A Multilevel Method for Image Segmentation

by

Adley Au

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Applied Mathematics

Waterloo, Ontario, Canada, 2010

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Image segmentation is a branch of computer vision that has received a considerable amount of interest in recent years. Segmentation describes a process that divides or partitions the pixels of a digital image into groups that correspond to the entities represented in the image. One such segmentation method is the Segmentation by Weighted Aggregation algorithm (SWA). Inspired by Algebraic Multigrid (AMG), the SWA algorithm provides a fast multilevel method for image segmentation.

The SWA algorithm takes a graph-based approach to the segmentation problem. Given an image Ω the weighted undirected graph $A = (N, E)$ is constructed with each pixel corresponding to a node in N and each weighted edge connecting neighbouring nodes in E. The edge weight between nodes is calculated as a function of the difference in intensity between connected pixels.

To determine whether a group of pixels should be declared as a segment in the SWA algorithm, a new scale-invariant measure to calculate the saliency of the group of pixels is introduced. This new measure determines the saliency of a potential segment by taking the ratio of the average similarity to its neighbours and its internal similarity. For complex images, intensity alone is not sufficient in providing a suitable segmentation. The SWA algorithm provides a way to improve the segmentation by incorporating other vision cues such as texture, shape and colour.

The SWA algorithm with the new scale-invariant saliency measure was implemented and its performance was tested on simple test images and more complex aerial-view images.

Acknowledgements

I would like to thank my supervisor Hans De Sterck for his guidance throughout this project. Also, I would like to thank Geoffrey Sanders, Josh Nolting, Tiffany Inglis and Killian Miller for their valuable input.

Thank you to Francis Poulin and Ed Vrscay for taking the time to read this work, and a special thanks to the applied mathematics graduate secretary Helen Warren.

Dedication

This is dedicated to my family, friends and colleagues.

Contents

List of Figures

Chapter 1

Introduction

Generally speaking, vision is an information processing task. Before the information can be used, it has to be translated into a form that can be recognized by the visual processor. In humans, this is done by the rods and cones of the retina. Light energy is reflected off of the scenery and penetrates the cornea of the eye, and is focused by the lens onto the retina. It is here that the light energy is transformed by an electrochemical reaction of the rods and cones into a signal, that travels down the optic nerve and into the brain where it can be processed. The brain then translates the signal into neurological patterns that are perceived as an image [40]. A similar process can also be achieved with a computer. Light enters a photosensitive device where it is converted into an electrical signal that can then be expressed as digital values that are sent to a computer and then are displayed as an image [41]. Although the computer is capable of capturing the image, it is unable to determine what it has captured. This is where computer vision comes in. Computer vision is the enterprise of automating and integrating a wide range of processes and representations used for visual perception [4].

It was originally thought that it would be a simple task to create algorithms for computers that mimic human vision, but early attempts by scientists failed miserably. It turns out that vision is very complex and requires a large amount of processes that are not well understood [4, 25, 41]. One way to view computer vision is by breaking it down into two levels: early-level vision and high-level vision. High-level vision is concerned primarily with the interpretation and the use of what is seen in the image rather than a direct recovery of the detailed physical properties of the visible environment, such as object boundaries and surface textures [3, 38]. High-level vision processes often require knowledge of the data within the image such as scenic information and specific objects. An aspect of early-level vision is feature analysis, which uses information about low-level vision cues such as colour, texture, shape, and luminance [3, 4]. This feature information can be used to obtain information about the physical properties of the image described above, that can be used for image processing tasks, such as segmentation. Early-level vision processes are not well known and are usually carried out at a subconscious level [4].

1.1 Image Segmentation

In computer vision, segmentation describes a process that divides or partitions the pixels of a digital image into groups that correspond to the entities represented in the image. The goal of segmentation methods is to produce a simplified and meaningful representation of the original image. Human vision is capable of segmenting images with a great deal of accuracy and precision. Automated segmentation procedures are not as accurate and general as their human counterparts, but they do have their advantages. Automated methods use complex mathematical analysis to determine suitable partitions of an image.

Because the process is automated, it can be run numerous times over multiple images with the hope of achieving consistent results. With the increasing processing power of computers, segmentation algorithms can now be executed in a reasonable amount of time.

Interest in segmentation techniques has increased dramatically in recent years. Since many general techniques do not discriminate as to what information the original image may contain, these segmentation methods have been able to branch into several disciplines and have many practical applications. In medical imaging, segmentation techniques help doctors locate tumours, diagnose patients and plan treatments [9, 29]. Fingerprint scanning [5, 44] and facial recognition techniques [36] also implement segmentation techniques to assist in the identification process.

Most segmentation techniques use the pixel intensity as the primary source of contrast, although some algorithms incorporate other low-level information such as texture and colour. A popular approach to segmentation is the graph-based approach. An image is regarded as a weighted undirected graph, with the nodes being associated with the pixels and the edge weights between nearest-neighbour pixels calculated as a function of the pixel intensities [14]. The salient regions of the image can be determined using a wide variety of measures such as graph cuts and eigenvectors [15]. The graph-based methods described in [15] may be relatively slow, and can be impractical.

A solution to speed up segmentation is to use a multilevel approach. Multilevel methods speed up the computation by using coarse approximations of the original problem. The main benefit of multilevel methods is that computational work is often linearly scalable in the number of variables, which in this case is the number of pixels in the image. This is valuable since the number of pixels in an image can be large.

The segmentation by weighted aggregation (SWA) method described in [33] presents a

fast multilevel graph-based method that incorporates multiple low-level vision cues such as intensity, colour and texture, along with higher level cues such as shape in order to find salient regions of an image.

In this thesis a new scale-invariant saliency measure is proposed for the SWA algorithm to replace the original measure described in [33]. This new measure properly takes scaling into account on the multiple levels. The goal of this thesis is to implement the improved SWA algorithm to solve segmentation problems related to satellite imaging.

1.2 Motivation

The purpose of our research is to implement a fast unsupervised segmentation algorithm which requires minimal human intervention during the segmentation process, at least within a certain class of images. Segmentation is a process in computer vision that has many applications and for these applications, there are segmentation methods that were developed specifically for them. The motivating factor of this research was to implement a segmentation method that can be used in a multitude of applications, without too much variation in the parameters.

The SWA algorithm provides a framework that can be customized quite easily. The large list of vision cues that can be included into the segmentation process makes it quite a versatile algorithm. The multilevel method provides a natural analogy to human vision [21]. Most segmentation methods are of the bottom-up nature, which use only the low-level vision cues of the image, such as colour and texture, to produce the segmentation. Since the SWA method is multilevel, high-level vision cues can also be used during the segmentation process. Top-down processes involve prior knowledge of the objects in the

image to guide the segmentation process [21]. High-level vision cues such as shape and orientation can then be included into the segmentation process. In some sense, the SWA algorithm combines bottom-up and top-down aspects to segment an image.

Vision is a difficult process for computers: to recreate vision, scientists must reinvent the most basic and yet unknown processes of specialized biological visual systems [4, 26]. Segmentation is an early-level vision process that is not well understood in humans. Thus, it can be difficult for scientists to duplicate a segmentation provided by a human expert. Hence, a completely unsupervised method may not be found for quite some time, but the SWA algorithm provides a strong framework for a semi-supervised segmentation method, where limited human intervention is required.

1.3 Thesis Summary

This thesis is divided into six chapters. **Chapter 2** provides a brief overview of common approaches to image segmentation. Advantages and disadvantages are described and some specific methods are also discussed. **Chapter 3** presents an in-depth overview of the multilevel algorithm that has been implemented. Each component of the algorithm is described in detail, including the rationale behind its inclusion into the algorithm. **Chapter 4** provides a set of manufactured images that were used to test the algorithm with a new scale-invariant saliency measure, followed by a discussion of the segmentation results obtained. **Chapter 5** includes a real application of the algorithm with sample segmentations. The application aims to identify airports and the different structures in airports such as aprons and taxiways from images captured by satellites. The segmentation results

for each image are discussed as well. **Chapter 6** summarizes the work that was conducted and details the conclusions that were drawn as well as future research directions.

Chapter 2

Background

2.1 Image Segmentation Methods

Segmentation is closely related to the Gestalt Laws of Organization that were developed in the early twentieth century to explain certain visual observations by humans [4, 10]. The Gestalt Laws describe grouping preferences based on features such as proximity, similarity, and continuity [4, 10]. These grouping characteristics are still used in many segmentation methods to identify objects in an image.

Image segmentation is a well-studied subject within computer vision and knowledge in the area has grown quickly. A wide variety of segmentation methods have become available with the same objective: to group pixels together based on some feature or characteristic [4]. Currently there is no unified method to solve the segmentation problem, but many specialized techniques have become available for specific applications. With a large number of segmentation methods available, it can be quite difficult to decide which method would provide the best results. Image segmentation methods can be grouped based on different

characteristics [4, 15, 19, 20, 27]. Some of these are listed below:

- **Region of Interest**:

  There are two approaches to partitioning an image into regions: region-based segmentation and boundary estimation using edge detection. Region-based methods group pixels that correspond to an object using some saliency criterion. Edge detection methods examine the rate of change in pixel intensities [20]. Sharp changes and discontinuities are good indications that an edge or boundary has been detected between two regions.

- **Vision level of the method**:

  Low-level segmentation methods use low-level vision cues to create groups and partition the image. High level methods integrate information about the objects within the image into the segmentation process.

- **Locality of the method**:

  Local methods use local information such as connectivity and spatial locations of pixels to guide the segmentation process. Local parts of the image are integral to the segmentation. Global methods focus on features of the entire image.

The rest of this section provides information about a few common segmentation methods. Many of the segmentation methods available overlap with respect to their characteristics, which can make it more difficult to decide which method to use in a specific situation. The list of segmentation methods provided is only a small subset of the wide variety of methods available. Some of the methods below can be related to the SWA algorithm as described in [16, 17, 23, 32, 33, 34, 43]. Thus, there also exist hybrid methods that incorporate different segmentation ideas to provide meaningful partitions.

2.1.1 Thresholding Methods

Thresholding techniques are one of the most practical ways to segment images. They are global techniques for partitioning images. Thresholding is a method to extract an object from its background and is normally used to partition grayscale images, but can also be used to segment colour images. One way to use thesholding is to define a range of intensity values (for a grayscale image) that correspond to the object of interest. Then, reject all pixels with an intensity value that does not fall within this range [31]. Multiband thresholding methods combine multiple original images that contain different information of the same scene to segment the image. An example of a multiband image is an RGB colour image. A generalized multithresholding method attempts to find $n-1$ thresholding values to partition the image into n groups. Thresholding methods are fast and are practical for simple images with sharp jumps in intensity. For more complex images with gradual changes in intensity, thresholding methods have a great deal of difficulty producing accurate segmentations.

2.1.2 Edge Detection

Edge detection methods have been studied extensively in computer vision. They work essentially by the operation of detecting significant local changes in an image. Edge detection methods usually contain three steps: filtering, enhancement and detection [20]. Filtering is commonly used to reduce the amount of noise in the image to improve the edge detector. Enhancement is used to reinforce regions in which a significant change in intensity is detected. This is done to help the edge detection process. The actual detection step identifies line segments where significant changes in intensity occur. It is these lines that

become edges of the segments. Edge detection techniques are commonly based on convolution operators. Four common edge detecting operators are the Roberts, Sobel, and Prewitt operators and the Laplacian operator [20]. The Sobel and Prewitt operators approximate the first derivative in a certain direction, and the Laplacian operator approximates the second derivative of the image intensities [20]. The Roberts operator is the simplest of the four and also approximates the first derivative in a certain direction [20]. Edges that are determined by these methods are often disconnected, which is a problem for segmentation, since a closed boundary is required to define each segment in the final partition.

2.1.3 Region Growing Methods

Region growing methods begin with seed points. The determination of where these seed points should be located, and how many seed points to begin with are important technical aspects of the algorithms. Each seed will correspond to a segment in the image. The region is grown iteratively, as each pixel is allocated to a seed region based on some measure of similarity [28]. Common measures include intensity and spatial proximity. This process is continued until all pixels are allocated. A modification to this method is as follows. Begin with a single seed in the image. For each non-seed pixel, if it is determined that it is not to be merged with a seeded region, it becomes a seed itself, and this process is continued until all pixels have been classified. A poor choice of the merging measure can lead to a severely over-segmented image in this approach. A popular region growing method is the watershed algorithm. Watershed methods simulate a flooding process by identifying initial markers, or "water" sources [37]. Then, for each non-marker pixel, its topographical distance to each water source is calculated, and it is grouped with the marker to which the distance turned out to be the smallest.

2.1.4 Split and Merge

Split and merge techniques begin with a naive partition of the image. In some cases, the initial partition can be the image itself. Then, the partition is split based on differences in specified characteristics within the partition. A common method for the partition splitting is the quadtree representation [20] which is an example of a regular decomposition method, that splits the region into a fixed number of equally sized regions. After each splitting, neighbouring regions are considered for merging [28]. Two regions merge to become a single region based on a measure of similarity between them. This process continues until no further splitting or merging occurs within the image.

2.1.5 Graph Clustering

A popular approach to segmentation is the graph-based approach. Given an image Ω, graph clustering methods construct the undirected weighted graph $A = (N, E)$, where each node in N corresponds to a pixel and each edge in E has a weight that is a measure of the similarity of the connected neighbouring pixels. Common measures of similarity are intensity and colour. Based on this weighted graph, a grouping of pixels is determined. A common method to group the pixels is the normalized cut method of [24, 35]. The normalized cut method groups pixels based on graph partitioning. The normalized cut measure that is used, computes the cost of the graph cut normalized by the edge weights [24, 35]. At each stage of the algorithm, the normalized cut measure bipartitions the graph. A decision is then made whether the resulting partitions are adequate or require further division. To further partition the image, the normalized cut method is applied to the segmented parts recursively.

Chapter 3

The Segmentation by Weighted Aggregation Algorithm

The Segmentation by Weighted Aggregation (SWA) algorithm first introduced in [32] was developed as an efficient multilevel segmentation method that in its simplest form approximates the solution for the normalized cut segmentation problem described in [24, 35]. In order to produce an efficient segmentation method, [32] was inspired by methodologies from the Algebraic Multigrid (AMG) framework [8]. Specifically, the SWA algorithm uses the idea of constructing coarse approximations of the original problem to accelerate the segmentation process and to improve the segmentation quality. Early attempts at constructing a fast multilevel segmentation method used only pixel intensity to segment the images.

In order to improve the segmentation of more complex images, more low and high-level vision cues were included to create greater contrast between two regions. This was done in later versions of the algorithm. In [16] the SWA algorithm was improved by incorporating

regional properties, such as pixel intensity variance and shape moments of groups of pixels. This chapter provides a description of the SWA algorithm and its components.

3.1 Algorithm Overview

The algorithm begins with an input image Ω. A graph A is constructed from the image with each node corresponding to a pixel. Graph edges join horizontal and vertical nearest neighbour pixels, with a weight that is determined by some affinity measure.

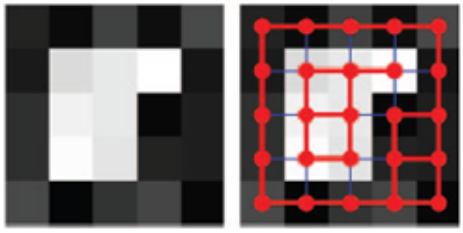


Figure 3.1: A Grayscale image shown with and without the fine graph [34]. Red edges indicate strong similarity, whereas the blue edges indicate dissimilarity.

In order to find salient regions, the graph is coarsened to subsequently coarser approximations of the original graph A. To determine the coarse graph a suitable subset of nodes are selected as representative nodes on the coarse level. The coarse edge weights are determined from the fine edge weights. Coarse nodes can be considered as overlapping blocks of fine nodes. At each coarse level the algorithm does a search for salient nodes. The coarsening procedure is repeated recursively until all nodes at the current level are salient. If a block is determined to be salient, the nodes one the finest level that comprise that block are assigned as a salient segment.

