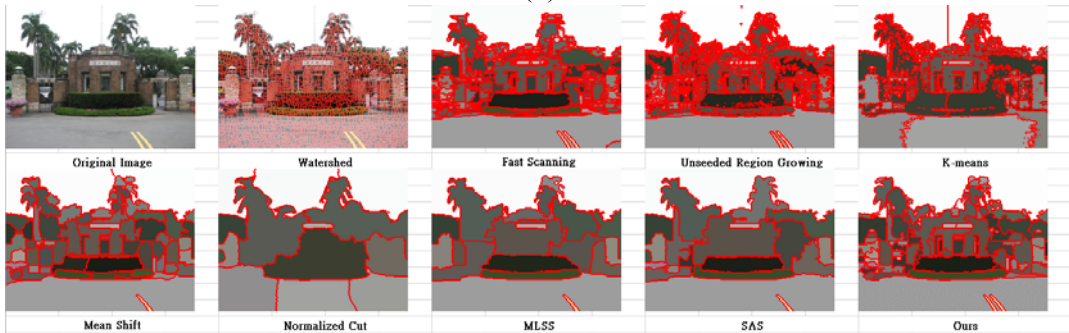
(l)



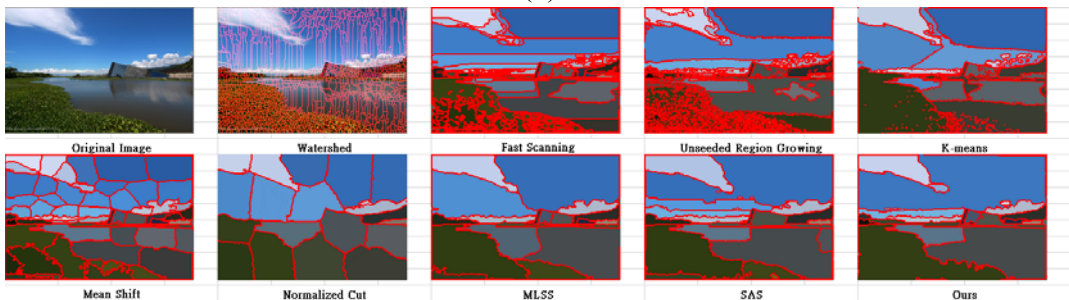
(m)



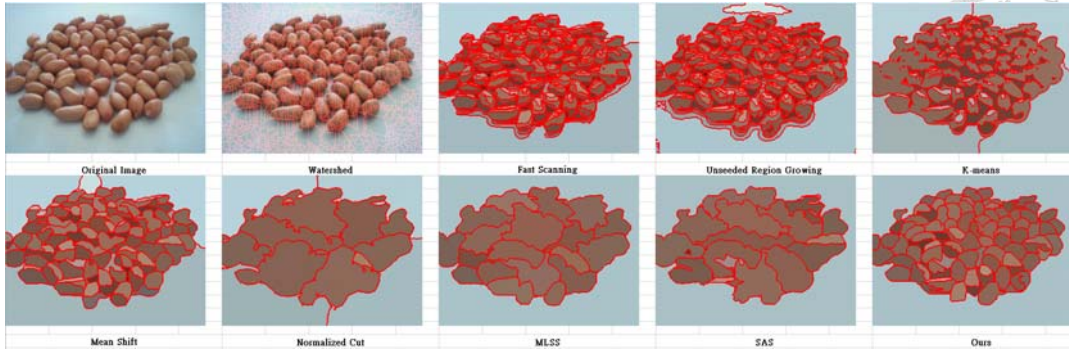
(n)



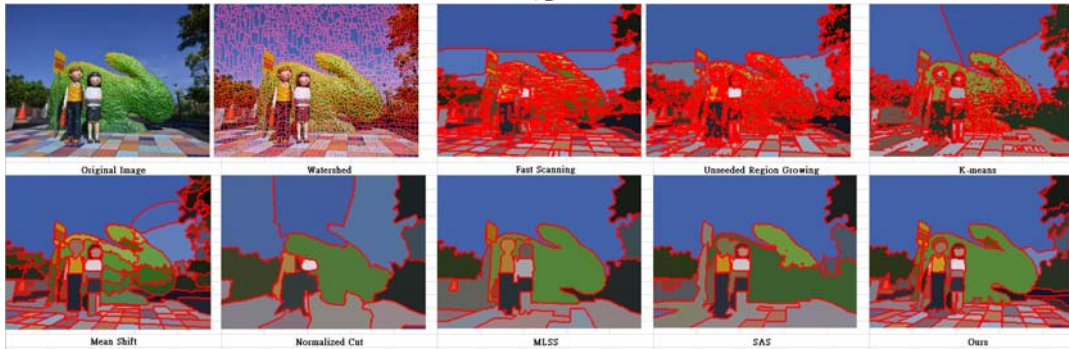
(o)



