12-2007

On the Stability of Region Count in the Parameter Space of Image Analysis Methods

Li Yu
Clemson University, liy@clemson.edu

On the Stability of Region Count in the Parameter Space of Image Analysis Methods

---

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Electrical Engineering

---

by
Li Yu
December 2007

Accepted by:
Dr. Adam Hoover, Committee Chair
Dr. Eric Muth
Dr. Ian Walker
Dr. Stanley Birchfield

ABSTRACT

In this dissertation a novel bottom-up computer vision approach is proposed. This approach is based upon quantifying the stability of the number of regions or count in a multi-dimensional parameter scale-space. The stability analysis comes from the properties of flat areas in the region count space generated through bottom-up algorithms of thresholding and region growing, hysteresis thresholding, variance-based region growing. The parameters used can be threshold, region growth, intensity statistics and other low-level parameters. The advantages and disadvantages of top-down, bottom-up and hybrid computational models are discussed. The approaches of scale-space, perceptual organization and clustering methods in computer vision are also analyzed, and the difference between our approach and these approaches is clarified. An overview of our stable count idea and implementation of three algorithms derived from this idea are presented. The algorithms are applied to real-world images as well as simulated signals. We have developed three experiments based upon our framework of stable region count. The experiments are using flower detector, peak detector and retinal image lesion detector respectively to process images and signals. The results from these experiments all suggest that our computer vision framework can solve different image and signal problems and provide satisfactory solutions. In the end future research directions and improvements are proposed.

DEDICATION

To my dear wife Yan Liu, my son Frank and my daughter Rebecca

ACKNOWLEDGEMENTS

In retrospect of my past seven years as a graduate student at Clemson, I owe gratitude to the following mentors and friends.

My thanks first go to Adam Hoover, for his guidance and advice for me to go through my research on computer vision and the related topics. Adam has brought me to the area of computer vision, and taught me the skills and knowledge to work in this field. I also owe him all the financial support that enables me to finish my Ph.D. study at Clemson University. I forever owe him the ability he has developed in me to be an independent researcher.

My thanks also go to Eric Muth, who has served on my committee through these years. Eric has shared many thoughts with me on my research from his psychological perspective, which are very interesting and stimulating. Also I feel thankful for all the cooperation I have received from him and his group. Thanks for Adam and Eric jointly securing the research funding that enables my research to finish.

My thanks go to Stan Birchfield for his strenuous work on my proposal, dissertation and serving on my committee. I owes thanks to him for giving valuable opinion on my research proposal, and reviewing my dissertation. Also thank him for sharing his unique thoughts on computer vision research.

My thanks go to Ian Walker who has faithfully served on my committee through these years. Ian has been an excellent collaborator with our research group, and has contributed so much to our joint research.

To my lab friends at this computer vision and data fusion lab, Reetal Pai, Niaz Abdul, Lee Brandon, Joseph Mathews, Joshua Hughes, Jeromie Rand, Kelly Waller, Tom Epton, Karsten Lowe, Marty Werner, Anirudh Bhupender, Danyan Ganjali and Yujie Dong for

their view and support on my work and constructive feedback in a most friendly and kind manner.

To the friends at the Psychology Department lab, Mandy Elkins, Stephanie Fishel, Jennifer Pappas, Alex Walker and Jason Moss for their wonderful support to our collaborative research.

Thanks to the Electrical and Computer Engineering Department of Clemson University for accepting me into the graduate program and offering all the resources available for me to complete my study and research and gaining an invaluable experience.

Finally I would like to express love and gratitude to my family. Thanks for my beloved wife, Yan Liu, for her understanding and support to my pursuits. She has undertaken the demanding job of taking care of our two children, Frank and Rebecca, so that I can continue my research. Without her sacrifice, this work would be impossible. She has also made progress in her own career path in this country despite the tedious housework. I also thank my parents for bringing up and preparing me for the study and research I have to undertake. And also thank them for coming to this country to help with our child care. I owe forever to my family.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Overview

This dissertation considers the problem of stability analysis of the parameter space of algorithms for data analysis. We are interested in problems where the analysis should arrive at a count of things within the data. For example, the segmentation of an image produces a number of regions identifying objects or areas of interest in the image. In a sense, the number of regions (the count) is part of the output. As another example, an analysis of a database could identify a specific number of trends or patterns within the data. Again, the number of trends (the count) is part of the output. As yet another example, an analysis of a signal could quantify the number of transitions it undergoes. All of these problems are similar, in that part of the goal of the automated data analysis is to identify a unique count of things in the output. We graphically show the idea of "N" count for a few examples in Figure 1.1.

All algorithms operate using parameters, such as thresholds, limits, windows, and iteration controls. As the values of these parameters are changed, an algorithm will produce different results, and hence a different output count. We define a parameter space as all possible values for all the parameters of a given algorithm. We hypothesize that within a

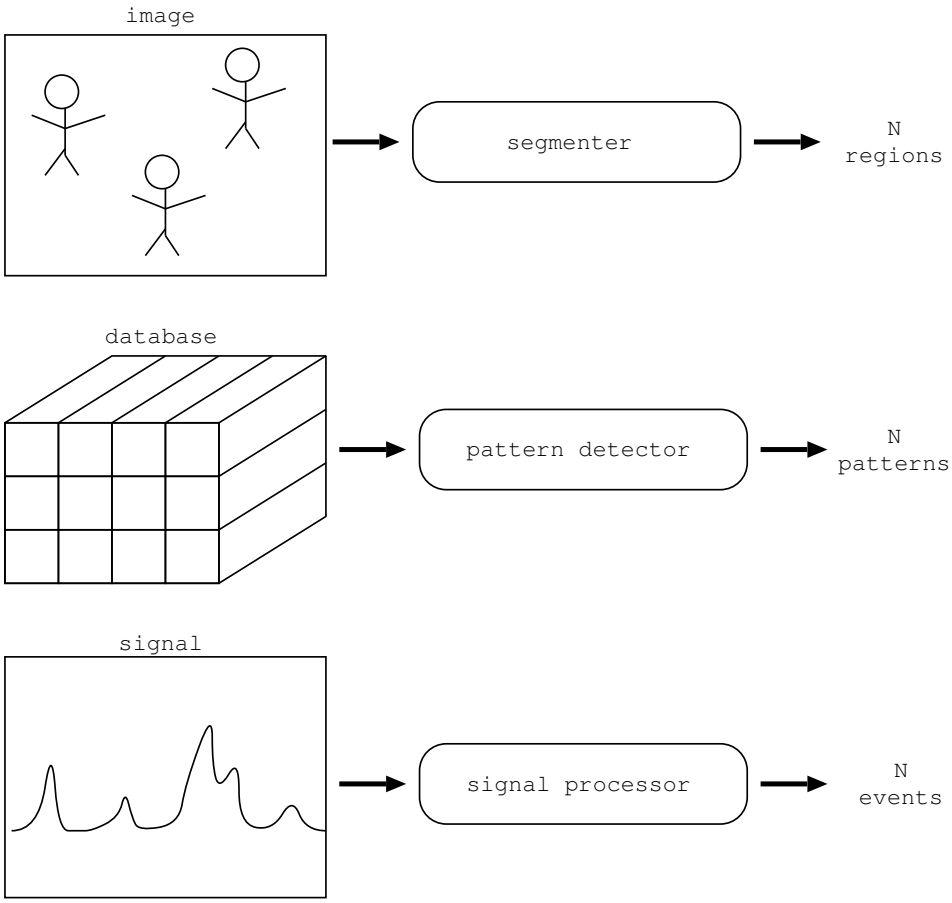


Figure 1.1: The examples of “N” count of things

parameter space, there should reside an area of stable output count. The area of stable output count should identify the best values for the parameters to use in processing the data. The shape of the stable area may identify which parameters are more and less sensitive for the given data. If the output count is stable for all values for a given parameter, this could indicate that the given parameter is useless. If there is no stable area within the entire parameter space, this could indicate that the algorithm is not suitable for processing the given data.

Our approach may be considered a brute force search for algorithms, parameters, and parameter values, to process a given data set. For example, consider the image segmentation problem. Given an image, one can consider processing it with every segmentation algorithm known to man. For each algorithm, we could try all possible parameter combinations. Counting up the regions produced in each segmentation, we would create a region count space. We could then search for the stability of region count within this space to identify the best algorithm and parameter values to use to segment that image. Searching further, we may find that multiple algorithms agree on region count, and therefore produce a larger area of stability in the region count space, and therefore provide greater confidence in the result.

One could argue that a brute force approach like we describe is completely impractical, due to the required processing time and computational complexity of the approach. However, one can point to recent successes in brute force approaches to solving problems that were at one time considered impossible. In the game of chess, a computer was first able to defeat the human champion of the world by employing a brute force approach to move analysis [13]. That same group is now studying the game of go, which is many orders of magnitude more complicated than chess, and believes that a brute force approach will eventually triumph [14].

Although these are solutions to games, they model problems that may be similar to other more daunting problems such as image segmentation. Given the historical trend

of computational power increasing each year ("Murphy's Law"), it seems reasonable to suppose what could be done if the brute force methodology were applied to algorithm and parameter selection for data analysis.

In this dissertation, we largely focus the development of our approach on images and the image segmentation problem. Two of our three experiments involve segmenting different types of images to identify a count of objects of interest. However, it is important to note that our methods are applicable to any sort of data. For one of our experiments, we apply our methods to the analysis of a 1-D signal. We would also like to point out that while the segmentation problem is being used in our experiments, we are not ultimately trying to build the "optimal segmenter". We assume we are provided with a segmenter (or any algorithm), and are interested in studying the stability of its output in its parameter space, and how that relates to its performance.

1.2 Related Work

From a high level, we can contrast our approach with several other methodologies. Perhaps the most closely related is clustering. Clustering methods typically separate and combine data until a criterion is reached. The result is that a specific count of clusters is identified. This can be thought of as a localized, gradient search through a parameter space. Instead of trying all possible combinations of parameters, and searching through the resulting region count space, the parameter space is traversed according to a criterion which quantifies the similarity of clusters. Although an output count is identified in the end, this is different from a brute force approach to identifying that count. We discuss clustering methods more in Section 1.5.

Another related methodology is scale-space analysis. In this framework, an image is processed with different parameter values that vary the scale (size) of the operations. The persistence of an image feature across multiple scales is measured, and used to derive the

final result. In this methodology, the parameter space is again being searched somewhat locally, rather than brute force. In addition, these methods use a data persistence measurement rather than count stability to identify the final result. We discuss scale-space methods more in Section 1.6.

Other researchers have looked at automated parameter selection. One of the most commonly applied areas is automated threshold selection in images. Another important application is edge detection in the existence of noise. The rest can be listed as application specific implementation of automated parameter selection in solving different kinds of practical, real-world problems.

People use this approach to remove the noise in the histogram of images and segment the images to get the best result. In this paper [40], a typical automated threshold selection method was developed for blood vessel segmentation. Two moving-window methods, either flat or Gaussian weighted windows, for local thresholding with robust automatic threshold selection were developed to segment blood vessels in 3-D angiograms. In forming the automatic thresholding scheme, both grey level intensity and edge strength were used together. The method was found to be robust to noise and modest in computation cost. The results show that this method can segment complex images with filamentous structures at relatively low computation cost. The automated threshold selection in this method is typical among this kind of application.

The other most commonly applied area is the edge detection with intrinsic noise. In edge detection, this approach is often used together with scale-space and wavelet to find the best parameters for generating the edges. In the paper [42], an optimal filter scale based upon edge detection theory and an automatic threshold to eliminate the ragged edge were developed to form a fast edge detecting algorithm. A Gaussian scale-space is used to form the optimal filter scale in detecting the specific edge. The method was applied in the X-ray cephalometric images to detect the edges of soft and bone tissues. The experimental results show that the method can achieve either the noise removing effect of low-resolution

filter or the edge detecting precision of high-resolution filter and make a better compromise between the precision of edge detection and the effect of noise removal. In this case the automated threshold detection is used in the edge detection to remove noise, and this is the case like most other automated threshold selection used in edge detection.

There are many other application specific situations where automated parameter selection is used. In this paper [15], automatic parameter selection and color histograms were used to track objects in video surveillance. The automatic determination of the number of bins in the color histograms was made possible in this work compared with arbitrary selection of such a parameter and it has achieved good result. The method has been applied to sequences of face tracking, car tracking and aerial video, and it demonstrated the robustness in tracking in different applications. In another paper [26], an automatic threshold selection method was developed in subblock matching-based conditional motion estimation for video compression. Block matching is a time consuming part in the encoding process, and a threshold automatically determined through an iterative algorithm was developed to differentiate an active block and an inactive block so that only active blocks are subjected to motion estimation. Experimental result shows that this method provides better tradeoff between computation and rate-distortion performance than some other contemporary methods. We only give above two examples about how the automatic parameter selection can be applied to solving different problems. Many more applications can be found for automatic parameter selection.

Our work differs from all of those in that we are not tied to any specific application or parameter type. We search globally in the parameter space, while the other methods search locally in the respective space. We demonstrate our brute-force method and other methods in Figure 1.2. We search in the parameter space of any algorithm with any range of parameters for the best values for that parameter set to operate best in that particular algorithm.

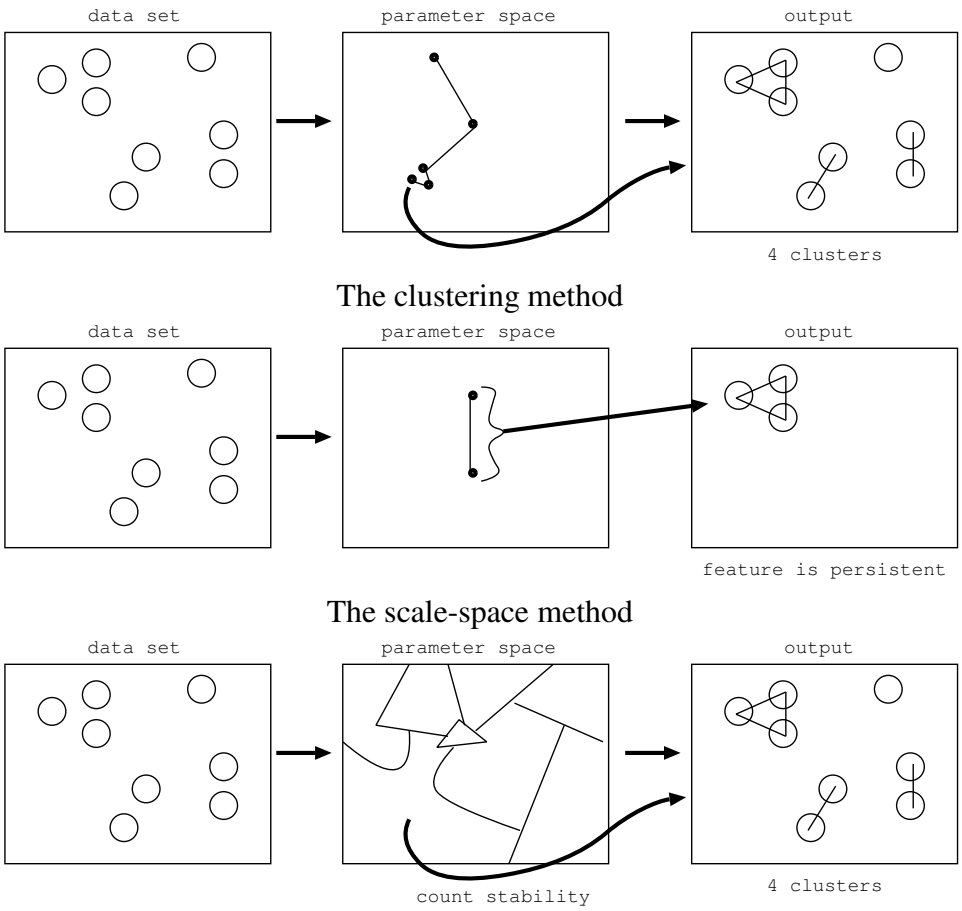
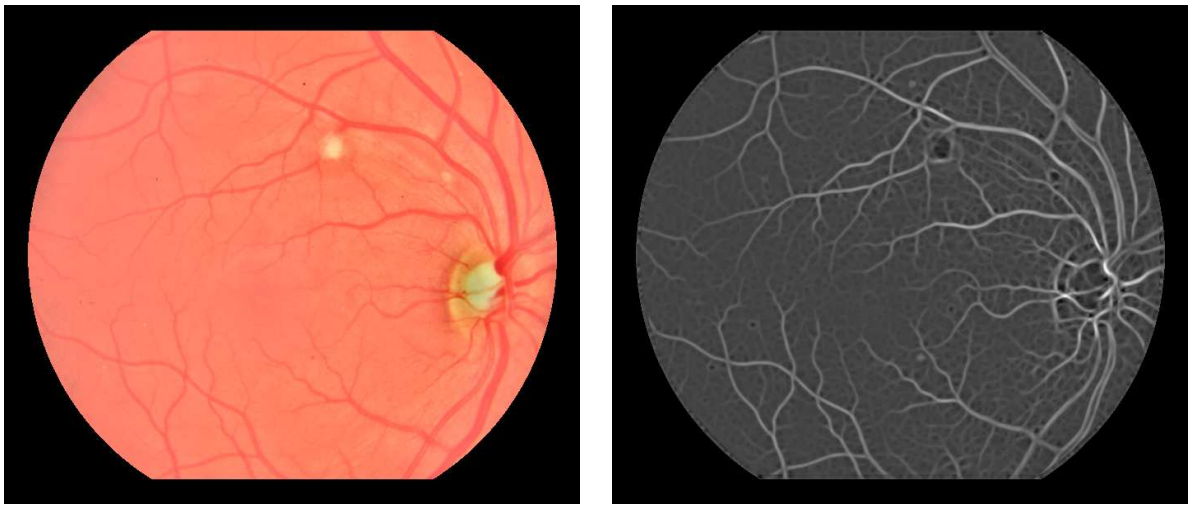


Figure 1.2: Our brute-force search method and other methods



original retinal image blood vessels highlighted

Figure 1.3: A retinal image with blood vessels highlighted

1.3 Computational models

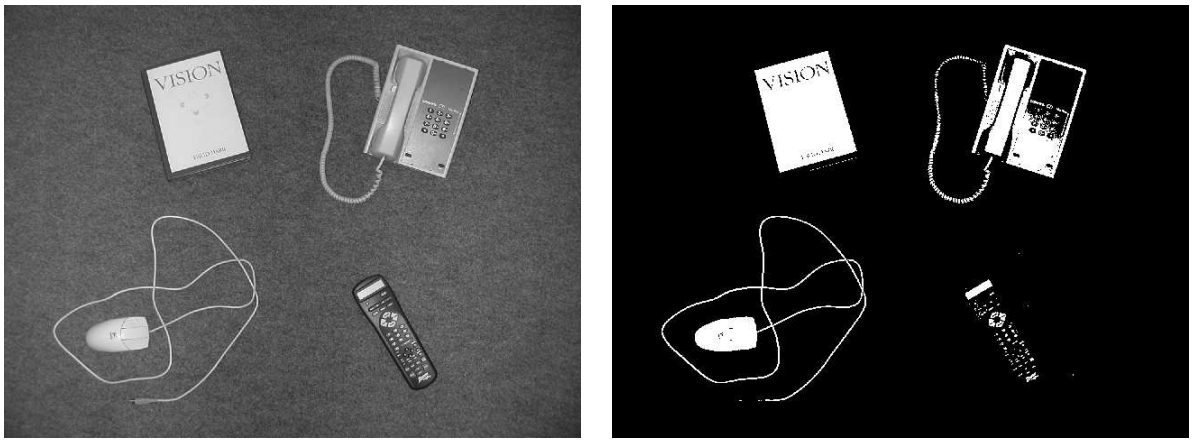
One of the ultimate goals of computer vision is object recognition, the capability to recognize the contents of images. It is presumed that this enables machines to perform visual tasks as people desire on behalf of or in support of human beings.

Figure 1.3 shows a retinal image of the human eye and an image of the eye with blood vessels highlighted through computer vision algorithms. The enhanced image therefore facilitates doctors in diagnosing the symptoms of hemorrhages in the patients' eyes [10].

To recognize an object in a computer-assisted visual task, there are three styles of computational approaches,

- model-based top-down approach,
- data-based bottom-up approach, and
- combinations of the above two approaches.

These three paradigms can all be employed to complete the perceptual process and achieve object recognition.



original image image with objects segmented

Figure 1.4: Bottom-up image processing

The bottom-up approach deals basically with the fundamental components of an image such as edges, blobs, regions, segmentations and illuminations. In other words, the bottom-up approach starts from the image, of which there is no a priori knowledge, through processing all pixels equally to find candidate pixels, regions or features, and therefore extract meaningful information to achieve high-level image understanding. Figure 1.4 shows an image and some meaningful objects extracted from the image in a bottom-up framework.

The top-down approach becomes an appropriate implementation when a known object has to be recognized in an image. In this paradigm, features can be extracted from the image and compared to those stored in a database to find a match. The processing of pixels is guided by known data, data model, template or process. Therefore, image understanding starts from a known feature and then continues through finding similarities between that particular feature and the information in an image with multiple features. Figure 1.5 shows an image of an individual's face recognized from an image with many individuals' faces together in a top-down framework.

David Marr was a pioneer in the bottom-up investigation of images. In his work, Marr modeled the vision process as an information processing task in which the visual information undergoes different hierarchical transformations at and between levels, generating representations which successively make three-dimensional features more explicit [22], as

target face image with many faces face recognized

Figure 1.5: Top-down image processing in face recognition

summarized by [38]. He specified three levels that any information-processing task must understand, the computational theory level which deals with the goal of computation and the logic of strategy to achieve the goal, the representation and algorithm level which deals with the input and output structure and algorithm for the transformation to implement the computational theory, and the hardware level to realize the representation and algorithm physically. He also outlined the three-stage process for human vision: from the original 2-D image to primal search to 2.5-D sketch to 3-D model representation. Primal search gets raw local properties such as zero-crossings and blobs, the 2.5-D sketch makes explicit the orientation and rough depth of surfaces and contours of discontinuities in a viewer-centered coordinate frame, and 3-D model describes shapes and their spatial organization in an object-centered coordinate frame through a modular hierarchical representation that includes volumetric space and surface primitives for ultimate object recognition.

Rutishauser et al. [28] have developed a bottom-up saliency-based region selection attention system for object recognition. They have found that the bottom-up attention performs better in three domains, learning and recognition in highly cluttered scenes, learning and recognition when several objects are in the same image, and online learning of landmarks suitable for robot navigation.

Jones et al. [16] have presented a simple computational model to implement a top-down, object-class-specific and example-based approach. The model has been used to test on edge-detection and view-prediction for three-dimensional objects and found to be consistent with human perceptual expectations. The model is also performing better on sensor noise and incomplete input image information than the conventional bottom-up strategies, and therefore in support of the hypothesis that human visual system may learn to perform low-level perceptual tasks in a top-down fashion. But only human faces are used in the paper as the example images, which makes the application scope of such a model limited.

Borenstein, Sharon and Ullman [3] have developed a scheme to combine bottom-up and top-down approaches into a single figure-ground segmentation process. The top-down ap-

proach uses object representation learned from examples to detect an object in a given input image and provide an approximation to its figure-ground segmentation while the bottom-up approach uses image-based criteria to define coherent groups of pixels that are likely to belong to either the figure or the background part. A global cost function is constructed to reflect the top-down and bottom-up requirements and a global minimum of such a cost function can be efficiently found by applying the sum-product algorithm. Khadhouri and Demiris [17] have also developed a scheme to combine the saliency of top-down elements and bottom-up components to construct a visual attention mechanism. Bottom-up part is used to initialize the top-down part so that a limited computational resources are needed to compute a selection of behaviors generated. The system has been implemented on a robot to examine its performance during the observation of object-directed human actions.

The bottom-up and top-down approaches have their respective advantages and disadvantages. The bottom-up approach can better overcome the local noise in the image background and provide robust low-level features for high-level analysis. The bottom-up approach is the only option in a scenario when you have no clue what you are looking for at all. However the bottom-up data-driven approach does have the problem of not handling very well some high-level features such as occlusions compared with its top-down counterpart. On the other hand, although a top-down approach can perform well with high-level model information, it may deteriorate and degrade dramatically when the image background is noisy with inherent clutter.

Although some problems can be solved purely top-down or bottom-up, previous research and experience show that neither top-down nor bottom-up strategies can explain the vision process or solve complex vision problems alone. So many researchers believe the appropriate combination of the top-down and bottom-up approaches may yield a more flexible and powerful vision solution than just use each of the approach individually.

One important link that lies between low-level image processing and high-level image understanding is the mid-level image processing. Perceptual organization, currently under

intense research, plays the role of a middle platform connecting low-level structures and high-level recognition. Scale-space is another important image processing technique fitting into most computer vision frameworks. These two techniques will be explained in the next two sections respectively. In the last section, we are giving an overview of our method, the region count stability in the scale-space. Our approach is a mid-level scale-space technique originating from low-level processing, similar to that of perceptual organization.

1.4 Perceptual organization

Perceptual organization is the technique to group low-level features that are probably coming from one object into an intermediate framework for further vision processing. These visual primitives are grouped and organized by basic geometric relations such as proximity, parallelism, continuation, symmetry, similarity, closure, common region and object-background separation to become more perceptually significant data structures. The grouping principles are supported by psychological laws such as Gestalt theory.

Compared with the large amount of research work done both in top-down and bottom-up approaches, relatively a small quantity of research is completed in perceptual organization, between the low-level feature extraction and the high-level object detection.

Engbers and Smeulders [6] have proposed six considerations for the design of generic grouping in computer vision: proper definition, invariance, multiple interpretations, multiple solutions, simplicity and robustness. Although these principles are generic, they have more theoretical importance than practical significance.

Yu [45] has developed a perceptual organization scheme to combine grouping cues, figure-ground cues and depth order cues in one process. The scheme also combine top-down and bottom-up information in a single grouping process. The whole process is an interactive feedback process between segmentation, grouping, figure-ground and recognition. Object segmentation is eventually achieved with cues from spatial and object attention.

Sarkar and Soundararajan [29] have developed a flexible learning perceptual organization framework based on graph partitioning. The grouping process is able to form large groups from a small number of salient relationships such as parallelism, continuity, common regions and perpendicularity. The robust performance of the grouping and learning frameworks has been statistically demonstrated on a variety of real images. It is concluded that large salient groups can be formed from a set of local relations defined over a small number of primitives.

Tu et al. [37] have proposed a general framework to parse images into regions and objects. In this framework, the detection and recognition of objects proceed simultaneously with image segmentation in a comparative and cooperative manner. Bottom-up proposals are combined with top-down generative models using Data Driven Markov Chain Monte Carlo algorithm to guarantee convergence. However, the objects to be recognized are restricted to human face and text in a scene.

Zhu [47] tries to build up perceptual organization framework through studying the visual patterns of images. In this paper, the issue of conceptualizing visual patterns and their components (vocabularies) is addressed through statistics mechanism. This paper has provided an interesting idea to understand and process images in a mostly statistical perspective.

Perceptual organization is the link between low-level segmentation and high-level detection and recognition. Based on the nature of the technique and the research work already finished in this area, it is evident that perceptual organization has the potential to incorporate the top-down and bottom-up approaches into one uniform framework and therefore overcome the deficiencies of individual approaches, and eventually enable figure-ground separation, object recognition, scene reconstruction, image or video change detection, spatio-temporal grouping, and many other areas in computer vision.

1.5 Clustering methods

Many methods are used in image segmentation such as histogram-based thresholding, edge detection and region growing. Clustering methods are an approach one of the methods used in image segmentation. K-means is a typical clustering algorithm that can be used to segment images. In this algorithm, K cluster centers are picked randomly or heuristically first, each pixel in the image is assigned to the cluster so that the variance between the pixel and cluster center of the cluster is minimal, and then the pixel membership and the cluster centers are repeatedly computed iteratively until a convergence criterion is reached. In this way, an image can be segmented into a given number of regions.

One of the representative work [32] using clustering method to segment images is the normalized cut. In their work, Shi and Malik treats image segmentation problem as a graph partitioning problem. The normalized cut criterion developed measures both the total dissimilarity between the different groups as well as the total similarity within the group so that a "fair" 2-way cut can be implemented recursively until a maximum number is reached or a stopping criterion is hit in the iterative process. The method uses eigenvectors to bipartition the graph. The method also explores the simultaneous k-way cut with multiple eigenvectors. The normalized cut has really improved from the original K-means clustering method.