The algorithm consists of two distinct phases. In the first phase, increasingly coarser graphs are constructed by grouping the fine level pixels into overlapping blocks. It is in

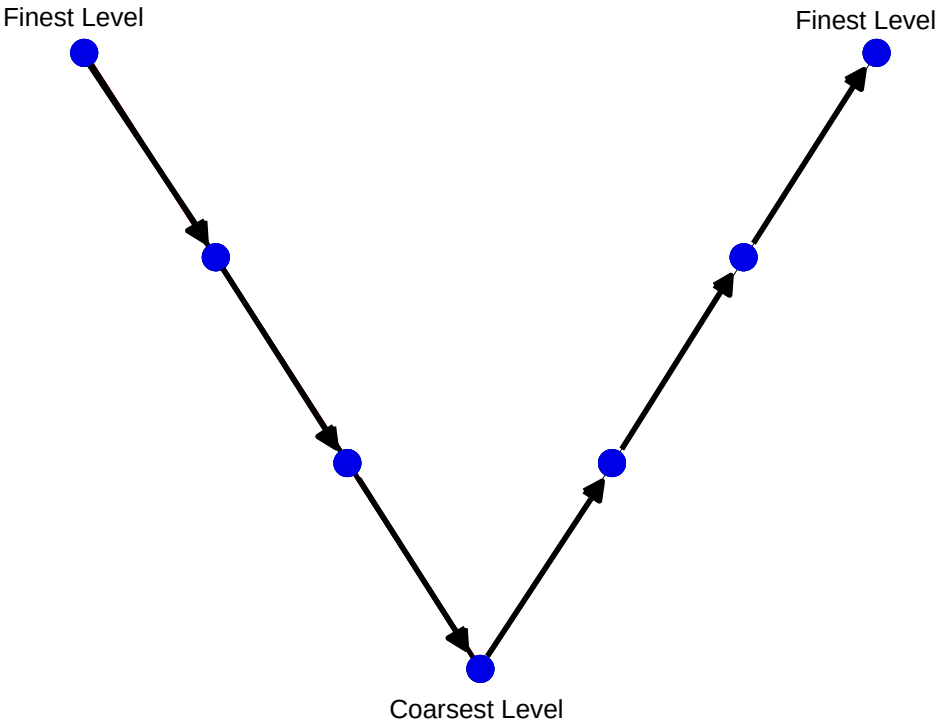


Figure 3.2: Multigrid V-cycle

this initial phase that the nodes are determined to be salient if they are distinct from their neighbours. In the second phase of the algorithm, the goal is to produce a partition of the original pixels. The difficulty lies in the allowance of overlapping blocks in the first phase of the algorithm. The overlapping blocks are sharpened at each finer-level graph until all pixels are associated with a single segment.

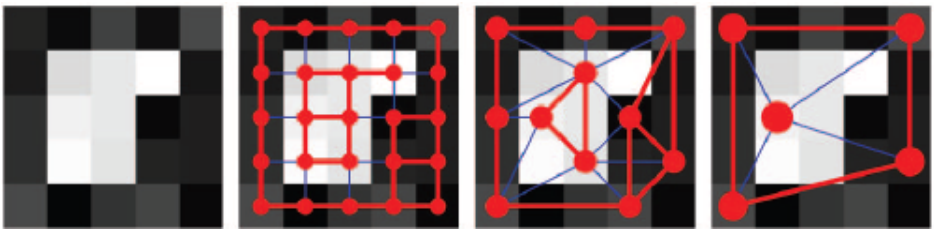


Figure 3.3: A grayscale image shown with increasingly coarser graphs. [34] Red edges indicate strong similarity, whereas the blue edges indicate dissimilarity.

As is usually done with multigrid, the two phases can be illustrated as a V-cycle similar to the illustration in Figure 3.2. The first phase of the algorithm begins at the finest level and travels down the side of the V until it reaches the coarsest level, and the second phase of the algorithm travels back up the other side of the V from the coarsest level to finest level.

3.2 Coupling Matrix

Given an image Ω that contains $N = m \times n$ pixels, we construct the graph A in which each node represents a pixel and each node is connected by an edge to its four nearest neighbours horizontally and vertically.

Each edge has a weight w_{ij} that is a measure of the similarity between the pixels i and j. An exponential function is chosen to ensure that the edge weights $w_{ij} \in [0, 1]$. The

size of the edge weight w_{ij} directly corresponds to the similarity between nodes i and j, meaning that a higher edge weight indicates a stronger similarity between nodes.

3.2.1 Grayscale Images

In digital imaging, grayscale images refer to a class of images for which each pixel has a single value that contains its intensity information. Grayscale intensities range from white to black, and contain various shades of gray in between. For grayscale images, the coupling matrix W is determined as a function of the difference between the intensity values of adjacent pixels. Let I denote the intensity vector for the image Ω: I_i, I_j denote the intensities of pixels i and j, respectively. The edge weight between i and j is then determined using the following expression,

$$w_{ij} = \begin{cases} e^{-\alpha|I_i - I_j|} & \text{if pixels i and j are neighbours,} \\ 0 & \text{otherwise,} \end{cases} \tag{3.1}$$

where $\alpha > 0$ is a user-defined parameter. For grayscale images, Equation (3.1) is an adequate measure of the similarity between two pixels. However, most real images are not grayscale and contain colour information which presents additional complexity.

3.2.2 Colour Images

The RGB colour model is the most common way to encode colours in digital imagery. It is an additive colour model with red, green and blue as the three primaries. Every colour in the model is represented as a triplet (r, g, b) with each entry corresponding to the intensity of a primary colour. A problem with this tristimulus model is that colour differences are

not easily quantified. A linear colour model does not correctly model hue, and the distance between two spatial locations in the RGB color model do not correlate well with colour differences [15].

A solution to this problem is to convert the RGB values to a uniform colour space that maps colour differences more intuitively. A uniform colour space is one in which the distance in spatial locations of the coordinates is a good representation of the differences in colour. One such space is called CIELAB. The CIELAB colour space was developed by the International Commission on Illumination (CIE) [1, 15, 39] in 1976. It was designed to approximate human vision accurately and has become widely used. The coordinates of CIELAB (L^*, a^*, b^*) correspond to the lightness or intensity of a pixel (L^*), its position between red/magenta and green (a^*),and its position between yellow and blue (b^*). Other more uniform colour spaces exist, but they are more complex and CIELAB was chosen for simplicity. More information about the conversion from RGB to CIELAB can be found in Appendix A.

Once the image Ω has been converted to the CIELAB colour space, the edge weights between pixels can be determined. From this point there are two directions in which one can proceed. The L^* component of the colour space contains the intensity values of the image Ω, thus one option to proceed is to let $I = L^*$ and use Equation (3.1) to determine the coupling matrix W for the image Ω. Depending on the application this may be a suitable option. An alternate option is to calculate the colour differences between two pixels. Since CIELAB is a more uniform colour space, the colour difference between two pixels can be determined by calculating the Euclidean distance between their colour vectors. Let C_i, C_j denote the CIELAB colour vectors of pixels i and j. Then the edge weight w_{ij} is determined by

$$w_{ij} = \begin{cases} e^{-\alpha||C_i - C_j||} & \text{if pixels i and j are neighbors,} \\ 0 & \text{otherwise,} \end{cases} \tag{3.2}$$

where $\alpha > 0$ is a user-defined parameter.

The construction of the coupling matrix W is the first step in the segmentation process. Note that W is a sparse matrix since only the nearest neighbours of a pixel i are of interest.

3.3 Multilevel Coarsening

A key component of the SWA algorithm is the coarsening procedure, which provides a systematic way to define the nodes of the coarse graph. The coarsening procedure used in the SWA algorithm closely resembles the coarsening used in Algebraic Multigrid (AMG). AMG is a fast multilevel matrix solver that was inspired by multigrid methods applied to partial differential equation discretizations [7, 8, 13]. The benefit of AMG is that it is an algebraic procedure that does not require the positions of the nodes to coarsen the grid.

The SWA algorithm was originally designed as an approximation to the normalized cut method described in [35]. The normalized cut method produces segments by solving the generalized eigenvalue problem,

$$Lu = \lambda W u, \quad \lambda > 0 \tag{3.3}$$

where W is the coupling matrix defined in (3.1) and L is the graph Laplacian defined below in (3.11).

AMG provides a fast method to solve (3.3), and it is intuitive that the SWA algorithm

below would use the same method to choose the coarse graph nodes. In particular, a modified version of the classical Ruge-Stüben AMG coarsening routine [8, 13] is used for the coarsening in our implementation of the SWA algorithm as explained in Subsection 3.3.1.

A candidate for a coarse level node should be one that has a large effect on many of its neighbours. The Ruge-Stüben AMG coarsening routine is based on the strength of connection between nodes.

Definition 1 *Let θ be a given constant with $0 < \theta < 1$, and consider row i in the coupling matrix W, i strongly depends on j if*

$$w_{ij} > \theta \max_{k \neq i} \{w_{ik}\}. \tag{3.4}$$

Also, if i strongly depends on j, we say that j strongly influences i.

Note that θ is a user-defined parameter.

Consider two consecutive levels, and call them the fine and coarse level. The points on the fine level are divided into two classes:

- C-points are fine-level points that become the coarse level nodes
- F-points are the fine-level points which were not chosen as C-points.

Denote the set of C-points C and F-points F.

Let $S_i = \{j : w_{ij} > \theta \max_{k \neq i}\{w_{ik}\}\}$ denote the set of points that strongly influence i, and let C_i denote the set of C-points that are neighbors of node i and strongly influence i. Also, let $S_i^T = \{j : i \, \epsilon \, S_j\}$ denote the set of points that i strongly influences.

The coarse node selection routine partitions the fine-level nodes into C-points and F-points. The concept of strength of connection is used to determine which fine-level nodes become C-points and which become F-points. The set of C-points, C, should be a maximal independent subset of points, such that no two C-points are strongly connected to each other.

3.3.1 Coarse Node Selection

The objective of the coarse node selection is to group the fine-level nodes into C-points and F-points. The selection process begins by assigning to each point i, a measure of its desirability to be a C-point. The desirability of point i, denote ξ_i is calculated by counting the number of points that i strongly influences. Note that these points are in fact the members of S_i^T, hence, the desirability of point i to become a C-point is the cardinality of S_i^T,

$$\xi_i = |S_i^T|. \tag{3.5}$$

Once ξ_i has been determined for all fine-level points, the first C-point is one of the points k such that $\xi_k \geq \xi_i$ for all fine-level points i. To avoid strong couplings between C-points, all the neighbouring points that k strongly influences become F-points. Next, we consider the unassigned fine-level points that strongly influence the newly defined F-points, because they are good candidates for becoming C-points in the maximal subset. So, for each unassigned point j that strongly influences a newly assigned F-point, increment ξ_j, and then choose one of the unassigned points k, with the largest ξ_k as the next C-point. This process is repeated until all points are assigned as C-points or F-points.

The classical Ruge-Stüben AMG coarsening routine in [8, 13] in addition to the above,

performs a second pass that creates more C-points, but this is not necessary for the SWA algorithm.

3.3.2 Interpolation Operator

Once the coarse-graph points C have been determined, we can construct the inter-graph operator. An inter-graph operator P is required to pass from one level to another. The number of rows in P corresponds to the number of fine-scale nodes, and the number of columns corresponds to the number of coarse nodes that were determined by the Ruge-Stüben AMG coarsening algorithm. The interpolation weights that make up P are determined as follows,

$$p_{ik} = \begin{cases} \frac{w_{ic_k}}{\sum_{n=1}^{K} w_{ic_n}} & \forall i \notin C, c_k \in C, \\ 1 & \forall i \in C,\ i = k, \\ 0 & \text{otherwise}, \end{cases} \tag{3.6}$$

where $K = |C|$. Note that the rows of P sum to 1.

The i^{th} entry of column j in P, P_{ij}, indicates how much the fine-level node i is associated with the j^{th} coarse-level node. Column j of P indicates how much each fine-level node belongs to the j^{th} overlapping aggregate of the fine-level nodes. The j^{th} C-point is a representative for the j^{th} aggregate.

3.3.3 Coarse Coupling Matrix

Once we have determined the inter-graph operator P, the coupling matrix W can be approximated to the coarse-graph using the Galerkin variational formula [8]:

$$W^{2h} = P^T W^h P. \tag{3.7}$$

The superscript denotes the scale of the grid: W^{2h} is the coarse graph approximation of W^h, the fine-level coupling matrix.

3.4 SWA Saliency Measure

To determine whether a block is a segment, a measure is used to determine its saliency. Saliency of a block refers to the quality of it standing out from its neighbours. Thus, an object that is deemed to be salient, is considered a segment. To determine the saliency of a block, a saliency measure Γ was proposed for the SWA algorithm [16, 32, 33]. For each potential segment U define the state vector $u = (u_1, u_2, \ldots, u_N)^T$ where N is the total number of pixels and

$$u_k = \begin{cases} 1 & \text{if } k \in \text{U}, \\ 0 & \text{if } k \notin \text{U}, \end{cases} \tag{3.8}$$

The saliency of a potential segment U is defined by

$$\Gamma(U) = \frac{\sum_{i>j} w_{ij}(u_i - u_j)^2}{\sum_{i>j} w_{ij}(u_i u_j)}. \tag{3.9}$$

For a segment U, the numerator of (3.9) sums the coupling values along the boundary of the segment and the denominator sums the internal couplings of the segment. Thus, the saliency Γ of a segment is the sum of the coupling weights along the boundary divided by the sum of internal weights. In matrix notation, (3.9) can be written as

$$\Gamma(U) = \frac{u^T L u}{\frac{1}{2} u^T W u}. \tag{3.10}$$

Here W is the coupling matrix determined by (3.1), and L is the weighted graph Laplacian, which is calculated as follows,

$$l_{ij} = \begin{cases} -w_{ij} & \text{if } i \neq j, \\ \sum_{k \neq i} w_{ik} & \text{if } i = j, \end{cases} \tag{3.11}$$

Note that, if we allow real nonboolean values to be assigned to u, the minimum positive value for Γ in (3.10) is obtained by the generalized eigenvector u that gives the minimal positive eigenvalue of (3.3), which is equivalent to the normalized cut solution proposed in [35]. By analogy, for boolean values of u, if $\Gamma(U)$ is determined to be sufficiently small then U can be considered a salient segment. In particular, we define a tolerance parameter γ and if $\Gamma(U) < \gamma$ then U is a segment. Potential segments U are determined by the coarsening procedure described in section 3.3, as explained further below.

The saliency measure Γ is calculated for each node in the multilevel hierarchy. At a given level l, the saliency is first calculated for all nodes. Once the saliency has been calculated for all nodes, the saliency of groups of nodes on level l should be considered. In practice, the saliency of selected groups of nodes on level l is computed on level $l+1$ after the coarse graph nodes have been determined. That is, the coarsening routine determines

which nodes at level l are to be grouped together. Thus, the multiple-node segments at level l are represented by single nodes at level $l+1$. Hence, it is sufficient to consider only single node partitions at a given level. Let $u = (\ldots, 0, u_i, 0, \ldots)^T$ where $u_i = 1$. In terms of calculating Γ, Equation (3.10) simplifies to

$$\Gamma = \frac{L_{ii}}{\frac{1}{2}W_{ii}}. \tag{3.12}$$

for each node i at level $l > 1$. Single-pixel partitions are not considered at the finest level in this implementation to avoid division by 0, since $w_{ii} = 0$ on the finest level. In most cases, single-pixel segments are not desired anyway. To avoid oversegmentation of the image, we let σ be a user-defined parameter to determine at which level to begin searching for segments. A larger value for σ leads to larger segments.