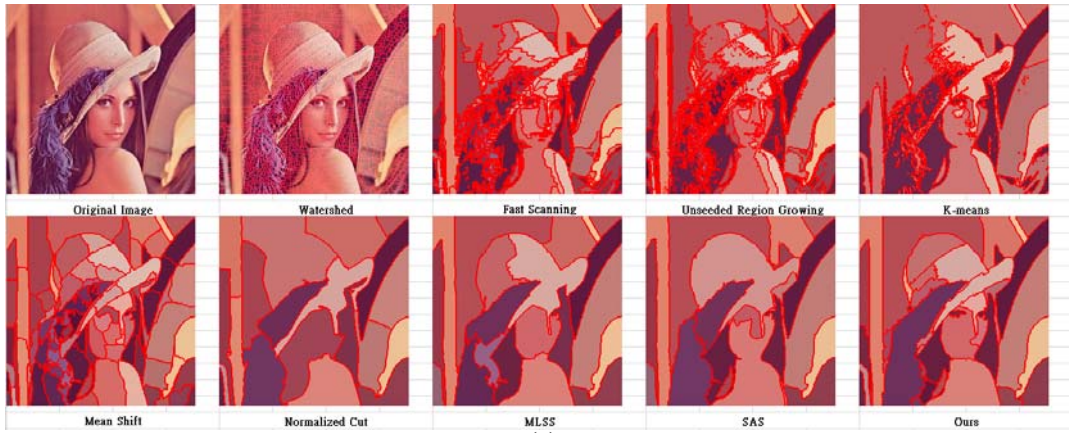
71 (p)



(q)

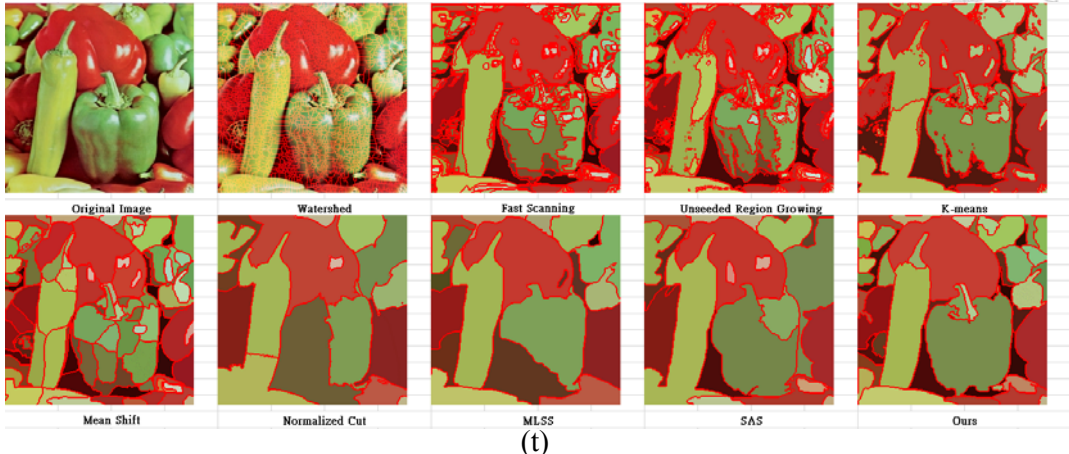


(r)

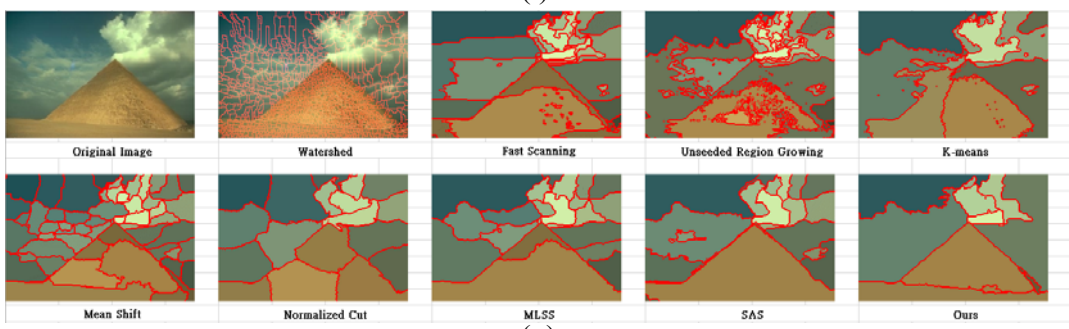


(s)