The clustering method is guaranteed to converge but does not guarantee the optimal solution because it only executes a local search rather a global search in the parameter space. The result of clustering method depends greatly on the initial guess of the cluster centers or the preassigned number of cluster K. In contrast, our stable region approach does not need a predefined number of regions. More importantly, we carry out a rigorous global search in the parameter space to locate the best parameter set and values for the image processing operation. From another perspective, our method is purely a bottom-up approach whereas the clustering method is a top-down approach.

1.6 Scale space

The concept of scale-space was first introduced by Iijima more than 40 years ago and became popular later in the 1970s and 1980s by the work of Witkin and Koenderink.

A scale-space is created when convolving an original image with a linear filter (such as Gaussian spatial filter or wavelet transform), or processing an original image with a nonlinear filter (such as a morphological operator), or by any other means (such as pyramid, quadtree, statistics etc.). According to Lindeberg [18], real-world objects appear and exist in different ways depending on the scale of observation, which shows the importance of scale in image processing and understanding. The scale-space is most necessary and useful, and sometimes indispensable when information of an original image cannot be derived and analyzed at the original scale and only a multi-scale structure is capable of extracting the necessary features from it. Since the idea of scale-space can be conveniently applied to any measurement in which the measurement depends on a parameter or a set of parameters that can be varied, many varieties of scale-space techniques can be developed by researchers with their special perspectives to solve various kinds of image processing problems.

Table 1.1 lists some important representative research work in scale-space theory in recent years. In the table, the first column lists the author and citation, the second column lists the scale-space kernel used in the work, which actually describes the parameter that is created and varied across multiple scales, and the third column lists the result, which is the parameter that is measured across different scales.

The most common scale-space kernel is a Gaussian filter used for smoothing. The parameter varied is the width of the Gaussian, which affects the amount of smoothing. Lindeburg [18] uses a Gaussian filter scale-space to segment blobs, measuring how well they correlate across different scales. He defines the terms significance and saliency of a blob to describe the correlation, and he uses the blob volume to quantify significance and saliency which refers to the regions of perceptual importance in an image. Hadjidemetriou, Grossburg and Nayar [12] have used Gaussian filtering to create a scale-space of histograms and

Table 1.1: Summary of Scale-space Techniques

Citation	Scale-space Kernel	Scale-space Parameter
Lindeberg [18]	Gaussian Spatial Filter	Significance and saliency measure in terms of blob life-time and intensity.
Hadjidemetriou, Grossberg and Nayar [12]	Gaussian Spatial Filter	Intensity Histogram.
Gauch [7]	Gaussian Spatial Filter	Intensity Watershed Hierarchies.
Ho and Gerig[8]	Laplace Scale-Space from Gaussian Spatial Filter	A statistical model based on training a 1-D scale-space system which preserves the across-boundary features but blurs the along-boundary features in the resultant Laplace scale-space.
Setarehdan and Soraghan[30]	Wavelet Transform	HFT is a scale-space edge detection technique using wavelet transform, which is used for endocardial and epicardial boundary estimation. B-spline approximation method is used to define the closed LV boundaries.
Wang et al. [39]	Wavelet Transform via Quadtree	Sharply focused objects in a low DOF image are detected by using wavelet frequency analysis and statistical methods. Average Intensity of Image Block and Variance of Wavelet Coefficients in the High Frequency Bands are evaluated to generate segmentation in a quadtree scale-space framework.
Acton and Mukherjee[1]	Area Open-Close and Area Close-Open Operators	Intensity of pixels in the scale-space is formed into a vector, and Fuzzy C-means clustering technique is used to group scale-space vectors based on a similarity measure, which leads to a classification.
Chen and Wang[4]	Product of Gaussian Spatial Filter and the Inverse Function of the Absolute Value of the Difference between Neighboring Pixels	Scale-space edge-preserving filter is used to protect edges as well as remove noise and can be applied to constructing further scale-space system.
Wilson and Li[41]	Quadtree Filtering	Multi-resolution formed by quadtree and Markov Random Fields are combined to analyse and describe image structures statistically, leading to segmentation of certain types of images.
Yu and Hoover[43]	Threshold, Region Growing and Gaussian Smoothing	Stability of the number of regions in the scale-space formed by thresholding and region-growing.
David G. Lowe[20]	Difference-of-Gaussian function	Standard deviation of Gaussian function σ

developed a matching algorithm making use of differences between histograms of consecutive image resolutions for image recognition. Gauch [7] has used Gaussian filtering and constructed a multi-scale watershed hierarchy to achieve image segmentation and object recognition. In the research, watersheds are computed by identifying the local minima, and drainage directions for each pixel are identified by calculating the image gradient. Partitions of image by watershed is achieved by marking the locations of intensity minima and associate every pixel with a local minimum. Multi-scale hierarchy is imposed on watershed regions by linking the paths of intensity extrema in the Gaussian scale-space, and also associate scale with watershed boundaries. Interactive or automatic image segmentation achieved via gradient watershed hierarchy in Gauch's scheme. Derivative of Gaussian scale-space has also been used by some people. Ho and Gerig [8] have utilized Gaussian spatial filter to construct a Laplace scale-space and train the scale-space model to achieve segmentation through edge protection. In the research, image profiles are completed by sampling in a coordinate system relative to the object boundary. A 1-D scale-space is constructed on these profiles which preserves the across-boundary features but blurs the along-boundary features in the resultant Laplace scale-space. A statistical model is built based on training the scale-space model which can be incorporated into a Bayesian segmentation framework.

Wavelet transform is another popular scale-space technique used in the research. Setarehdan and Soraghan [30] have used wavelet transform to construct a edge detection scale-space to denoise and protect boundaries of a special group of medical images. In their research, HFT(Hibrid Fuzzy Temporal)-FMED(Fuzzy Multi-scale Edge Detection and Tracking) is a scale-space edge detection technique using wavelet transform, which is used for endocardial and epicardial boundary estimation. It is based on the intensity and motion information of a moving edge in a fuzzy-based framework. The *a priori* information on the temporal, spatial and frequency domain properties of the boundaries are used in wavelet transform to denoise the boundaries. Finally, a uniform cubic B-spline approximation

method is used to define the closed LV boundaries. Wang et al. [39] have also built a wavelet and quadtree scale-space framework to generate segmentation of sharply focused objects in low depth-of-field images through a statistical approach. Sharply focused objects in a low DOF image are detected by using wavelet frequency analysis and statistical methods. Average Intensity of image block and variance of wavelet coefficients in the high frequency bands are evaluated to generate segmentation in a quadtree scale-space framework.

Morphological operator has also been investigated by researchers in recent years. Acton and Mukherjee [1] use morphological operators to construct a scale-space and then apply fuzzy C-means to grouping each pixel by its vector in the scale-space. Many other scale-space approaches have been investigated. Chen and Wang [4] have developed a special filter (product of Gaussian filter and inverse function of absolute difference value of neighboring pixels) to construct a scale-space to protect edge and remove noise. Wilson and Li [41] have used quadtree filtering to build a scale-space together with Markov Random Fields to analyse and describe image structure statistically and achieve segmentation.

Many other ideas in scale-space have been proposed, but are within the types we have discussed above.

1.7 Overview of our approach

We are trying to develop a computational model for perceptual processing in computer vision through the application of scale-space technique. We are coming up with a unique approach based on the psychological idea of “stable perception”, which characterizes the count of objects perceived as the stable view of an image.

Figure 1.6 shows an image containing coins sitting on a piece of plain cloth. If a casual viewer were asked to describe what the image contains, a likely answer would be “some coins”. If the viewer were asked to make a specific count of “things of interest” in the

image, the answer would likely be "16". The question we are trying to address is how that specific count of objects could be used to drive the visual processing of the image. We seek to use the search for a "stable count" of segmented regions to drive automated image processing.

For example, the coins in Figure 1.6 readily stand out for several reasons. First, they are substantially brighter than the cloth background. Second, they are roughly the same size and shape. Third, they are organized into a regular pattern, in this case a 4×4 grid. Fourth, there is nothing else in the image to draw focus of attention away from seeing the coins. Some or all of these properties cause the viewer to quickly and easily see a segmentation of the image in terms of the 16 coins.

In terms of image processing, one could apply a simple thresholding algorithm to segment this image. A wide range of threshold values would all yield roughly the same segmentation. In each of these segmentations, the count of segmented regions would always be 16. If we could automatically find this "stable region count" across the range of segmentations, it would provide a positive indicator that (a) simple thresholding is a good choice of algorithm to segment this image, and (b) a good value for the threshold parameter is somewhere within the range that produces the "stable region count".

Other image processing algorithms are also likely good candidates for producing a good segmentation for this simple image. For example, a circular template detector or a Hough transform would also likely work fairly well. Each of those algorithms has its own parameters, and each parameter has a possible range of values. We would seek ranges of parameter values over which the final region count in the segmentations remains constant.

Figure 1.7 shows another image with a scenery of yellow plants, dark green trees, patches of white cloud and blue sky. This image is not as definitive as the above coins image in terms of region count, but a rough number of regions are still identifiable. In other words, although we cannot find an exact number of regions as the number of 16 in the coins image, we can still get a fairly close range of numbers that most viewers agree upon. When

Figure 1.6: A coins image

looking at such an image, a viewer should find the four distinct objects in the image, but maybe five or more regions are detected as the patches of cloud may represent more than one object. Therefore we expect to find in the scale-space at least four objects, but maybe five or even more in the stable view.

Looking deeper into the mechanism to segment such an image, there are three reasons worth attention. First, the different objects (clouds, trees, plants and sky) are in different shapes and patterns from each other. Second, the objects in the image are of different colors from each other. Third, the different objects are fairly uniform in color and intensity in itself but different in these properties from others. So an algorithm differentiating in color or intensity will probably provide a good segmentation of this image. If a proper algorithm is selected to apply to the image, with appropriate parameters determined and appropriate range of values found, where there is a stable number of regions counted in the parameter scale-space, a good segmentation of the image will be yielded through the implementation of such an appropriate scheme.

Figure 1.7: A natural scene image

This is a low-level to mid-level approach for image segmentation, which also leads to high-level recognition. We are looking for a stable "N object count" in an image by subjecting the image in a range of a parameter or a set of parameters (in their scale-space) to extract the number of regions (objects) in the image and locate the parameter range where there is the least change in the region count, and this area indicates psychologically the meaningful objects in an image. We capture this idea using a scale-space formulation, and detect "stable" segmentations as local minima in the scale-space. This work was originally motivated by the problem of detecting some types of common lesions in retinal images (many lesions appear to be abnormally bright areas) [43].

This is a bottom-up approach in parallel with and similar to perceptual organization. Since the bottom-up approach deals primarily with the basic features of images, which are the low-level data structures and the building blocks of images, starting from the bottom-up approach will eventually lead to the correct understanding of images.

We explore building up the scale-space by expanding the number of degree-of-freedom from one dimension to two or even higher dimensions, and the parameter from just a basic

threshold to others such as hysteresis threshold, intensity mean and standard deviations and other local structural and statistical properties.

Our scheme should be considered as a low-level to mid-level method for the fact that the parameters used in the scale-space, such as thresholding, segmentation, primitive statistics etc., are primarily rudimentary image data structures, meanwhile the object count is really an important aspect of stable perception which has some bearing on the human cognition, and therefore is a high-level mechanism or at least has some high-level connection. By placing any image into such a parameter scale-space, insightful information can be derived from the image and provided for high-level object recognition in computer vision.

We try to investigate how this idea works in several 1-D and 2-D scale-spaces with algorithms such as thresholding and region merging, hysteresis thresholding, variance-based region growing. We try to get the best figure-ground separation by looking for the stability of region count in the scale-space.

Chapter 2

Methods

We consider the problem where an image is to be segmented into a number of regions, where each region indicates an object or area of interest. A number of algorithms are available to perform the segmentation. Each algorithm has a different set of parameters that control its operation. Each parameter has a range of values. Changing the values of the parameters of an algorithm, for a given input image, produces a different result. For each result we count up the number of regions segmented. This analysis produces a "region count space", which we call an "R-space" for brevity. The R-space can be analyzed to locate areas of "stability", where the number of regions segmented does not change in the local area. We propose that stable areas in the R-space point to good segmentations, and hence appropriate algorithms, parameters, and parameter values.

In this chapter we formalize the concept of an R-space. We then demonstrate the computation of an R-space using three different algorithms on a handful of images. We then discuss the computation of stability in an R-space, from a general point of view. We have found that the computation of stability depends to some degree upon the specific segmentation problem being considered. Varying factors can be useful during stability computation, so that the analysis can be tuned to the problem at hand. We demonstrate several of these

factors, and show examples. Finally, we discuss our framework from a general point of view, before proceeding to specific case problems and experiments in the next chapter.

2.1 Region count space (R-space)

An image can be modeled as an intensity function **I(x)**. This intensity function **I(x)** is defined in **x**, which is in a 2-dimensional spatial domain **D**. Sometimes images can represent a 3-dimensional scene of the real world, but the image itself is still in a 2-dimensional area. In this research, the algorithms are designed based upon 2-dimensional image data but can be extended to any higher dimensional data. Therefore we use **I** to symbolize an image. This image data can be replaced by any other data of higher dimensions in terms of the application of the algorithms.

The algorithms are symbolized using $\mathbf{A_j}$ where $\mathbf{j} = \mathbf{1}...\mathbf{J}$. Since a countless number of algorithms can be developed in this general framework, **J** is any integer number. For each algorithm, there are one or more parameters which control the operation of the algorithm. These parameters include the properties (such as threshold, growth and variance) to be measured and a measurement range of these properties (determined by the maximum and minimum values of these measurements). Sometimes objects in an image are in a range of a certain reasonable guess, for example, they cannot be zero or above a thousand, so we often use the feasible range to replace the maximum range to save computation time. We symbolize the parameter sets as $\mathbf{P_k}(\mathbf{A_j})$ where $\mathbf{k} = \mathbf{1}...\mathbf{K_j}$, and $\mathbf{K_j}$ is an integer. We use $\mathbf{A_j}$ in the symbolization of **P** because parameter sets are dependent on the algorithms; that is, every algorithm is composed of a specific set of parameters. The selection of parameter sets is a major component of design in an algorithm development. The parameters are characterized by their values and ranges. We use $\mathbf{V}(\mathbf{P_k})$ to symbolize the values of any parameter. As the parameter values are dependent on the parameter sets selected, we denote the parameter value **V** as a function of parameter set **P**.

After processing with an algorithm, a segmentation of the image is derived and we count the number of regions in the segmented image. So our methods consider a transform from the image space $\mathbf{I} \subset \mathbf{D}$ to a region count space $\mathbf{R} \subset \phi$. The processing steps are summarized as follows,

$$\mathbf{I} \subset \mathbf{D} \rightarrow \mathbf{I} \times \{\mathbf{A_j}, \mathbf{P_k}, \mathbf{V}\} \rightarrow \mathbf{R} \subset \phi \tag{2.1}$$

The whole method is actually a transformation from the image space **D** to the region count space ϕ. The transformed region function **R** is in a region count space ϕ determined by image **I**, algorithm **A**, parameter set **P** and parameter values **V**. **A** will determine if an algorithm is appropriate for an image **I**; **P** will determine if a parameter is important for **A** or **I**; **V** will eventually determine the final range of parameters that lead to strong and meaningful segmentation from the original image **I**. So the methods we are developing are actually transformations from the spatial domain of **D** to the parameter domain of ϕ through a series of intermediate steps of **A**, **P** and **V**, for the given **I**. We put **A**, **P** and **V** in a set symbol $\{\ \}$ because of their inter-dependencies.

All image processing methods have parameters and variables that control their operation. If an algorithm is well-suited to process an image, and the image contains an obvious "stable count" of objects of interest, then there should be a wide range of parameter values for that algorithm that all produce the same region count.

The region count space ϕ is a space composed of the region count of all segmented images over all kinds of parameter values in all algorithms. In other words, ϕ-space is composed of all kinds of individual R-spaces. We speculate that if any image is subjected into such a space, with all algorithms tested on it, all parameters tried on it and all value ranges checked out, there should be a stable region count area somewhere in there in the space of ϕ. This stable region count area will yield valuable information for the further analysis of that particular image. An example (simulated) 1-dimensional stable region count space is shown in Figure 2.1.

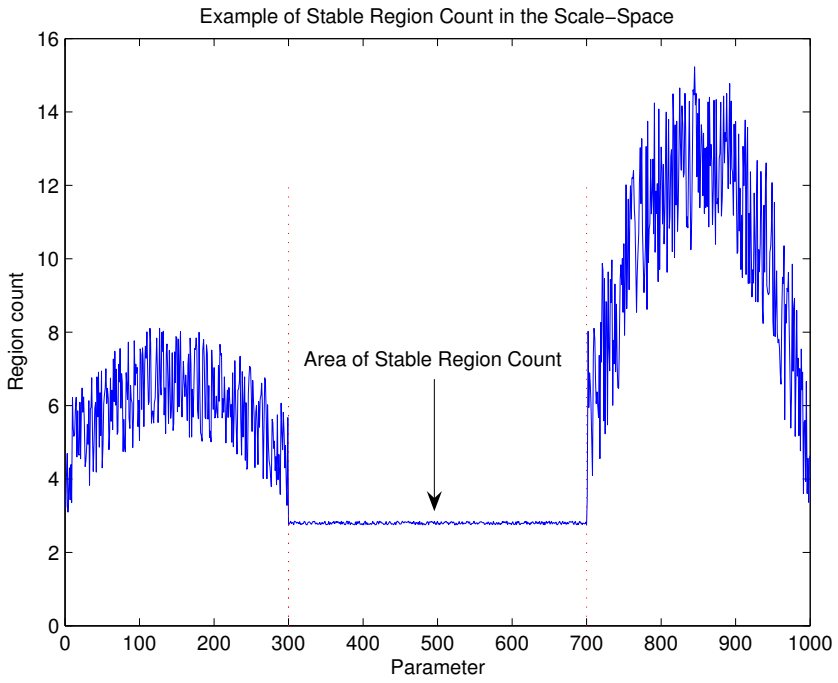


Figure 2.1: An example of a stable region count in a parameter-space.

By this means, we transform an image from its original domain to a multi-dimensional scale-space region domain and investigate the stability of region count in such a domain. We propose that the parameters at the areas with stable region count in the scale-space will provide a meaningful view of the original image.

2.2 Example R-spaces

In this section we demonstrate the construction of an R-space (region count space). To construct the R-space, we use parameter sets and algorithms. We have tentatively selected a few algorithms and related parameters to demonstrate how to construct the R-spaces. The algorithms selected are thresholding and region merging, hysteresis thresholding and variance-based region growing .

We select three parameter sets to form three algorithms, and in each parameter set, we focus our attention to two dimensional R-space, so we have two parameters in a set. For

each parameter value, we use integer values for the parameters to build up the R-space. The details of the methods and ranges of parameter values are given in the next sections.

Through the mathematical theory introduced in the previous section, we can create a R-space such that the region count **R** is a function of any pairs of parameters such as threshold, variance, growth or any others deemed as appropriate. We call this function the region count space, or R-space. This R-space is a R-space created with the parameter sets selected and implemented with different algorithms. We introduce three R-spaces created with our idea of stable region count and selected parameters.

2.2.1 Thresholding and region merging

The purpose of this algorithm is to segment bright objects or areas in an image. All dark areas are lumped together and left unlabeled. A threshold parameter controls what is considered bright. A second parameter is used to merge together nearby bright areas. For example, looking at clouds, one can envision segmenting them as a single patch of cloudy area or a distinct number of individual clouds. We call this algorithm a thresholding and region merging algorithm.

Algorithm

The basic steps of the algorithm are as follows:

- Threshold the original image using a value T.
- Grow the thresholded image by an amount S.
- Compute the number of regions R.

The thresholding step is done using the basic function:

$$O_T[r,c] = \begin{array}{ll} 1 & \text{if } I[r,c] \geq T \\ 0 & \text{if } I[r,c] < T \end{array} \tag{2.2}$$

where $I[r,c]$ is the original input image and $O_T[r,c]$ is the output thresholded image.

At each thresholding stage, we also complete a growing process from the smallest growth to the largest. The growing step is done using:

$$O_{TS}[r,c] = \begin{array}{ll} 1 & \text{if any } O_T[r \pm \{1...S\}, c \pm \{1...S\}] = 1 \\ 0 & \text{otherwise} \end{array} \tag{2.3}$$

where $O_{TS}[r,c]$ is the output image after growing. The idea behind this step is to combine regions that are close together, where the parameter S controls how close two regions need to be in order to be combined. Figure 2.2 shows an example image demonstrating growing at different values of S. This example serves as good motivation for this step. The image is a retinal image, showing some lesions as bright spots. This area of interest could be perceived as 10-15 smaller lesions, as 2-4 mid-sized lesions, or as a single large lesion. It is difficult to say which of these is the most correct perception, so that no fixed value for S seems definitive. Therefore as T is varied, we consider the change in the number of regions across the whole range of S. Finally, the region counting step uses standard 8-connected component analysis to identify the integral number of regions in the image.

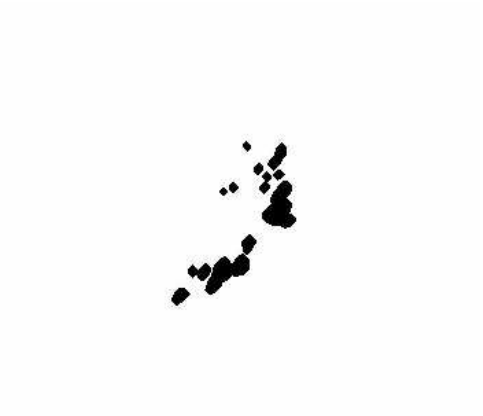
Using these steps, the number of regions R is a function of the threshold T and the amount of growing S:

$$R = F(T,S) \tag{2.4}$$

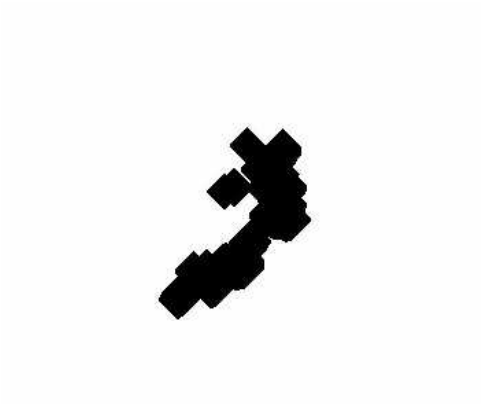
Implementation of algorithm

We threshold an image at every interval of $dT = 2$ $(0, 2, 4, ..., 254)$, and then at each threshold grow the segmented image at intervals of $dS = 5$ $(0, 5, 10, ..., 50)$ through region merging. These choices are made to save the processing time of the image without losing im-

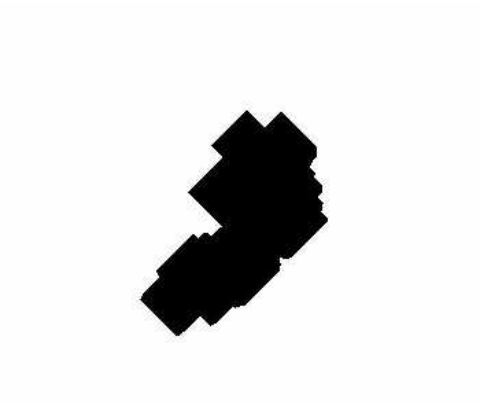
original image



$S = 5$



$S = 25$



$S = 50$

Figure 2.2: An example of region count as a result of thresholding and region growing.

portant information to be extracted. The minimum and maximum ranges ensure that there is not any strong perception beyond these limits.

Example image

We use the coins image in Figure 2.4 to demonstrate the R-space created with the above algorithm.

R-space

The R-space created with the above algorithm and parameter pair is shown in Figure 2.3. In this R-space, it is easy to see the plateau corresponding to a region count of 16. The center of the plateau is pointed out by a vertical broken line. We discard any region count number above 50 because usually there are not that many objects to be detected in an image.

Segmentation corresponding to the largest plateau

From the center of the plateau, we get the parameter pair as (198,10). We use these parameters to segment the original image, and we get the segmented image as shown in Figure 2.4.

2.2.2 Hysteresis thresholding

The purpose of the hysteresis thresholding algorithm is to segment bright objects or areas that may not stand out very well from the background. We expect that parts of objects may be darker than others, and possibly darker than the average background, so that a single threshold will not adequately separate foreground from background. Therefore, two thresholds are used. At least one or more pixels in an object must be brighter than the first threshold, while any connected pixels must be greater than the second, lower threshold.

Algorithm

The basic steps of hysteresis thresholding are as follows:

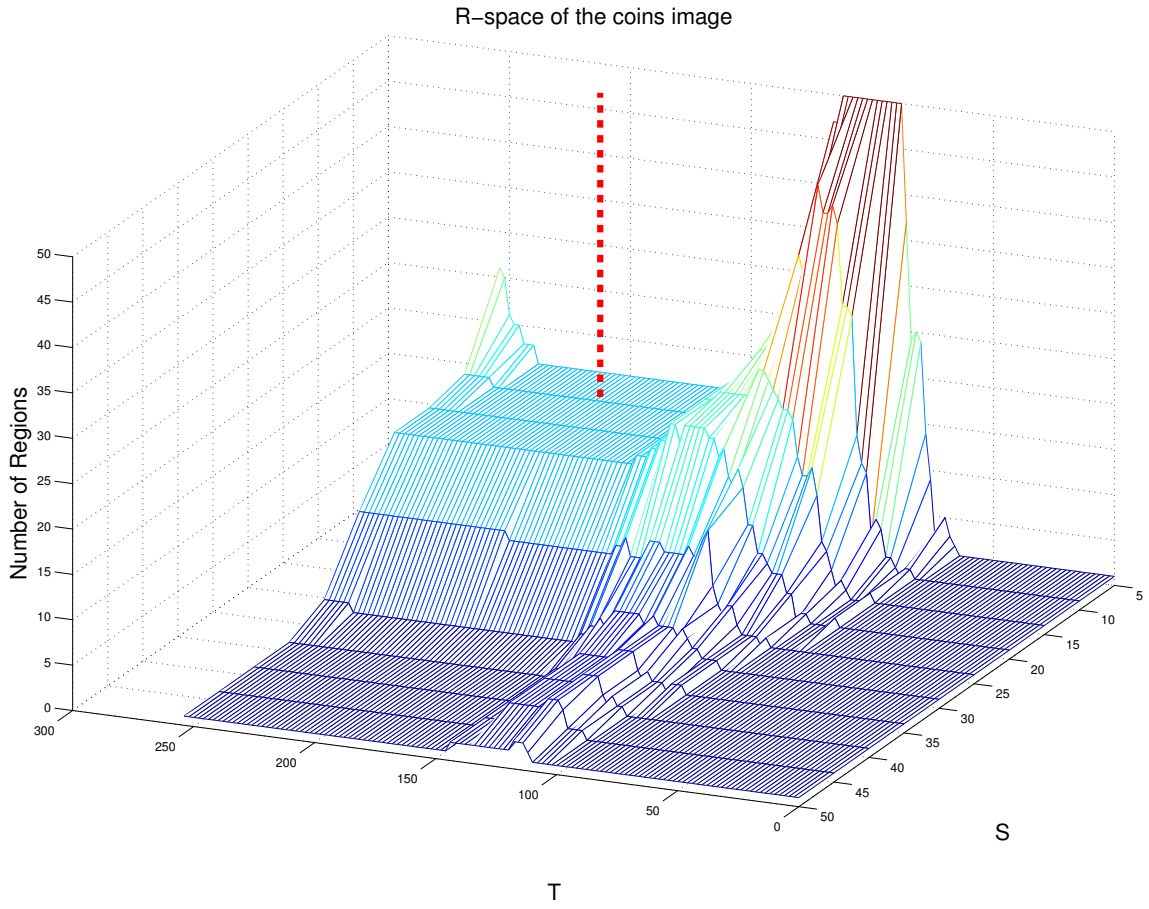
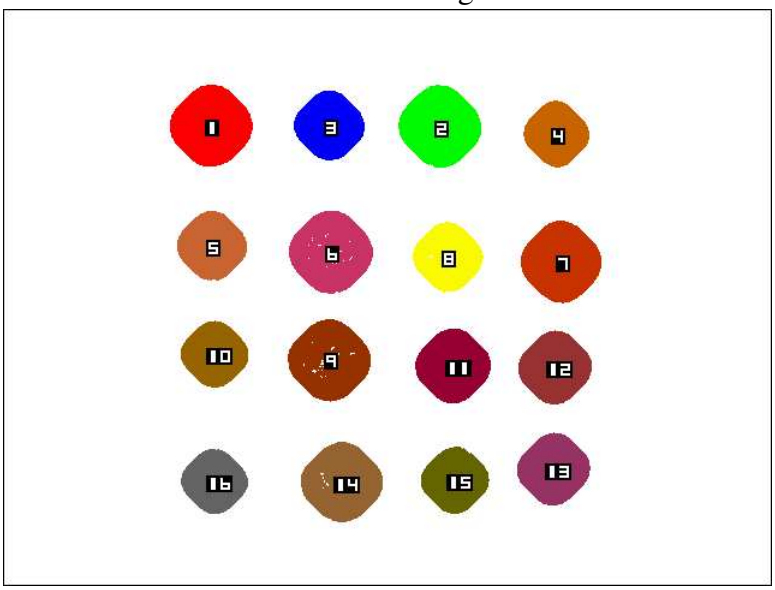


Figure 2.3: The R-space of the coins image, using the thresholding and region merging algorithm.