3.4.1 Scale-invariant Saliency Measure

In this section we propose a new scale-invariant saliency measure that is a modification of (3.10) from [16, 32, 33]. Recall that for a boolean segment, the numerator of (3.10) is the sum of the coupling weights along the boundary of a block and the denominator is the sum of the internal weights. Expression (3.10) does not properly take into account the boundary length and the number of internal connections when calculating the saliency. For example, consider two segments with equal intensity on a uniform background. Assume that both segments have the same area, but different boundary lengths. The segments should be equally salient, but Expression (3.10) may give very different results for the two segments. Hence, (3.10) is not a good indicator of the saliency of segments. As a new measure of salience, we propose the following,

$$\Gamma_{new} = \frac{\text{averaged coupling weight along the boundary}}{\text{averaged internal weight}} \tag{3.13}$$

Quite simply, we would like to normalize the sum of the external couplings as well as the sum of the internal couplings independently. To do this we define two more operators, G and V, where

$$v_{ij} = \begin{cases} 0 & \text{if } w_{ij} = 0, \\ 1 & \text{if } w_{ij} \neq 0, \end{cases} \tag{3.14}$$

and

$$g_{ij} = \begin{cases} -v_{ij} & \text{if } i \neq j, \\ \sum_{k \neq i} v_{ik} & \text{if } i = j, \end{cases} \tag{3.15}$$

Thus, G is the unweighted graph Laplacian and V is a boolean matrix with the same sparsity structure as W, also known as the adjacency matrix. In order to improve the saliency measure, the numerator of (3.10) is normalized by the number of boundary connections, and the denominator of (3.10) is normalized by the number of internal connections. A description as to why G and V can be used to determine the number of boundary and internal connections can be found in Appendix B. In the new measure, saliency of a block i is determined by taking the ratio of the average similarity of block i to its neighbouring blocks and the average internal similarity in block i:

$$\Gamma_{new} = \frac{\left(\frac{L_{ii}}{G_{ii}}\right)}{\left(\frac{W_{ii}}{V_{ii}}\right)} \tag{3.16}$$

A benefit of the normalization is that the scale-invariant saliency measure can be used for segments of different length-to-area ratios.

The new scale-invariant saliency measure calculated above, normally lies between zero and one on all levels. Conversely, the original saliency measure proposed in [16, 32, 33] takes on a larger range of values that vary between levels. Another advantage of the scale-invariant saliency measure is that the same tolerance parameter γ can be used on all of the levels since the range in the Γ_{new} values is normally between zero and one. Having prior knowledge of the range of values that the saliency measure can take on and knowing that the range is similar on all levels, is a significant benefit when determining the saliency tolerance parameter γ. In Appendix C the two saliency measures are compared for segmenting a test image. The saliency values for both the original and scale-invariant saliency measures are shown for each level, and are then compared and discussed. Appendix C illustrates the behaviour of the saliency measures in one experiment of several that were conducted to draw the conclusions above.

The computation cost and memory required for this new saliency measure is higher than the original saliency measure in (3.10). Two new matrices need to be stored, G and V at each level. Thus, G and V must also be coarsened using Expression (3.7) at each level that requires two matrix-matrix products and two matrix-vector products. Also, for each calculation of Γ two more division operations are required. However, given the extra cost of the scale-invariant measure we feel that the benefits of standardizing the saliency measure to Γ_{new} is worth the extra computational cost.

3.5 Multiscale Coupling Update

Early versions of the SWA algorithm used only the pixel intensity at the finest level to produce the segmentation of the image. This is a simplified approach to segmentation

and vision in general. Real images are complex and cannot be segmented successfully using the pixel intensity alone. In human vision, texture and shape are two very important components in identifying objects. Texture is concerned with the spatial distribution of the image intensities and discrete tonal features [19] and shape refers to the spatial descriptors of a region, such as length and width. These characteristics are regional properties and cannot be captured at the finest level of the image.

Regional properties are incorporated into the algorithm by modifying the coarse-graph weights. This is carried out by using some measures to compare the characteristics of neighboring nodes. This allows regions of similar properties to be grouped together and nodes with different properties to not be grouped together. The modification of the coarse-graph weights affects the grouping at coarser levels. In this algorithm it is relatively simple to incorporate regional properties since the algorithm already considers different scales by nature. Recall that, at a coarse level, each node represents a subset of nodes at some finer level. The subset of nodes at the finer level is grouped together to form a block. Thus, to incorporate regional properties it is sufficient to determine the properties of the coarse-level nodes. Since the regional characteristics are just averages of the characteristics of the nodes that comprise the regions, they can be calculated efficiently during the coarsening process.

Suppose Q is a property vector for the nodes at a particular level, and Q^c is the property vector at the next coarser level. Note that Q^c contains the average of the regional properties of the nodes on the finer levels that make up the coarse-level block. Recall that the regional properties are average values of the nodal properties. To compute the regional properties, a modified version of the inter-graph operator P can be used. In particular, a column-normalized version $\hat{P}$ of P is needed. The columns of $\hat{P}$ are normalized to one.

Thus, the region properties are determined by the following,

$$Q^c = \hat{P}^T Q \tag{3.17}$$

Regional properties that are considered in this algorithm are the average intensity, intensity variance, and the geometric shape moments. These measures are used to affect the coupling on coarser levels through an exponential weighting (See Equation (3.30)), and hence to affect the final segmentation of the image.

3.5.1 Average Intensity

Calculating the average intensity of a block allows us to segment regions whose boundaries have gradual transitions in intensity [33]. The inclusion of the average intensity of a block reduces the effect that noise has on the segmentation. The average intensity between blocks is calculated after the first level, since at the finest level, the average intensity is just the pixel intensity vector I. For colour images, the intensity I is determined by $I = L^*$, where L^* is luminance matrix of the CIELAB colour space. The average intensity values of I are calculated using the formula,

$$I^c = \hat{P}^T I, \tag{3.18}$$

where the superscript c denotes the next coarse level. This is done at each level of the algorithm.

3.5.2 Variance

The variance of the pixel intensities is a common statistic used to measure textures in image segmentation [19, 33]. In uniform regions of the image, the variance is very small and it becomes larger in areas where the pixel intensity changes.

Definition 2 *If a random variable* $\mathbf{X}$ *has an expected value of* $\mu = \mathbf{E}(\mathbf{X})$ *then the variance,* $Var(\mathbf{X})$ *of* $\mathbf{X}$ *is given by*

$$Var(\mathbf{X}) = \mathbf{E}\left[(\mathbf{X} - \mu)^2\right]. \tag{3.19}$$

With a little bit of algebra it can be shown that (3.19) is equivalent to the following:

$$Var(\mathbf{X}) = \mathbf{E}\left[\mathbf{X}^2\right] - (\mathbf{E}\left[\mathbf{X}\right])^2 \tag{3.20}$$

To calculate the intensity variance of a block k, the intensity values and squared intensity values need to be averaged. The intensity values are averaged using Equation (3.18). The squared intensity values are first calculated at the finest level, by squaring the intensity of each pixel. Then, the coarse versions of the intensity-squared vector I_2 is accumulated using (3.17), and we get the following,

$$I_2^c = \hat{P}^T I_2 \tag{3.21}$$

where c represents the next coarse level.

Thus, the variance in intensity vector is determined by the following,

$$Var = I_2 - I^2. \tag{3.22}$$

3.5.3 Elongated Shapes

In the previous subsections, we determined two regional properties, the average intensity and the variance of intensity. In this section we are concerned with the shape of the blocks. In particular, we would like to differentiate regions that have an elongated shape from those that have a more regular shape. Elongated shapes will be used in Chapter 5 to find roads in satellite images. The second-order principal moments of the aggregate can be used to determine how elongated an object is [11, 16]. The x and y coordinates of each node along with their product and squared values are used to determine the length and width of the block. The following eigenvalue problem is solved for block k:

$$\Lambda_k e = \omega e, \tag{3.23}$$

where Λ_k is the covariance matrix of the coordinate statistics for block k,

$$\Lambda_k = \begin{bmatrix} (\overline{x^2})_k - (\overline{x})_k^2 & (\overline{xy})_k - (\overline{x})_k(\overline{y})_k \\ (\overline{xy})_k - (\overline{x})_k(\overline{y})_k & (\overline{y^2})_k - (\overline{y})_k^2 \end{bmatrix}. \tag{3.24}$$

The second order principal moments for a block k are represented by, $(\overline{x})_k$, $(\overline{y})_k$, $(\overline{x^2})_k$, $(\overline{y^2})_k$, and $(\overline{xy})_k$. The corresponding eigenvalues ω of (3.23) are used to calculate how elongated block k is by looking at their ratio. We define

$$R_k = \frac{\min(\omega)}{\max(\omega)}. \tag{3.25}$$

A value for R_k that is close to zero corresponds to a block k that is very elongated, whereas an R_k value of close to one represents a block with a more regular shape. In experiments we found that it is useful to modify the R_k values to accentuate the differences between regular and elongated shapes:

$$\hat{R}_k = \begin{cases} 0 & \text{if } R_k < \tau_1, \\ 1 & \text{if } R_k > \tau_2, \\ \frac{R_k - \tau_1}{\tau_2 - \tau_1} & \text{otherwise} \end{cases} \tag{3.26}$$

where $\tau_1 = 0.2$ and $\tau_2 = 0.8$.

To set up (3.24) at each level, the second order moments need to be averaged. Let

$$Z = \begin{pmatrix} X & Y & X_2 & Y_2 & XY \end{pmatrix} \tag{3.27}$$

where the k^{th} component of X, Y, X_2, Y_2, and XY contain the corresponding averaged quantities of each node k on the current fine level. Note that X_2, and Y_2 are vectors that contain $(\overline{x^2})_k$ and $(\overline{y^2})_k$ as their k^{th} components. Also note that XY is a single vector with $(\overline{xy})_k$ as its k^{th} component. Then, using (3.17), we get for the next coarse level:

$$Z^c = \hat{P}^T Z \tag{3.28}$$

Let $\hat{R}^{[s,l]}$ denote the ratio that has been calculated at level s and coarsened to level l. This produces a multilevel shape vector for block k on level l:

$$\hat{R}_k = (\hat{R}_k^{[1,l]}\ \hat{R}_k^{[2,l]}\ \ldots\ \hat{R}_k^{[l-1,l]}\ \hat{R}_k^{[l,l]}) \tag{3.29}$$

where $\hat{R}_k^{[l,l]}$ is calculated as in (3.26). Note that $\hat{R}_k^{[l,l]}$ is only calculated for $l > 2$ so that meaningful shapes are incorporated into the algorithm.

A multilevel shape vector is beneficial because information about a block k and all the subblocks that comprise it are stored and used to modify the coupling at coarser scales. Thus, a block i that is comprised of several elongated subblocks, but is not necessarily elongated itself, has the capacity to merge with a block j that is also comprised of elongated subblocks, and is elongated itself as shown in Figure 3.4. Without a multilevel shape vector this may not be possible.

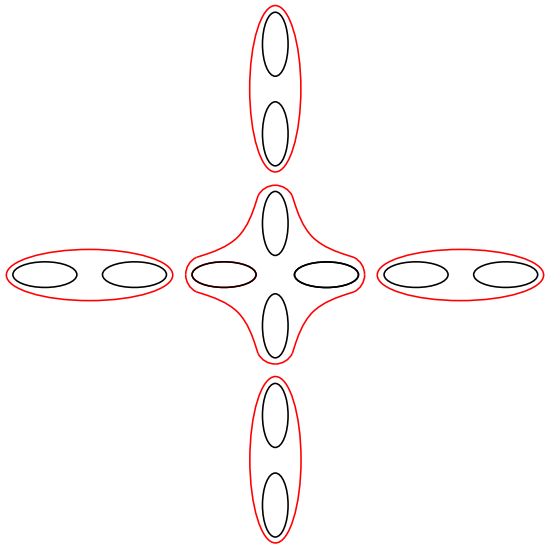


Figure 3.4: Sketch for segmentation of intersecting streets using the multilevel elongation measure. Black shapes indicate the fine-level blocks and the red shapes indicate the coarse-level blocks.

3.5.4 Weight update

In order to make these statistics useful, they need to be incorporated into the coupling matrix. Before this can occur, the update values should be combined in such a way that

they are of the same scale. This is done using the exponential function similar to what was used to determine the initial values of the coupling matrix.

$$w_{ij}^{c} \leftarrow w_{ij}^{c} \cdot e^{-\tilde{\alpha}|I_i - I_j|} e^{-\beta|Var_i - Var_j|} e^{-\rho||\hat{R}_i - \hat{R}_j||}, \tag{3.30}$$

where $\tilde{\alpha}$, β, and ρ are parameters chosen by the user and $||\hat{R}_i - \hat{R}_j||$ is the Euclidean distance between $\hat{R}_j$ and $\hat{R}_j$. Note that the average intensity and intensity variance of a block k is determined for all levels l such that $l > 1$. The shape measure $\hat{R}_k$ of block k updates the coupling matrix W for level $l > 2$.

3.6 Assignment of Nodes to Segments

This section describes the beginning of the second phase of the algorithm that is illustrated in Figure 3.2. Once we have reached the coarsest level, and all salient nodes have been detected on various levels of the algorithm, the overlapping blocks need to be sharpened in order to provide a meaningful partition at the finest level.

In Section 3.4 it was explained that only single-node partitions are considered on a given level when determining saliency. Suppose a node k on level l has been identified as salient. Construct the elementary vector, $u^{[l]} = e_k$, where e_k has a one in the k^{th} entry and zeroes everywhere else. Then, this vector is interpolated back to the finest level to determine the pixels that belong to the segment. The vector $u^{[l]}$ is interpolated from level l to level $l-1$ using

$$u^{[l-1]} = Pu^{[l]}. \tag{3.31}$$

Suppose that $S = [U_1 U_2 \dots U_r]$ is an array of state vectors for all segments that have been interpolated to level l from coarser levels. Since node k is assigned entirely to this new segment, the positive values in the k^{th} row of S are changed to 0. That is, $s_{ij} = 0$ if $i = k$. Thus, node k is removed from each segment currently in S. Then $U_{r+1} = u^{[l]}$, and $S = [U_1 U_2 \dots U_r U_{r+1}]$ is interpolated to the next finer level, where the same procedure is repeated.

3.7 Sharpening

After each interpolation, the vector $u^{[l-1]}$ may have values in [0,1]. The goal here is to bring $u^{[l-1]}$ closer to a binary state vector. We do this by modifying the interpolated $u^{[l-1]}$ before interpolating it to the next finer level, by the following formula:

$$u_i^{[l-1]} = \begin{cases} 1 & \text{if } u_i > 1 - \delta, \\ 0 & \text{if } u_i < \delta, \\ u_i^{[l-1]} & \text{otherwise,} \end{cases} \tag{3.32}$$

with δ a user-defined parameters. The sharpening procedure occurs once all segments have been found and are recursively interpolated back to the finest level.

3.8 Determining the Final Association of Pixels

At the finest level it is required that each pixel i be associated with a single segment q. Suppose $S = [U_1 U_2 \dots U_r]$ at the finest level. After the interpolation process it may be

the case that a single pixel is associated to more than one segment. In order to create the final partition, we associate the pixel with the segment in which its weight turned out the largest. At the end of this procedure, each pixel is associated with exactly one segment and the result is output to the user.

3.9 Pseudo Code

The modified SWA algorithm outlined in this chapter is illustrated in pseudo code below. The algorithm was implemented recursively.