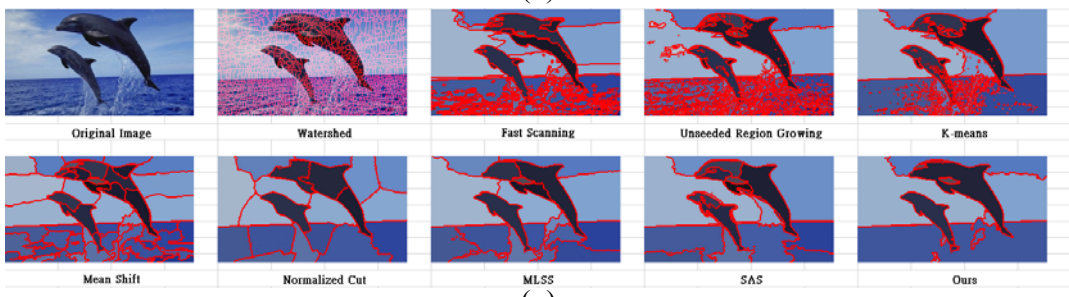




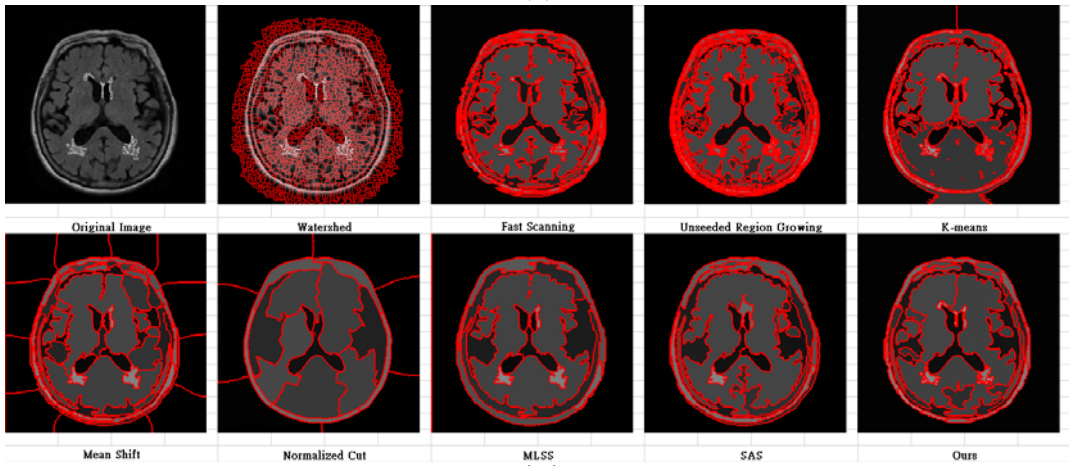
(t)



(u)



(v)



(w)

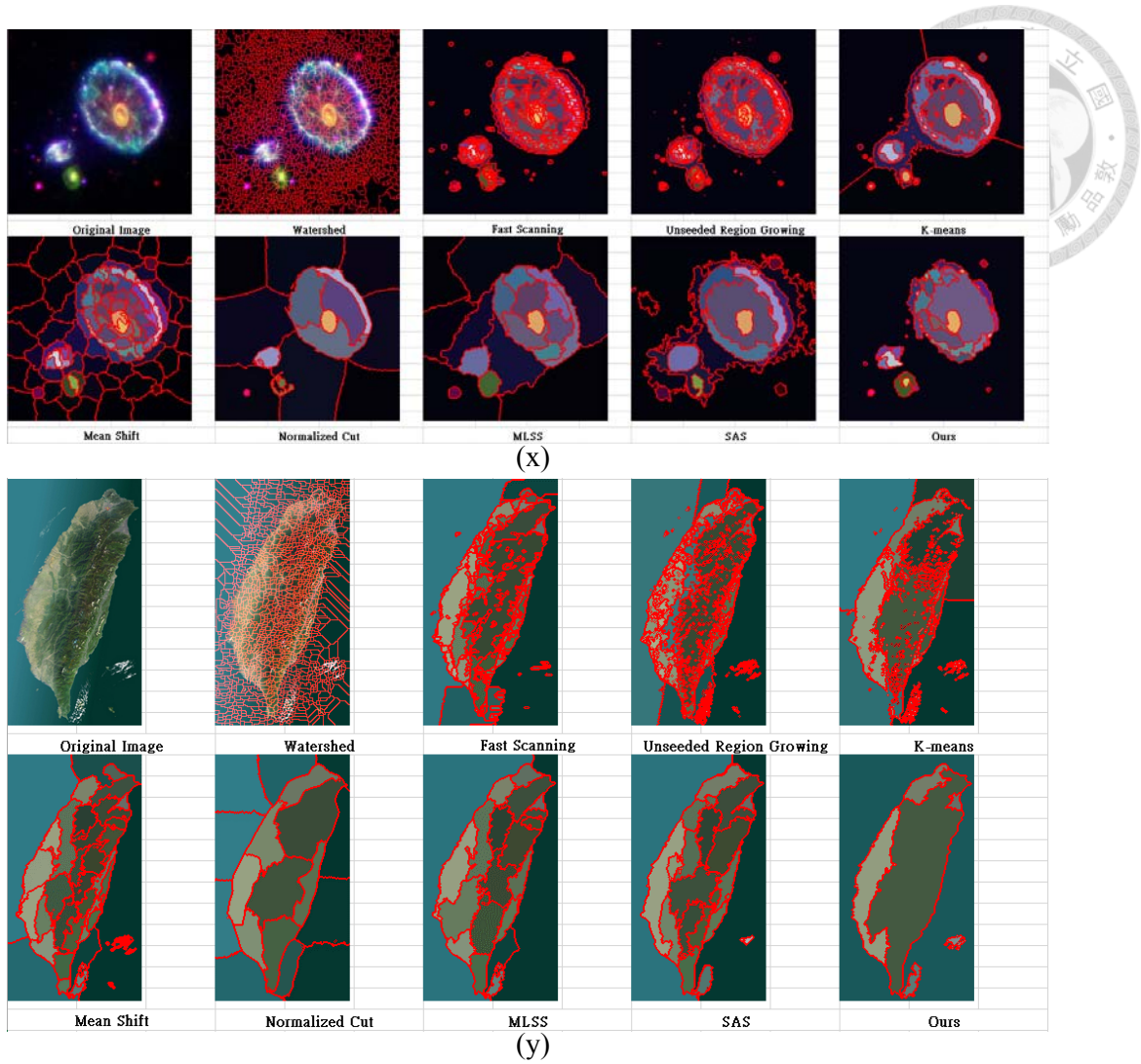
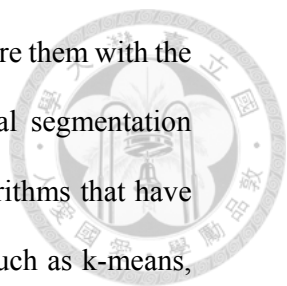


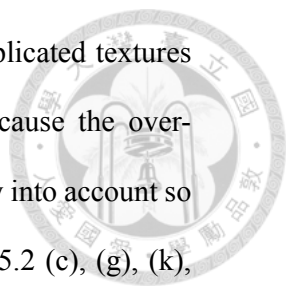
Fig. 5.2 The visual comparison between our proposed method and the other state-of-the-art methods.