The coins image.



Segmentation by the largest plateau.

Figure 2.4: The coins image and the segmentation corresponding to the largest plateau in the R-space of the coins image. The numbers in the segmentation image indicate region labels.

- Threshold the original image using a value T_1, marking foreground pixels as 1 and background pixels as 0.
- Scan through the image and find pixels with an intensity value less than T_1 but greater than T_2 ($T_1 > T_2$), and mark these pixels as 1 if they are connected with foreground pixels in the first thresholding run (with T_1).
- Process the whole image iteratively until all pixels fitting the criteria are processed.
- At each set of thresholds T_1 and T_2, compute the number of regions R in the resulting binary image.

The initial thresholding step is done using the basic function:

$$\begin{aligned} O_{T_1}[r,c] = \quad & 1 \quad \text{if } I[r,c] \geq T_1 \\ & 0 \quad \text{if } I[r,c] < T_1 \end{aligned} \tag{2.5}$$

where $I[r,c]$ is the original input image and $O_{T_1}[r,c]$ is the output thresholded image. The second thresholding step is done using:

$$\begin{aligned} O_{T_1T_2}[r,c] = \quad & 1 \quad \text{if any } I[r \pm 1, c \pm 1] \geq T_2 \text{ and } O_{T_1}[r \pm 1, c \pm 1] = 0 \\ & 0 \quad \text{otherwise} \end{aligned} \tag{2.6}$$

Using these steps, the number of regions R is a function of the higher threshold T_1 and the lower threshold T_2:

$$R \quad = \quad F(T_1, T_2) \tag{2.7}$$

Implementation of algorithm

We threshold an image at every interval of $dT_1 = 5$ $(0, 5, 10, ..., 255)$, and then at each threshold grow the segmented image at intervals of $T_2 \leq T_1$ with a step of 10 (for example, for $T_1 = 120$, $T_2 = 0, 10, 20, ..., 120$) through hysteresis thresholding. These choices are made to save in processing time of the image without losing important information to be extracted. The minimum and maximum ranges ensure that there is not any strong perception beyond these limits.

Example image

We use a scenery image of sky, clouds and trees in Figure 2.6 to demonstrate the R-space created with the above algorithm.

R-space

The R-space created with the above algorithm and parameter pair is shown in Figure 2.5. In this R-space, the largest plateau shows a region count of 2. The center of the plateau is pointed out by a vertical broken line. There are 2 other smaller plateaus in the R-space. Their heights are 3 and 4. These 2 plateaus separate the white clouds into 1 and 2 more pieces, which are reasonable too. So all the plateaus in the R-space can give good segmentations to the image although the largest one gives relatively the best segmentation.

Segmentation corresponding to the largest plateau

From the center of the plateau, we get the parameter pair as (215,75). We use these parameters to segment the original image, and we get the segmented image as shown in Figure 2.6.

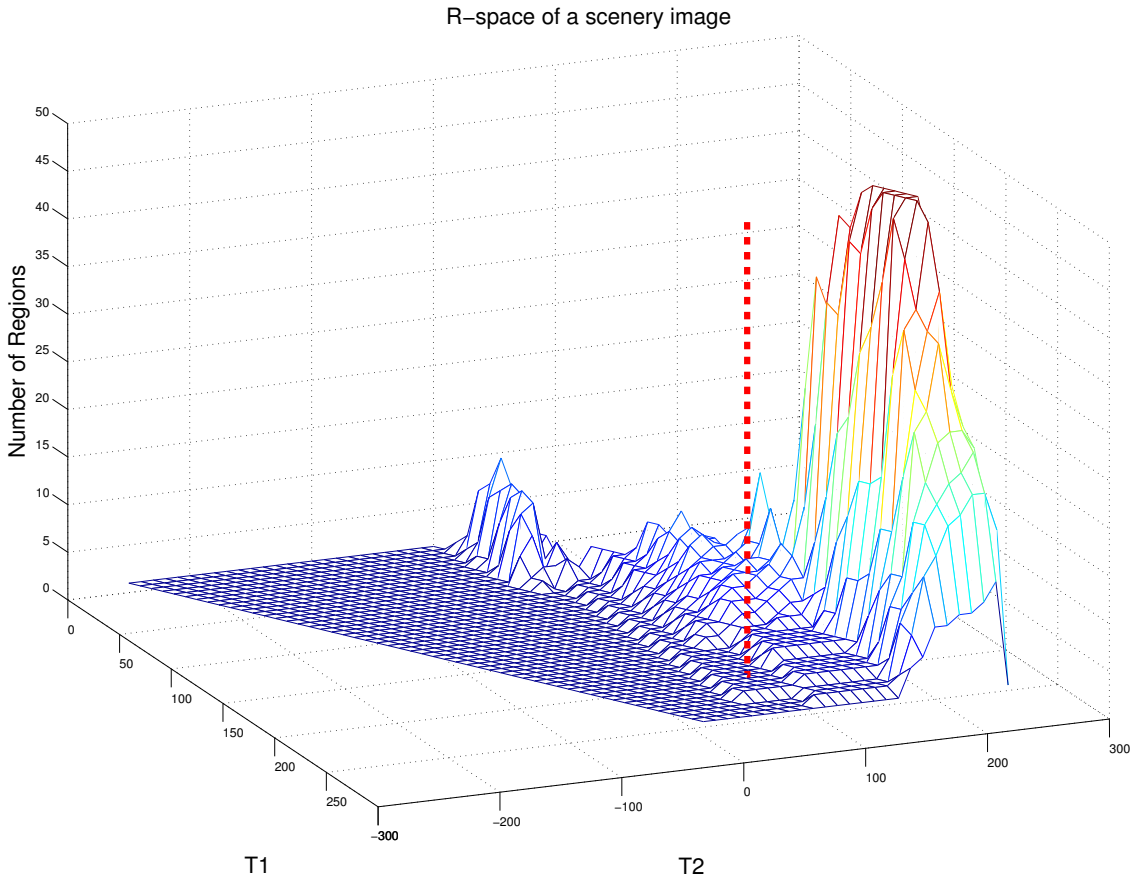
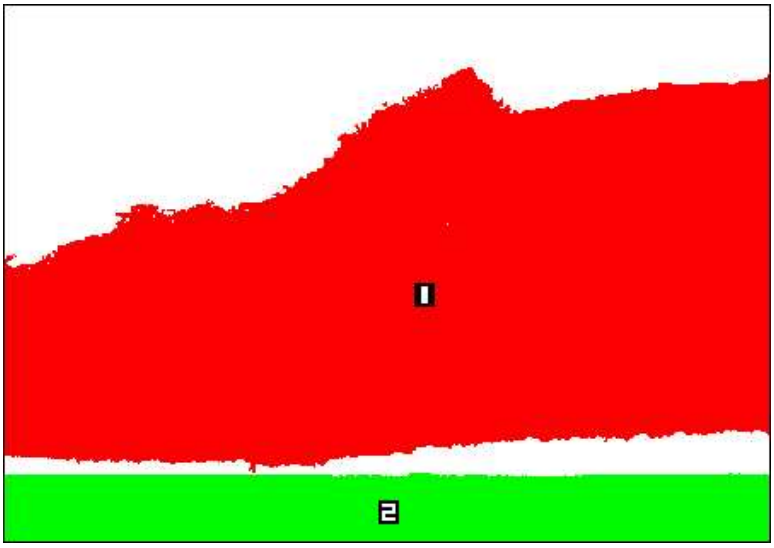


Figure 2.5: The R-space of the scenery image, using hysteresis thresholding algorithm.

Example image of sky, clouds and plants



Segmentation by the largest plateau

Figure 2.6: The segmentation corresponding to the largest plateau in the R-space of the scenery image. The numbers in the segmentation image indicate region labels.

2.2.3 Variance-based region growing

The purpose of the variance-based region growing algorithm is to segment objects or areas in an image that have homogenous color. The first parameter sets the threshold to seed pixels by their color variance (how similar the pixel colors are in the local area), and the second parameter sets the threshold to group together other pixels which are connected with the seeded pixels. The algorithm is designed to group regions or areas in an image by their colorness rather than by brightness as used in the previous algorithms.

Algorithm

The basic steps of the variance-based region growing algorithm are as follows:

- Calculate the variance (standard deviation) of the all the pixels within a $N \times N$ window in the image as,

$$V = \frac{1}{N^2} \sum_{i,j=1}^{i,j=N} \sqrt{(R_{ij} - R_{mean})^2 + (G_{ij} - G_{mean})^2 + (B_{ij} - B_{mean})^2} \qquad (2.8)$$

  where R_{ij}, G_{ij} and B_{ij} are the red, green and blue intensities of the pixel and R_{mean}, G_{mean} and B_{mean} are the mean intensities of red, green and blue colors in the $N \times N$ window where $N = 9$.

- Threshold the image using variance V_1, marking foreground pixel with labels and background pixel as 0.

- Scan through the image, from the pixel with the smallest variance to that with the largest variance, and find pixels whose variance is smaller than V_2, and mark these pixels with labels if they are connected with foreground pixels in the first thresholding run (with V_1).

- Process the whole image iteratively until all pixels fitting the criteria are processed.

- At each set of V_1 and V_2, compute the number of regions R in the resulting labeled image.

The initial seeding step is done using the basic function:

$$O_{V_1}[r,c] = \begin{array}{ll} label & \text{if } V[I[r,c]] \leq V_1 \\ 0 & \text{if } V[I[r,c]] > V_1 \end{array} \tag{2.9}$$

where $I[r,c]$ is the original input image and $O_{V_1}[r,c]$ is the output thresholded image. The second grouping step is done using:

$$O_{V_1V_2}[r,c] = \begin{array}{ll} label & \text{if any } V[I[r \pm 1, c \pm 1]] \leq V_2 \\ 0 & \text{otherwise} \end{array} \tag{2.10}$$

Note that the grouping starts from the pixel with the smallest variance to that with the largest variance.

Using these steps, the number of regions R is a function of the first seeding variance V_1 and the second grouping variance V_2:

$$R = F(V_1, V_2) \tag{2.11}$$

Please note that we denote the variance method for convenience, the parameter V_1 and V_2 are actually standard deviations which is the square root of the variance defined in probability.

Implementation of algorithm

We have implemented this algorithm using variance region growing. We threshold an image at every interval of $dV_1 = 2$ $(1, 3, 5, ..., 99)$, and then at each threshold grow the segmented image at intervals of $dV_2 = 5$ ($V_2 = 1, 6, 11, ..., 496$) through variance region

growing. These choices are made to save in processing time of the image without losing important information to be extracted. The minimum and maximum ranges ensure that there is not any strong perception beyond these limits.

Example image

We use a forest and lake image in Figure 2.8 demonstrate the R-space created with the above algorithm.

R-space

The R-space created with the above algorithm and parameter pair is shown in Figure 2.7. In this R-space, the largest plateau shows a region count of 3. The center of the plateau is pointed out by a vertical broken line.

Segmentation corresponding to the largest plateau

From the center of the plateau, we get the parameter pair as (51,201). We use these parameters to segment the original image, and we get the segmented image as shown in Figure 2.8.

2.2.4 Discussion

In this section we have finished some exploratory research by segmenting a number of images using the algorithms described above. We have presented the detailed implementation of the algorithms. We have created the R-spaces of all three algorithms, each with an image as an example. In the R-spaces, we find a plateau that gives the best segmentation of this image. Sometimes multiple plateaus are found, which all give reasonable segmentations. We need to further explore other features of the R-space.

Since noise comes with any real-world images, we use a region size filter to remove unwanted noise, that means, we don't count a region as one if it is smaller than a certain size. For most of the images, we use a region filter size of 30. It should be noted that the

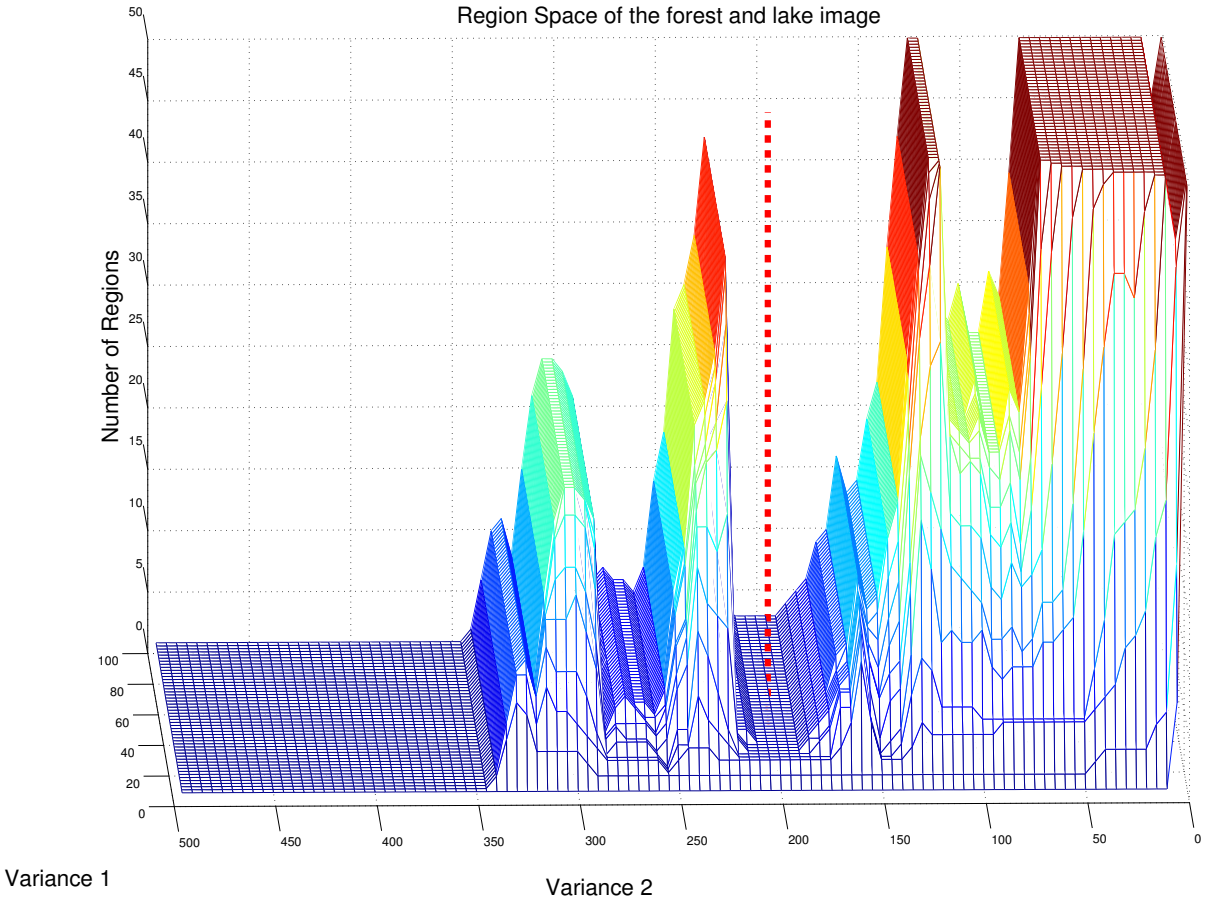
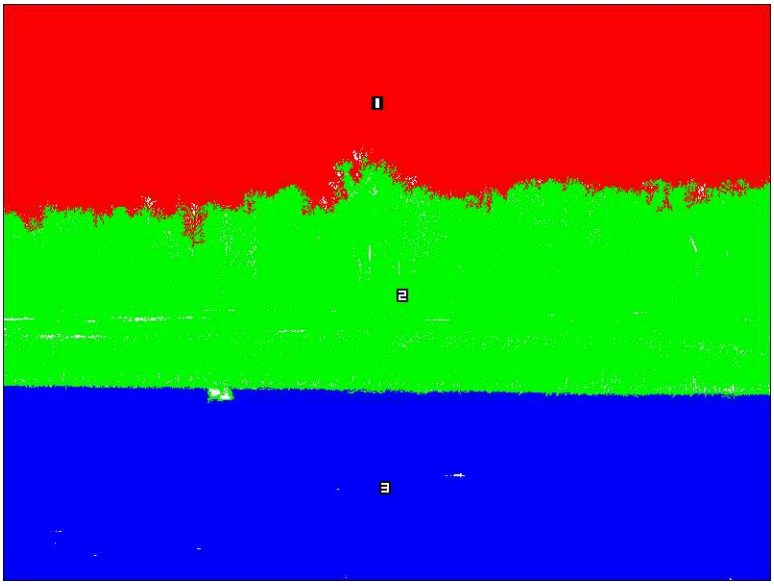


Figure 2.7: The R-space of the forest and lake image, using the variance-based region growing algorithm.

A forest and lake image.



Segmentation by the largest plateau.

Figure 2.8: The segmentation corresponding to the largest plateau in the R-space of the forest and lake image. The numbers in the segmentation image indicate region labels.

region size filter is not such a sensitive parameter, and it is not necessary to tune it for every image. For a batch of similar images, a rough size of the filter is specified depending on what is the noise to be discarded. In rare cases in which the size of the filter matters a lot, it is always possible to incorporate this parameter as one dimension in the R-space and search for a plateau in such a R-space.

2.3 Stability

From the R-space, we wish to know where the stable region count occurs in the R-space. The parameter sets at these places of stable region count should produce a better or more reasonable view of the image than at those places of no or smaller stable region count. We call this analysis the stability analysis of the R-space.

According to our theory, a stable view of the image comes from the size of the stable region count in the R-space. Therefore our primary goal is to search for the largest flat area (plateau) in the R-space. Besides the size of the plateau, we also need to investigate whether the other "features" of the plateau such as the height of the plateau, the location and surrounding of the plateau, the uniqueness of the plateau, and any other properties of the plateau (such as compactness) are informative.

We create R-spaces of more images with the three algorithms. We search for the region count in such R-spaces showing the number of regions segmented at any set of parameters in the R-space. There are "plateaus" appearing in such R-spaces. We refer to a "plateau" as a flat area in the R-space where the region count is a constant. We manually select the points in those plateaus to check the corresponding segmentations of the original images and find whether they make perceptual sense to us.

From the R-spaces created with all three algorithms, we wish to find a strong correlation between a plateau in the R-space and good segmentations of the original images in the application of every algorithm. Sometimes there are multiple plateaus, sometimes there is

only one, and other times there is none. We wish to verify that whenever there is a plateau or multiple plateaus, there is one or more reasonable segmentations or views corresponding to the parameters in that plateau or plateaus. The results are listed in the following sections.

2.3.1 Big plateau corresponds to strong segmentation

Our most obvious finding in the important findings in the correlation between a plateau and a segmentation is that a large, stable and unique plateau usually corresponds to a strong segmentation. This phenomenon can be seen in Figures 2.9 through 2.11. The three results are obtained with three different algorithms, two for the coins image and one for a scenery image. These results show a stable region count.

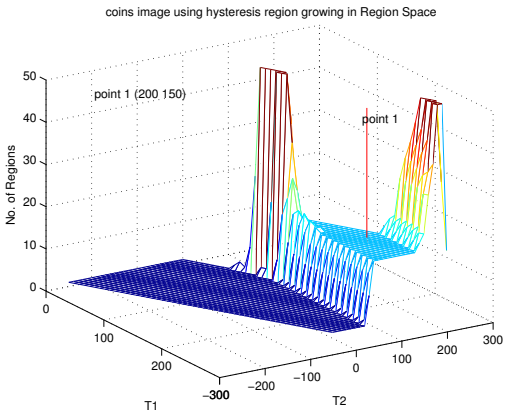
From the figures, it is easy to see that all of them have only one large obvious plateau in the the R-space. Using the parameter set in the center of these plateaus we can get very good segmentations of the original images. The correlation between the plateau and the segmentation is very clear.

2.3.2 Multiple plateaus indicate good segmentations

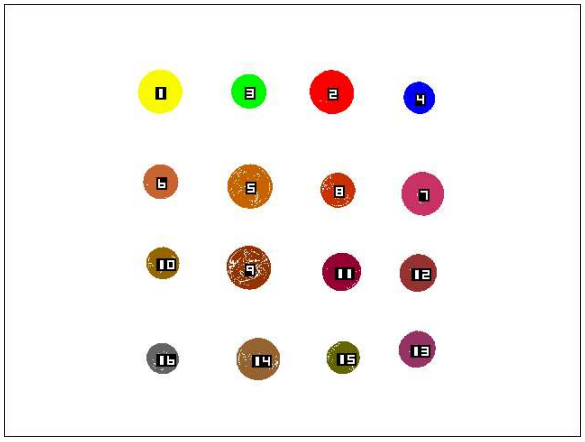
We have processed several images using the three algorithms. We see multiple plateaus of different sizes rather than one big plateau. Analyzing these plateaus indicates that one or more of them represents good segmentations too. We discuss four examples in detail in Figures 2.12 through 2.15. These figures all show multiple plateaus in their respective R-spaces, one or more such plateaus lead to meaningful segmentations of the original images. The images are from coins, retinal, and scenery, and all three algorithms are used. These results again support our idea of stable region count.

Figure 2.12 shows a retinal image processed with the thresholding and region merging algorithm. This algorithm is a threshold-based noise removing approach to segment images. Because there is a growth (merging) factor in the labeled image with this kind of algorithm, the segmented image is not easy to see. But excluding the growth factor, the

A coins image



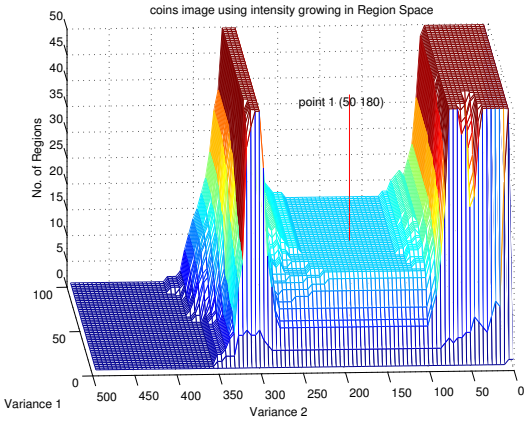
R-space



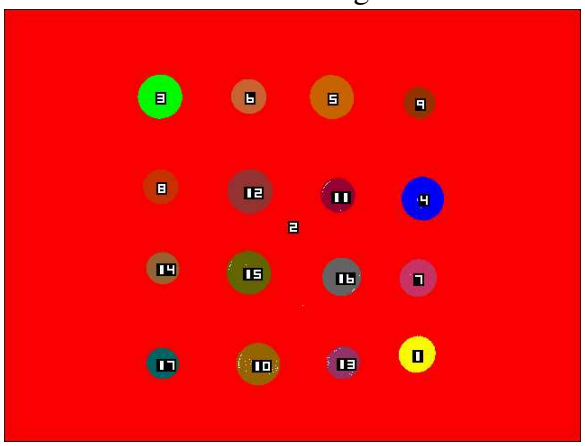
Segmentation from plateau

Figure 2.9: An example of a large, unique plateau in an R-space corresponding to a good segmentation. (Coins image processed by the hysteresis thresholding algorithm.)

A coins image



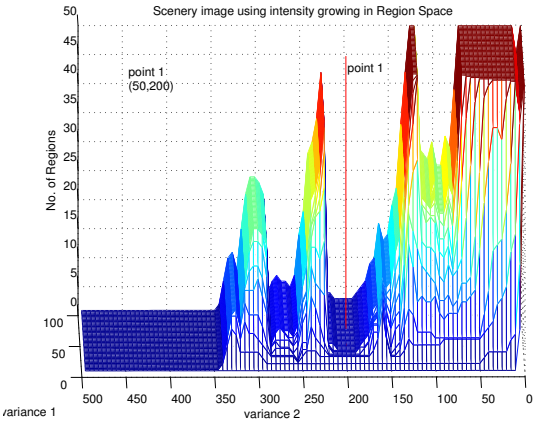
R-space



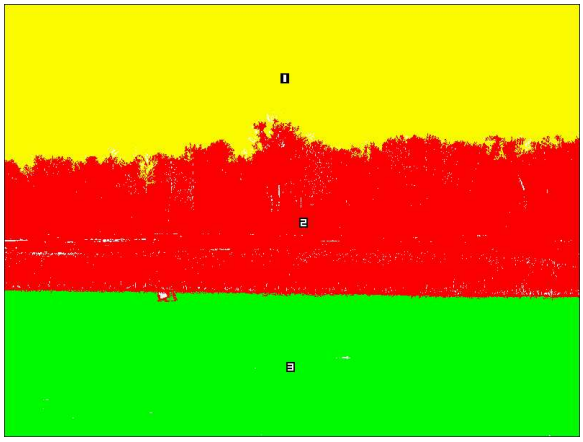
Segmentation from plateau

Figure 2.10: Another example of a large, unique plateau in an R-space corresponding to a good segmentation. (Coins image processed by the variance-based region growing algorithm.)

A scenery image

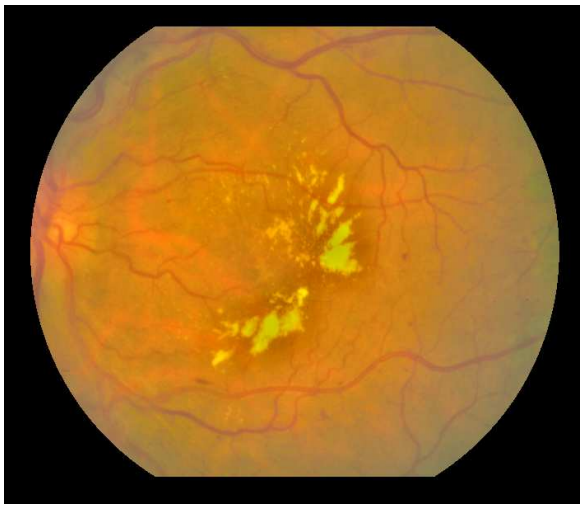


R-space

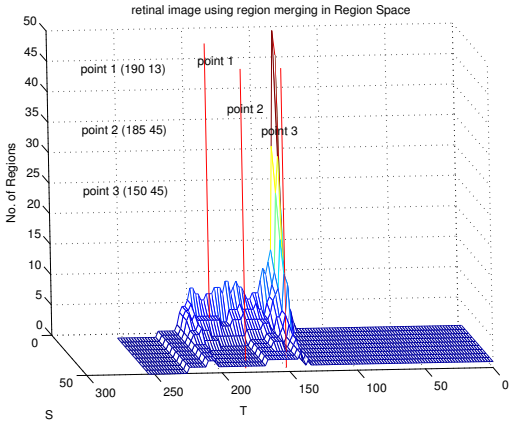


The only possible segmentation

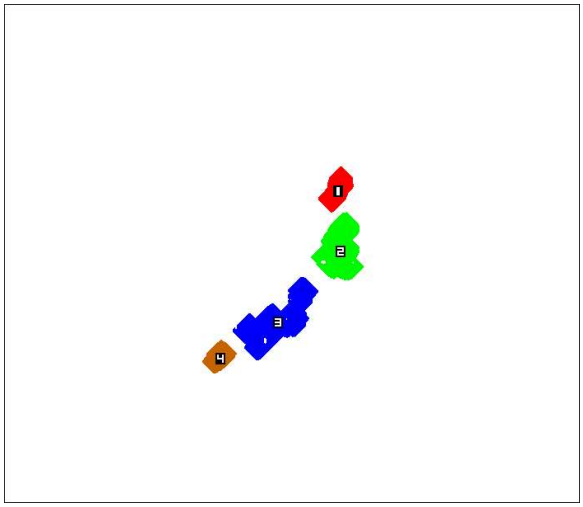
Figure 2.11: A third example of a large, unique plateau in an R-space corresponding to a good segmentation. (A forest and lake image processed by the variance-based region growing algorithm.)