Algorithm: Determine the Segments of a colour image Ω

---

1. Calculate matrices W, V, L, G according to Equations (3.2), (3.14), (3.11), (3.15) for image Ω and set level=0.
2. U $\leftarrow$ ***vcycle***(W, L, V, G)
   (a) level = level +1.
   (b) if the current level has only one node,
      i. Calculate the saliency using Equation (3.16). Denote the set of salient nodes S
      ii. Let $U^{coarse} = \emptyset$ and skip to step 2(c)ix.
   (c) else
      i. if level $\geq \sigma$ calculate the saliency of all nodes using (3.16). Denote the set of salient nodes S. If all nodes are salient, let $U^{coarse} = \emptyset$ and skip to step 2(c)ix.
      ii. Select the set of coarse nodes and determine the inter-graph operator P as outlined in Section 3.3.
      iii. Coarsen W and V, $W^c = P^T W P$, and $V^c = P^T V P$.

iv. Update W^c with regional properties using Equation (3.30).

v. Calculate L^c and G^c using Equations (3.11) and (3.15).

vi. $U^{2h} \leftarrow \boldsymbol{vcycle}(W^c, L^c, V^c, G^c)$

vii. Interpolate U^{2h} to the current level ($U^h = PU^{2h}$).

viii. Sharpen $U^h \rightarrow U^{coarse}$.

ix. Construct elementary vectors U_i for salient nodes $i \in S$ and construct $U^{fine} = [\ldots U_i \ldots]$. Modify rows i of U^{coarse} (see Section 3.6).

x. Construct $U = [U^{fine}|U^{coarse}]$.

3. Determine final pixel association from U

Chapter 4

Algorithm Performance on test images

In this chapter we will present some segmentation results of our implementation of the SWA approach. To test the algorithm, simple test images were produced with clearly identifiable segments. There are two reasons to run the algorithm on simple test images. The first is that we would like to illustrate the performance of the segmentation algorithm on simple images before applying it to more complicated ones. The particular images in this chapter were chosen to display the usefulness of the properties that were included in the algorithm, namely, the intensity, intensity variance and shape moments. The second reason is to experiment with the extensive list of user-defined parameters that are available in this algorithm. The number of parameters is quite large and each has an impact on the segmentation results. Thus, the choice of parameters are a significant step in the segmentation process.

The segmentation algorithm used to obtain the results in Chapters 4 and 5 was im-

plemented in MATLAB except the coarsening routine described in Section 3.3 which was coded in C. The segmentation results were obtained on a 32-bit Windows XP machine with an Intel T2250 CPU and 3 gigabytes of DDR2 ram.

4.1 Parameter List

An extensive list of user-defined parameters are used for the SWA algorithm, with each having a significant role in the segmentation process. With many degrees of freedom, it can be a difficult task to fine-tune the parameters to produce a meaningful segmentation. A short summary of the parameters included in our implementation of the algorithm is provided along with a general description of their usage.

1. $\boldsymbol{\alpha} > 0$ is used to scale the contrast between pixels when determining the initial edge weights for colour and grayscale images in Equations (3.1) and (3.2).

2. $0 < \boldsymbol{\theta} < 1$ is the coarsening strength threshold that determines whether nodes are strongly connected or not for the coarsening. Its usage is shown in Equation (3.4).

3. $\tilde{\boldsymbol{\alpha}} \geq 0$ rescales the coupling values at coarser levels according to the difference in average intensities of two neighboring blocks after the first level. This is described in Equation (3.30).

4. $\boldsymbol{\beta} \geq 0$ rescales the coarse-level coupling values to incorporate the differences in the intensity variance of neighboring blocks after the first level. This is shown in equation (3.30).

5. $\boldsymbol{\rho} \geq 0$ rescales the differences in elongation between blocks. Elongated shapes are differentiated from more regular shapes. This is not incorporated until the third level of the algorithm. The coupling weights of neighbouring are modified according to Equation (3.30).

6. $0 < \boldsymbol{\delta} < 1$ is the sharpening parameter for the segments determined by the algorithm. The sharpening is done according to expression (3.32).

7. $\boldsymbol{\sigma}$ is a parameter used to prevent over-segmenting the image. It indicates at which level in the algorithm to begin searching for salient segments.

8. $\boldsymbol{\gamma}$ is the saliency tolerance parameter. If the saliency value of a node is below this tolerance γ, then is it considered a segment. That is, if $\Gamma(U) < \gamma$ then U is a segment.

Note that the elongation parameter was not used for any segmentations in this chapter ($\rho = 0$).

4.2 Test Images

4.2.1 Example 1: Grayscale Circles

The first example is a grayscale image with regions that are identifiable strictly by intensity. The regions have uniform intensities with abrupt changes in contrast at the boundaries between regions. The first image that was used to test the algorithm is shown in Figure 4.1. It shows three nested circles of different intensities on a black background. The different regions in this image have uniform intensities. Thus, the main change in intensity

occur at the boundaries between regions. Figure 4.2 displays the segmentation of the original image.

Figure 4.1: Grayscale image with nested circles (256x256). Intensity values from the inside out: 151, 76, 236, 21

The red regions denote the segments that were found by the segmentation algorithm. Figure 4.2 shows that the algorithm was successful in segmenting the simple image into its constituents. The parameters that were used to determine this partition were: $\theta = 0.25$, $\alpha = 100$, $\tilde{\alpha} = 0$, $\beta = 0$, $\rho = 0$, $\delta = 0.15$, $\sigma = 6$, and $\gamma = 0.1$. The runtime to conduct the segmentation was 89.53 seconds. Since the regions in the image have uniform intensities, it was unnecessary to include variance over the blocks as it would not influence the segmentation. The average intensity of blocks was also left out since the fine pixel contrast was sufficient to segment Figure 4.1 as the regions have uniform intensities. The large value of α emphasizes the jumps in intensity along the region borders and thus

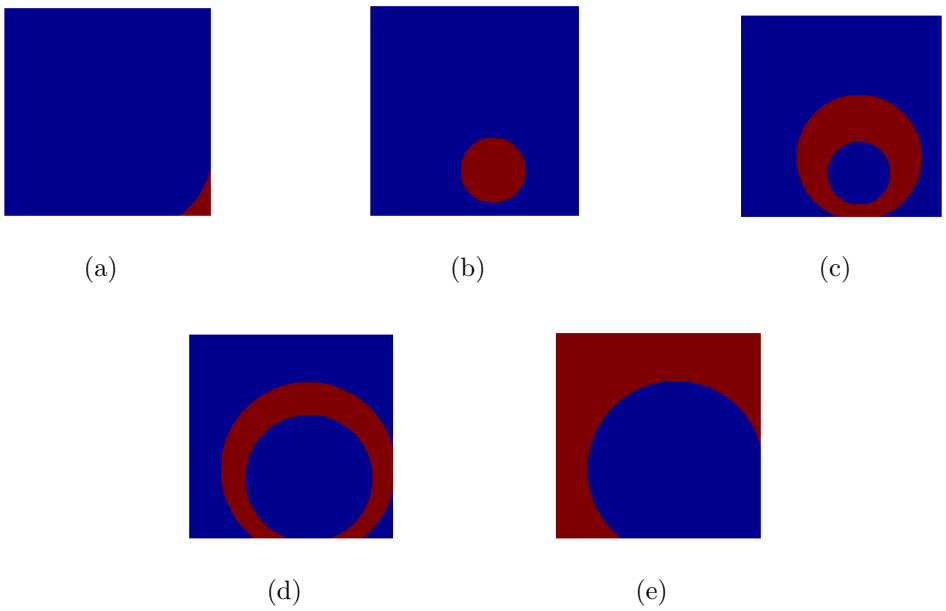


Figure 4.2: The five captured segments of Figure 4.1 (red).

increases the contrast between nodes in the initial calculation of the edge weights. This essentially gives pixels connected across the boundary a coupling weight of nearly zero. For this image, all five segments were found at the 10^{th} level of the algorithm. Each region of Figure 4.1 was identified, but that does not mean that the algorithm finds the partition that is required by the user. One user may require a total of four segments, in which all pixel with similar intensities are grouped together, thus it can be argued that Figure 4.2 is an over-segmentation of the original image in Figure 4.1. The segments of Figure 4.2a and Figure 4.2e were declared as two different segments by the SWA algorithm, yet in the original image they have the same intensity. Ultimately, the desired segmentation of this image would be based on the user's requirements. For the purposes of this thesis, the results in Figure 4.2 are considered successful.

4.2.2 Example 2: Colour Image

In this subsection a colour image is segmented. The RGB values of Figure 4.3 are first converted to the CIELAB colour space, which is more accurate in determining colour differences to construct the coupling matrix using (3.2). The parameters that were used to determine the segmentation are identical to the ones used to segment Figure 4.1, $\theta = 0.25$, $\alpha = 100$, $\tilde{\alpha} = 0$, $\beta = 0$, $\rho = 0$, $\delta = 0.15$, $\sigma = 6$, and $\gamma = 0.1$. The resulting segmentation is displayed in Figure 4.4, with the magenta regions representing the segments. It took 56.97 seconds to segment Figure 4.3.

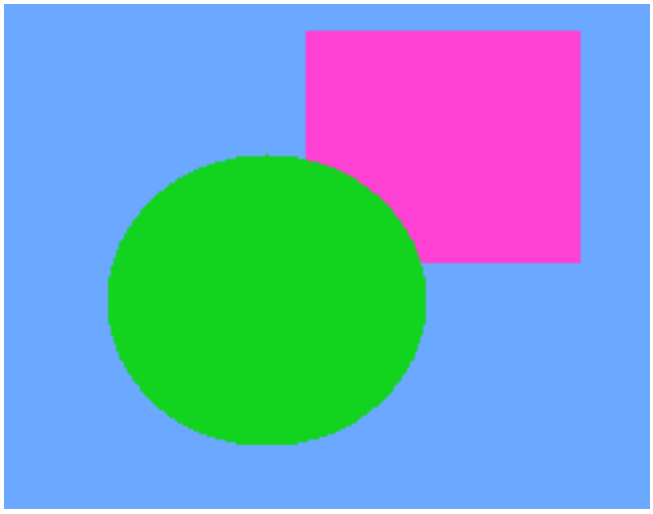


Figure 4.3: RGB colour image (242x189).

The segmentation in Figure 4.4 shows that the colour image was accurately segmented. The correct segmentation illustrates that colour images can also be successfully segmented using the SWA algorithm. Being able to take advantage of colour within an image is beneficial as more information about the image can be utilized. It is the uniformity of the CIELAB colour space that allows the SWA algorithm to segment colours properly, by accurately modelling colour differences.

(a) (b) (c)

Figure 4.4: The three captured segments of Figure 4.3 (magenta).

4.2.3 Example 3: Perturbed Images

In this next example, we would like to illustrate the advantage of the average intensity block coupling update. Figure 4.5 is a grayscale image of nested squares with regions differentiated by intensity alone. Figure 4.6 displays the segmentation obtained using the following parameters $\theta = 0.25$, $\alpha = 100$, $\tilde{\alpha} = 0$, $\beta = 0$, $\rho = 0$, $\delta = 0.15$, $\sigma = 6$, and $\gamma = 0.1$. Note that these are the same parameters that were used to segment the image in Figure 4.1.

We now modify Figure 4.5 to make the regions less uniform. Perturbations are added to the image in two ways. First, we add a random number uniformly distributed between zero and seven to the intensity of each pixel. The random noise is shown in Figure 4.7a. Second, we sample $f(x) = 3\sin x$ for $x \in [0, 4\pi]$ and add it to the intensity of each pixel. The added perturbation is shown in Figure 4.7b.

Figure 4.8 shows the image of Figure 4.5 with the added perturbation. Let I_{old} be the intensity values of Figure 4.5, and I_{random}, I_{sine} the intensity values of Figure 4.7a and Figure 4.7b, respectively. Then the intensity values of Figure 4.8 is given by

$$I_{new} = I_{old} + I_{random} + I_{sine}. \tag{4.1}$$

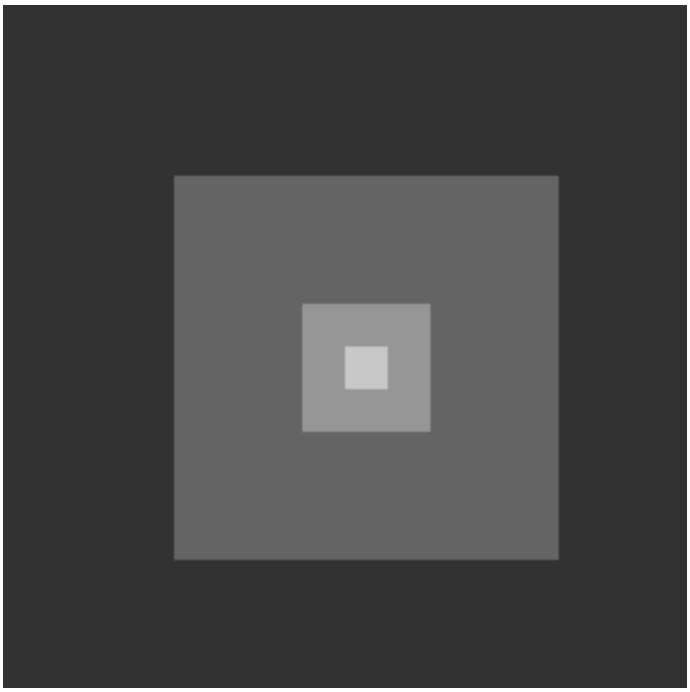
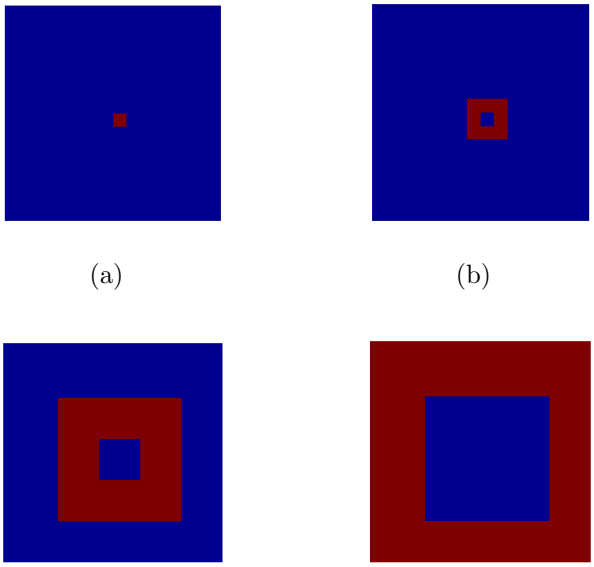


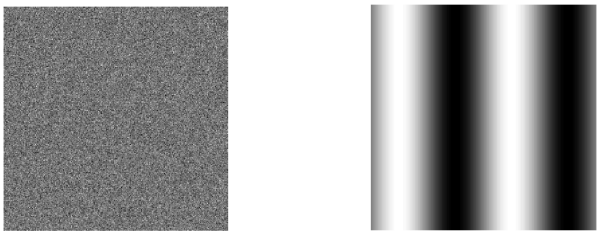
Figure 4.5: Grayscale image of nested squares (256x256). Intensity values from the inside out: 201, 151, 101, 51.



(a) (b)

(c) (d)

Figure 4.6: The four captured segments of Figure 4.5 (red).



(a) Random noise. (b) Pertubation (sin(x)).

Figure 4.7: Perturbation added to Figure 4.5. The pixel intensities in (a) and (b) were rescaled to enhance the perturbations such that they are visible in the plots.

Visually it is clear that the different regions still exist and the resulting partition should resemble the segmentation that was determined in Figure 4.6. If the parameters used to segment Figure 4.5 are used to segment Figure 4.8, we obtain a severely over-segmented result. The large value for α magnifies the contrast between neighboring pixels within a region at the finest level. Due to the inclusion of the random noise, nodes did not group together on the coarser levels. In fact, all 7891 nodes at the 9^{th} level were determined to be segments. Clearly, this is not a good result. A solution to this problem is to decrease α and increase $\tilde{\alpha}$, which scales the average intensity differences between regions at coarse levels. Averaging the intensities within a region reduces the effect that the manufactured noise has on the affinity between nodes. Using the parameters: $\theta = 0.25$, $\alpha = 1$, $\tilde{\alpha} = 30$, $\beta = 0$, $\rho = 0$, $\delta = 0.15$, $\sigma = 9$, and $\gamma = 10^{-5}$, we were able to obtain the segmentation results in Figure 4.6. Also, if we add the perturbation generated in Figure 4.7 to Figure 4.1 we obtain the image in Figure 4.9. Using the same parameters to segment Figure 4.9 as Figure 4.8, we obtain the segmentation that was presented in Figure 4.2.