We select many different image segmentation algorithms and compare them with the proposed method. These algorithms include time-honored but classical segmentation algorithms (such as watershed and unseeded region growing), and algorithms that have been commonly used for image segmentation over the last ten years (such as k-means, mean shift, and normalized cut). Finally, we also compare the proposed algorithm with segmentation algorithms (such as MLSS and SAS) that have been shown to deliver excellent performance over the last few years (Note that the number of regions to be segmented must be set for MLSS and SAS. In the comparison figures in Fig. 5.2, this parameter is set to 20.).

From Fig. 5.2, we can infer that for any natural image, the proposed natural image segmentation algorithm performs better than most of the other segmentation algorithms. The following are the advantages of the proposed method for image segmentation over the other methods:

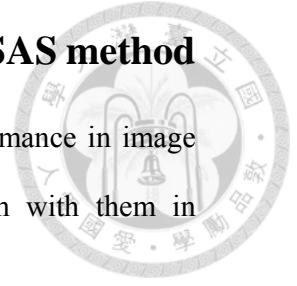
1. By selecting different edge detectors according to specific conditions, the proposed method effectively solves the over-segmentation problem due to the shadow of the objects and the difference in the brightness of the objects. The results are shown in Figs. 5.2 (a) and (u).
2. Even if the foreground and the background in an image have an unclear boundary or similar color, the proposed method can completely segment the foreground from the background because the proposed method focuses particularly on boundary information. The results are shown in Figs. 5.2 (b), (f), (s), and (u).

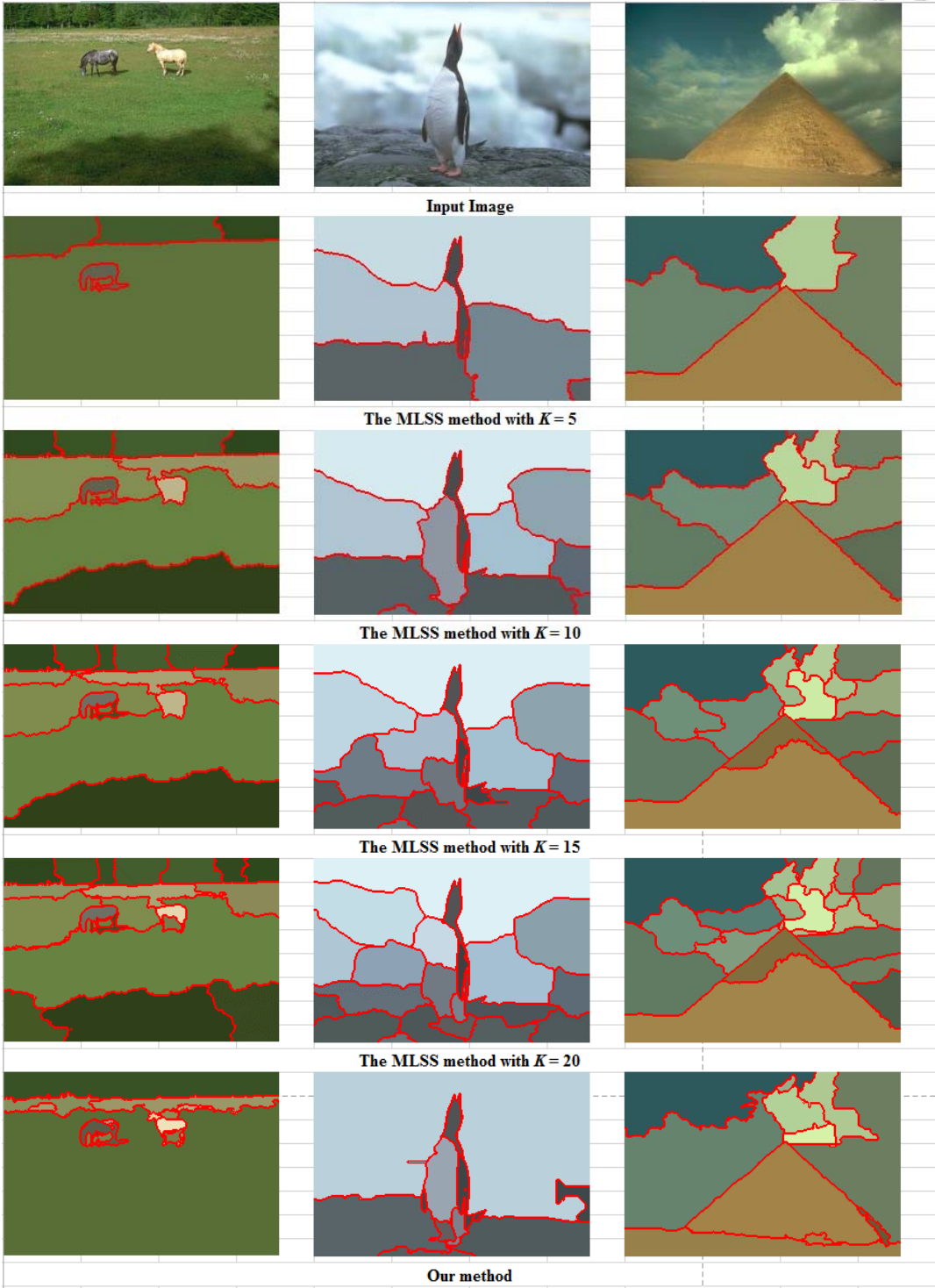
- 
3. Early segmentation methods normally fail to properly process complicated textures in images, such as animal fur, grassland, and ocean waves, and cause the over-segmentation problem. The proposed method takes image complexity into account so as to gain significant improvement. The results are shown in Figs. 5.2 (c), (g), (k), (m), (n), and (v).
 4. The sensory system of the human eyes segments excessively disordered or unclear backgrounds into the same region. The proposed method is integrated with saliency detection to detect the background regions in the image and merges these regions together. The results are shown in Figs. 5.2 (d), (f), and (i).
 5. The proposed method can be used for processing various types of images. For example, Fig. 5.2 (w) represents biomedical images, (x) represents astronomical images, and (y) represents satellite cloud images. Further, the proposed method can obtain excellent segmentation results for different types of images.