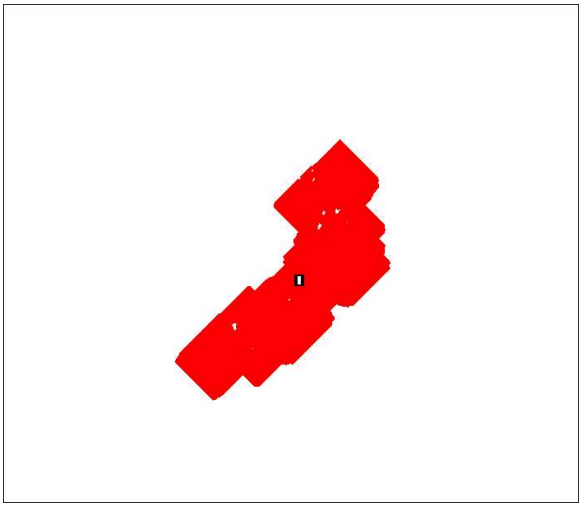
A retinal image



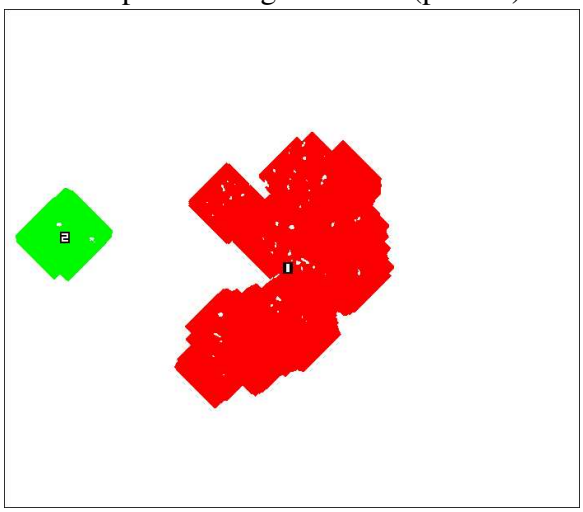
R-space



One possible segmentation (point 1)



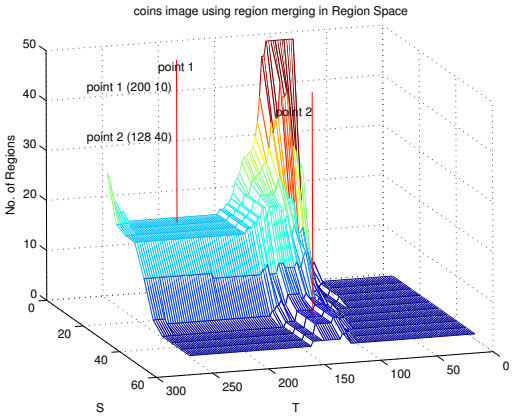
Another possible segmentation (point 2)



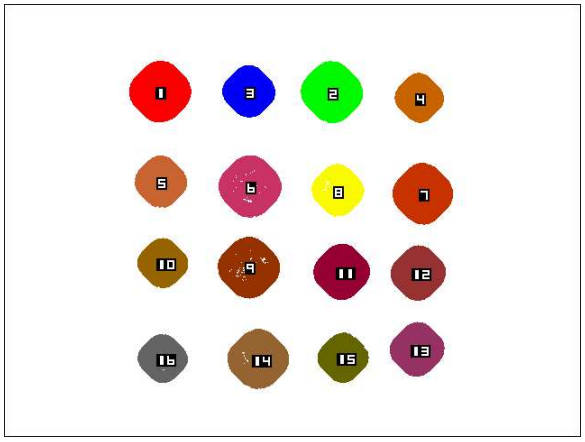
Another possible segmentation (point 3)

Figure 2.12: An example of an R-space having multiple large plateaus. Each plateau indicates a reasonable segmentation. (Retinal image processed with the thresholding and region merging algorithm.)

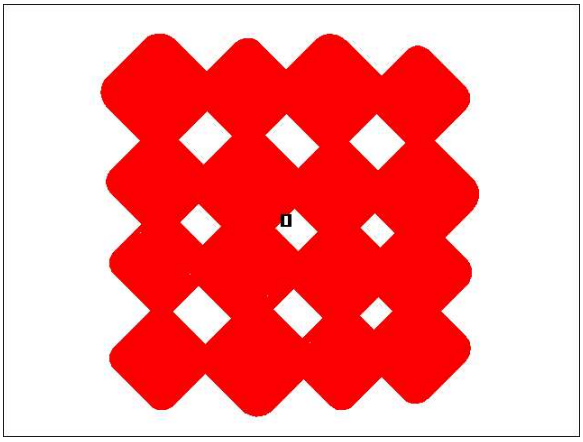
A coins image



R-space



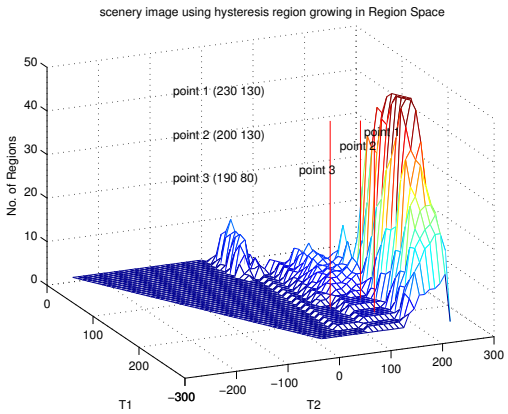
One possible segmentation (point 1)



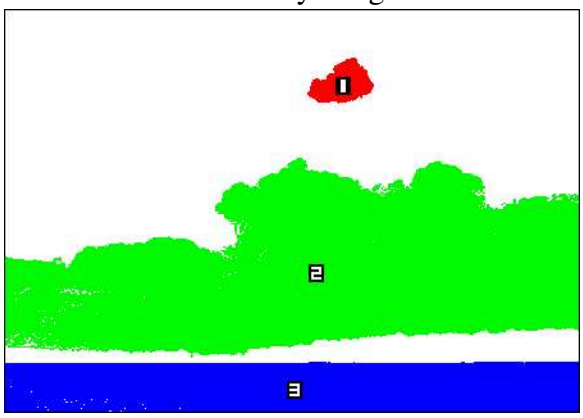
Another possible segmentation (point 2)

Figure 2.13: Another example of an R-space having multiple large plateaus. Each plateau indicates a reasonable segmentation. (Coins image processed with the thresholding and region merging algorithm.)

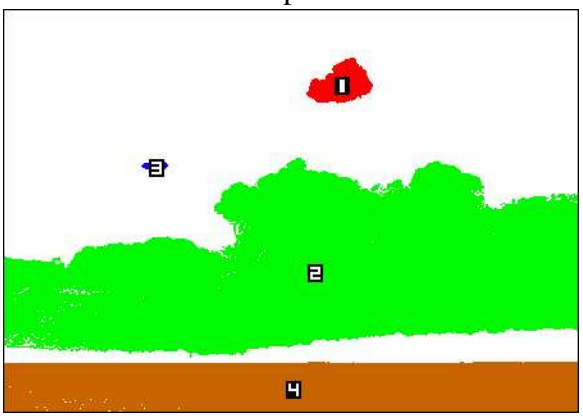
A scenery image



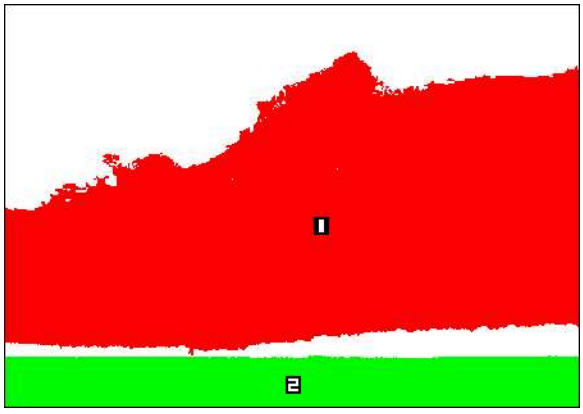
R-space



One possible segmentation (point 1)



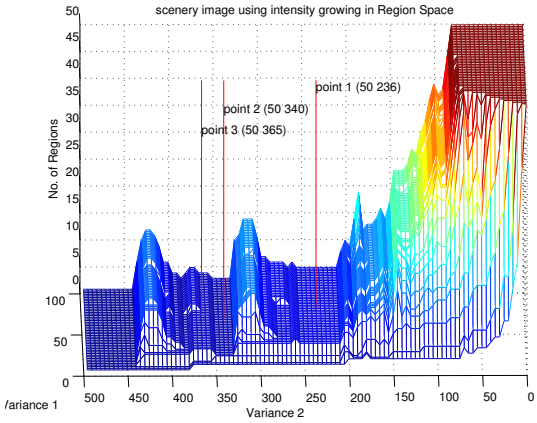
Another possible segmentation (point 2)



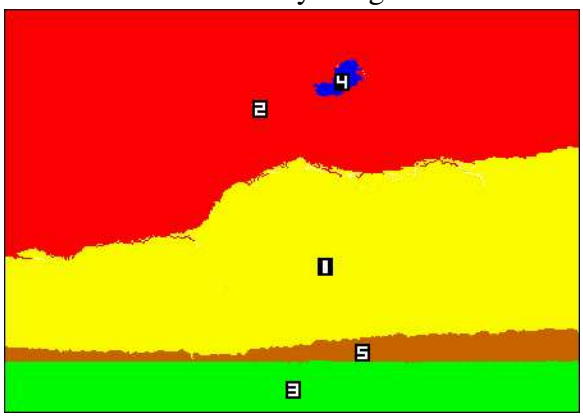
Another possible segmentation (point 3)

Figure 2.14: A third example of R-space having multiple large plateaus. Each plateau indicates a reasonable segmentation. (Image of sky, clouds and trees processed with the hysteresis thresholding algorithm.)

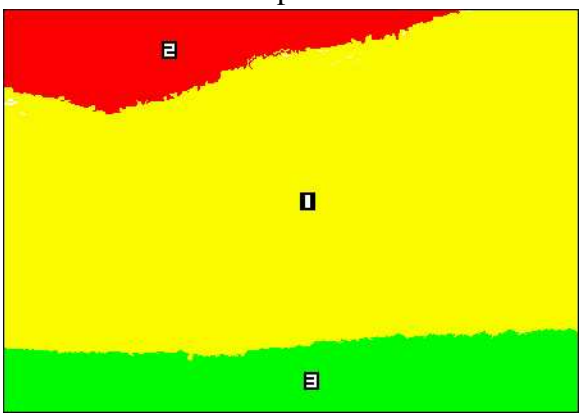
A scenery image



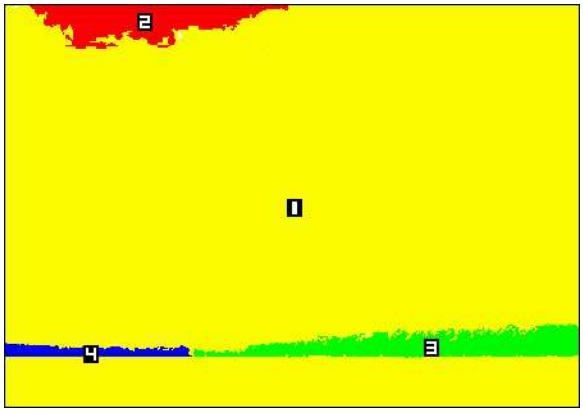
R-pace



One possible segmentation (point 1)



Another possible segmentation (point 2)



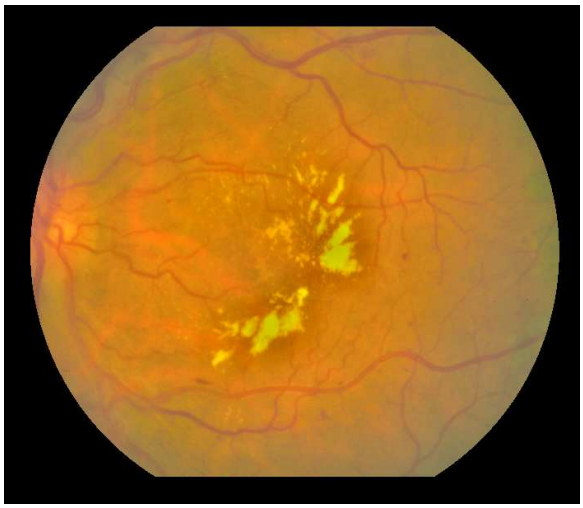
Another possible segmentation (point 3)

Figure 2.15: A fourth example of R-space having multiple large plateaus. Each plateau indicates a reasonable segmentation. (Image of sky, clouds and trees processed with the variance-based region growing algorithm.)

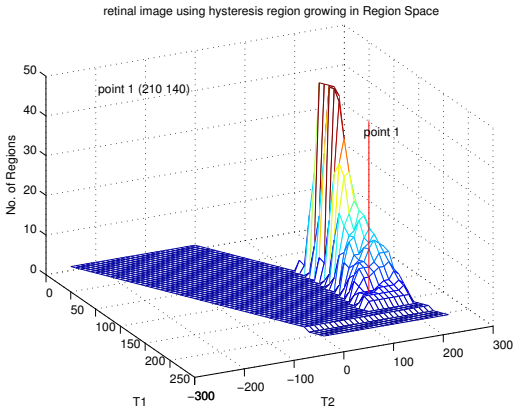
three segmented images show clearly that the lesions and optic nerve (only in one of the labeled images) in the original retinal image have been captured by the algorithm. The first segmentation (point 1) considers the legions as four separate parts, the second (point 2) considers all four parts as one integral legion, and the third segmentation (point 3) not only finds the legions but also the optic nerve. From a human point of view, all these segmentations make good sense of the retinal image.

Figure 2.13 shows a coins image processed with the thresholding and region merging algorithm. In this segmentation, the coins image is viewed by the algorithm in two ways, one segmentation of sixteen separate coins and another segmentation of coins connected as one integral part. Both views make sense to human perception. When we look at such an image, we first see a bunch of coins in some background, and then when we look closely, we see sixteen different coins in there.

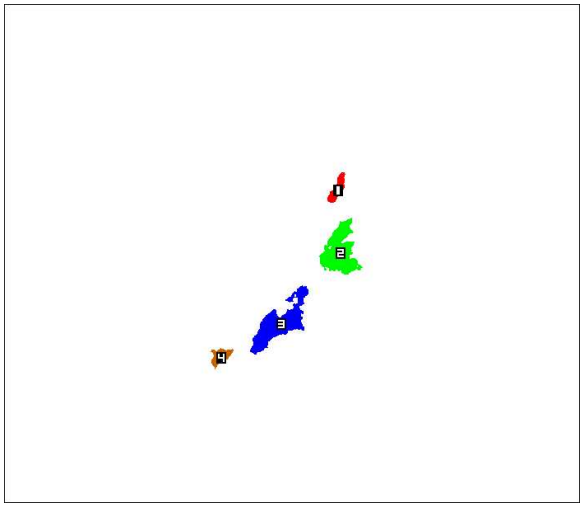
Figures 2.14 and 2.15 are both for the same image of sky, clouds and trees. One is processed with hysteresis thresholding algorithm and the other with variance-based region growing algorithm. Even processed with two different algorithms, they show similarity in R-space. Both have a few small plateaus and all these plateaus correspond to some meaningful segmentations. Except for one segmentation showing a result of separated forest (point 3 of Figure 2.15) and another segmentation showing a result of connected yellow plants and green forest (point 2 of Figure 2.15), others only differ in the way the white clouds are organized into one piece, two pieces or three pieces. When such an image is put to a human, he or she should also have different views to organize these pieces of clouds. Another point worth noting is that the larger the size of the plateau, the more the segmentation seems to make sense, and that means the more stable and stronger perceptual view for humans.



A retinal image



R-space



The only possible segmentation

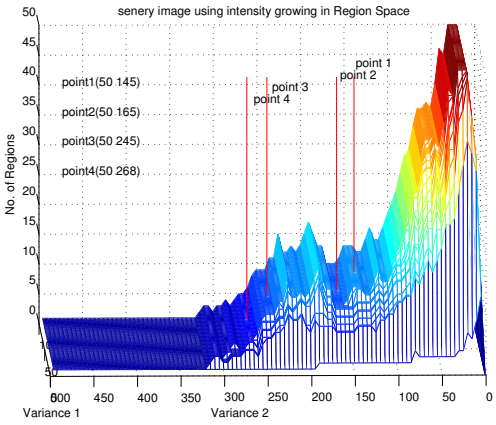
Figure 2.16: An example of an R-space having a small plateau corresponding to a reasonable segmentation. (Retinal image processed by the hysteresis thresholding algorithm.)

2.3.3 Small plateau indicates good segmentation

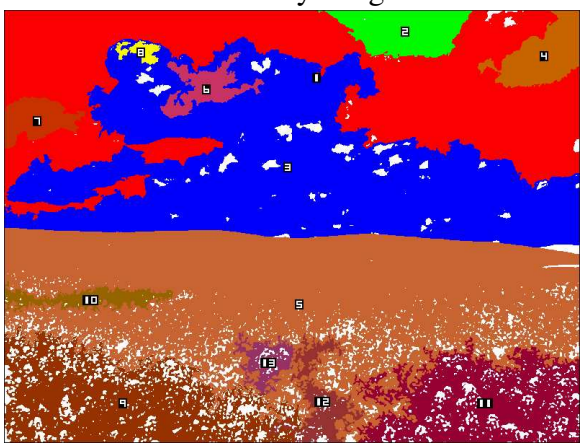
There are times when only fairly small plateaus are present in the R-space. Checking the corresponding segmentations of these plateaus reveal to us that the plateaus represent moderate if not ideal or strong view of the images. Figures 2.16 through 2.18 show such cases.

Figure 2.16 shows the legions captured by the hysteresis thresholding algorithm, which is one of the meaningful segmentations by the region-merging algorithm. It does not get

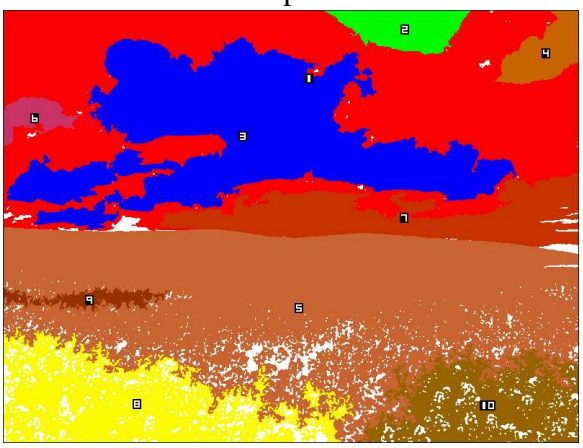
A scenery image



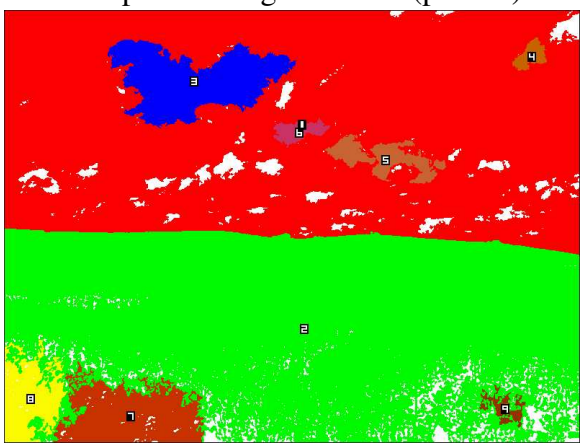
R-space



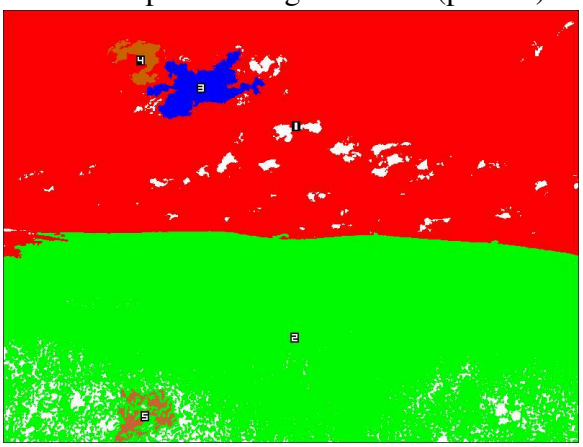
One possible segmentation (point 1)



Another possible segmentation (point 2)



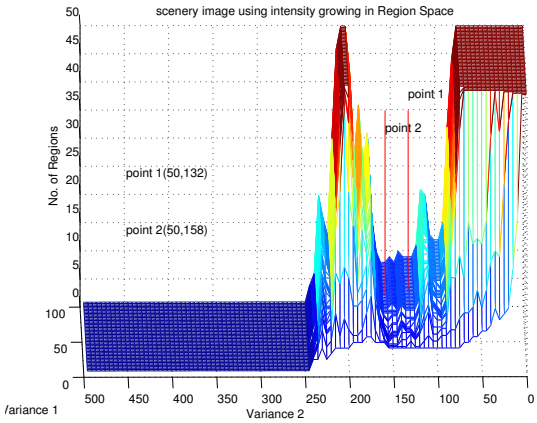
Another possible segmentation (point 3)



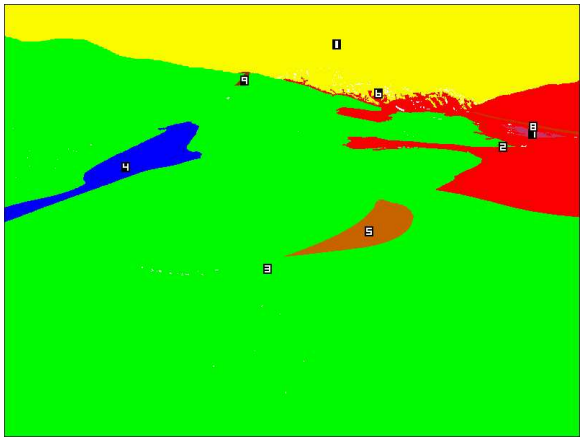
Another possible segmentation (point 4)

Figure 2.17: An example of an R-space having multiple small plateaus. Each plateau indicates a reasonable segmentation. (Image of sky, clouds and plants processed by the variance-based region growing algorithm.)

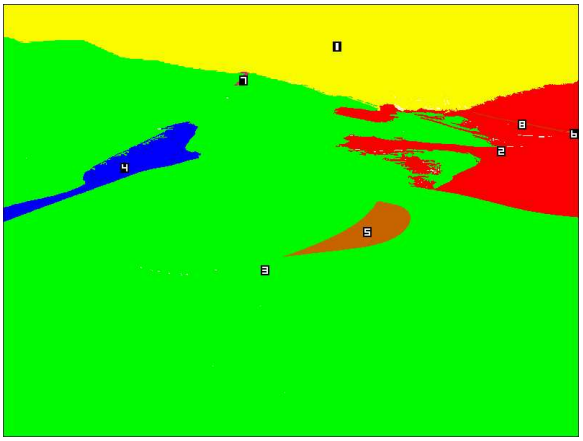
A scenery image



R-space



One possible segmentation (point 1)



Another possible segmentation (point 2)

Figure 2.18: Another example of an R-space having multiple small plateaus. Each plateau indicates a reasonable segmentation. (Image of sand dune processed by the variance-based region growing algorithm.)

the optic nerve as another possible view of the image. Figure 2.17 show several possible segmentations of the image of sky, clouds and plants, and segmentations make human perceptual sense. Due to the yellow flowers in the image, the region count is not stable, and that may be the reason why plateaus are so small. Figure 2.18 shows two meaningful views of the sand dunes separated from the sky. How to organize the visual components on the right side of the image may be the reason contributing to the small sizes of the plateaus.

2.3.4 Valley indicates reasonable segmentation

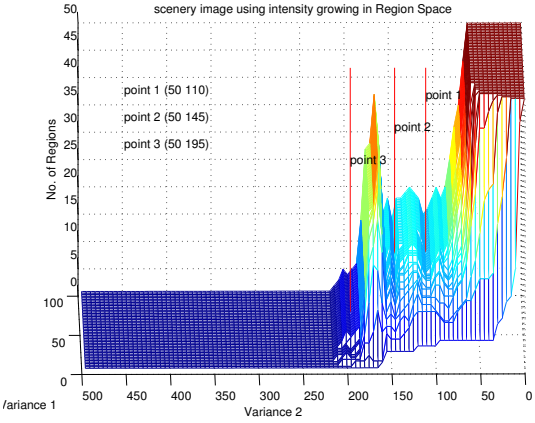
In an unusual case we don't find any plateau in the R-space, instead we find a "V" shaped valley. The height of the valley does not exceed five. A parameter pair in this valley also gives us fairly good segmentations as shown in Figure 2.19. In this figure, the sky, sea, sand and chairs can be differentiated by a pair of parameters in the valley. So this case has given us an alternative if a plateau is not present in the R-space.

2.3.5 Cases of no correlation

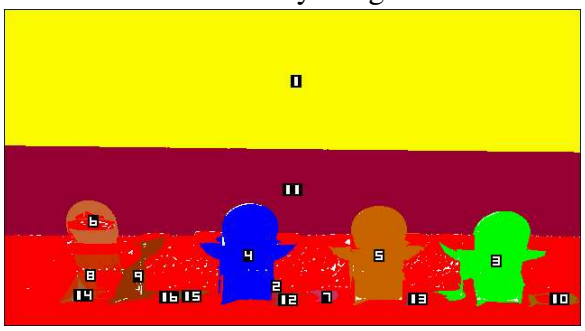
On the other hand, for some other images we have processed with these three algorithms, we did not find any reasonable view or segmentation. There are usually two reasons for this. One reason is that there is no plateau at all, so there is no stable view associated with the algorithm. The other reason is that there is some kind of plateau, but the view or segmentation associated with the plateau does not make sense to a human observer. In both cases the algorithm is not a good choice for that particular image. We show a few results of processed images which do not show the plateau-segmentation correlation below.

Although Figures 2.20 through 2.24 don't reveal the kind of views we have anticipated, they do reveal some important information for the future improvement of our method. Since the region-merging and hysteresis thresholding algorithms are basically threshold-based methods, white clouds in Figures 2.20 and 2.21 are detected, sand traps and bright sky are also detected in Figure 2.22 by several different plateaus. In Figure 2.23, although chairs

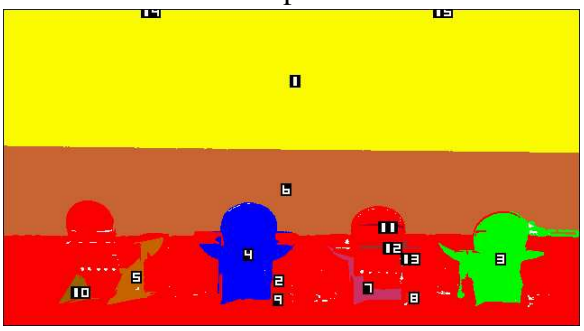
A scenery image



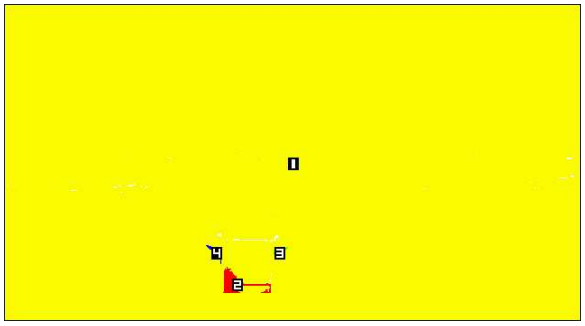
R-space



One possible segmentation (point 1)



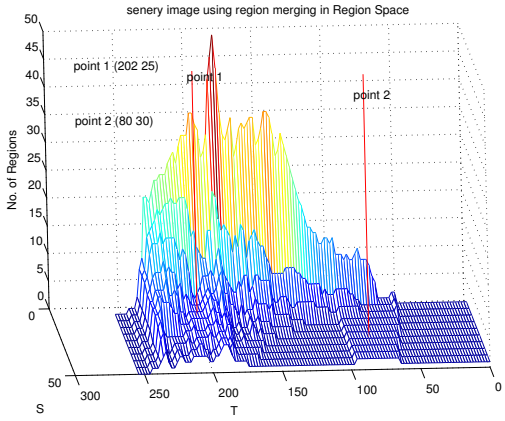
Another possible segmentation (point 2)



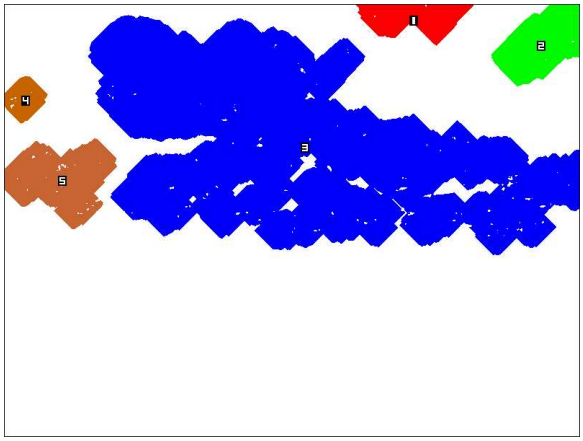
Another possible segmentation (point 3)

Figure 2.19: An example of valley in stead of plateau in the R-space. The corresponding segmentation is reasonable. (Image of beach, sky and chairs processed by the variance-based region growing algorithm.)