In image processing, random perturbations are normally modeled using Gaussian noise. Gaussian noise is generated using a Gaussian probability distribution function (pdf) with mean μ_g and variance σ_g^2. A special case of a Gaussian pdf is the normal distribution, which has a mean of 0 and variance of 1. Gaussian noise was not used in this example, but we expect that the algorithm would obtain similar results if Gaussian noise is added to the image.

4.2.4 Example 4: Texture

Figure 4.10 is used to illustrate how intensity variance is an important block property for identifying textures in the segmentation process. Figure 4.10 shows four distinct regions.

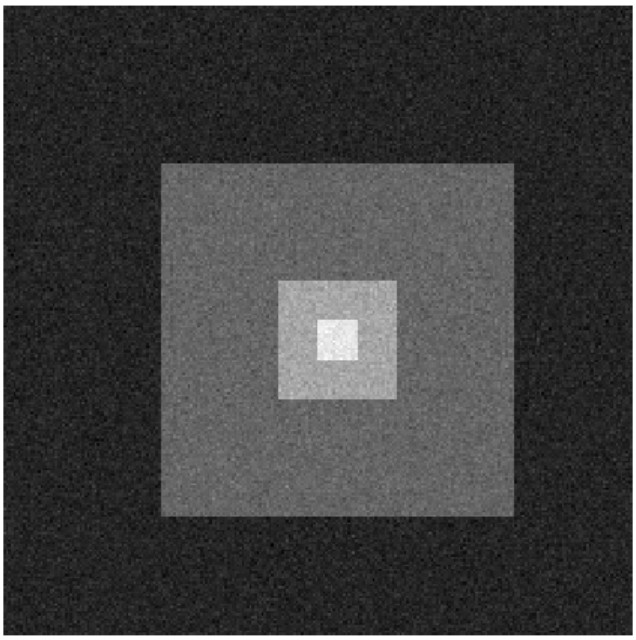


Figure 4.8: Grayscale image of Figure 4.5 with noise (256x256).

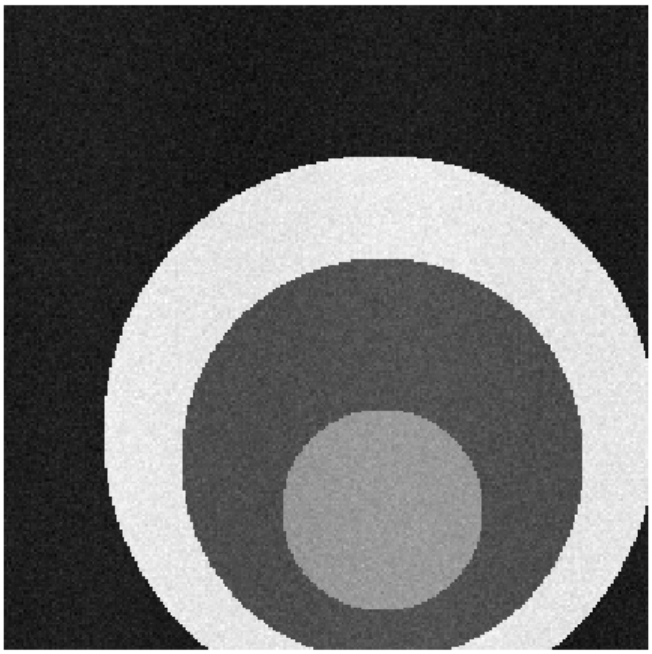


Figure 4.9: Grayscale image of Figure 4.1 with noise (256x256).

There are two distinct checkerboard regions separated by two uniform regions. The average intensity over each of the four regions is the same. The uniform regions have a pixel intensity of 20 and the checkerboard region alternates between 0 and 40. Since the average intensities over the regions are the same, the algorithm requires a different measure to distinguish the regions. Over the checkerboard region there is a lot of variation between the pixel intensities, whereas the uniform regions by construction have none. The concept of intensity variance is able to capture these differences and is used in our implementation as a measure of texture.

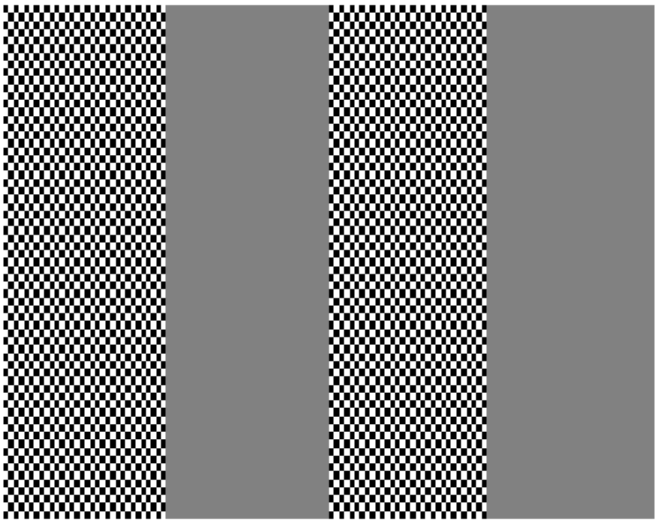
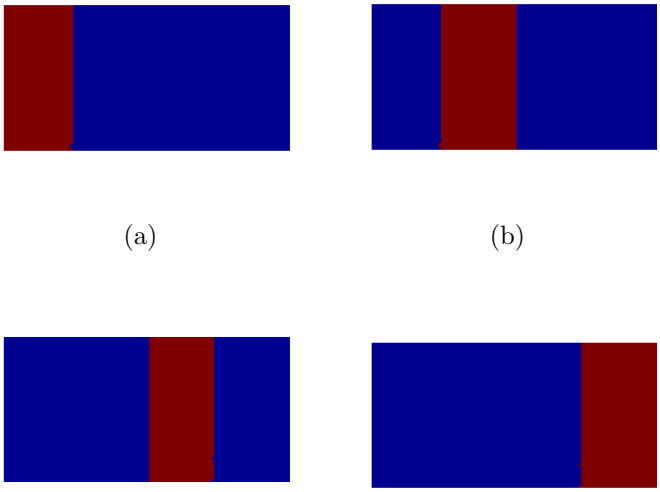


Figure 4.10: Grayscale image with textured regions (128x65).

In Figure 4.11 the red regions denote the salient blocks. The parameters used to determine this partition were $\theta = 0.25$, $\alpha = 4.2$, $\tilde{\alpha} = 1$, $\beta = 5$, $\rho = 0$, $\delta = 0.15$, $\sigma = 8$, and $\gamma = 10^{-30}$. The segmentation of Figure 4.10 obtained in Figure 4.11 is very promising and it took 6.17 seconds to segment. Overall, the segmentation algorithm was able to capture the regions with a high variance in pixel intensity. In Figure 4.11 we observe that a few



(a) (b)

(c) (d)

Figure 4.11: The four captured segments of Figure 4.10 (red)

boundary pixels were not properly segmented. This anomaly could have been avoided by decreasing the α parameter and increasing β. At the uniform and checkerboard region boundary, the white pixels in the checkerboard region are closer in intensity to those of the uniform gray region than to the dark pixels in the checkerboard region. This alteration of the parameters would decrease the affinity of the mislabeled pixels with respect to the difference in intensity, and increase the affinity of the pixels to differences in intensity variance. This example illustrates the difficulty of fine-tuning the parameters to obtain the desired segmentation results. Once a reasonable partition of the image has been obtained, additional human intervention may be required to guide the algorithm into producing an optimal result.

Chapter 5

Segmentation of Aerial-view Images

Evidence from the previous chapter has shown that our implementation of the SWA approach is capable of segmenting relatively simple images. The next step is to observe if the algorithm is capable of producing a correct segmentation for more complex images. The purpose of this chapter is to apply the segmentation algorithm to real images and observe the resulting partitions. The focus of the input data will be on aerial-view images taken from nadir. That is, the images used in our research are taken directly below the device that captures it.

In recent years, satellite imagery and, more broadly, aerial imagery has become widely available due to affordable and easy to use software used for viewing and editing, with access to large image databases such as Google Earth [6]. Aerial-view images have become a significant source of information in industrial applications. For example, geologists have been successfully using satellite imagery for a number of years for mineral exploration as a cost-effective method for identification of possible future excavation sites [12]. Aerial-view images can also be used for road extraction for road related services, such as navigation

software and location-based services [42]. Segmentation techniques can be applied to aerial-view images to update land-use databases [22]. Segmentation methods have also been used to assist researchers to create geo-specific road databases for the updating of driving simulators [18]. Aerial-view images are also an important source of information for the forestry industry, city planning and the military.

The goal of this chapter is to segment airports and different structures in airports. The American Federal Aviation Administration (FAA) and other government bodies would like to keep an updated and accurate database of airports and their runways. For airport safety, they would like to be aware of new runways and runways that are no longer operational. Keeping an up-to-date database can be very time-consuming as there is currently no automated method to update it. Satellite images can be used to update these databases in time by analyzing temporal sequences of images, but with the high resolution of the images and the large areas of interest, processing this raw data can be computationally expensive. Automatic segmentation methods are a natural choice to address this problem, since the goal of these methods is to simplify the input image into a meaningful partition.

The data used for our research was derived from images from Google maps and and gathered by the Geoeye satellite Ikonos [2]. The sample images that were derived from Google maps were captured using a screenshot tool on a Windows Vista machine. This is not the ideal way to receive satellite imagery because there is limited information with respect to the spatial resolution of the images and how they were captured. Also, since RGB colour spaces are device-dependent [15, 39], the image data may not be the same depending on the computer that captures it. The high-resolution images that were provided by Geoeye were captured by the satellite Ikonos that was put into orbit in 1999. The Ikonos satellite is capable of collecting panchromatic (black and white) images with 82-cm resolution and

multispectral imagery with 4-m resolution [2]. The panchromatic and multispectral images can be combined to produce high-resolution colour images with a spatial resolution of 1-m per pixel. This process of combining images is called pansharpening. The Ikonos cannot capture features that are under one meter in size for colour images and a resolution of 82-cm for panchromatic images.

The images that were used for segmentation in this chapter have a colour depth of 24 bits. The satellite images provided by Geoeye were down-sampled from 48-bit to 24-bit to reduce to amount of time required to test the segmentation algorithm.

5.1 Farm Image

Figure 5.1 contains an RGB colour image of a farm in the Waterloo region that was captured using a cropped screenshot obtained from the Google maps application. The goal of this example is to show that the algorithm is able to divide the original image into meaningful segments. In order to begin the segmentation, the RGB values of the image in Figure 5.1 were first converted to the CIELAB colour space. This colour space was then used to construct the coupling matrix using Equation (3.2).

The results from the segmentation are shown in Figure 5.2. Each solid colour represents a single segment. The parameters used for the segmentation were: $\theta = 0.3$, $\alpha = 3.5$, $\tilde{\alpha} = 3$, $\beta = 3$, $\rho = 2$, $\delta = 0.15$, $\sigma = 9$ and $\gamma = 10^{-70}$ and it took 58.33 seconds to segment. Note that the coarse-level elongation measure was used ($\rho = 2$).

The segmentation in Figure 5.2 appears to be a relatively good partition of the input image. The crop areas were identified and successfully partitioned. The small service road in the bottom right of Figure 5.1 was also detected. The region near the farm property

Figure 5.1: RGB image of a farm (180x180).

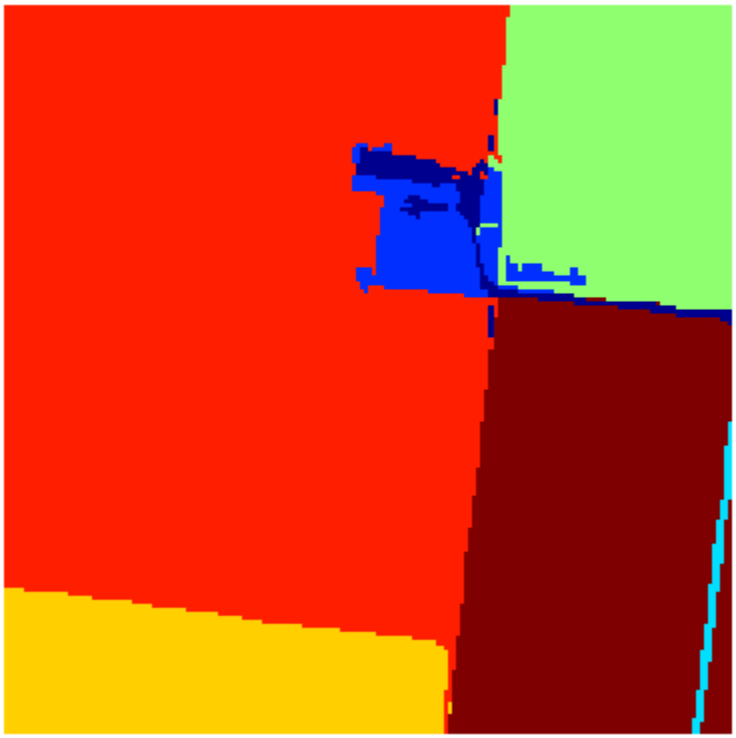


Figure 5.2: Seven captured segments determined (180x180). Each colour represents a segment.

had some difficulty in the segmentation, but it appears that the road leading up to it was captured. The false identification of some points near the farm property may be attibuted to the low spatial resolution of the image.

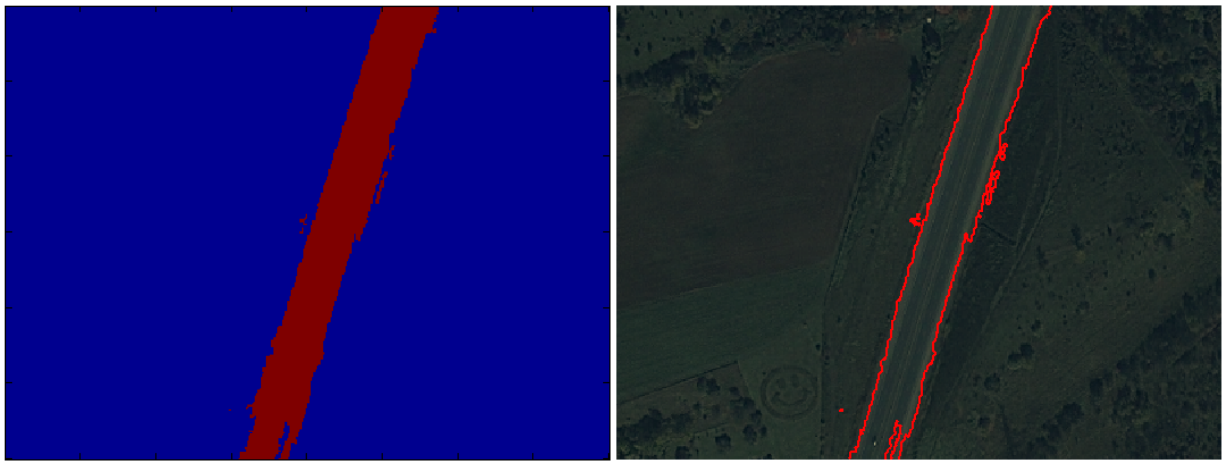
5.2 Road Image

The next image to be discussed is an image that was taken by the Ikonos satellite. Due to the high spatial resolution of the image, only a small window was taken, but at the full resolution of the original satellite image. Figure 5.3 displays the image that the algorithm was applied to. The contrast and brightness of the image in Figure 5.3 was changed for better presentation. The goal for this image is more specific: the region of interest is the road running from top-to-bottom on Figure 5.3. There are several reasons to want to segment roads. One reason is for roadmaps and navigation systems. An automated method would be able to update a database on the construction of new roads. A successful segmentation would require the road to be identified accurately.