5.2 Compared with the MLSS method and the SAS method

Since the MLSS method and the SAS method have better performance in image segmentation nowadays, we will focus on comparing our algorithm with them in parameter adjustment.

Both of the MLSS method and the SAS method need the parameter K , which decides the final number of regions in an image. For example, we adopt the MLSS method with parameter K and $K = 5$, therefore the image will be segmented into five regions. Fig. 5.3 and Fig. 5.4 show that the segmentation result of the MLSS method and the SAS method with parameter $K = 5$, $K = 10$, $K = 15$, and $K = 20$.







Input Image



The MLSS method with $K = 5$



The MLSS method with $K = 10$



The MLSS method with $K = 15$

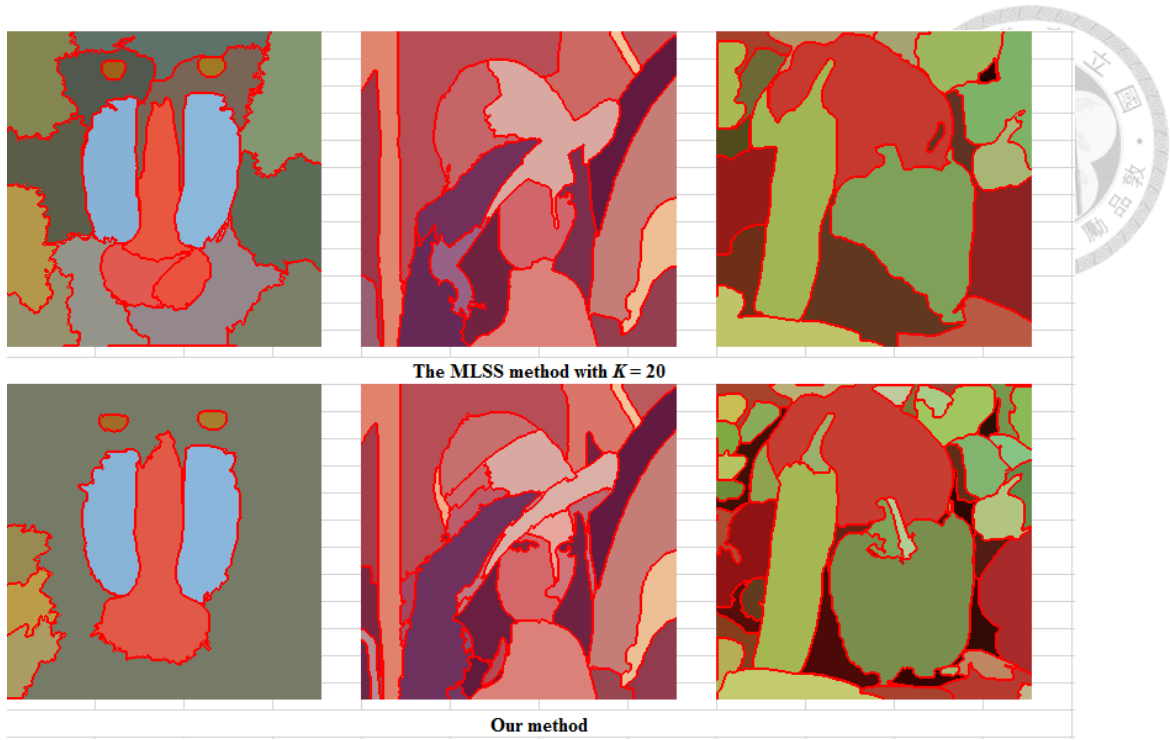
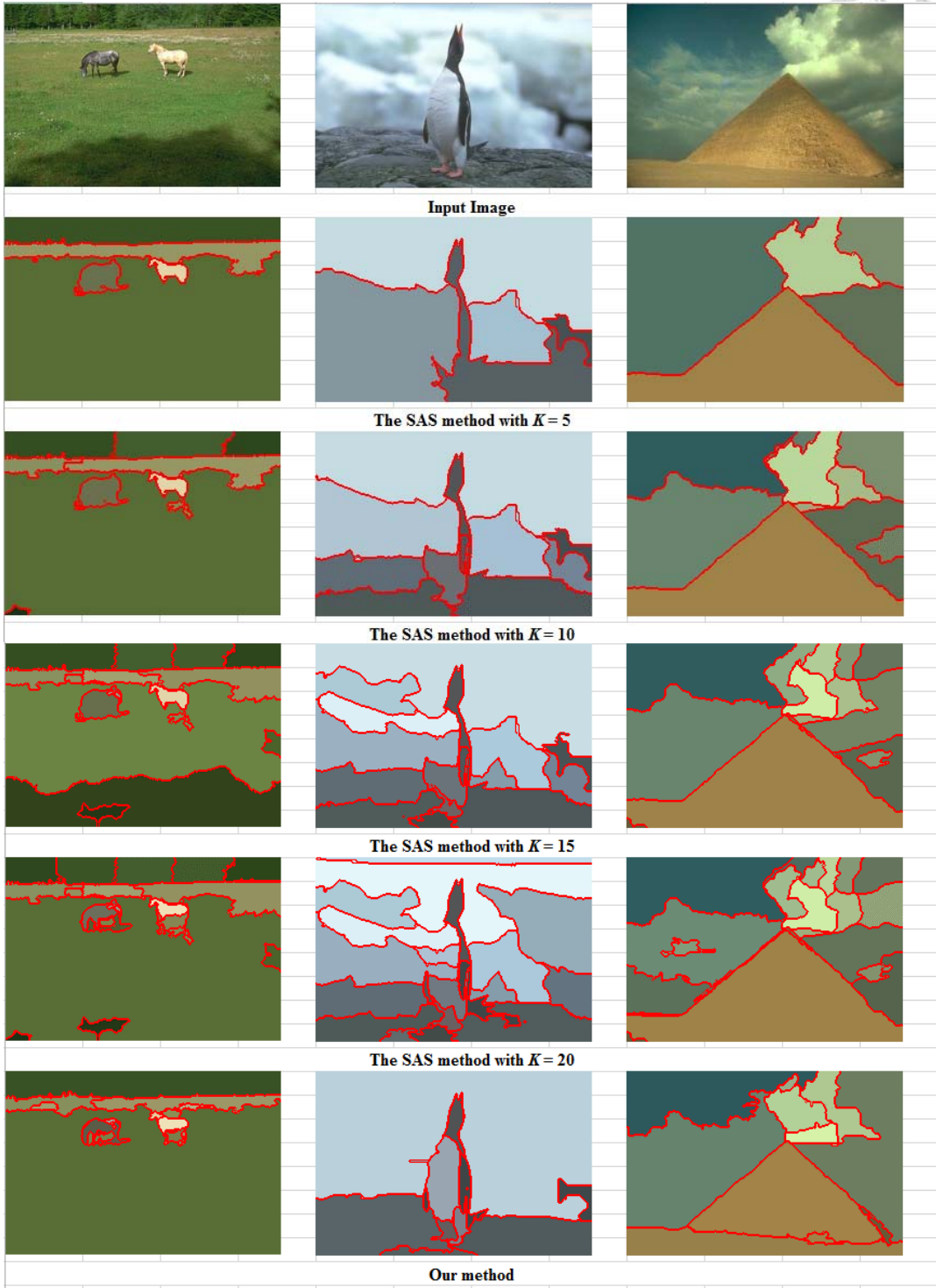


Fig. 5.3 Compare our segmentation method with the MLSS method.