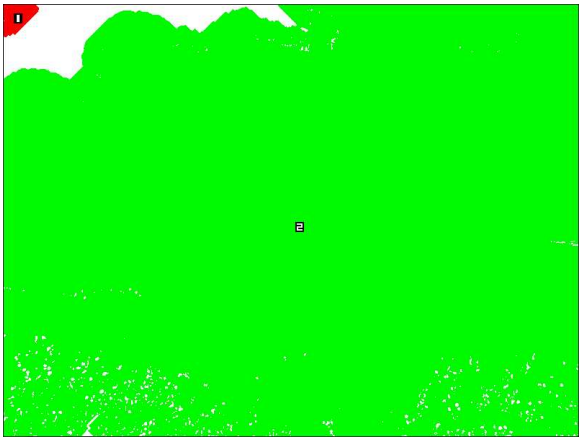
A scenery image



R-space



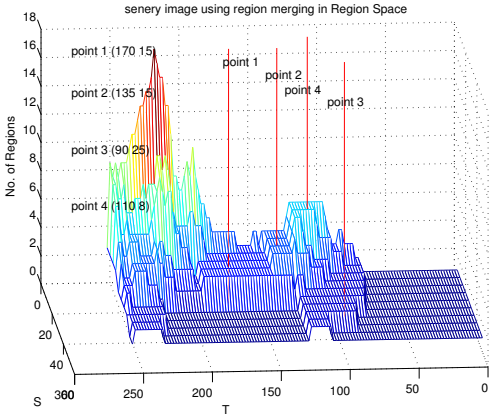
One possible segmentation (point 1)



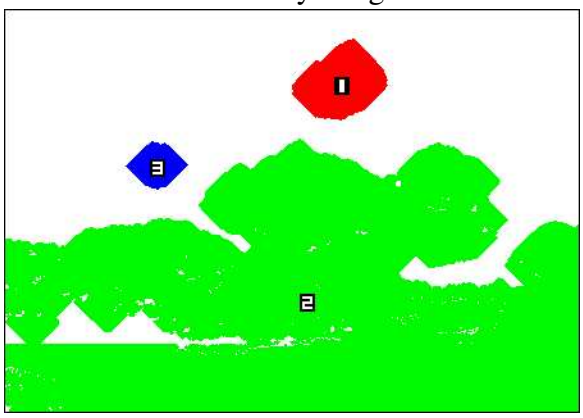
Another possible segmentation (point2)

Figure 2.20: An example of no correlation. (Image of sky, clouds and plants processed by the thresholding and region merging algorithm.)

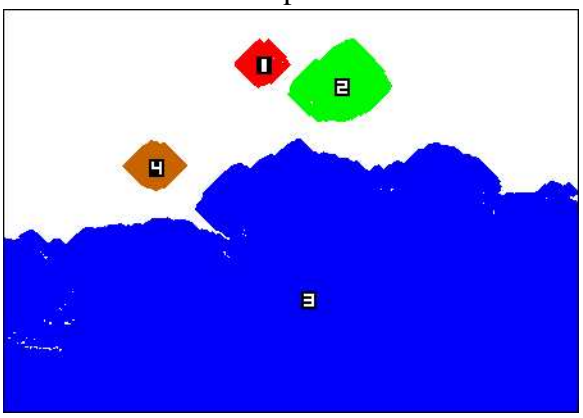
A scenery image



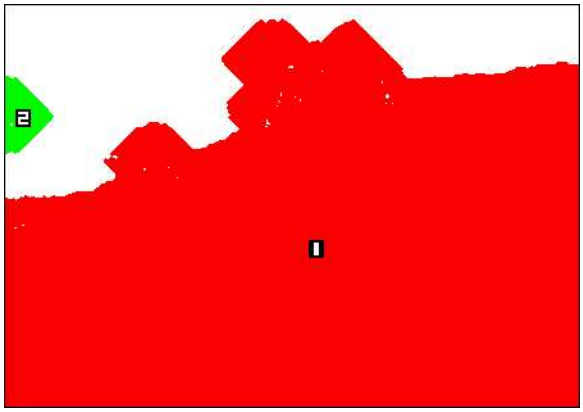
R-space



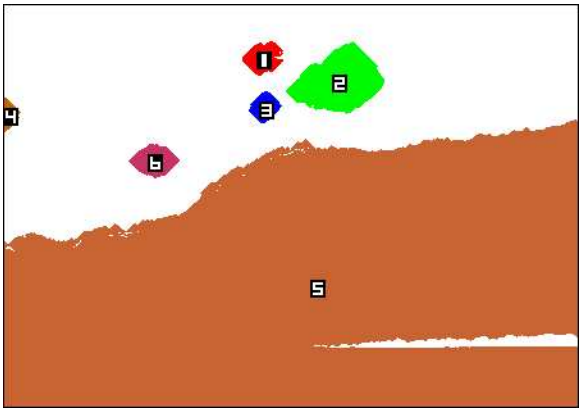
One possible segmentation (point 1)



Another possible segmentation (point 2)



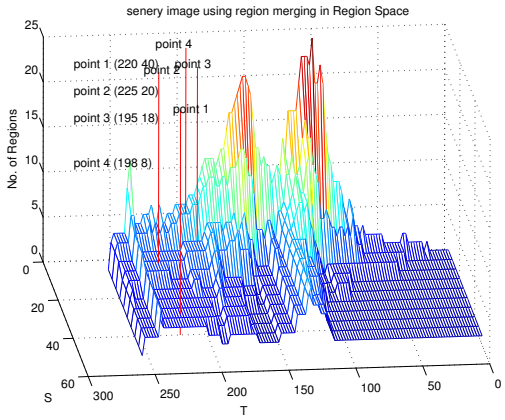
Another possible segmentation (point 3)



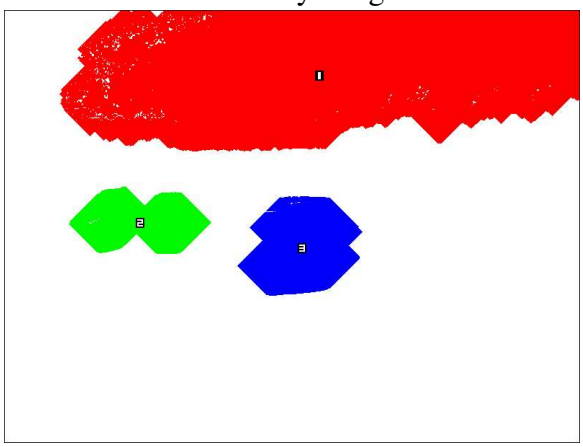
Another possible segmentation (point 4)

Figure 2.21: Second example of no correlation. (Image of sky, clouds, trees and plants processed by the thresholding and region merging algorithm.)

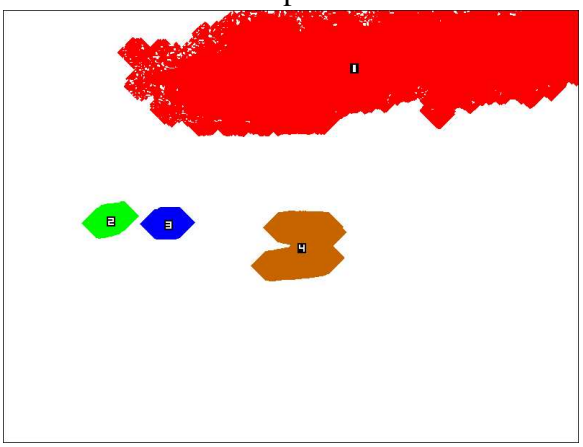
A scenery image



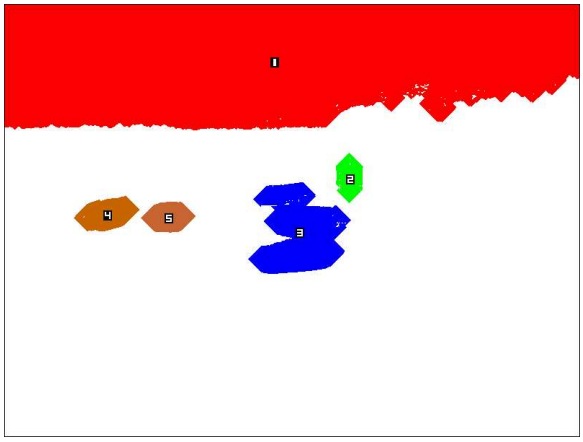
R-space



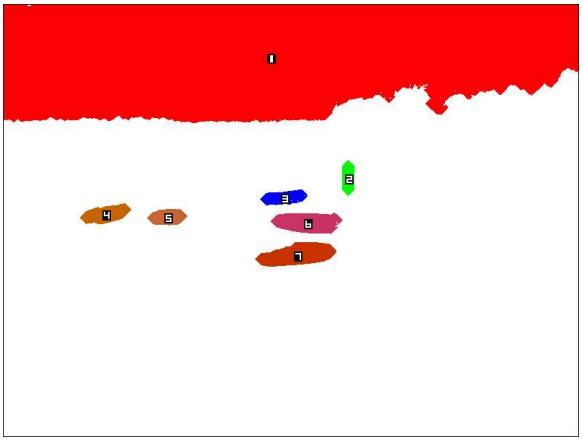
One possible segmentation (point 1)



Another possible segmentation (point2)



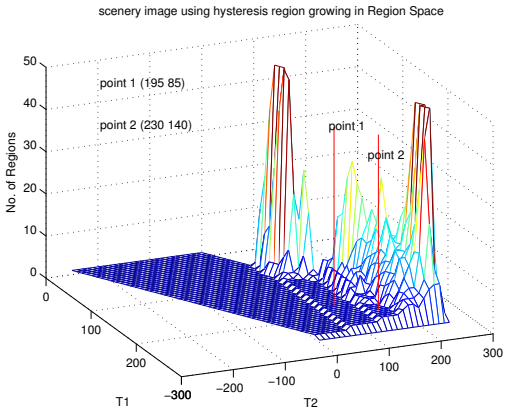
Another possible segmentation (point 3)



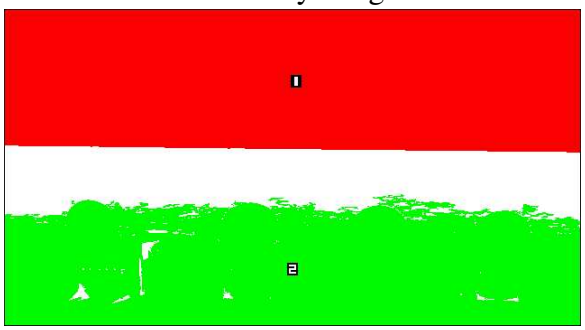
Another possible segmentation (point 4)

Figure 2.22: A third example of no correlation. (Image of golf course processed by the thresholding and region merging algorithm.)

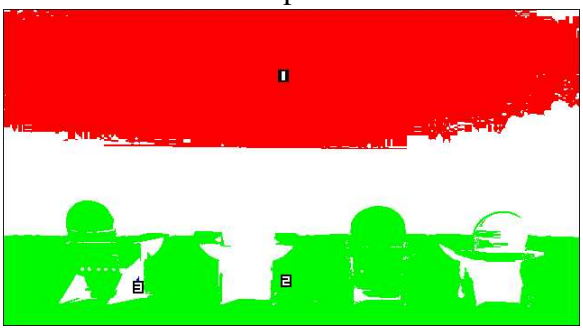
A scenery image



R-space

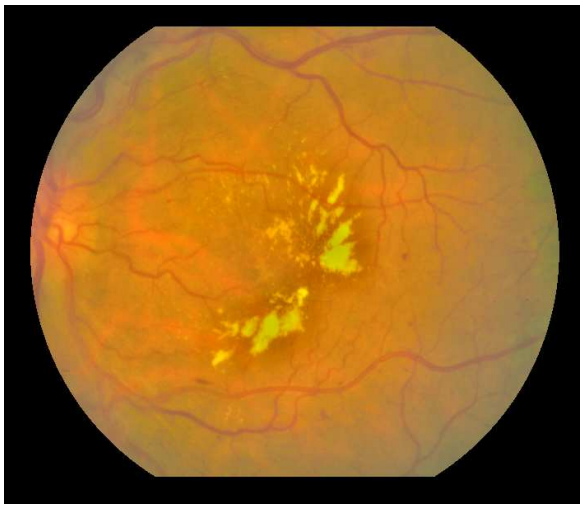


One possible segmentation (point 1)

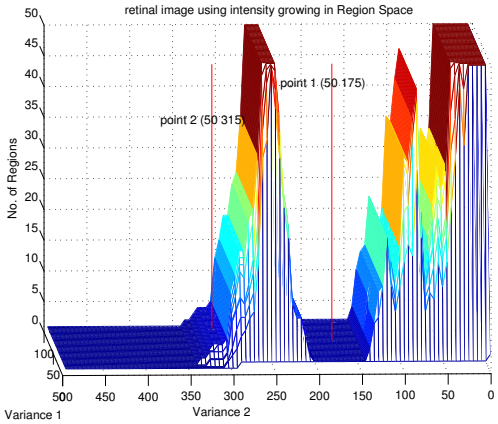


Another possible segmentation (point 2)

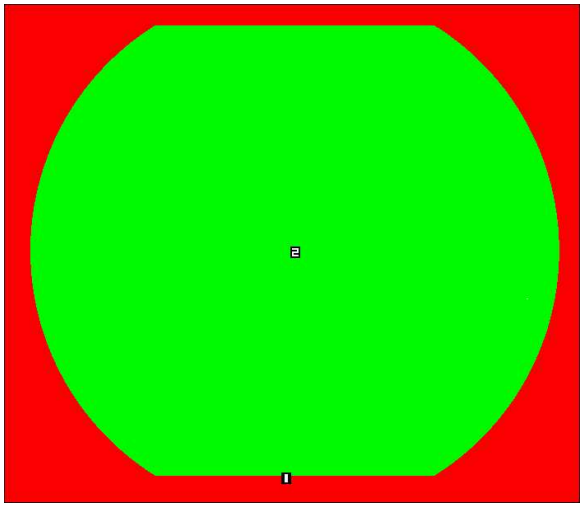
Figure 2.23: A fourth example of no correlation. (Image of beach, sky and chairs processed by the hysteresis thresholding.)



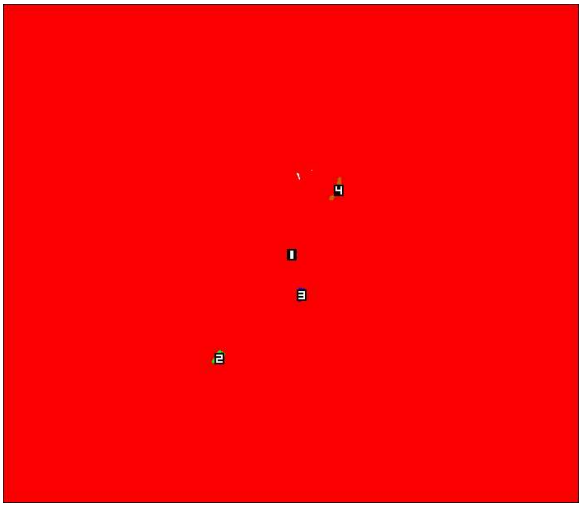
A retinal image



R-space



One possible segmentation (point 1)



Another possible segmentation (point 2)

Figure 2.24: A fifth example of no correlation. (Retinal image processed by the variance-based region growing algorithm.)

are not separated from the beach, they are separated with the beach from the white clouds, bright sky and the sea. These are just proving the fact that threshold-based algorithms are first and foremost a white-blob detector. For the retinal image shown in Figure 2.24, the plateaus only reflect that the retinal is different from the background or the remnants of legions are extraordinarily different in color from the rest of the image. This segmentation result shows that the variance-based approach is actually a color differentiator.

It should be noted that the "no correlation" statement is made at a relatively high standard. In those cases, the best segmentations have been achieved within the capabilities of the respective R-spaces (parameter pairs and algorithms). The results are not satisfactory because the R-space formulated is not appropriate for that particular image. So it is vitally important to select the most appropriate algorithm and the right parameter set for the images.

We have created all these R-spaces of different images with the three algorithms in the hope of finding a systematic approach for locating the best parameter set for segmentation. However, we didn't find a unified approach to analyze the R-space. Instead, we have to search for the stability of R-space in an application specific manner. In other words, we need to develop an individual R-space analysis method for each specific group of images or each specific algorithm. Although our idea of stable region count still works, the framework has to adapt to different applications.

2.4 Discussion

From the above data and analysis we notice that if there is only one plateau in the R-space, that plateau usually is what we are looking for. Sometimes there are multiple plateaus, one or more of which make sense. In this case we tentatively speculate that the larger the area of the plateau, the better the segmentation. The plateaus should better lie in some big changes (mountains) in the R-space. Plateaus usually are not what we need if they connect

with the start and end of the R-space. On the other hand, the amplitude of the plateau is also an important factor. A region count of zero means nothing is in there, so it is not what we are looking for. A region count of one is usually not significant unless you are searching for only one object in that image. By the same reasoning, the region count equalling to the maximum number (we set some threshold so that beyond this range the number is not a possible region count) in the R-space is not what we are interested in because a very large number of objects in an image is not likely.

Based upon the observations noted in Section 2.3, we hypothesize that it should be possible to automatically identify the best parameter values of the given algorithm for the given input image. We assume that the given algorithm is appropriate to process the given image; that is, we assume that the algorithm is capable of producing a strong segmentation if the appropriate parameters are used. The problem is to decide how to use the region count space to identify the best parameter values.

We have tried to explore searching for parameter values by looking for plateaus with desirable properties, including area, height, surroundings and uniqueness. These properties could be combined into a single search. For example, a score for each property could be computed, and then the scores could be combined into a single value, relating to overall "stability" of the segmentation at the given parameter values. Conversely, the properties could be searched in some preferred order. For example, the area of a plateau could be the deciding factor, unless no plateau of appreciable area is found, in which case the search proceeds to uniqueness.

However, through all our efforts, we cannot combine all these properties of area, height, surroundings and uniqueness into one single score for all these heterogeneous images although we strongly believe the usefulness of these properties in helping us to find the right parameter values. We just cannot prove the usefulness of these properties through this diverse group of images, but we believe these can be proved through a group of images that are somewhat similar. We have also failed in searching the best parameters by the pre-

ferred order. But we do find the plateau area is the dominating factor in locating the best parameter values in the R-space.

We have also tried several variations on these ideas, computing combined scores in various manners and conducting searches in various preferred orders and methods. We evaluate each of these variations using hand graded results from the region count spaces presented in Figures 2.9 to 2.19. We try to use the different variations to get the best parameter pairs but only succeed in using the plateau area as the criterion for the stability analysis.

We plan to use the only usable stability criterion — the plateau area to analyse some specific groups of images.

Chapter 3

Experiments

In this chapter we apply our framework to three specific problems. The first problem is flower detection in images of flowers. The second problem is real-time peak detection in a 1-dimensional signal over time. The third problem is lesion detection in retinal images. For each problem, we describe the algorithm, parameter set, implementation, and computation of the region count space. We then describe an automated method to compute stability. The stability computation is used to produce a unique output segmentation. We evaluate our results according to the criteria for success for the given segmentation problem.

3.1 Flower detector

For this problem, we want to detect and segment flowers in any given image. In order to simplify algorithm development (We are not interested in building the most powerful flower detector ever constructed, we are only interested in demonstrating the application of our framework to the problem.), we restrain the problem to the detection of brightly colored (non-white) flowers. Thus, the algorithm primarily depends upon finding groups of brightly colored, non-white pixels. We selected 44 images to test our methods. Each image contains a readily identifiable number of flowers or flower clusters, so that the correctness of our automatically produced segmentation can be evaluated. We evaluate our methods

by human visual inspection, as well as by comparison of the count of flowers against the human-determined correct count. The following sections describe in detail the algorithm, R-space computation, data set and results.

3.1.1 Algorithm

For flower detection, we use an algorithm called RGB-distance variance-based region growing. The basic steps of RGB-distance variance region-based growing are as follows:

- Calculate the variance (standard deviation) of all the pixels with respect to neighboring pixels (within a fixed 9×9 window) in the image.
- Seed the pixels in the entire image (don't seed any pixel with a high green component), marking foreground pixel with labels from 2 and up, and mark background pixels and unlabeled pixels as 0.
- Scan through the image, from the seeded pixel with the smallest variance to that with the largest variance.
- Grow a region by finding pixels whose RGB distance D_1 from a neighbor pixel already in the region is smaller than T_1.
- Find the above pixels whose RGB distance from the average RGB value of the grow region D_2 is smaller than T_2 and mark these pixels with the same labels as the grow region.
- Process the whole image iteratively until all pixels fitting the criteria are processed.
- At each set of T_1 and T_2, compute the number of regions R in the resulted labeled segmentation.

The RGB distance related to T_1 is computed as:

$$D_1 = \sqrt{(R_p - R_n)^2 + (G_p - G_n)^2 + (B_p - B_n)^2} \tag{3.1}$$

where R_p, G_p and B_p represent the red, green and blue intensity of the current pixel to be grown, and R_n, G_n and B_n represent the red, green and blue intensity of the neighboring pixel already in the region grown.

The second RGB distance related to T_2 is computed as:

$$D_2 = \sqrt{(R_p - R_r)^2 + (G_p - G_r)^2 + (B_p - B_r)^2} \tag{3.2}$$

where R_r, G_r and B_r represent the average red, green and blue intensity of the region already grown. These values are updated every time a pixel joins the region.

In order to select seed pixels for growing regions, we follow several criteria. First, during the seeding process, the grouping always starts from the pixel with the smallest variance to that with the largest variance.

Second, we compute the "colorness" of the pixels by calculating the distance between the pixel's RGB values and the perfect grey vector. In computing the "colorness" of the pixels, we define,

$$U = \frac{(R_{mean} \times 255 + G_{mean} \times 255 + B_{mean} \times 255)}{(255^2 + 255^2 + 255^2)} \tag{3.3}$$

where R_{mean}, G_{mean} and B_{mean} are the mean red, green and blue intensity of the 3×3 window with the pixel being computed at the center. The perfect grey vector is $V_R = 255 \times U$, $V_G = 255 \times U$ and $V_B = 255 \times U$. So the "colorness" is computed as,

$$Colorness = \sqrt{(R_{mean} - V_R)^2 + (G_{mean} - V_G)^2 + (B_{mean} - V_B)^2} \tag{3.4}$$

Any pixel with a "colorness" value lower than a certain value (we set our value at 100) cannot be seeded.

Third, we discard any pixels whose green component is greater than red and blue components, that is when $(G_{mean} - V_G) > (R_{mean} - V_R)$ and $(G_{mean} - V_G) > (B_{mean} - V_B)$. We do so because we don't want to segment green leaves and grass and we only want to seed the brightly colored flowers.

Any pixel with a variance higher than a certain value (we set our value at 100) cannot be seeded either. The reason behind this criterion is that flower pixels tend to have a lower variance than other non-flower pixels.

For any seeding pixel, we have a "nearness" measure. We construct a 5×5 window centering on the seeding pixel. If there is another seeding pixel within this window, the current pixel cannot be seeded. If we grow from this seed, we will probably grow 2 regions in the same flower.

We use a region size filter to control the validity of the regions segmented. The maximum region allowed is 70% of the total image area because a flower can rarely occupy more than such a portion of an image. The smallest region allowed is 0.5% of the total image area, and this measure is to smooth out smaller region noise.

Using these steps, the number of regions R is a function of the first grouping RGB distance T_1 and the second grouping distance T_2:

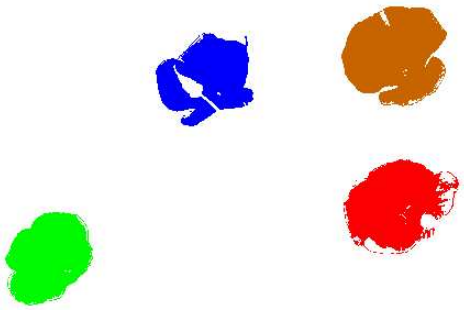
$$R = F(T_1, T_2) \tag{3.5}$$

3.1.2 R-space

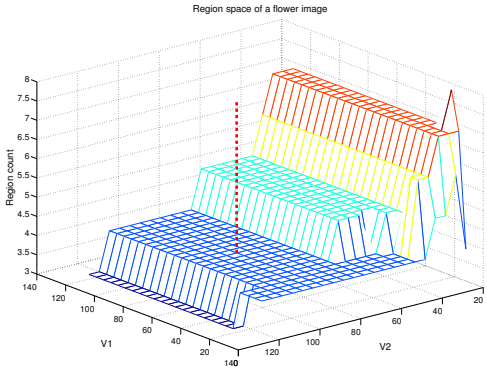
In the implementation of the algorithm to the application of flower detection, the ranges of parameters T_1 and T_2 are selected as $T_1 = 5, 10, 15...125$ and $T_2 = 25, 30, 35...125$. Our R-space will be created in this parameter range.

An R-space example of the RGB-distance variance-based region growing algorithm implemented on a flower image is given in Figure 3.1. The image has 4 separate flowers. The most stable part of the R-space shows a value of 4. This can be seen in the 3-D plot

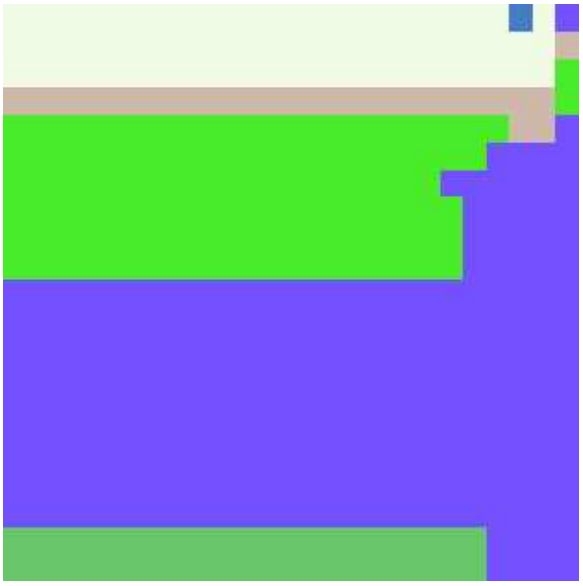
A flower image



Segmentation at the largest plateau ($T_1 = 65$, $T_2 = 95$, $R = 4$).



3-D plot of R-space



R-space viewed as image

Figure 3.1: An example of variance-based region growing with different pairs of standard deviations.

as the largest plateau. It is even more obvious when viewing the R-space as a 2-D image, where the continuous central area has a value of 4. The parameter values selected for producing the final segmentation are $(T_1 = 65, T_2 = 95)$, which is roughly in the middle of that plateau. It is worth mentioning that in the 2-D image, the color of the blocks represents the height of the plateau. If the colors of two blocks are the same, it means they are at the same height (same number of regions detected) even though they are not connected.