The parameters used to determine the segmentation of Figure 5.3 were: $\theta = 0.3$, $\alpha = 5.2$, $\tilde{\alpha} = 2.5$, $\beta = 2.5$, $\rho = 2$, $\delta = 0.15$, $\sigma = 12$, and $\gamma = 10^{-12}$. The segmentation of Figure 5.3 took 362.68 seconds and the results are displayed in Figure 5.4a.

In Figure 5.4a two segments were found. The top-to-bottom road segment was identified by the algorithm, along with a background segment. At the bottom of Figure 5.4a there is a small blue region that is surrounded by red. This appears to be an area where the segmentation algorithm incorrectly labeled the pixels. By observing the image in Figure 5.4b we can see that the red segment from Figure 5.4a was able to capture the road in its entirety. However, several pixels along the boundary were mislabeled and grouped

Figure 5.3: RGB colour image taken by Ikonos (400x300).



(a) Two segments of Figure 5.3 (400x300).

(b) Road segment of Figure 5.3 (400x300).

Figure 5.4: The captured segments of Figure 5.3 shown in two different ways. (a) The segments are displayed as two distinct colours, blue and red. (b) A red line is drawn on top of the original image to indicated the boundary between segments.

along with the road. A solution to this problem may be to fine-tune the parameters until the mislabeled area is correctly grouped with the road segment. For the purposes of this research, it demonstrates a good result: the largest part of the road was identified, but some fine-tuning of the parameters may be required to properly segment the mislabeled region.

In Sections 5.1 and 5.2 we demonstrated that the algorithm is customizable by the user. Depending on the features of interest, the algorithm can be trained to the application by changing the user-defined parameters. In the first example the goal was to provide a meaningful partition of the entire farm input image shown in Figure 5.1, and in the second example, the goal was to identify the road in Figure 5.3. The algorithm was able to satisfy both objectives, but with different parameters.

The cost of having several user-defined parameters is felt in the amount of time required to fine-tune them. The amount of human supervision for this algorithm can be costly as there are a large number of user-defined parameters. This is especially evident when supervision is required in the parameter selection phase to segment an image that is of a considerable size.

5.3 Airports

The segmentation of airport taxiways and aprons is an important application for the algorithm. Several institutions would like to use an algorithm capable to segmenting these areas. The shape moments measure described in Chapter 3 is useful for segmenting the elongated paved areas because the taxiways and aprons are elongated in nature. In the next example we take a look at a high-resolution image of the Region of Waterloo airport

provided by Geoeye and taken by the Ikonos satellite. The original high-resolution satellite image was quite large, and for testing the algorithm a reduction in its size was conducted due to memory and time concerns. It should be noted that due to its linear work and memory complexity, the SWA algorithm is in principle, able to segment high-resolution images on computers with sufficient computing power and with an efficient implementation. For our test implementation, the high-resolution satellite image was scaled down to 30% of the original size using imresize.m, a built-in Matlab function that scales the image using a nearest neighbour interpolation. The resolution of the scaled-down image was 691x541. Figure 5.5 displays the high-resolution image that was provided by Geoeye. To compute the coupling matrix of the scaled-down version of the image in Figure 5.5, the RGB values of Figure 5.5 were first converted to the CIELAB colour space and Equation (3.2) was used.

Figure 5.5: RGB image of the Region of Waterloo airport taken by Ikonos (2301x1801).

Figure 5.6 displays the segmentation of the scaled-down version of Figure 5.5 with each colour representing a single segment. In this image, nine segments were determined. The

parameters used to produce the segments were: $\theta = 0.3$, $\alpha = 4.2$, $\tilde{\alpha} = 2.5$, $\beta = 2$, $\rho = 1$, $\delta = 0.15$, $\sigma = 11$ and $\gamma = 10^{-40}$. The algorithm runtime to determine the segments of the scaled-down version of Figure 5.5 was 3546.43 seconds.

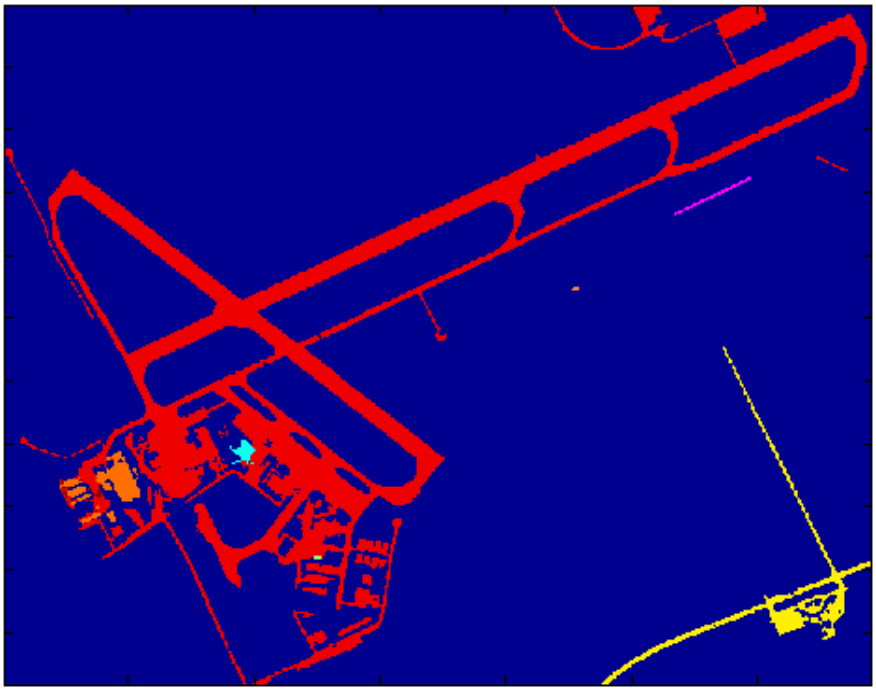


Figure 5.6: The nine segments determined for Figure 5.5 (691x541). Each colour represents a different segment.

The results shown in Figure 5.6 are promising. The algorithm was able to identify the runway, and essentially all paved areas. There appears to be some under-segmentation as a few of the airport buildings were segmented together with the runways and a paved service area next to the main runway was also missed. With some fine-tuning of the user-defined parameters this may be improved. The under-segmentation of Figure 5.5 can also be attributed to scaling the input image down by 70% of the original satellite image resolution. To test this hypothesis, the segmentation algorithm would have to be run on the original high-resolution image.

5.3.1 High-Resolution Runway

In the next example, a small sample of the airport runway shown in Figure 5.7 is taken at full resolution and the algorithm is applied to it with the results illustrated in Figure 5.8. To incorporate colour into the segmentation process, the RGB values were converted to the CIELAB colour space, as before.

Figure 5.7: RGB image of the Region of Waterloo airport runway (501x351).

Figure 5.8 displays the resulting segmentation. Each colour represents a single segment. In this image, nine segments were determined. The parameters used to produce the segments were: $\theta = 0.3$, $\alpha = 4.2$, $\tilde{\alpha} = 2.5$, $\beta = 2$, $\rho = 1$, $\delta = 0.15$, $\sigma = 11$ and $\gamma = 10^{-40}$. The algorithm runtime to segment Figure 5.7 was 833.94 seconds. Note that these are the same parameters that were used to segment Figure 5.5.

The algorithm was able to segment Figure 5.7 with success. The intersecting runways were found as one segment, which is a strong result. At the higher resolution more fine details from the image are represented in the segmentation. The two painted lines on the

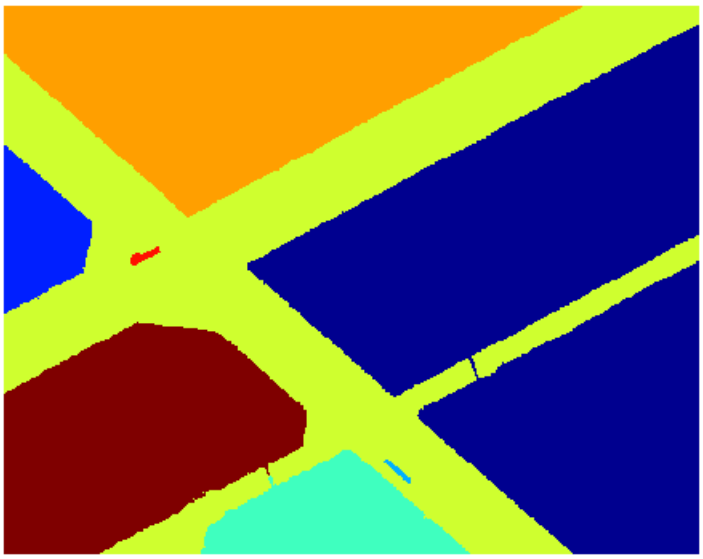


Figure 5.8: The nine segments found in Figure 5.7 (501x351).

runway were declared as segments, which is an over-segmentation of the image for our purposes. Also, because of the high resolution of the image there are noticeable gaps in the runway appearing in the segmentation in Figure 5.8. The segmentation of Figure 5.7 was an overall success, but depending on the desired outcome, some fine tuning of the parameters may be required.

5.4 Road Network Extraction

Roads are an important aspect of everyday life and with the rapid growth of road-related services such as navigation systems, it has become an important task to keep up-to-date road databases. Current methods for updating road information rely heavily on manual work and can become expensive and time-consuming [42]. In this section we discuss a possible method for the extraction of road connectivity information using the segmentation obtained by the SWA algorithm. At the end of the SWA algorithm, the pixels in an image

are partitioned into segments. Recall that the SWA algorithm detects the saliency of groups of pixels as they emerge as single nodes on progressively coarser graphs on incremental levels. By tracking the interpolation of the salient node back to the finest level of the algorithm from the coarse level at which it was detected, the coarse sub-graphs of the segment can be identified at each level.

Figure 5.9 illustrates the image of Figure 5.7 scaled to 10% of the spatial resolution for simplicity. The image in Figure 5.9 was segmented using the following parameters: $\theta = 0.3$, $\alpha = 6$, $\tilde{\alpha} = 5$, $\beta = 2$, $\rho = 2$, $\delta = 0.15$, $\sigma = 7$ and $\gamma = 10^{-12}$. The segmented image is illustrated in Figure 5.10, with the segment of interest in orange.

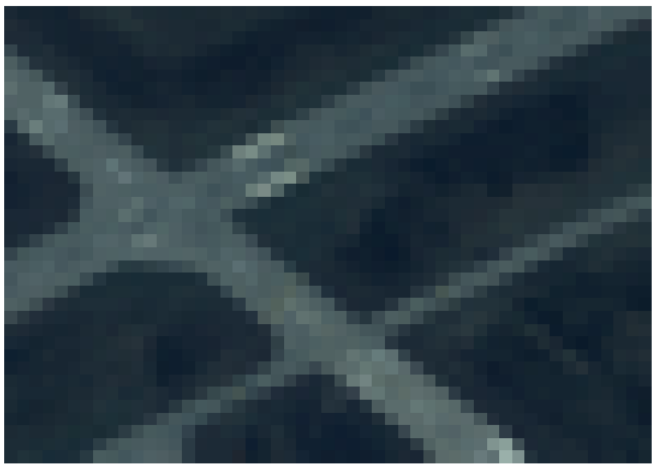


Figure 5.9: Scaled runway image (51x36).

Figures 5.11 to 5.18 illustrate the coarse graphs of the orange segment from level eight to the finest level. The red dots indicate the centres of mass of each block on the indicated level. The black lines between the red dots indicate a graph edge. Edges between connected nodes that are far apart are not displayed: edges between nodes that are further apart than

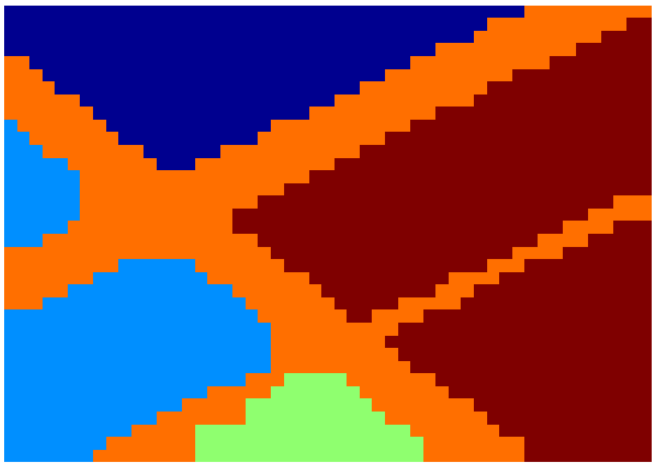


Figure 5.10: Segmented image. 4 segments found.

a scalar multiple of the sums of the maximum eigenvalue of (3.23) for each node are not shown. In particular, for nodes i and j, if $||(\overline{x},\overline{y})_i - (\overline{x},\overline{y})_j|| < \mu|\max(\omega_i) - \max(\omega_j)|$ with ω_i the eigenvalues of Λ_i described in (3.24) (and μ a user-defined parameter), then the edge is shown.

Using the coarse-graph hierarchy from Figures 5.11 to 5.18, we are able to extract some information about the road connectivity networks from the links between sub-blocks. In Figures 5.13, 5.12 and 5.11 (levels 6,7 and 8) there are not enough nodes in the coarse graph to extract sufficient information about the road connectivity. In Figures 5.15 and 5.14 (levels 4 and 5) there are several undesirable edges in the graph that cannot be easily removed. In Figures 5.17 and 5.16 (levels 2 and 3) the local connectivity of the nodes provides a nice representation of the network of nodes that comprise the road segment. However, to produce this representation, a large amount of nodes and a significant amount of parameter tuning is required.

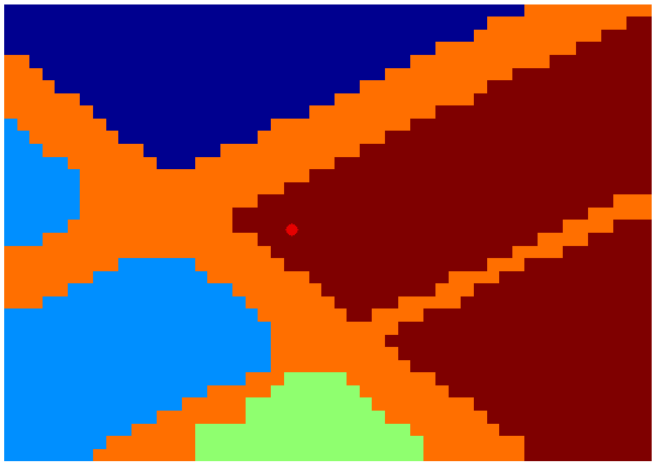


Figure 5.11: Segmented image with coarse graph overlay (level 8, $\mu = 1$).

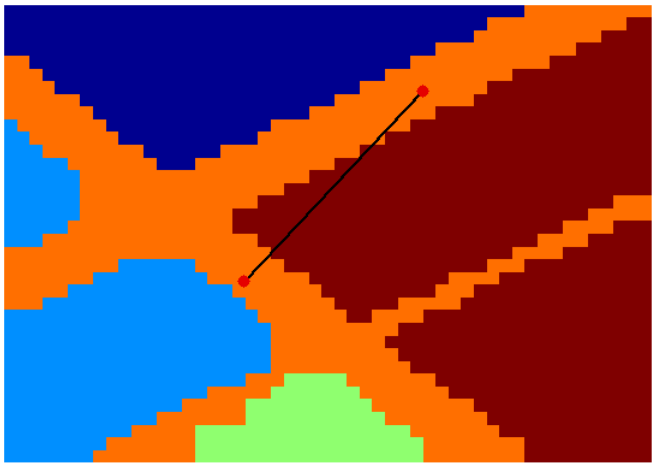


Figure 5.12: Segmented image with coarse graph overlay (level 7, $\mu = 1$).