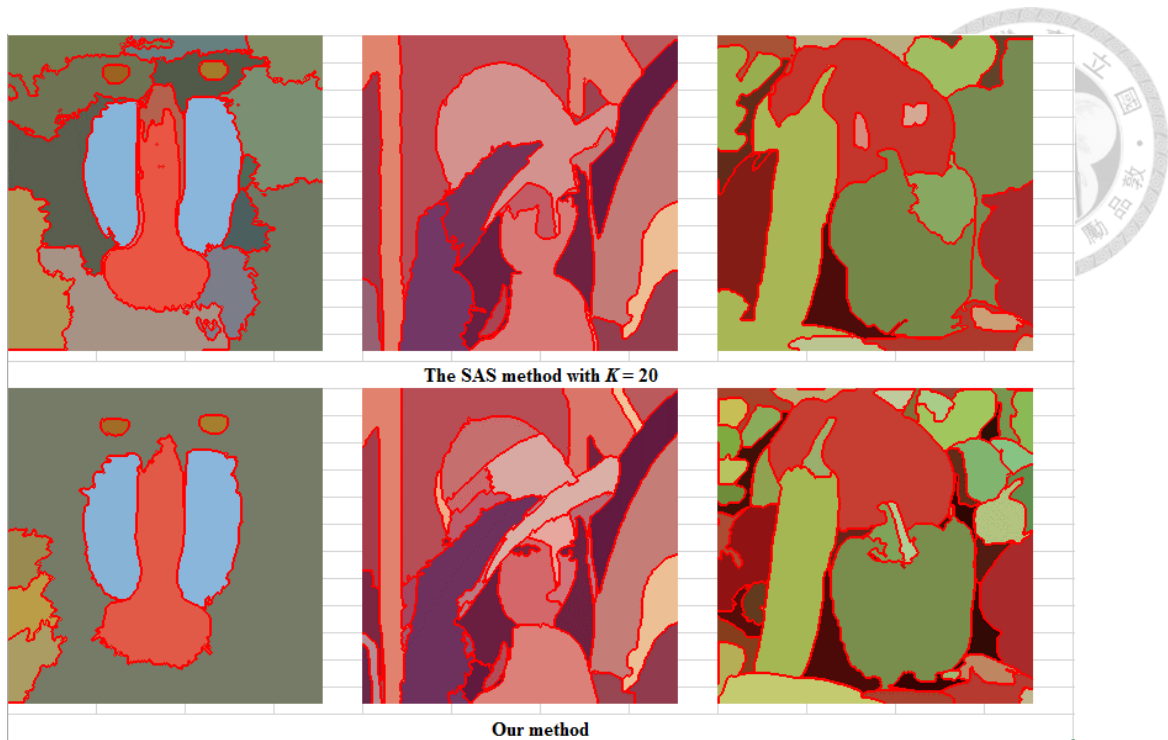


Fig. 5.4 Compare our segmentation method with the SAS method.

From Fig. 5.3 and Fig. 5.4, we can observe that

1. For the MLSS method and the SAS method, parameter K plays an important role in the performance of segmentation result.
2. For each different kind of natural image, when we adopt the MLSS method or the SAS method, we should choose an appropriate parameter K to make the segmentation result perform well.

Compared with the MLSS method and the SAS method, our proposed method do not need any parameter adjustment and perform more accurately in natural image segmentation.

Chapter 6 Conclusion and Future Work



6.1 Conclusion

In this thesis, we provide an overview of some important works of image segmentation and superpixel-based segmentation algorithms in recent years.

We propose a natural image segmentation framework with highly accurate performance and without any parameter adjustment. Our framework is based on ERS superpixels, modified edge detection, Log-Gabor texture clustering and saliency-oriented region merging. Constructing the graph representation using ERS superpixel makes our algorithm become efficient. Moreover, the borders of ERS superpixels match the true edges of the objects very well. In our stage 1, Clustering superpixels according to their texture features applying Log-Gabor filter and the proposed modified edge detection which combines the Sobel filter and LOG filter make the segmentation result more accurate. In our stage 2, we use saliency detection to merge regions in the foreground and background, which would match human perception much more.

The simulations on the database of natural images show that the proposed algorithm outperforms the state-of-the-art segmentation methods because of its parameterless adjustment and highly accurate segmentation result.



6.2 Future Work

1. Efficiency

Because of a lot of calculation based on loops, we could further speed up the program by implementing the code fully by C++. (We implement the code by MATLAB now.) Additionally, since we apply the same measure (gradient values and texture features) to all superpixels, the process can be implemented in parallel computing.