3.1.3 Data set

We have collected 44 flower images from various public websites. In these flower images, there are different colors of flowers, different numbers of flowers, different clusters and different backgrounds.

Six example flower images are presented in Figure 3.2. In the 44 flower images, the number of flowers varies from 1 to 12. Table 3.1 shows the number of flowers in each image for all of the 44 flower images.

3.1.4 Results and evaluation

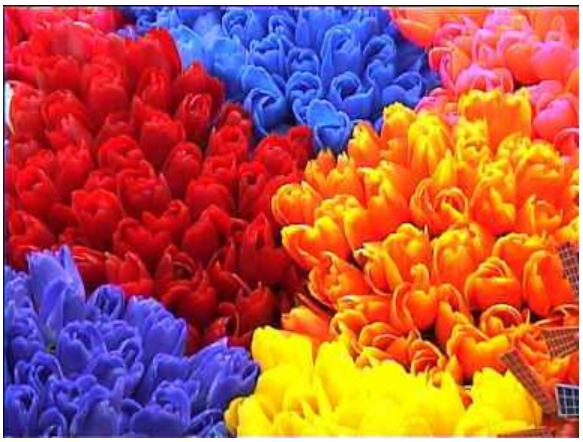
We processed these flower images using the algorithm of RGB-distance variance-based region growing. To verify our methods, we list the ground truth count of the number of flowers in these images and the number count from computer generated result in Table 3.3. The table shows that only a difference of 6 out of 44 images between the computer result and the ground truth. That is 13.6% in disagreement and 86.4% in agreement. In most of the successful detection cases, the number of flowers is a definite countable number. In the 6 failed cases, however, 2 of them are patches of flowers, the number of which is not a definite number even for a human viewer; the rest of the cases are that the number of flowers is definite, but the background is noisy and some pixels there grows as if it were a flower pixel. It is interesting that although the numbers of flowers don't match in these failed cases, the segmentations produced are the best for these images.

We also try to see where are all the parameter pairs (T_1, T_2) are located. In Figure 3.3, it is easy to see that the parameters corresponding to the center of the largest plateaus are scattered in almost the entire parameter ranges. That is to say, our framework is not one that finds a common good parameter set for a group of similar images, but instead it can adapt to difference in a group of specific images and produce the best parameter values for each image differently from others.

1 flower

1 flower

2 flowers



6 flower patches

1 flower

4 flowers

Figure 3.2: Example of flower images.

Table 3.1: Table of All Flower Images

Image Name	Image Description	Number of Flowers
Flower image No.1	Red flower	1
Flower image No.2	Yellow flower	1
Flower image No.3	Yellow flowers	4
Flower image No.4	Bouquets of flowers of several colors	6
Flower image No.5	Bouquets of flowers of several colors	2
Flower image No.6	Yellow sunflower	1
Flower image No.7	Red and yellow flowers	2
Flower image No.8	Magenta flower	1
Flower image No.9	Yellow flowers in green leaves	12
Flower image No.10	Yellow flower in green leaves	1
Flower image No.11	Red flower	1
Flower image No.12	Red flower	1
Flower image No.13	Red flower	1
Flower image No.14	Red flower	1
Flower image No.15	Pink flower	1
Flower image No.16	Pink flower	1
Flower image No.17	Red flowers	2
Flower image No.18	Yellow flower	1
Flower image No.19	Orange flower	1
Flower image No.20	Yellow flower	1
Flower image No.21	Yellow flower	1
Flower image No.22	Yellow flower	1
Flower image No.23	Yellow flower	1
Flower image No.24	Yellow flower	1
Flower image No.25	Yellow flower	1
Flower image No.26	Red flower	1
Flower image No.27	Red flower	1
Flower image No.28	Red tulips	1
Flower image No.29	Pink flower	1
Flower image No.30	Yellow,red and violet flowers	4
Flower image No.31	Pink flower	1
Flower image No.32	Yellow and red flowers	3

Table 3.2: Table of All Flower Images (continued)

Flower image No.33	Yellowgreen and pink flowers	7
Flower image No.34	Yellow flowers	7
Flower image No.35	Pink flowers	1
Flower image No.36	Pink flowers	2
Flower image No.37	Red flowers	3
Flower image No.38	Orange flowers	2
Flower image No.39	Yellow and red flowers	4
Flower image No.40	Orangered flowers	8
Flower image No.41	Violetred flowers	1
Flower image No.42	Red flowers	3
Flower image No.43	Red flowers	7
Flower image No.44	Pink flowers	1

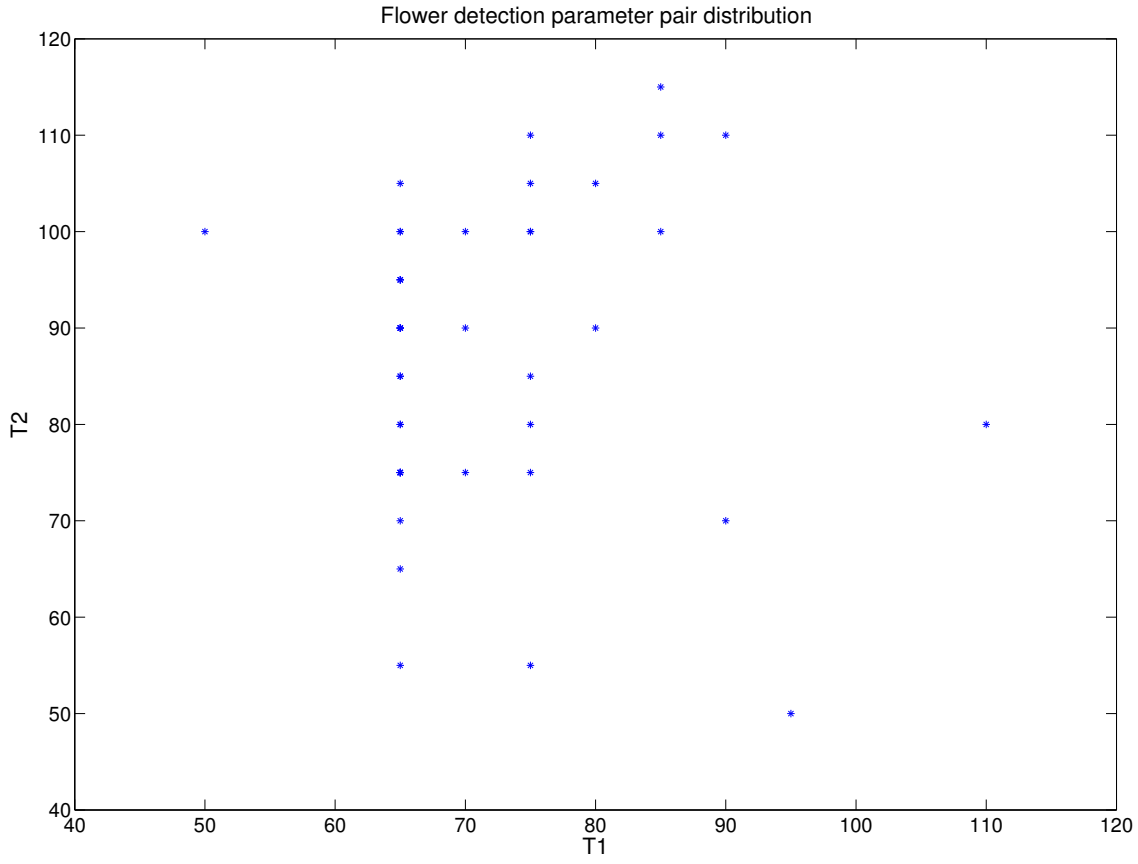


Figure 3.3: Parameter distribution of the flower detection.

Table 3.3: Table of Flower Count Ground Truth and Computer Results

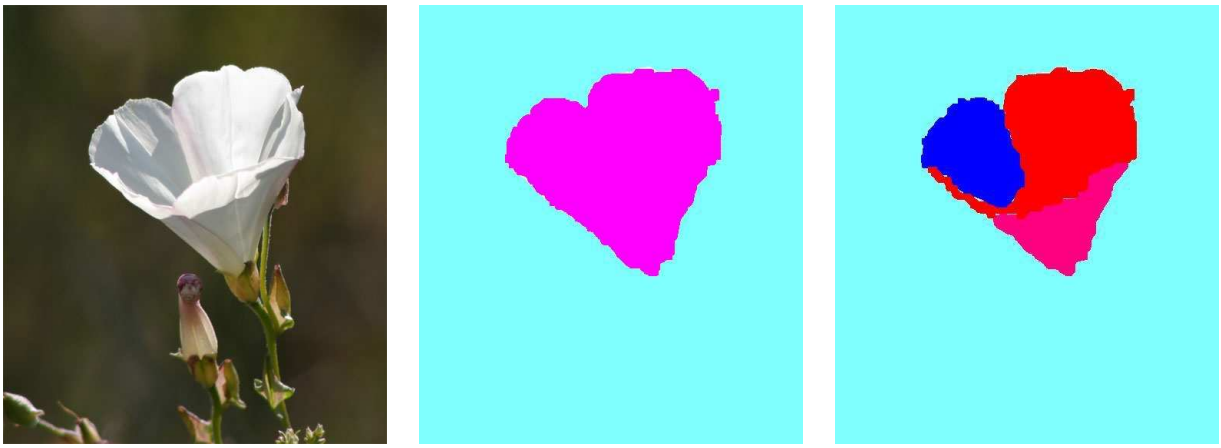
Image Name	Ground Truth	Computer Result	Match
Flower image No.1	1	1	
Flower image No.2	1	1	
Flower image No.3	4	4	
Flower image No.4	6	8	failed
Flower image No.5	2	2	
Flower image No.6	1	1	
Flower image No.7	2	2	
Flower image No.8	1	1	
Flower image No.9	12	12	
Flower image No.10	1	2	failed
Flower image No.11	1	1	
Flower image No.12	1	1	
Flower image No.13	1	1	
Flower image No.14	1	1	
Flower image No.15	1	1	
Flower image No.16	1	1	
Flower image No.17	2	2	
Flower image No.18	1	1	
Flower image No.19	1	2	failed
Flower image No.20	1	1	
Flower image No.21	1	1	
Flower image No.22	1	1	
Flower image No.23	1	1	
Flower image No.24	1	1	
Flower image No.25	1	1	
Flower image No.26	1	1	
Flower image No.27	1	1	
Flower image No.28	1	2	failed
Flower image No.29	1	1	
Flower image No.30	4	2	failed
Flower image No.31	1	1	
Flower image No.32	3	3	

Table 3.4: Table of Flower Count Ground Truth and Computer Results (continued)

Flower image No.33	7	1	failed
Flower image No.34	7	7	
Flower image No.35	1	1	
Flower image No.36	2	2	
Flower image No.37	3	3	
Flower image No.38	2	2	
Flower image No.39	4	4	
Flower image No.40	8	8	
Flower image No.41	1	1	
Flower image No.42	3	3	
Flower image No.43	7	7	
Flower image No.44	1	1	

Besides all the above objective evaluation metrics, we have also designed an interesting experiment of subjective evaluation by human participants. In the objective test, we only match up the number of flowers detected. We need to know if the quality of the segmentations are good as well. In the failed cases of the objective test, even though the numbers don't match, the segmentations are still the best. It is also possible that even though the numbers match, but the related segmentations are not the best. As a result of all these speculation, we need human viewers to compare the segmentations at the parameter values selected by the computer with that in the other parts of the R-space.

We generate the region space of these flower images, and analyse the stability of these region spaces. The largest plateau (flat area) in the region space is the parameter set where we believe there is good segmentation of the original image. We create segmentations of the image at the center of the largest plateau as well as two other segmentations at medium and small plateaus. The medium and small plateaus are picked randomly from a list of plateau sizes in the R-space. We have created a website where the original image stands side by side with three segmentations corresponding to the three plateaus, big, medium and small. The three segmentations are displayed in random order on the webpage.



Original image Good segmentation Bad segmentation

Figure 3.4: Good and bad segmentations in the instruction.

After that, we have asked 11 people to view the 44 flower images on our website and fill out a table telling us which segmentation is the "best segmentation" of the original image. In the instructions given to the participants, we define the "best segmentation" as,

- One single flower should be best segmented into one segmentation;
- For multiple flower clusters, same color flowers should remain together in the segmentation;
- The single flower shape should be as full as the original.

We also show the participants an example of good segmentation and bad segmentation as in Figure 3.4.

Among these 11 people, 5 people are faculty or undergraduate or graduate students from Psychology Department, whom we assume to have little image processing knowledge or background; 6 others are graduate students from Electrical and Computer Engineering Department, whom we assume to have some image processing knowledge or background. We believe the different backgrounds of the participants could help us to get a more objective evaluation and avoid the bias that may result from knowledge or lack of knowledge in the image processing field.

We compare the human responses with computer results (the segmentations at the largest plateau in the R-space), and count how many they match each other and how many they don't. The polls show that all 11 people agree our machine results at an overall rate of 90.90%. If we separate the psychology group from the ECE group, 5 psychology people agree with our machine results at an average rate of 88.63% (from 81.81% to 95.45%), while ECE people agree at an average rate of 92.80% (from 81.81% to 97.73%). ECE people score better than the psychology people at a mere difference of 5%, which shows that their knowledge may have a somewhat positive impact on their selection. The polls show that our method can generate the best segmentations that most human viewers agree with at an percentage of roughly 90%.

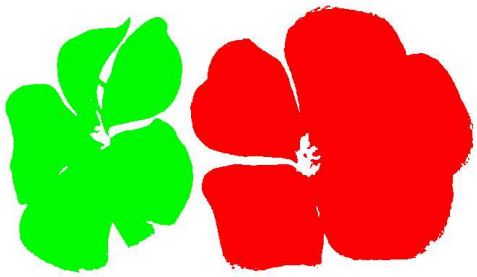
We show a sample of flower images and segmentations in Figure 3.5 and Figure 3.6. Most other segmentations are very similar to the samples shown. All the flower images and segmentations in this experiment can be viewed somewhere in the Parl system of Electrical and Computer Engineering Department. Interested parties should contact Dr. Adam Hoover for information as to where they are.

In this experiment, we have collected some flower images from various public sources for processing with our stable count framework. We have designed an objective test and a subjective evaluation test. The objective test measures matching of the number of flowers detected automatically and visually. The subjective test measures how much the human viewers agree with the segmentations produced by the computer algorithm. The results from both tests show our framework can produce the best segmentation. Therefore our stable count idea can be used to process a specific group of images such as flower images.

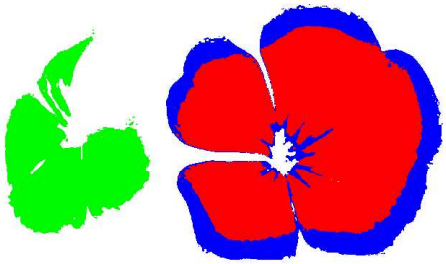
3.2 Peak detector

For this problem, we want to show how our framework can be applied to problems besides image segmentation. We develop a synthetic peak generator and peak detector for

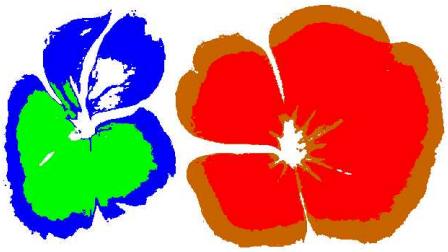
A flower image (2 flowers)



Segmentation corresponding to the largest plateau



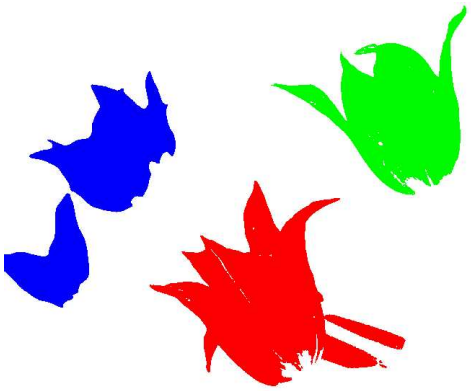
Segmentation corresponding to a medium plateau



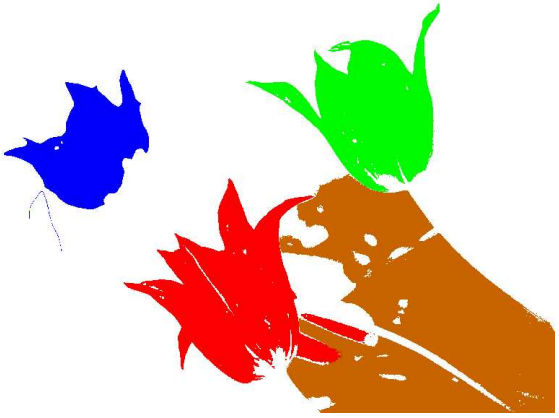
Segmentation corresponding to a small plateau

Figure 3.5: A flower image and segmentations from different sizes of stable count plateaus.

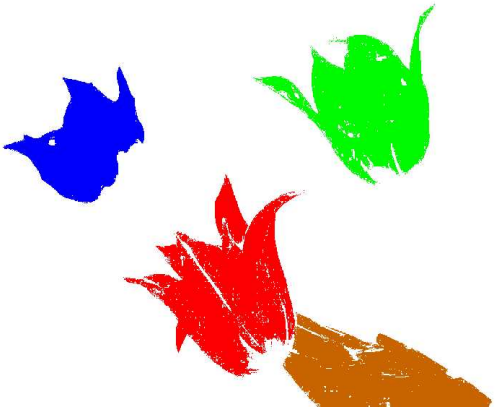
A flower image (4 flowers)



Segmentation corresponding to the largest plateau



Segmentation corresponding to a medium plateau



Segmentation corresponding to a small plateau

Figure 3.6: A flower image and segmentations from different sizes of stable count plateaus.

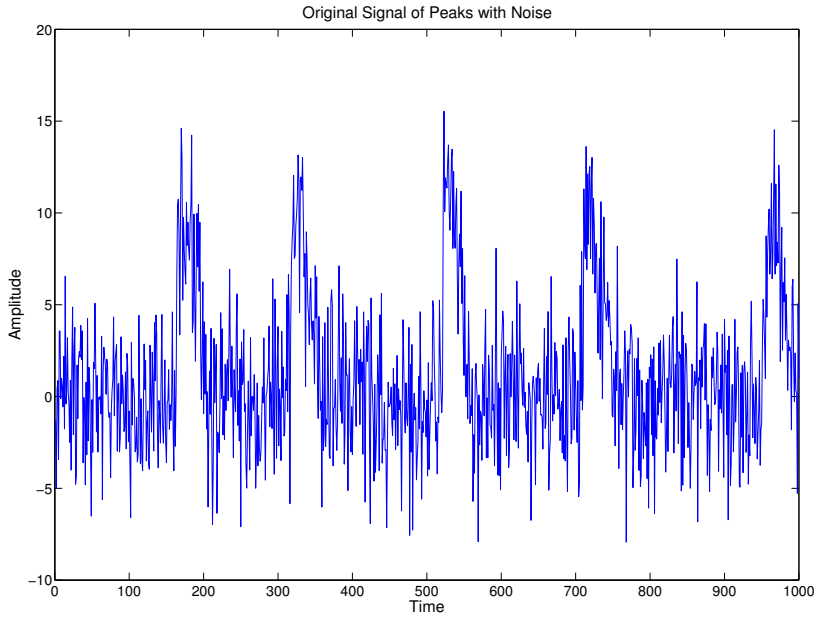


Figure 3.7: A 1-D signal with Gaussian noise with 5 peaks.

a 1-dimensional signal over time. This type of operation is common in signal processing problems. Figure 3.7 shows a generated 1-D signal with 5 peaks. The signal is mixed with some Gaussian noise. Our peak detector uses a threshold to detect when a peak has occurred, along with a Gaussian filter to smooth out noise in the signal. We use our methods to analyze the R-space of the peak detector to select parameter values automatically. We generate a number of different example signal inputs, with varying amounts of noise and peak thresholds. We evaluate our results according to the correct determination of peaks.

3.2.1 Algorithm

We call this algorithm peak detector. The basic idea of the algorithm is to smooth a 1-D signal over time and threshold to identify peaks or pulses. This algorithm comes from the idea of stable view or stable count, but is applied to general signal processing.

The basic steps of this thresholding and smoothing algorithm are as follows:

- Use Gaussian function to smooth each data point of the signal with neighboring data values. Each Gaussian smoothing is characterized by a parameter of standard deviation σ.
- At each σ, separate the possible range of the whole signal at certain equally spaced thresholds T, from low to high.
- At each set of values σ and T, find the number of peaks R the signal has.

The smoothing is implemented using the following Gaussian function:

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.6}$$

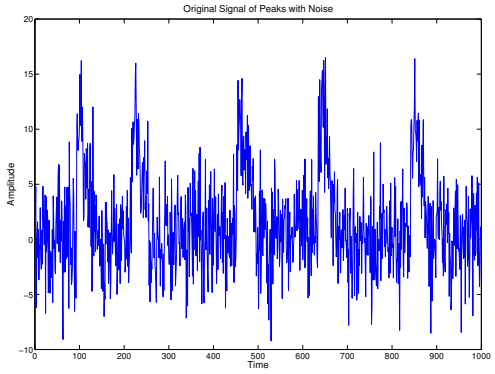
where we use the standard Gaussian function with $\sigma = 1$ and $\mu = 0$. Suppose we have a signal $S(i)$, $i = 1...1000$. The signal is tainted with uncertain amount of Gaussian noise. We use a window size W to process the signal with Gaussian smoothing. Therefore the smoothed signal is computed as follows:

$$S_{Smoothed}(i) = \frac{1}{\sqrt{2\pi}} \sum_{j=1}^{\sigma} e^{-\frac{(j-1)^2 W}{2\sigma^2}} S(i-j+1) \tag{3.7}$$

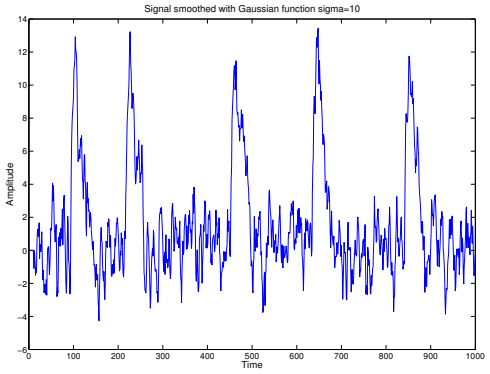
where $i = 1...1000$ and $j = 1...\sigma$. To be counted as a peak, the signal value at that point should be higher than the two adjacent points and higher than the threshold T.

Using these steps, the number of peaks is a function of the threshold T and the standard deviation σ.

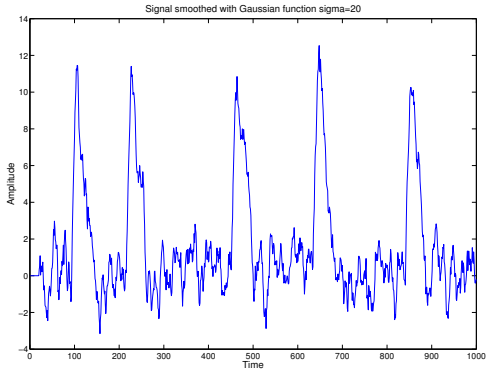
$$R = F(T, \sigma) \tag{3.8}$$



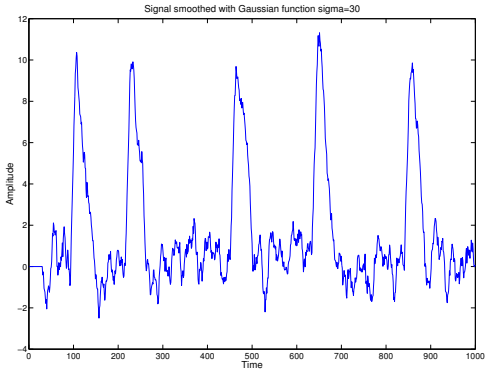
A signal with uncertain Gaussian noise



Signal smoothed with $\sigma = 10$



Signal smoothed with $\sigma = 20$



Signal smoothed with $\sigma = 30$

Figure 3.8: An example of a noisy signal and the effect of Gaussian at different scales.

An example of the effect of the smoothing algorithm on a signal with Gaussian noise is given in Figure 3.8.

The smoothing is showing effect after increasing the σ. To detect the real significant peaks while not duped by those small peaks caused by noise, we only need to count the number of peaks by searching for the largest plateau in the 2-D space along the threshold (height of the signal) axis and smoothing (σ) axis. The stable count of peaks in this space will pinpoint to the number of real peaks in the signal.

3.2.2 Data set

In this experiment of peak detection, we use some generated signals that have some Gaussian noise rather than images. The generated signals: (1) contain any number of peaks if

necessary; (2) contain peaks generated by a Gaussian function; (3) have random spacings between peaks; (4) have added Gaussian noise of different means and standard deviations.

The noise of the signal is added by multiplying a standard Gaussian signal by a signal of noise σ. We try to use different combinations of peak heights, peak numbers, noise standard deviations to construct the noisy signals. A list of signals we have tested is in Table 3.5.

Table 3.5: Table of Some Generated Signals

No.	Peak height	Number of Peaks	Standard deviation σ_{Noise}
1	16	6	3.0
2	26	5	6.0
3	12	5	2.0
4	18	5	2.0
5	16	6	3.0
6	8	6	1.0
7	28	2	1.0
8	28	2	1.0
9	28	3	5.0
10	18	2	4.0
11	15	5	5.0
12	12	7	3.0
13	12	7	1.0
14	12	7	5.0
15	10	2	3.0
16	10	2	5.0
17	16	6	3.0
18	10	2	2.0
19	12	2	4.0
20	16	6	3.0
21	12	3	6.0
22	12	2	8.0
23	12	8	8.0
24	10	5	1.0
25	10	6	2.0
26	10	6	3.0
27	12	3	5.0
28	10	2	6.0
29	16	6	6.0
30	10	5	6.0

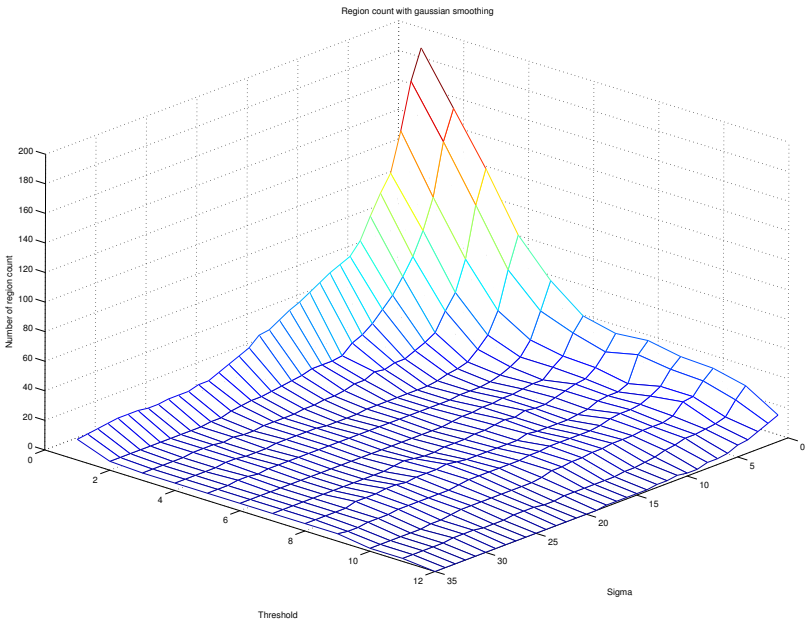
For the 30 generated signals in the table, we have used different combinations of peak height, number of peaks and noise standard deviations. Changing the number of peaks and peak height is relatively easy for any algorithm to adapt to, but the standard deviation of the noise creates more difficulty for detection. In this table we have tested a maximum $N_\sigma = 8$, which can create quite a large noise for detection. We believe these different noisy signals have represented most real situations.

3.2.3 R-space

The R-space of the signal in Figure 3.7 after implementation of the above algorithm is shown in Figure 3.9. As the R-space is not so easy to see in 3-D plot, we use an image to show the R-space with the intensity of pixels representing the region count (the Z amplitude in 3-D plot). The largest plateau is the one on the right side, but its region count is 0, so it can be ignored. The next largest plateau is the one located in the lower center, and the reading in the Z axis is 5. The number of peaks in the noisy signal in Figure 3.7 is 5 too. So the algorithm has correctly detected the number of peaks in the signal through the largest (valid) plateau in the R-space.