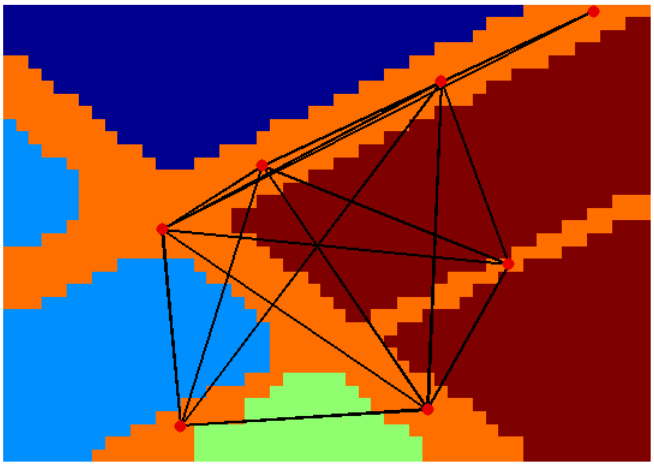


Figure 5.13: Segmented image with coarse graph overlay (level 6, $\mu = 1$).

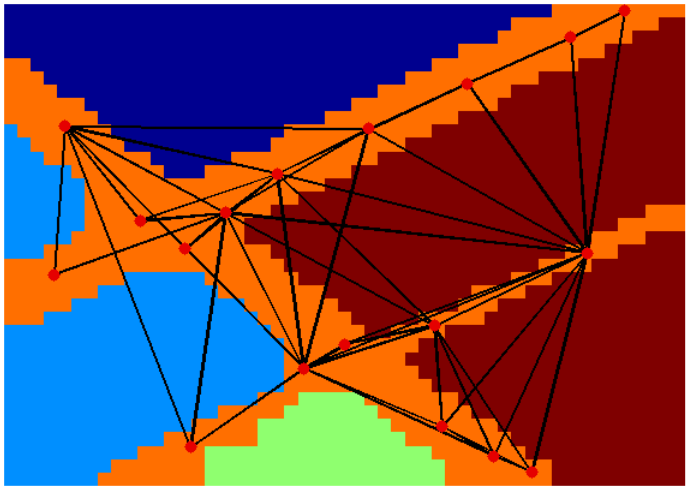


Figure 5.14: Segmented image with coarse graph overlay (level 5, $\mu = 1$).

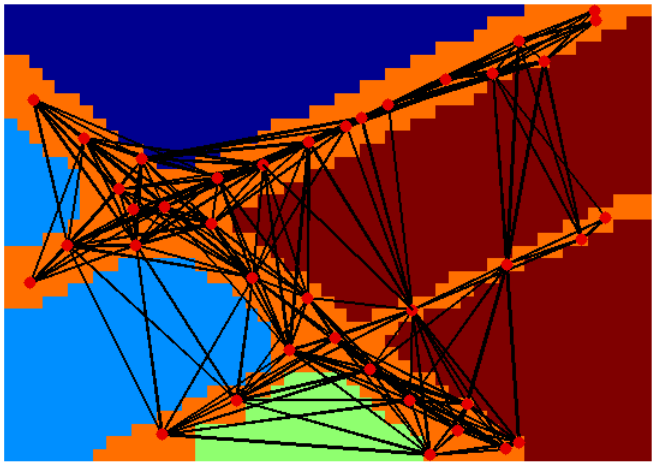


Figure 5.15: Segmented image with coarse graph overlay (level 4, $\mu = 1$).

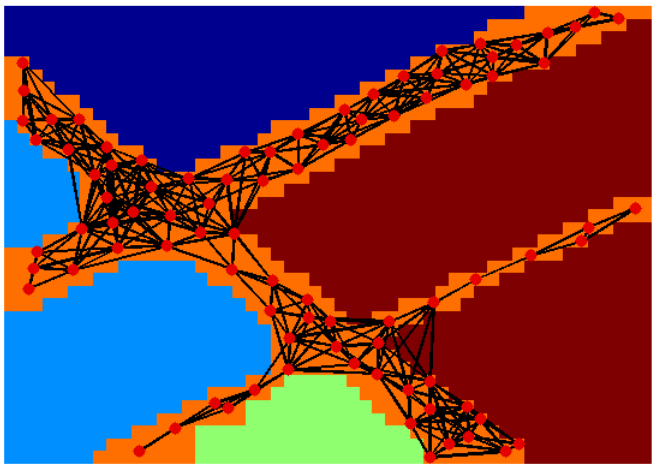


Figure 5.16: Segmented image with coarse graph overlay (level 3, $\mu = 1$).

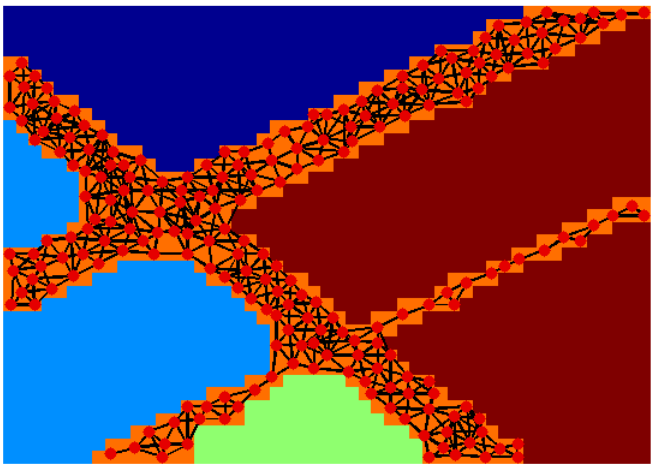


Figure 5.17: Segmented image with coarse graph overlay (level 2, $\mu = 1$).

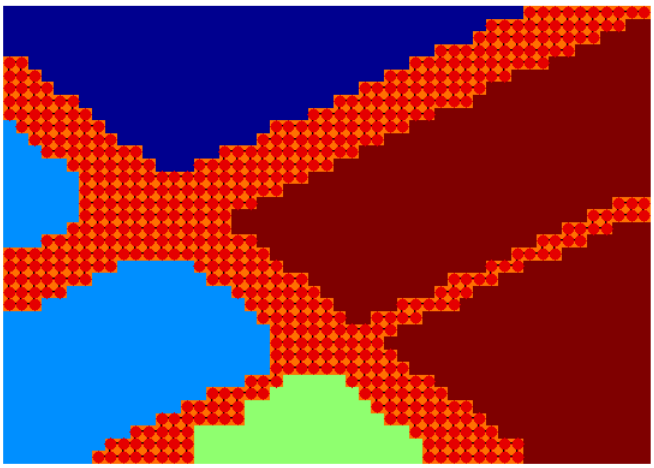


Figure 5.18: Segmented image with fine graph overlay (level 1, $\mu = 1$).

Chapter 6

Conclusion and Future Work

In this thesis the Segmentation by Weighted Aggregation algorithm for image segmentation was studied and improved. A detailed description of the algorithm and its components was given, including a new scale-invariant saliency measure. The new scale-invariant saliency measure was introduced to replace the original SWA saliency measure and was found to provide more consistent segmentation results. The scale-invariant measure simplifies the user-defined saliency threshold determination since the saliency of a block is no longer dependent on the scale at which it was calculated . The SWA algorithm was implemented in Matlab and several types of images were investigated to test the segmentation performance of the algorithm. The images that were segmented ranged from simple grayscale images with uniform regions to high-resolution images taken by a commercial satellite. Vision is generally a very difficult problem for computers, since complex vision processes in humans are still not well known, but the SWA algorithm has provides a strong framework for continued research.

The modified SWA algorithm was tested on several different types of images. It was

found that the SWA algorithm performed well in the segmentation of the simple test images. Without much fine-tuning of the parameters a proper segmentation was found in a short period of time. For real images, the SWA algorithm required more human intervention in terms of the parameter selection, but eventually a favourable segmentation was obtained in most cases.

The SWA algorithm is a powerful tool for image segmentation, and more generally computer vision. The SWA algorithm provides a fast segmentation method that can be used to segment a broad variety of images. One of the most significant aspects of the algorithm, is its ability to be extended without much difficulty. The SWA algorithm originally only used the pixel intensity information to segment images, with other image features included in later versions. Due to the multilevel nature of the algorithm, it is capable of incorporating high-level vision cues such as shape and orientation easily. The SWA algorithm is not confined to using only low-level vision in its determination of segments. In that sense, it is more like human vision than most segmentation methods available at the time this research took place. Another benefit of the method is that it can be trained to the application which the user is interested in by adding relevant high-level cues and fine-tuning the user-defined parameters. Although the number of parameters available allows for a great deal of customization, it also presents a great difficulty in fine-tuning the segmentation. Each parameter affects the segmentation in a different way, thus, determining the desired segmentation can take a lot of time.

A modified version of the SWA method was implemented in this research and it was able to provide accurate segmentation results for each image that was tested. In particular, we saw that the results from satellite imagery showed significant promise in terms of the segmentation and in identifying certain specific regions and road connectivity.

6.1 Future Work

Several future directions are available in this research. First, it would be beneficial to optimize the implementation to realize the benefits of a multilevel method. Multilevel methods can achieve linear complexity as a function of the number of pixels in terms of computation work and storage.

Another direction would be to improve the measures that are used to calculate the similarity between blocks, such as introducing a multiscale measure for the intensity variance regional property. A multiscale measure for the intensity variance retains more information about the block during the coarsening process, and the additional information about the sub-blocks can improve the characterization of the texture within the block. The addition of more high-level cues such as the orientation of the shape moments to provide more information about texture and orientation is another way to improve the segmentation process. The SWA framework is constructed such that including new high-level cues to update the similarity between blocks or improving the measures that are currently being used, can be done quite easily. The characteristics of a block can be accumulated and used to update the coupling between blocks similar to the process described in Section 3.5.

Significant future work would be to find a more automated segmentation method, that is to decrease the amount of supervision in the algorithm. A trade off takes place at this point, since it is usually the case that human intervention improves the segmentation. A future study would have to decide which parameters can be hidden from the user and which the user should retain control of. The general philosophy behind the SWA approach is to increase the number of high-level cues and hence the parameters up to a point where, for a given class of images, there is no longer a need to change the parameters. Thus, a fully

automated method is achieved, for that class of images. Further study is required to see up to which degree this goal is attainable.

APPENDICES

Appendix A

RGB conversion to CIELAB

The conversion from RGB coordinates to CIELAB is used to provide a better estimation of colour differences. The conversion is not straightforward and it involves several steps. The RGB coordinates need to be mapped to the device-independent CIEXYZ colour space [1, 15], which is the original 1931 CIE colour space specification, before they can be converted to the CIELAB colour space. The conversion to the CIEXYZ colour space is required because RGB colour spaces are device-dependent. The first step in the conversion is to normalize the RGB values (R, G, B). For 8-bit images, divide by 255:

$$R = \frac{R}{255}, \tag{A.1}$$

$$G = \frac{G}{255}, \tag{A.2}$$

$$B = \frac{B}{255}. \tag{A.3}$$

To reduce the amount of quantization error, the RGB coordinates must first be γ-encoded

[30]. The γ-encoding function is a nonlinear transformation of the RGB values to reduce the quantization errors. In general, we get that,

$$\begin{pmatrix} R_1 \\ G_1 \\ B_1 \end{pmatrix} = \begin{pmatrix} f(R) \\ f(G) \\ f(B) \end{pmatrix}, \tag{A.4}$$

where $f(\cdot)$ is the γ-encoding function. For the standard sRGB colour space developed by Hewlett-Packard and Microsoft,

$$f(C) = \begin{cases} \frac{C}{12.92} & \text{if } C \leq 0.04045, \\ \left(\frac{C+0.055}{1.055}\right)^{2.4} & \text{if } C > 0.04045, \end{cases} \tag{A.5}$$

where C is a particular colour coordinate of the sRGB colour space. Then, each component is transformed using the following linear system,

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{pmatrix} \begin{pmatrix} R_1 \\ G_1 \\ B_1 \end{pmatrix}. \tag{A.6}$$

The device independent CIEXYZ coordinates are transformed using the following nonlinear relationship,

$$L* = 116 \left(\frac{Y}{Y_n}\right)^{\frac{1}{3}}, \tag{A.7}$$

$$a* = 500 \left[\left(\frac{X}{X_n}\right)^{\frac{1}{3}} - \left(\frac{Y}{Y_n}\right)^{\frac{1}{3}}\right], \tag{A.8}$$

$$b* = 200 \left[\left(\frac{Y}{Y_n} \right)^{\frac{1}{3}} - \left(\frac{Z}{Z_n} \right)^{\frac{1}{3}} \right]. \tag{A.9}$$

Here X_n, Y_n and Z_n refer to the X, Y, Z coordinates of a reference white patch in the CIEXYZ colour space. The CIE standard illuminant D65 is often used [30]. The coordinates for the D65 point are: $X_n = 95.04$, $Y_n = 100.00$, $Z_n = 108.88$. Then (L^*, a^*, b^*) is the coordinate vector of the colour in the CIELAB colour space.

Appendix B

Interpretation of G and V operators

The following examples illustrate the reasoning behind the construction of the scale-invariant saliency measure described in (3.16). First, a one-dimensional case is shown, followed by a two-dimensional explained in stencil form. To simplify notation, let $P(:,k)$ denote the k^{th} column of P.

B.1 1-D case

Let the one-dimensional stencil for the (3.15) operator be given by

$$G = \begin{pmatrix} -1 & 2 & -1 \end{pmatrix} \tag{B.1}$$

and the one-dimension stencil for (3.14) by

$$V = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}. \tag{B.2}$$

Consider a non-overlapping aggregate k with three elements. Then the column k of P is given by

$$P(:,k) = \left(\begin{array}{ccc|ccc|ccc} \dots & 0 & & 1 & 1 & 1 & 0 & \dots & \end{array} \right)^T. \tag{B.3}$$

The coarse operator (3.7) satisfies

$$G^c_{kk} = P(:,k)^T G P(:,k) \tag{B.4}$$

$$= \left(\begin{array}{ccc|ccc|ccc} \dots & 0 & & 1 & 1 & 1 & 0 & \dots & \end{array} \right) G \left(\begin{array}{c} \vdots \\ \hline 0 \\ \hline 1 \\ 1 \\ \hline 1 \\ \hline 0 \\ \hline \vdots \end{array} \right) \tag{B.5}$$

$$= \left(\begin{array}{ccc|ccc|ccc} \dots & 0 & & 1 & 1 & 1 & 0 & \dots & \end{array} \right) \left(\begin{array}{c} \vdots \\ \hline -1 \\ \hline 1 \\ 0 \\ 1 \\ \hline -1 \\ \hline \vdots \end{array} \right) \tag{B.6}$$

$$= 2. \tag{B.7}$$

Thus, $G^c_{kk} = 2$ which gives the boundary length of the segment k. For the operator V, we get the following:

$$V^c_{kk} = P(:,k)^T V P(:,k) \tag{B.8}$$

$$= \left(\begin{array}{ccc|ccc|cc} \dots & 0 & & 1 & 1 & 1 & 0 & \dots \end{array} \right) V \begin{pmatrix} \vdots \\ 0 \\ \hline 1 \\ 1 \\ 1 \\ \hline 0 \\ \vdots \end{pmatrix} \tag{B.9}$$

$$= \left(\begin{array}{ccc|ccc|cc} \dots & 0 & & 1 & 1 & 1 & 0 & \dots \end{array} \right) \begin{pmatrix} \vdots \\ 1 \\ \hline 1 \\ 2 \\ 1 \\ \hline 1 \\ \vdots \end{pmatrix} \tag{B.10}$$

$$= 4 \tag{B.11}$$

We get that $P(:,k)^T V P(:,k) = 4$, which is twice the number of internal undirected edges in the aggregate k.