2. Specular-free image

Although we propose an algorithm to solve over-segmentation due to shadow problem, there are some challenges of image segmentation caused by illumination, such as highlights or specular reflections.

In image segmentation, the areas with highlights or specular reflections might be segmented to a lot of single regions, which is not perceptual, because highlights or specular reflections are still the portions of an object. Specular-free image is a concept first proposed by Tan *et al* [30], and they want to remove the specular reflections in the images. Therefore, if we can adopt the concept of specular-free image to remove the specular reflections in the input images, the segmentation results might be more perceptual.

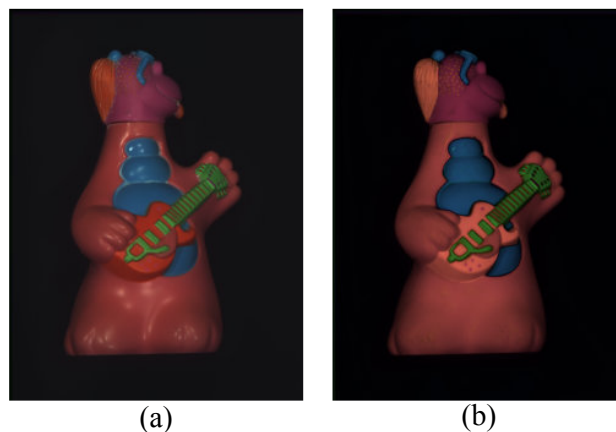


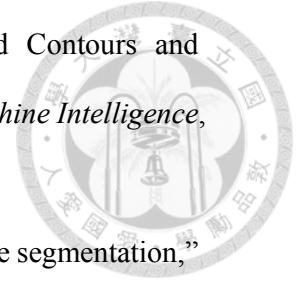
Fig. 6.1 (a) Original image. (b) Specular-free image.

REFERENCE



A. Image Segmentation

- [1] J. Shi and J. Malik, “Normalized Cuts and Image Segmentation,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888-905, Aug. 2000.
- [2] P. Felzenszwalb and D. Huttenlocher, “Efficient Graph-Based Image Segmentation,” *Int’l J. Computer Vision*, vol. 59, no. 2, pp. 167-181, Sept. 2004.
- [3] V. Kwatra, A. Schodl, I. Essa, G. Turk, and A. Bobick, “Graphcut Textures: Image and Video Synthesis Using Graph Cuts,” *ACM Trans. Graphics*, vol. 22, no. 3, pp. 277-286, July 2003.
- [4] D. Comaniciu and P. Meer, “Mean Shift: A Robust Approach toward Feature Space Analysis,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603-619, May 2002.
- [5] A. Vedaldi and S. Soatto, “Quick Shift and Kernel Methods for Mode Seeking,” *Proc. European Conf. Computer Vision*, 2008.
- [6] L. Vincent and P. Soille, “Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 583-598, June 1991.
- [7] Y. Weiss, “Segmentation Using Eigenvectors: A Unifying View,” *Proc. IEEE Int’l Conf. Computer Vision*, 1999.
- [8] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, “Contour Detection and Hierarchical Image Segmentation,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898-916, May 2011.
- [9] T. Cour, F. Benezit, J. Shi, “Spectral segmentation with multiscale graph decomposition,” in *CVPR*, pp. 1124-1131, 2005.