3.2.4 Results and evaluation

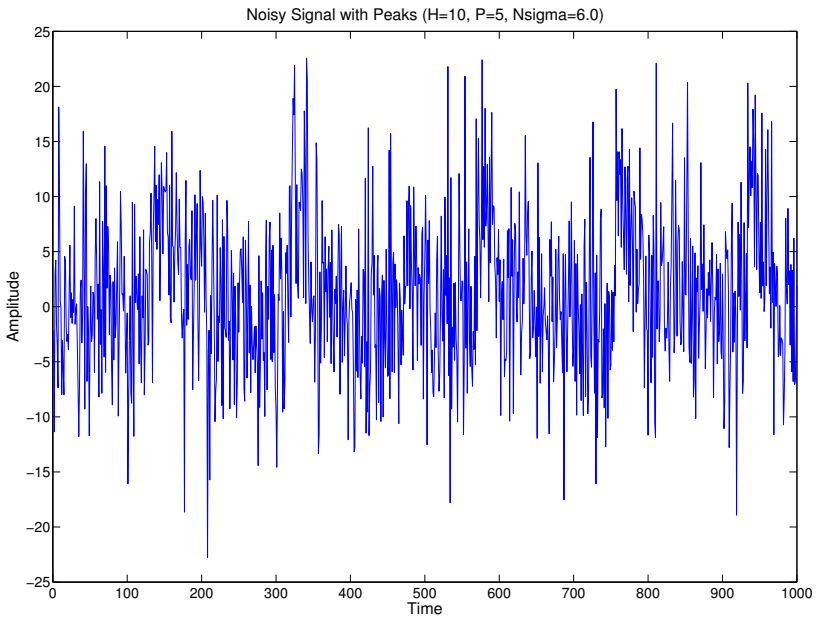
We implement the thresholding and smoothing algorithm with more than a dozen combinations of noises with different means and standard deviations. In all these various situations, the algorithm can correctly detect the number of peaks in these noisy signals. We show a 5-peak signal in Figure 3.10, and the R-spaces both in 3-D and image in Figures 3.11. The N_σ in this case is 6.0 ($SNR = 0.6$), so we can only vaguely identify 5 peaks. From the image of R-space, the largest plateau on the lower right side only has a region count of 0, so it is not the plateau we are looking for. The next largest plateau is the one on the lower center to the lower left of the zero plateau. It is the valid largest plateau and it reads 5. So the 5 peaks are detected in the R-space by checking this plateau. We show another signal



3-D plot of R-space

R-space viewed as image

Figure 3.9: The R-space of the signal after implementation of the peak detection algorithm.

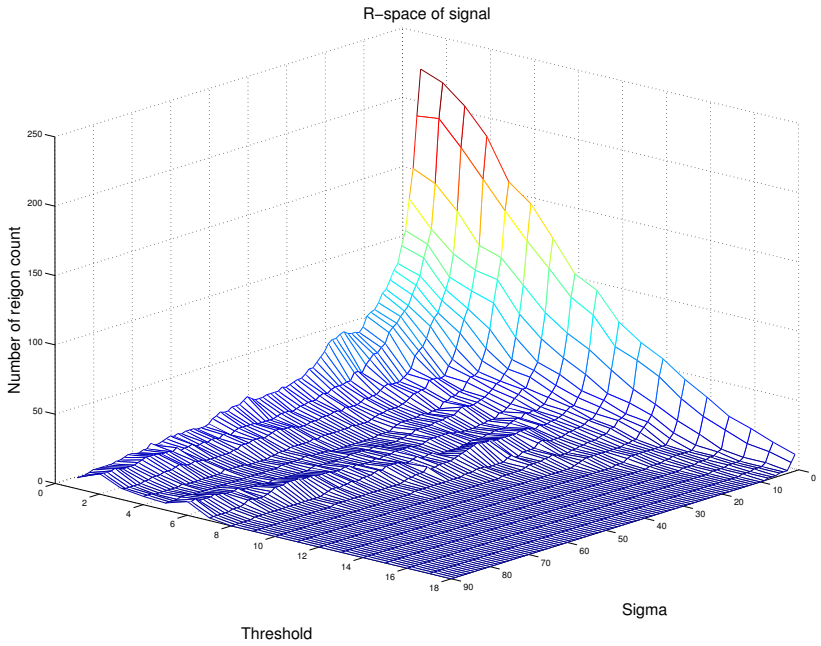


The generated noisy signal (peak height=10, 5 peaks, $\sigma_{Noise} = 6.0$ and $SNR = 0.6$)

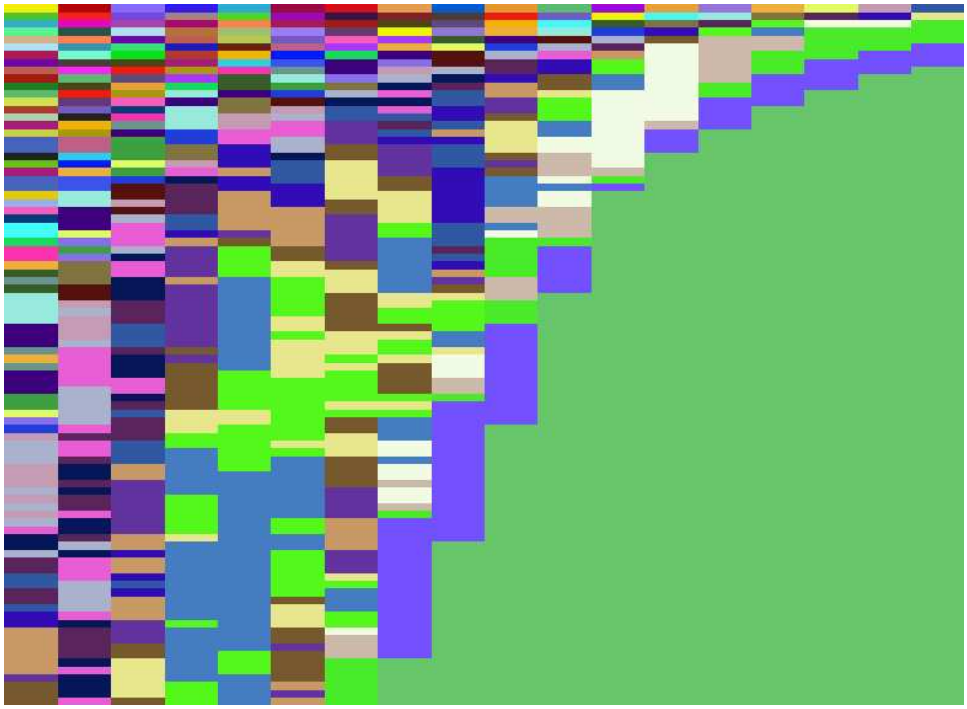
Figure 3.10: A noisy signal with 5 peaks.

with 8 peaks in Figure 3.12, and the R-spaces both in 3-D and image in Figure 3.13. The N_σ in this case is 8.0 ($SNR = 0.55$), and the 8 peaks are less obvious than the previous 5-peak signal. From the R-space image, the largest plateau on the right side has a region count of 0. The next largest plateau is the one occupying 3 columns in the lower middle of the image to the left of the 0-plateau, and its region count is 8. So the number of peaks is correctly detected by the largest (valid) plateau in the R-space. The number of peaks of all the generated noisy signals listed in Table 3.5 are successfully detected.

We design this simulation experiment just to show that our idea of stable count can be utilized to solve problems other than image segmentation. The solution can be applied to finding the number of peak seasons in business sales, or the number of rain seasons in a year, or the number of extreme summer temperatures in a whole season, or the number of critical water levels in a reservoir, or any other "similar problems". From this experiment, we show that our idea of stable count can not only work on images but also on general signals with noise.

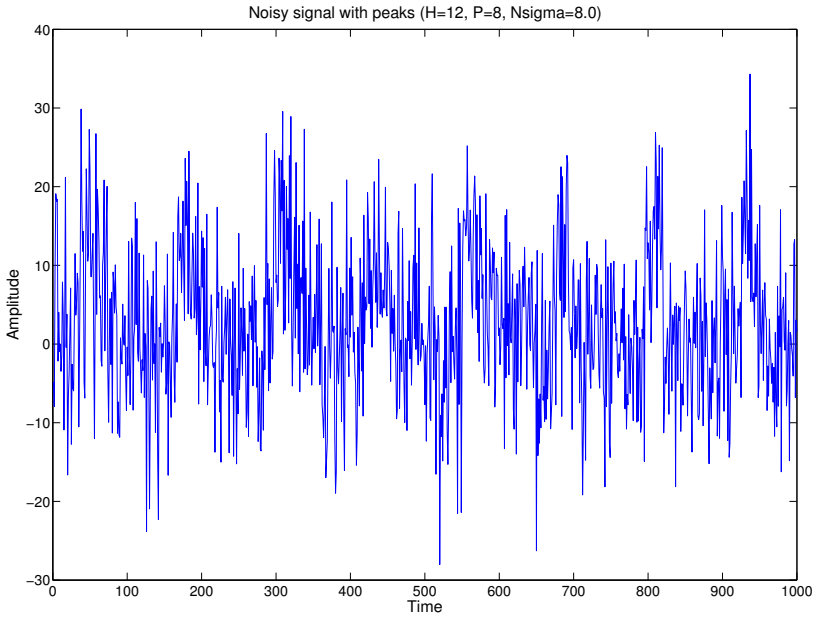


The R-space of the signal with 5 peaks.



R-space viewed as image.

Figure 3.11: The region space of 5-peak signal after the algorithm is implemented.

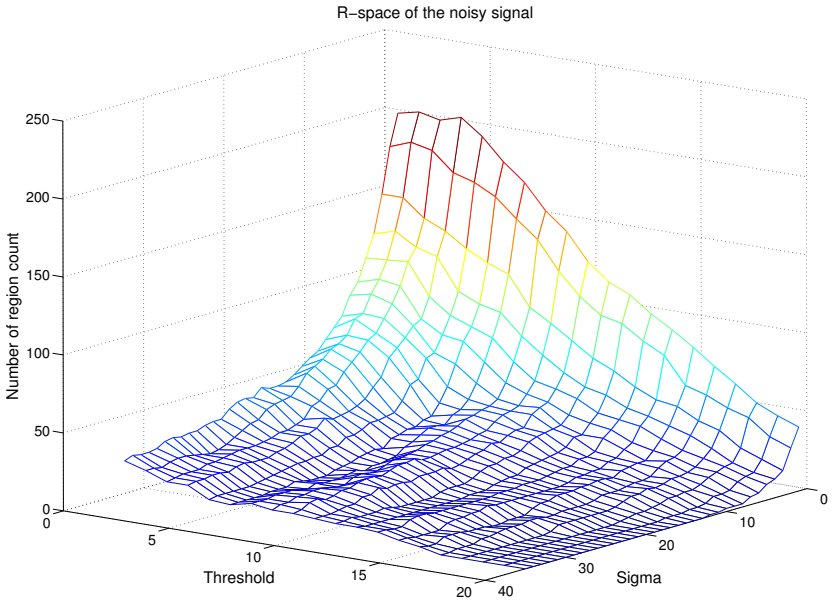


The generated noisy signal (peak height=12, 8 peaks, $\sigma_{Noise} = 8.0$ and $SNR = 0.55$)

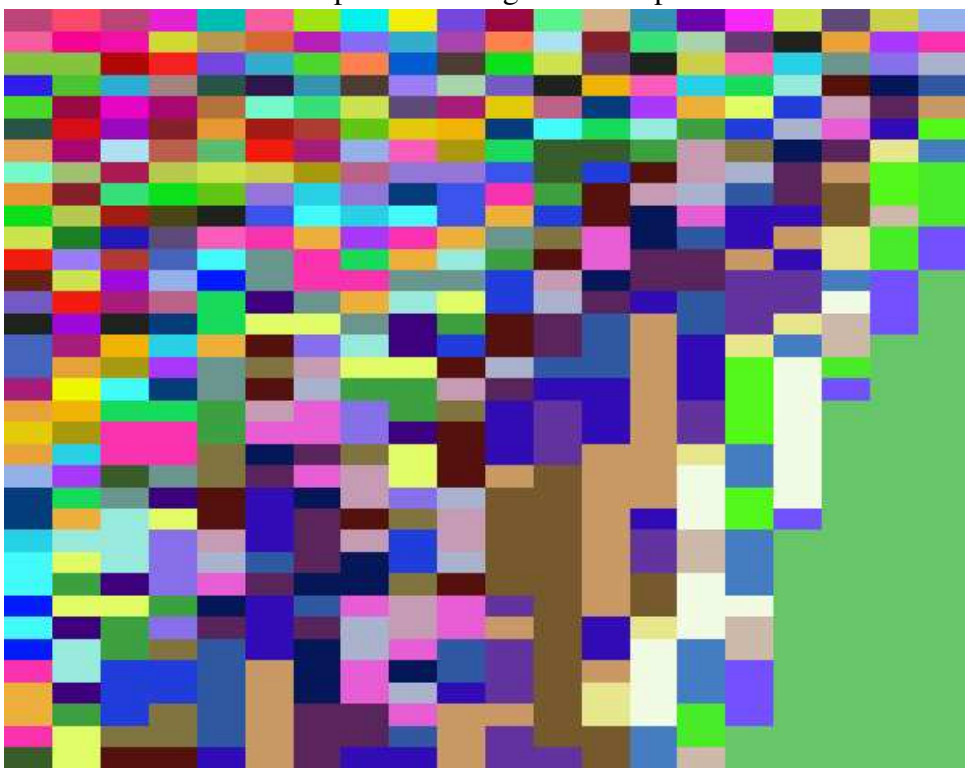
Figure 3.12: A noisy signal with 8 peaks.

3.3 Retinal image lesion detector

For this problem, we want to detect bright lesions in any given retinal image. This problem is different from the previous problems in that it is much more difficult to define the ground truth. In a few images, there is a uniquely identifiable count of lesions, upon which most viewers would agree. But for most images, there is not necessarily only one unique count. Some lesion patches may be viewed as either a high count of smaller areas, or a low count of larger areas, depending upon how the viewer combines the ill-defined boundaries of overlapping lesions. Therefore we formulate a stability analysis that is more than simply identifying the largest plateau in the R-space. Instead, we look for the stability of region count across a range of scales, where the scale is determined by how much nearby lesions are merged together. Basically, our algorithm thresholds the image, and then merges nearby areas.



The R-space of the signal with 8 peaks.



R-space viewed as image.

Figure 3.13: The region space of 8-peak signal after the algorithm is implemented.

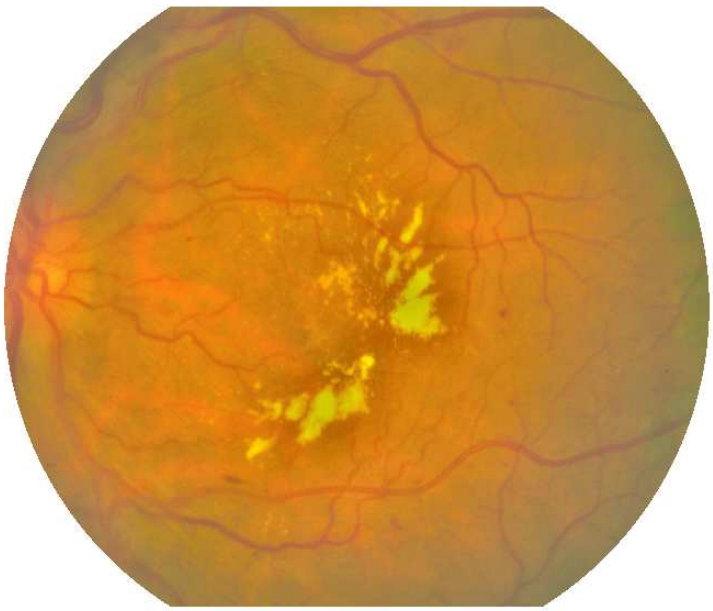


Figure 3.14: A retinal image.

Figure 3.14 shows a retinal image with lesions. The lesions in this image are the abnormally bright areas in the center. There is stable view of lesions in this image, but the exact number of region count is hard to determine because the lesions are so close to each other that it is difficult to count several neighboring lesions as a combined 1 or their individual numbers (This is true for most retinal images containing lesions.). As the intensities of these lesions are not identical, but rather span a certain range, a simple thresholding algorithm will not work. Therefore, we use our stable region count framework by constructing an R-space through counting the number of regions segmented as the threshold is varied. The stability analysis depends on finding a range for the threshold in which the number of regions segmented shows the least change.

3.3.1 Algorithm

We get the region count by thresholding and region growing. The steps are:

- Threshold the original image using a value T.

- Grow the thresholded image by an amount S.
- Count the number of regions R.

Therefore, we get region count R as a function of threshold T and growth S:

$$R = (T, S) \tag{3.9}$$

Since the number of region count is not a definite number at different growth S, we consider the change in the number of regions across the whole range of S as threshold T is varied. In this 1-D R-space of T, we seek to find a range of values of T with multiple scales of S where the number of regions R remains relatively constant. The wider the range, the more perceptually "stable" the segmentation. We define the function

$$\Delta R = \int \frac{dR}{dT} dS \tag{3.10}$$

The value ΔR denotes the difference in R across all S as T changes. Integrating S this way allows us to quantify how the number of regions changes across all "blurrings" of the threshold T. A ΔR plot of a retinal image is shown in figure 3.15.

We desire to find the local minima in the function ΔR. Depending on the image content, ΔR can be noisy. Some local minima are not of interest, where the stable range of $F(T, S)$ is narrow. In order to bring out the more important local minima, we apply Gaussian smoothing to ΔR:

$$G(\Delta R, \sigma) = \Delta R e^{-\frac{\Delta R^2}{2\sigma^2}} \tag{3.11}$$

Figure 3.16 shows four plots of G with different amounts of smoothing (at values of $\sigma = 0.5$, 1.0, 2.0, 4.0), computed from the ΔR in Figure 3.15. Equation 3.11 is in the classic form for scale-space processing (see for example [33], pg. 88). One way to use Equation 3.11

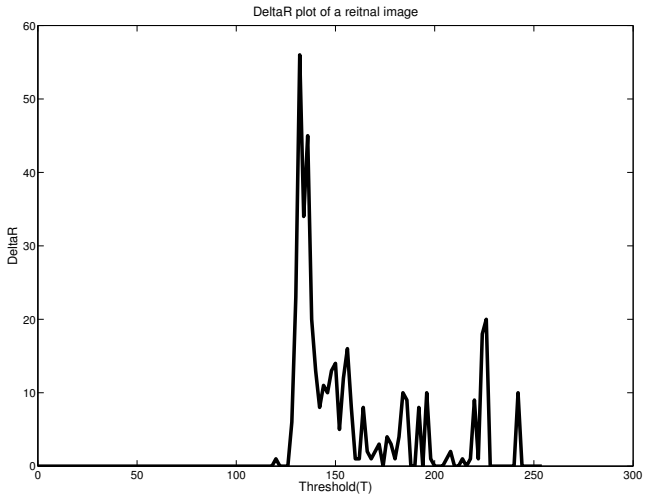


Figure 3.15: A plot of ΔR.

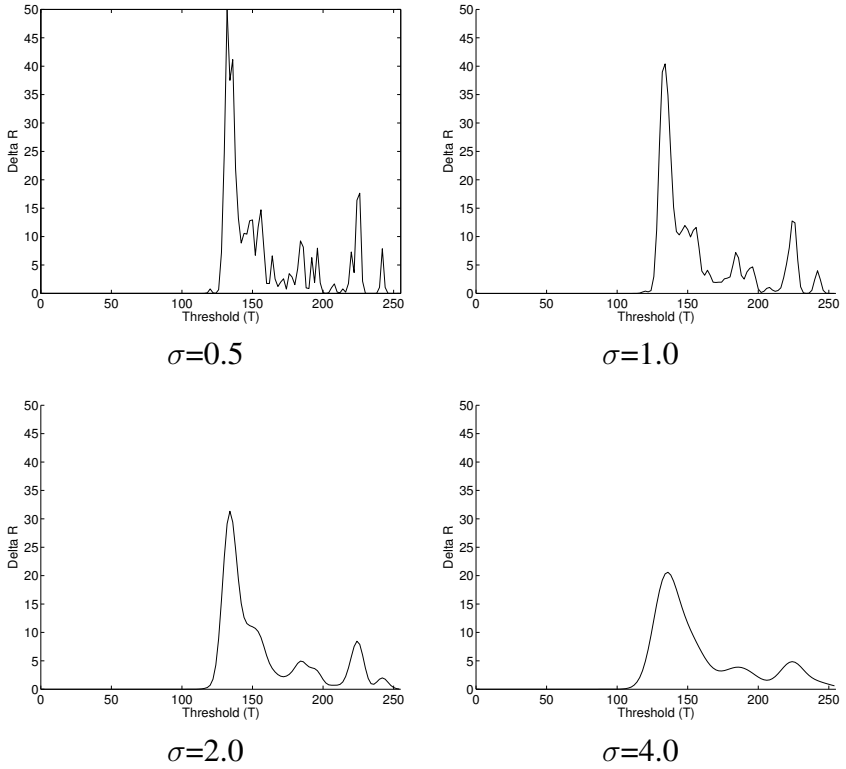


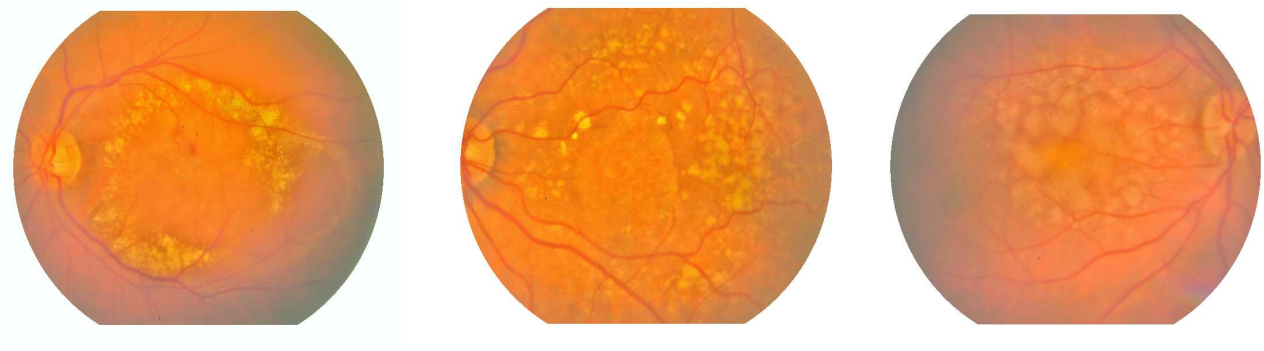
Figure 3.16: The effect of varying σ on $G(\Delta R, \sigma)$.

is to increase σ until only one local minimum is found in $G(\Delta R, \sigma)$. This should produce the "most stable" segmentation. However, in some cases multiple "perceptually strong" segmentations may be possible. In this case σ should be increased until only a small number of minima are found in $G(\Delta R, \sigma)$. We detail our specific use of Equation 3.11 in the results and evaluation section.

3.3.2 Data set

To verify our idea of stable count, we select the STARE database of retinal images[10, 11]. The group of retinal images contains 397 images (Each image is 605×700 8-bit pixels in resolution.) with different kinds of lesions such as drusen, exudates, cotton wool spots and other commonly occurring symptoms. Their shape, boundaries, cardinality (count) and patterns are difficult to definitively identify.

A few retinal examples are shown in Figure 3.17. The three example retinal images contain common lesions. Figure 3.17(a) shows a large crescent-shaped patch of lesion of medium contrast. Figure 3.17(b) shows dozens of small lesions of varying shapes, contrasts, and amounts of merging with neighbors. Figure 3.17(c) shows a large amorphous patch of lesion of faint contrast, with no easily discernible boundaries. (Note: For all the images in this paper, the global contrast has been adjusted for clarity in printing.) Often, an ophthalmologist refers to the presence of these types of lesions through a qualifying adjective, such as "many drusen" or "few drusen", because it is impossible to identify individual lesion boundaries. Work in automating the detection of these lesions is therefore hampered by the fact that even a trained human expert cannot produce a definitive pixel-level segmentation, or even a specific count of lesions against which detected "blobs" can be compared.



(a) (b) (c)

Figure 3.17: Example retinal images, showing variability in lesion size,shape and contrast.

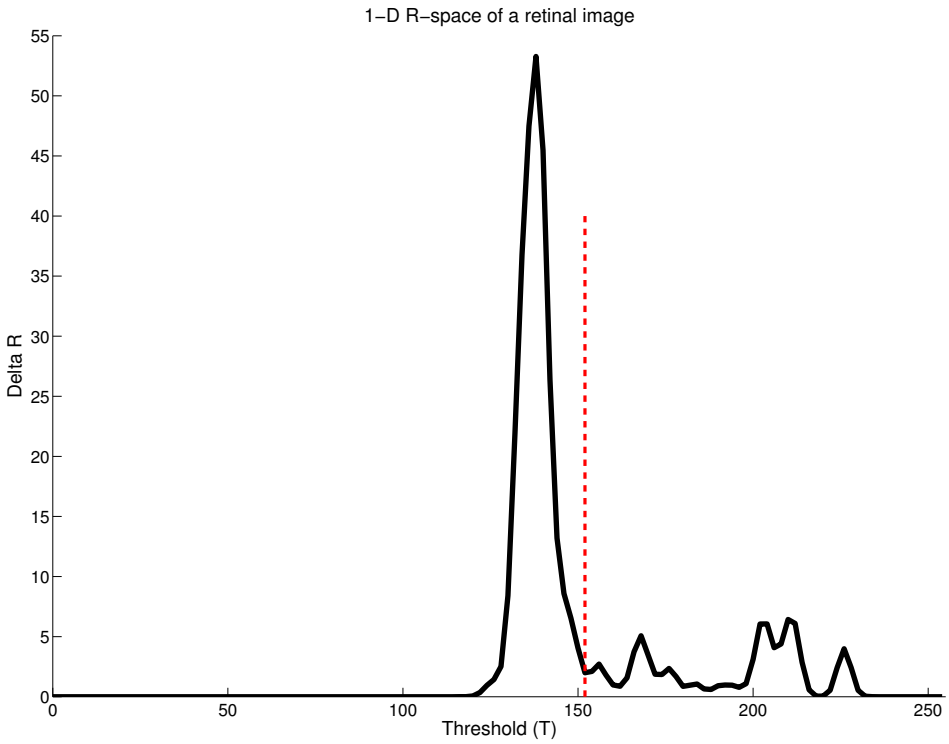
3.3.3 R-space

We show an example of a retinal image and the 1-D R-space in Figure 3.18. The R-spaces of other retinal images are very similar to this example. In the R-space, there is always a global peak and some local peaks and valleys right before and after the global peak. The global peak is where the greatest change is taking place in perception and in segmentation (The number of regions changes dramatically and segmentations change greatly as a result.). We select the final threshold T by finding the global maximum in $G(\delta R, \sigma = 1.0)$, and then finding the local minimum closest to that peak and which has a value lower than 65% of the value at the global maximum. Our selection of T finds the threshold near this volatile range at which the segmentation is most stable.

From the Figure 3.18, after averaging region counts across all growth S and applying the Gaussian smoothing to ΔR, the stability occurs at the lowest $G(\Delta R, \sigma)$ after the peak at approximately threshold $T = 152$, as indicated by the broken red line. The thresholds of all retinal images are determined in this manner.

3.3.4 Results and evaluation

We implement Equation 3.10 by thresholding the image at every interval of $dT = 2$ $(0, 2, 4, ..., 254)$, and growing at intervals of $dS = 5$ $(0, 5, 10, ..., 50)$. These ranges (instead of $dT = 1$ and $dS = 1$) were chosen to speed up the processing of an image. We



The 1-D R-space of the retinal image.