- [10] L. Najman and M. Schmitt, "Geodesic Saliency of Watershed Contours and Hierarchical Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 12, pp. 1163-1173, Dec. 1996.
- [11] Z. Lin, J. Jin and H. Talbot, "Unseeded region growing for 3D image segmentation," *ACM International Conference Proceeding Series*, vol. 9, pp. 31-37, 2000.

B. Superpixel

- [12] A. Moore, S. Prince, J. Warrell, U. Mohammed, and G. Jones, "Superpixel Lattices," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008.
- [13] S. Avidan and A. Shamir, "Seam Carving for Content-Aware Image Resizing," *ACM Trans. Graphics*, vol. 26, no. 3, Article 10, 2007.
- [14] O. Veksler, Y. Boykov, and P. Mehrani, "Superpixels and Supervoxels in an Energy Optimization Framework," *Proc. European Conf. Computer Vision*, 2010.
- [15] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274 - 2282, May 2012.
- [16] Y. M. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa, "Entropy rate superpixel segmentation," *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2097-2104, 2011.

C. Superpixel-Based Segmentation

- [17] Z. Li, X. M. Wu, and S. F. Chang, "Segmentation Using Superpixels: A Bipartite Graph Partitioning Approach," in *CVPR*, pp. 789-796, 2012.
- [18] T. Kim and K. Lee, "Learning full pairwise affinities for spectral segmentation," in *CVPR*, pp. 2101-2108, 2010.



D. Clustering Techniques

- [19] S.X. Yu and J. Shi, "Multiclass Spectral Clustering," *Proc. IEEE Int'l Conf. Computer Vision*, 2003.
- [20] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 2, pp. 849–856, 2002.
- [21] I. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," In *ACM SIGKDD*, pp. 269-274, 2001

E. Computer Vision

- [22] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Schölkopf, "Learning with Local and Global Consistency," *Proc. Neural Information Processing Systems*, 2003.
- [23] P. Arbelaez, "Boundary Extraction in Natural Images Using Ultrametric Contour Maps," *Proc. IEEE Workshop Perceptual Organization in Computer Vision*, 2006.

F. Saliency Detection

- [24] C. Yang, L. Zhang, H. Lu, X. Ruan, and M.-H. Yang, "Saliency detection via graph-based manifold ranking," in *CVPR*, 2013.
- [25] L. Grady, M. Jolly, and A. Seitz, "Segmentation from a box," in *ICCV*, 2011.

G. Edge Detection

- [26] P. Acharjya, R. Das, and D. Ghoshal, "A Study on Image Edge Detection Using the Gradients," *International Journal of Scientific and Research Publications*, vol. 2, Issue 12, Dec. 2012.
- [27] L.S. Davis, "A survey of edge detection techniques," *Computer Graphics and Image Processing*, vol 4, no. 3, pp 248-260, 1975.

H. Theorems and Mathematics

[28] Field, David J. "Relations between the statistics of natural images and the response properties of cortical cells," *JOSA A* 4.12 (1987): 2379-2394.

[29] G. Golub and C. Van Loan. Matrix computations. Johns Hopkins University Press, 1996.

I. Specular-Free Image

[30] R. T. Tan, K. Ikeuchi, "Separating reflection components of textured surfaces using a single image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 2, 2005.



PUBLICATION



- [1] C. J. Lin, J. J. Ding, and **I. F. Lu**, “Real-time interactive segmentation with superpixel Pre-segmentation,” *CVGIP*, Kenting, Taiwan, Aug. 2014
- [2] J. J. Ding, **I. F. Lu**, C. J. Lin, and Z. W. Lin, “Improved superpixel for interactive image segmentation,” *National Symposium on Telecommunications*, Taichung, Taiwan, Nov. 2014
- [3] J. J. Ding, J. Y. Wu, and **I. F. Lu**, “Very fast image segmentation algorithms using block-wise parallel structures,” *Joint IWAIT&IFMIA Conference*, Tainan, Taiwan, Jan. 2015
- [4] J. J. Ding, C. J. Lin, **I. F. Lu**, and Y. H. Cheng, “Real-time interactive image segmentation using improved superpixels,” *IEEE International Conference on Digital Signal Processing*, Singapore, Jul. 2015

To be submitted:

- [5] **I. F. Lu**, J. J. Ding, H. Y. Ko, “High Accuracy and High Robust Natural Image Segmentation Algorithm without Parameter Adjusting,” in *CVGIP*, 2015.
- [6] **I. F. Lu**, J. J. Ding, H. Y. Ko, “High Accuracy and High Robust Natural Image Segmentation Algorithm without Parameter Adjusting,” *IEEE Trans. Image Processing*, 2015.