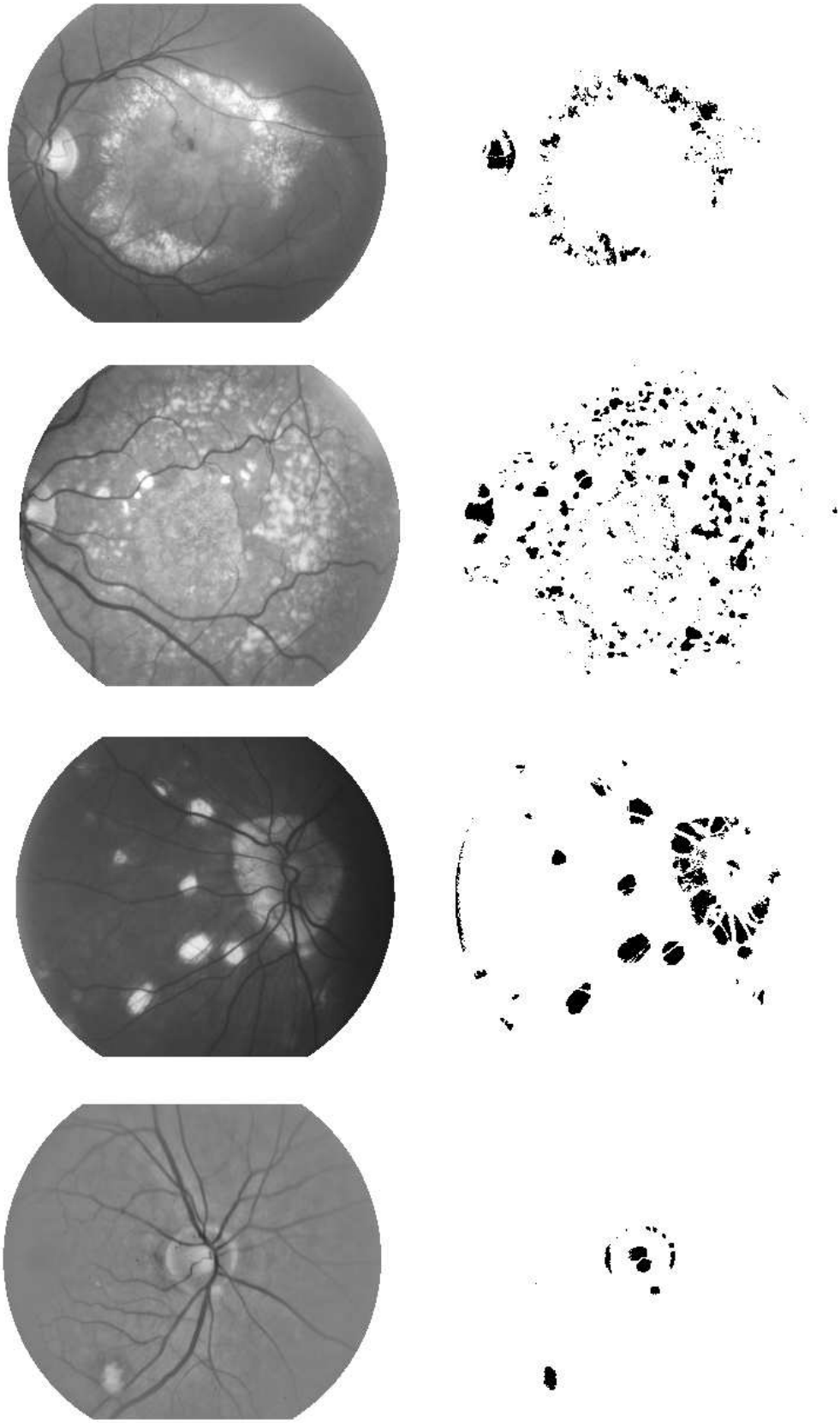
Figure 3.18: Example 1-D R-space of the retinal image.

chose 50 as a maximum value for S because in our experience growing beyond this limit does not match any strong perception of the image.

We implement Equation 3.11 using σ = 0.5, 1.0, 2.0, 4.0, to search for reasonable quantities of local minima. Eventually we decided to use a fixed $\sigma = 1.0$, giving reasonable results in most cases.

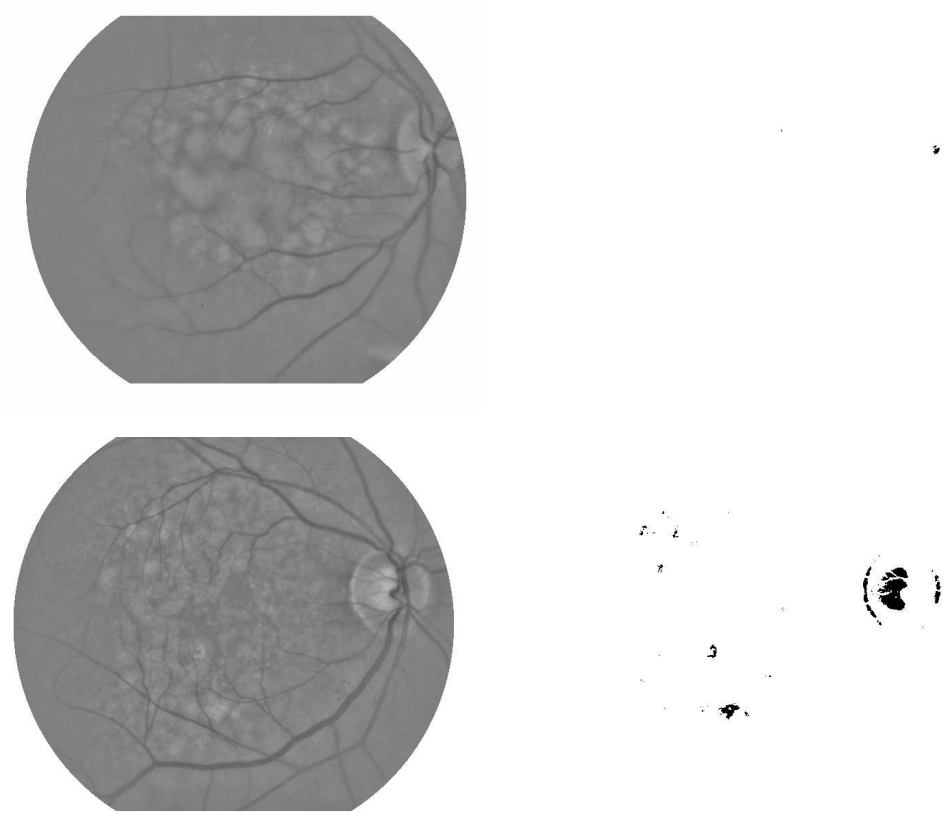
To subjectively evaluating our results, we decide if the segmentation has captured all of the lesions in the retinal images. If all of the lesions are not captured, we consider the segmentation a failure. We show some successful segmentations in Figure 3.19. In the four retinal images shown in the figure, the size, distribution, intensities, locations, number, background of lesions are different from each other. However, our algorithm has successfully segmented all the lesions from the image. All the 382 successful cases are similar to the ones shown in this figure. We show 2 failed segmentation cases in Figure 3.20. In both cases like all other failed cases, the lesions are faint and difficult even for a human view to perceive "strong segmentation". As a result, the algorithm cannot catch all the lesions but instead only segments out the normally bright lesions without the other large amount of faint lesions. In this manner, we find satisfactory segmentations on 382 of 397 retinal images.

Through this experiment we have built a 1-D R-space for 397 retinal images and test our framework of stable region count. We search for the best threshold by analyzing the stability of the R-space. The thresholds selected by our R-space can achieve a 96.22% success rate through our reasonable subjective evaluation. This experiment has proved that our stable region count framework can be applied to processing specific groups of images such as retinal images.



(a) retinal image (b) segmentation

Figure 3.19: Example successful segmentations.



(a) retinal image (b) segmentation

Figure 3.20: Example failed segmentations.

3.4 Summary of results

In this chapter, we have presented some experimental results of studies of three specific problems. These problems come from different domains, use different data types (images and signals), employ different algorithms and exhibit different stability analysis. With the plateau size as our primary indicator for position of good parameter values, we have found good segmentations in the experiments of peak detector and flower detector although the retinal image experiment uses the same idea but a little different implementation. In the retinal image experiment, although the R-space has shrunk from 2-D to 1-D, the threshold selected is a point in the thresholding scale where there is the least change. In the peak detector case, the correct count of peaks are always detected by finding the largest contiguous area in the R-space and read the height of the plateau. In the flower image experiment, besides the objective region count match rate at around 86%, we demonstrate through human evaluation that people are around 90% in agreement with our machine generated segmentations are the best among all results. These best segmentations come solely from the parameter values at the largest plateau in the R-space.

Through all these studies and experiments, we can see the viability of the idea of stable count in the application of images and signals we have randomly selected or generated. The idea and implementations can also be applied to other problems.

Chapter 4

Conclusions

Why can the human eye see the world around it so conveniently and instantly? Why it is so easy for the human eye to see while so difficult for the machine to simulate? How should we model a human vision system? We answer this kind of questions by presenting our stable view computational model to simulate the human vision system. Our computational model is a bottom-up framework similar to and in parallel with the perceptual organization framework which organizes and groups low-level features from one object for high-level vision processing. In our model, we create a carefully designed scale-space by counting the number of regions in the image subjected to the processing by a pair of specially selected parameters. The dimension of the parameters can be expanded to three or more depending on the need of the algorithm used. In such a region count scale-space, we analyse the stability all over the place and try to find the largest flat area called plateau as the best parameter set to segment the image and provide the best understanding of the image. We have used several experiments to validate the viability and versatility of the idea and implementations in our model. The experiments have been very successful in corroborating the feasibility and adaptability of our computational model of human vision system.

We have developed a computer vision framework that resembles that of perceptual organization, but also different from it. The similarity comes from the fact that both our

framework and perceptual organization are the middle link between low-level image features and high-level image understanding and recognition. The nuance lies in the fact that the perceptual organization uses the low-level features such as proximity, parallelism, symmetry, similarity etc. to group data structures, while we subject the image into a scale-space formulated by parameters that are very effective in low-level data grouping, and then count the number of regions generated. Our idea is easy and convenient to implement and performs as well as perceptual organization.

We have completed several studies and experiments to process different kinds of real-world images and generic signals with several algorithms derived from the same idea of stable region count. Each algorithm uses appropriate low-level image processing parameters to formulate a scale-space. We analyse the stability in the scale-space. From the results and analysis in the previous chapter, we can see that the idea does help to achieve good segmentations of different groups of real-world images as well as to detect significant peaks in artificially generated noisy signals.

Although we think our research is successful and inspiring, it is interesting to investigate further along the following directions.

1. During the studies and experiments, so far we can only use the significant role played by the plateau area to determine the best segmentation of images. As shown in our studies, the height, plateau surroundings, uniqueness of plateaus, compactness of plateaus and other relevant factors all have some role to play in yielding the best segmentation, and therefore the stable view. We don't have the time and resources to investigate all these factors, find out their intrinsic relationships as well as their overall combined effect on providing the best and reasonable segmentations of images. This is one of the interesting directions for research in the future.

2. In our research we can automatically determine the best parameter pairs from a score based only on the plateau area size. As we have noted before, the height, surroundings, uniqueness, compactness and other factors all have some role to play in the

process to get the best segmentation. It is interesting to explore the viability of a combined dynamic score composed of all these components on the process to get the best segmentations of various kinds of images automatically. This is another interesting direction to pursue.

3. In the research we have found that some algorithms are not only working on the images they are designed for, but also on those they are not originally intended for. So it is interesting to explore whether an algorithm can cross design boundary and works on other images. We can pursue an automatic determination of algorithm suitability by creating a standard segmentation of an image and compare the segmentations of different algorithms with that standard copy and measure the total difference in the number of pixels, position of pixels and other relevant criteria. If the difference is higher than a certain threshold, we declare it doesn't work, otherwise it works.

4. We can also pursue whether an algorithm works on an image best over all the other algorithms in the case that multiple algorithms work on a certain image. We can achieve this automated algorithm selection by comparing the segmentations of each image processed with different algorithms with its standard segmentation, as proposed in the last research direction. We list the difference from small to high, and rank the algorithm with smallest one as the best algorithm, and so on. In this way we can automatically rank the performance of all the algorithms.

5. Our computational model is a bottom-up mid-level computer vision framework. Although it is a bottom-up module, it can receive high-level cues and adjust itself accordingly, and it can also be used interactively with low-level features and high-level cues. The model is not stand-alone, but can be combined with other frameworks to achieve the best result. So it is very interesting to investigate some other computational models and combine ours with theirs to see how well they could solve the real-world problems together.

Appendix A

Saliency

We have done some work in parallel with the stability algorithm. This work is based on the idea of saliency.

We compare our stability approach with one based on saliency. Figure 1.1 shows the saliency image for one of the retinal images. Notice that the lesions tend to have a medium saliency, between the high saliency of the blood vessels and the low saliency of the retinal background. In order to find the best threshold for segmenting the retinal image, we examine the distribution of saliency of pixels in the resulting segmentation for all possible thresholds. We select the threshold whose distribution most closely matches a normal distribution.

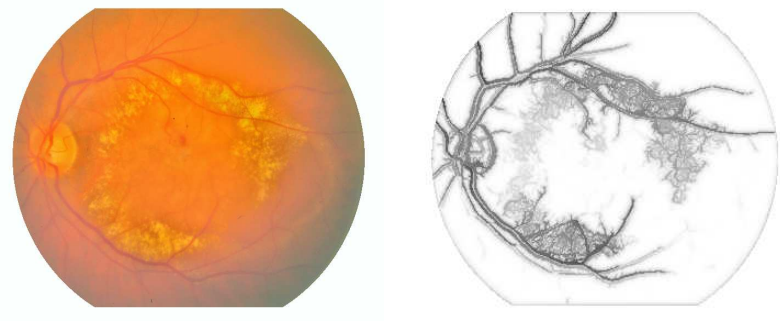


Figure A.1: Saliency image for a retinal image (left). The darker a pixel, the higher the saliency.

Formally, we complete the following steps. We compute the saliency image once, due to its high computational cost, using the publicly available code of Alter and Basri [2]. The result is an 8-bit image, which we will denote $S[r,c]$. The following steps are repeated for all thresholds at intervals of dT:

1. The original retinal image is thresholded to produce $O_T[r,c]$ according to Equation 2.9.

2. Let N be the number of pixels thresholded in $O_T[r,c]$, and let p_i, $i = 1...N$, be the locations of these pixels. The mean and standard deviation of the saliency of the thresholded pixels are computed:

$$\begin{aligned} \overline{s} &= \frac{1}{N}\sum_{i=1}^{N} S[p_i] \\ \sigma &= \sqrt{\frac{1}{N}\sum_{i=1}^{N}(S[p_i]-\overline{s})} \end{aligned} \tag{A.1}$$

3. The skew and kurtosis of the distribution of saliency of the thresholded pixels are computed:

$$\begin{aligned} S_{sk} &= \frac{1}{N\sigma^3}\sum_{i=1}^{N}(S[p_i]-\overline{s})^3 \\ S_{ku} &= \frac{1}{N\sigma^4}\sum_{i=1}^{N}(S[p_i]-\overline{s})^4 \end{aligned} \tag{A.2}$$

   Both the skew and kurtosis are zero for a perfect normal distribution. Skew is a measure of the asymmetry of the distribution about the mean line, and kurtosis is a measure of the distribution of spikes above and below the mean line.

The sum of the skew and kurtosis, $\mid S_{sk} \mid + \mid S_{ku} \mid$, provides a measure of the normalness of the distribution of saliency of the pixels. Figure 1.2 shows a plot of this sum versus the threshold for the example in Figure 1.1. We select the leftmost local minimum that is no more than double the global minimum as the best threshold.

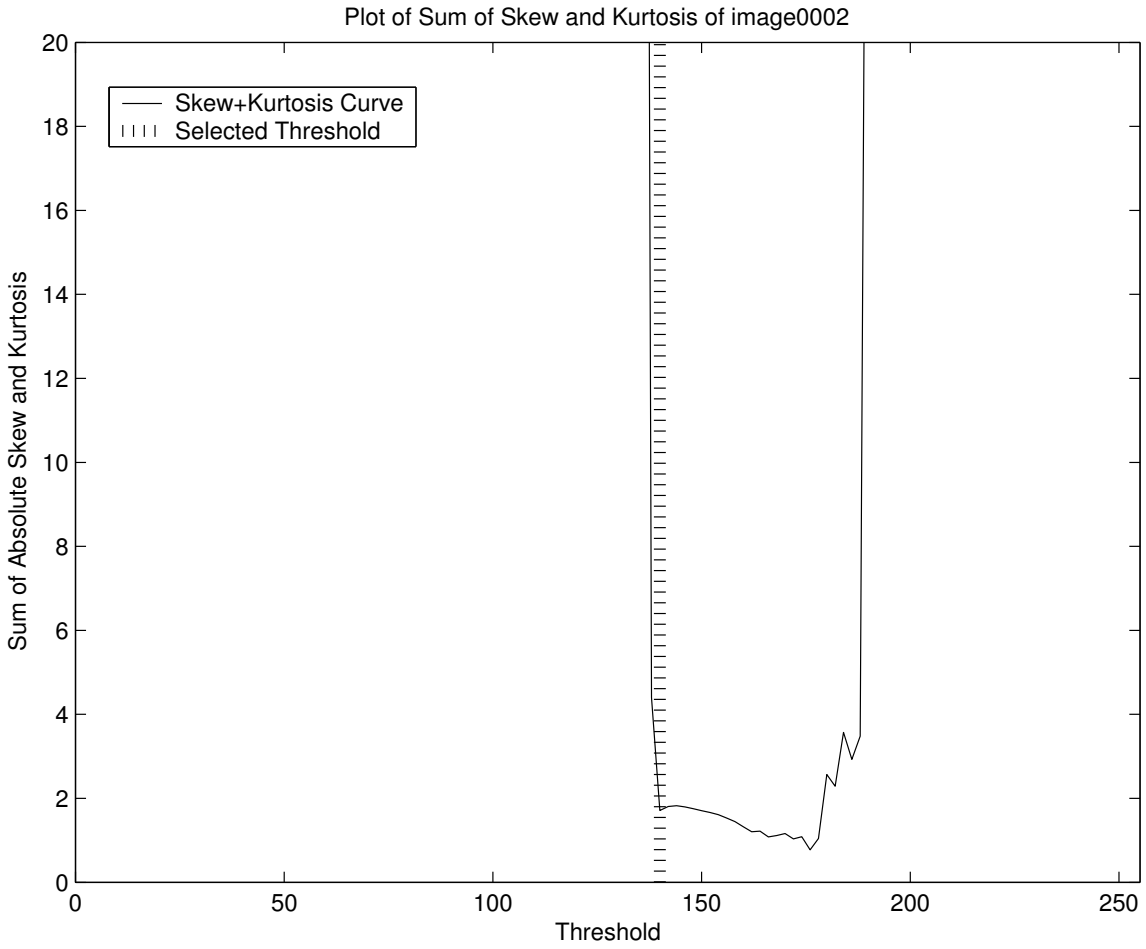


Figure A.2: The skew and kurtosis of the saliency of the thresholded pixels, as a function of the threshold.

Bibliography

[1] S. T. Acton, D. P. Mukherjee, "Scale Space Classification Using Area Morphology", *IEEE Transactions on Image Processing*, Vol.9, No.4, 623-635, April, 2000.

[2] T.D. Alter and R. Basri, "Extracting Salient Curves from Images: An Analysis of the Saliency Networks", in *International Journal of Computer Vision*, 27(1), 1998, pp.51-69.

[3] Eran Borenstein, Eitan Sharon and Shimon Ullman, "Combining Top-down and Bottom-up Segmentation", *2004 Conference on Computer Vision and Pattern Recognition Workshop(CVPRW04)*, Vol.4, 2004, pp.46-53.

[4] C. Chen, C. Wang, "A Simple Edge-preserving Filtering Technique for Constructing Multi-Resolution Systems of Images", *Pattern Recognition Letters*, Vol.20, pp.495-506, 1999.

[5] J.-S. Chang, H.-Y. M. Liao, M.-K. Hor, J.-W. Hsieh and M.-Y. Chern, "New Automatic Multi-level Thresholding Technique for Segmentation of Thermal Images", in *Image and Vision Computing*, vol. 15, 1997, pp.23-34.

[6] Erik A. Engbers and Arnold W.M. Smeulders, "Design Considerations for Generic Grouping in Vision", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.25, No.4, pp.445-457, April 2003.

[7] J. M. Gauch, "Image Segmentation and Analysis via Multiscale Gradient Watershed Hierarchies", *IEEE Transactions on Image Processing*, pp.69-79, Vol.8, No.1, January, 1999.

[8] S. Ho and G. Gerig, "Scale-Space on Image Profiles About an Object Boundary", *Lecture Notes in Computer Science*, Springer-Verlag Heidelberg, Volume 2695/2003 pp.564-575, August, 2003.

[9] A. Hoogs and J. Mundy, "An Integrated Boundary and Region Approach to Perceptual Grouping", *International Conference on Pattern Recognition*, 2000.

[10] Adam Hoover, Vlentina Kousnetsova and Michael Goldbaum, "Locating Blood Vessels in Retinal Images by Piece-wise Thresh Probing of Matched Filter Response", *IEEE Transactions on Medical Imaging*, Vol.19 No.3, pp.203-210, March 2000.

[11] Adam Hoover and Michael Goldbaum, "Locating the Optic Nerve in a Retinal Image Using the Fuzzy Convergence of the Blood Vessels", *IEEE Transactions on Medical Imaging*, Vol.22 No.8, pp.951-958, August 2003.

[12] E. Hadjidemetriou, M. D. Grossberg and S. K. Nayar, "Multi-resolution Histograms and Their Use for Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.26, No.7, July 2004.

[13] Feng-hsiung Hsu, *Behind Deep Blue: Building the Computer That Defeated the World Chess Champion*, Princeton University Press, 2004.

[14] Feng-hsiung Hsu, "Cracking Go", *IEEE Spectrum*, pp.50-55, Vol.44, Issue 10, October, 2007.

[15] A. Jacquot, P. Sturm and O. Ruch, "Adaptive Tracking of Non-Rigid Objects Based on Color Histogram and Automatic Parameter Selection", *IEEE Workshop on Motion and Video Computing*, pp 103-109, Vol.2, January, 2005.

[16] Michael J. Jones, Pawan Sinha, Thomas Vetter and Tomaso Poggio, "Top-down Learning of Low-level Vision Tasks", *Current Biology*, Vol.7, 1997, pp.991-994.

[17] Bassam Khadhouri and Yiannis Demiris, "Compound Effects of Top-down and Bottom-up Influences on Visual Attention During Action Recognition", *International Joint Conferences on Artificial Intelligence 05'*, Edinburgh, Scotland, 2005, pp.1458-1463.

[18] Tony Lindeberg, *Scale-Space Theory in Computer Vision*, Kluwer Academic Publishers, 1994.

[19] J. Liu and Y.-H. Yang, "Multi-resolution Color Image Segmentation", in *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 16, no. 7, July 1994, pp.689-700.

[20] David G. Lowe, "Distinctive image features from scale-invariant keypoints", *International Journal of Computer Vision*, 60, 2(2004), pp.91-110.

[21] S. Mahamud, L. R. Williams, K. K. Thornber and K. Xu, "Segmentation of Multiple Salient Closed Contours from Real Images", in *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 25 no. 4, April 2003.

[22] David Marr, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, W. H. Freeman and Company, San Francisco, 1982.

[23] David Mumford, Pattern theory: a unifying perspective. In Knill,D. and Richard, W., editors, *Perceptions as Bayesian Inference*, pp.25-62, Cambridge University Press.

[24] J. C. Olivo, "Automatic Threshold Selection Using Wavelet Transform", in *Graphical Models & Image Processing*, vol. 56, 1994, pp.205-218.

[25] N. Papamarkos and B. Gatos, "A New Approach for Multilevel Threshold Selection", in *Graphical Models and Image Processing*, vol. 56 no.5, 1994, pp.357-370.

[26] G.B. Rath and A. Makur, "Subblock Matching-Based Conditional Motion Estimation With Automatic Threshold Selection for Video Compression", *IEEE Transactions on Circuits and Systems for Video Technology*, pp.914-924, Vol.13, No.9, September, 2003.

[27] S. V. Raman, S. Sarkar and K. L. Boyer, "Tissue Boundary Refinement in Magnetic Resonance Images Using Contour-based Scale Space Matching", *IEEE Transactions on Medical Imaging*, vol. 10, June,1991, pp.109-121.

[28] Ueli Rutishauser, Dirk Walther, Christ of Kosh and Pietro Perona "Is Bottom-up Attention Useful for Object Recognition?", *IEEE International Conference on Computer Vision and Pattern Recognition 04'*, Vol.2, 2004, pp.37-44.

[29] Sudeep Sarkar and Padmanabhan Soundararajan, Supervised Learning of Large Perceptual Organization: Graph Spectral Partitioning and Learning Automata, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.22, No.5, pp.504-525, 2000.

[30] S.K. Setarehdan and J. J. Soraghan, "Automatic Cardiac LV Boundary Detection and Tracking Using Hybrid Fuzzy Temporal and Fuzzy Multiscale Edge Detection", *IEEE Transactions on Biomedical Engineering*, Vol.46, No.11, November, 1999.

[31] A. Sha'ashua and S. Ullman, "Structural Saliency: The detection of globally salient structures using a locally connected network", in the proceedings of *International Conference on Computer Vision*, pp.321-327, 1988.

[32] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.20, No.8, pp.888-905, August 2000.

[33] M. Sonka, V. Hlavac and R. Boyle, *Image Processing, Analysis and Machine Vision*, 2nd edition, Chapman & Hall, 1999.

[34] B. Sumengen, B. S. Manjunath and C. Kenney, "Image Segmentation Using Curve Evolution and Region Stability", in the proceedings of *International Conference on Pattern Recognition*, 2002.

[35] B. Sumengen, B. S. Manjunath and C. Kenney, "Image Segmentation Using Curve Evolution and Flow", in the proceedings of *International Conference on Image Processing*, Rochester, NY, USA, 2002.

[36] Diane M. Szaflarski, "How We See: The First Steps of Human Vision", *Access Excellence Classic Collection*, http://www.accessexcellence.org/AE/AEC/CC/vision_background.html.

[37] Zhuowen Tu, Xiangrong Chen, Alan L. Yuille and Song-chun Zhu, "Image Parsing: Unifying Segmentation, Detection and Recognition", *Proceedings of the Ninth IEEE International Conference on Computer Vision(ICCV 2003)*, Vol.2, pp.18-25, Nice, France, October, 2003.

[38] David Vernon, *Machine Vision, Automated Visual Inspection and Robot Vision*, Prentice Hall, September, 1991.

[39] J. Z. Wang, J. Li, R. M. Gray and G. Wielderhold, "Unsupervised Multi-resolution Segmentation for Images with Low Depth of Field", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.23, No.1, pp.85-90, January 2001.

[40] M.H.F. Wilkinson, T. Wijbenga, G. de Vries and M.A. Westenberg, "Blood vessel segmentation using moving-window robust automatic threshold selection", *Proceedings of 2003 International Conference on Image Processing*, pp.1093-1096, Vol.2, September, 2003.

[41] R. Wilson, C. Li, "A Class of Discrete Random Fields and Its Application to Image Segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp.42-56, Vol.25, No.1, January 2002.

[42] C. Xu and S. Ma, "Adaptive Edge Detecting Approach Based on Scale-Space Theory", *"IEEE Instrumentation and Measurement Technology Conference"*, pp.130-133, Ottawa, Canada, May, 1997.

[43] Li Yu and Adam Hoover, "Threshold Selection as A Function of Region Count Stability", *IEEE Workshop on Perceptual Organization in Computer Vision 2004' in association with CVPR2004'* , Washington, D.C., June, 2004.

[44] Li Yu and Adam Hoover, "Segmentation Methods Through the Stability of Region Count in the Scale-Space", Image Processing and Computer Vision Conference (IPCV'06) jointly with the 2006 World Congress in Computer Science, Computer Engineering and Applied Computing (WORLDCOMP'06), Las Vegas, June, 2006.

[45] Stella X. Yu, "Computational Models of Perceptual Organization", *Ph.D. Thesis CMU-RI-TR-03-14 Carnegie Mellon University*, Pittsburgh, PA 15213, May, 2003.

[46] T. Zhao and R. Nevatia, "Stochastic Human Segmentation from a Static Camera", *IEEE Workshop on Motion and Video Computing*, 2002.

[47] Song-Chun Zhu, "Statistical Modeling and Conceptualization of Visual Patterns", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.25, No.6, pp.691-712, June 2003.