B.2 2-D case

Let the stencil for the unweighted graph Laplacian G be given by

$$G = \begin{pmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{pmatrix}, \tag{B.12}$$

and the stencil for V be given by

$$V = \begin{pmatrix} & 1 & \\ 1 & & 1 \\ & 1 & \end{pmatrix}. \tag{B.13}$$

Consider a square non-overlapping aggregate k with nine elements, then the k^{th} column of P in stencil form gives

$$P(:,k) = \left(\begin{array}{c|ccc|c} 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right). \tag{B.14}$$

The coarsening procedure gives

$$G^c_{kk} = P(:,k)^T G P(:,k) \tag{B.15}$$

$$= \left(\begin{array}{c|ccc|c} 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right) G \left(\begin{array}{c|ccc|c} 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right) \tag{B.16}$$

$$= \left(\begin{array}{c|ccc|c} 0 & -1 & -1 & -1 & 0 \\ \hline -1 & 2 & 1 & 2 & -1 \\ -1 & 1 & 0 & 1 & -1 \\ -1 & 2 & 1 & 2 & -1 \\ \hline 0 & -1 & -1 & -1 & 0 \end{array}\right) \left(\begin{array}{c|ccc|c} 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right) \tag{B.17}$$

$$= 12, \tag{B.18}$$

which is the boundary length of block k. For V we get

$$V_{kk}^c = P(:,k)^T V P(:,k) \tag{B.19}$$

$$= \left(\begin{array}{c|ccc|c} 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right) V \left(\begin{array}{c|ccc|c} 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right) \tag{B.20}$$

$$= \left(\begin{array}{c|ccc|c} 0 & 1 & 1 & 1 & 0 \\ \hline 1 & 2 & 3 & 2 & 1 \\ 1 & 3 & 4 & 3 & 1 \\ 1 & 2 & 1 & 2 & 1 \\ \hline 0 & 1 & 1 & 1 & 0 \end{array}\right) \left(\begin{array}{c|ccc|c} 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right) \tag{B.21}$$

$$= 24, \tag{B.22}$$

which is twice the number of internal undirected edges in block k. This means that the diagonal elements of G^c give the boundary lengths of the blocks, and the diagonal elements of V^c give twice the number of internal connections of the blocks. This can also be seen from (3.9) with $w_{ij} = 1$ for all neighboring pixels.

Appendix C

Comparison of the Original and Scale-invariant Saliency Measures

We would like to compare the behaviour of the original saliency measure Γ that is calculated using Equation (3.12) and the new scale-invariant saliency measure Γ_{new} that is defined by Equation (3.16). A simple image is shown in Figure C.1. It is segmented using both saliency measures. The parameters used to segment the image with both the original saliency measure and the scale-invariant measure were: $\theta = 0.3$, $\alpha = 1$, $\tilde{\alpha} = 0$, $\beta = 0$, $\rho = 0$, $\delta = 0.15$, $\sigma = 7$, and $\gamma = 0.1$. The segments for Figure C.1 are shown in Figure C.2.

Both saliency measures were able to capture the segments accurately. For each measure, Figure C.2a was detected at the 9^{th} level and the segments in Figures C.2b and Figure C.2c were both detected at the 11^{th} level. For more analysis, plots of the original saliency measure and scale-invariant saliency measure values for each node and at each level are shown in Figures C.3 to C.7. Each plot contains the saliency corresponding to each node on the indicated level. The saliency values were sorted to illustrate the behaviour of the

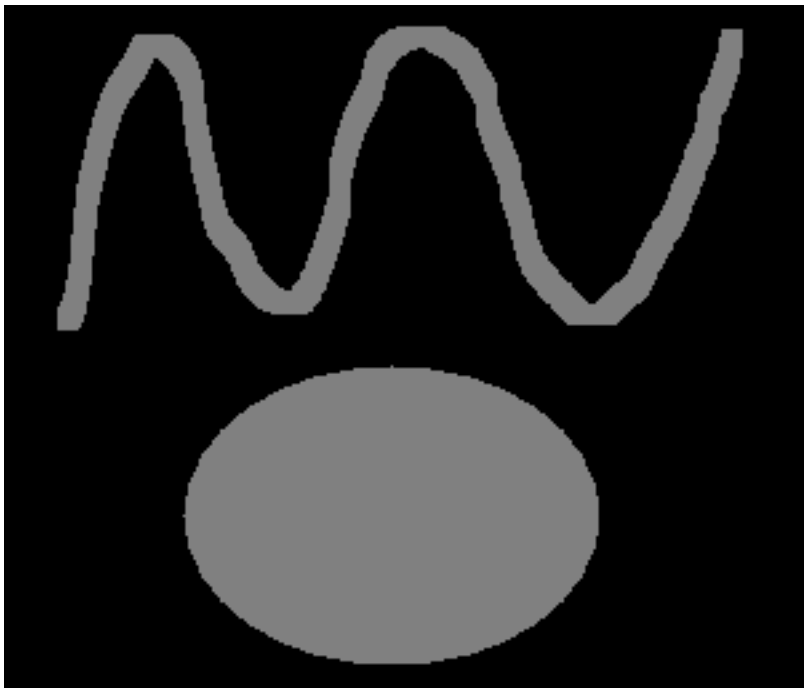
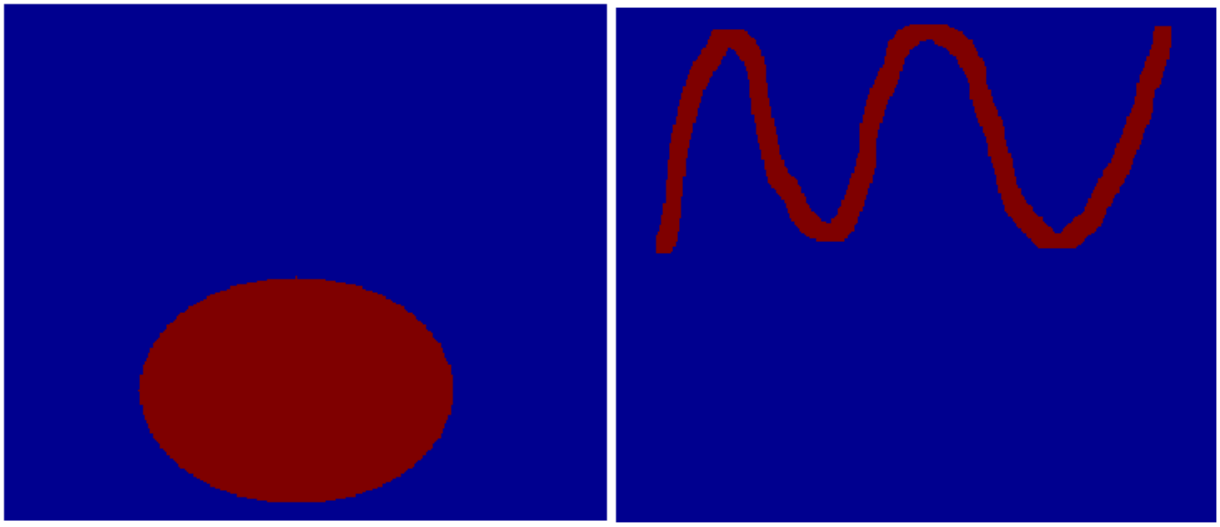


Figure C.1: Grayscale image (300x256).

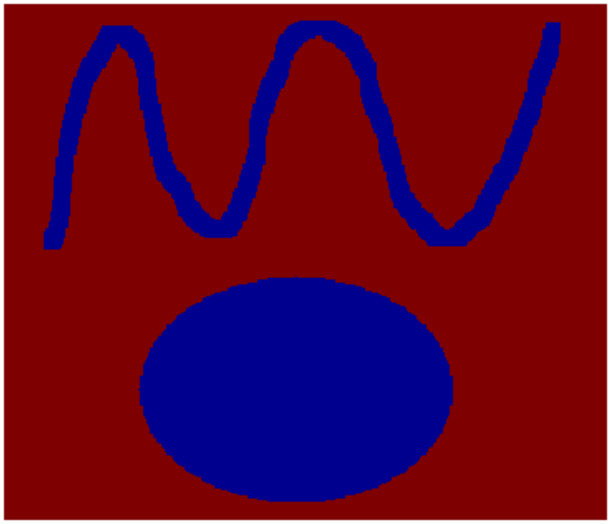
two measures more clearly. The values were sorted to better illustrate the behaviour of the saliency measures.

The most immediate feature of the plots in Figures C.3 to C.7 is that the scale-invariant measure used to segment Figure C.1 obtain values that are consistently between zero and one for each node on each level. On the other hand, the original saliency measure takes on a larger range of values which varies between different levels of the SWA algorithm. Thus, for the scale-invariant measure, we should be able to use the same γ on all levels, but it is not clear that this is the best course of action for the original measure. This feature of the scale-invariant method can be beneficial in determining the saliency threshold parameter γ.

The scale-invariant measure does a better job distinguishing salient and non-salient

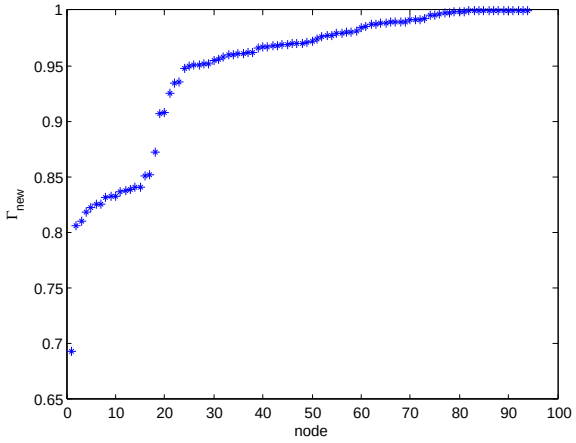


(a) (b)

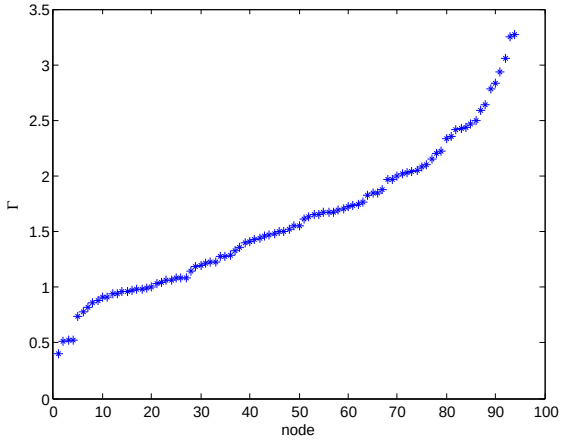


(c)

Figure C.2: The three captured segments of Figure C.1 (red)

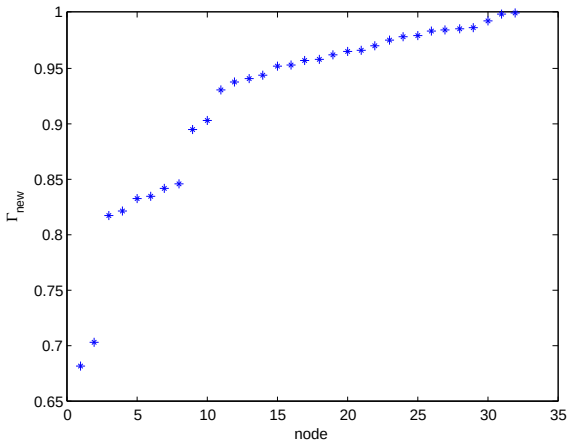


(a) Scale-invariant Measure

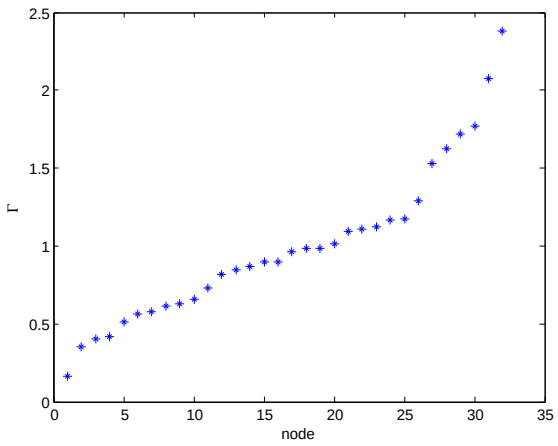


(b) Original Saliency Measure

Figure C.3: Sorted saliency measure values for the segmentation of Figure C.1 (level 7).

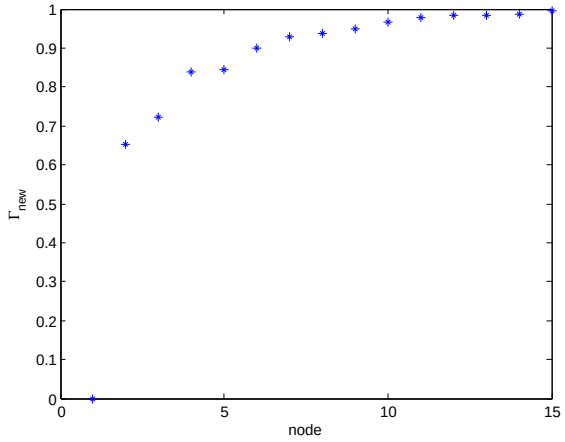


(a) Scale-invariant Measure

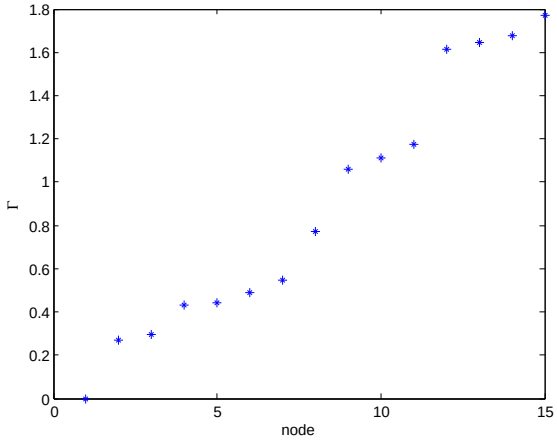


(b) Original Saliency Measure

Figure C.4: Sorted saliency measure values for the segmentation of Figure C.1 (level 8).

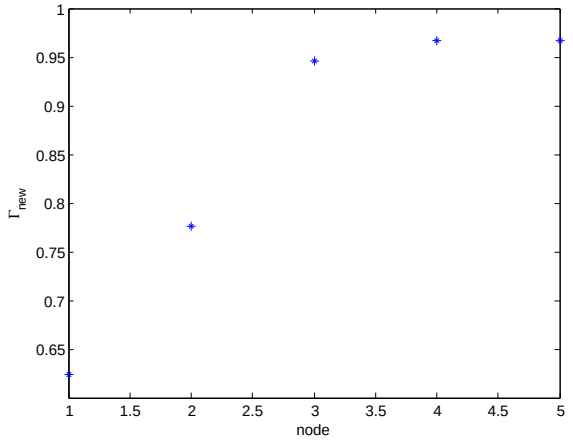


(a) Scale-invariant Measure

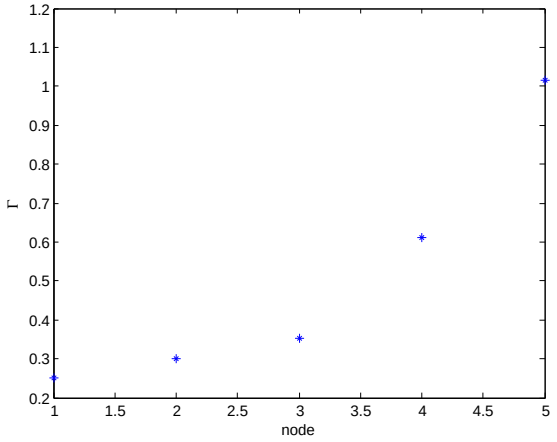


(b) Original Saliency Measure

Figure C.5: Sorted saliency measure values for the segmentation of Figure C.1 (level 9).

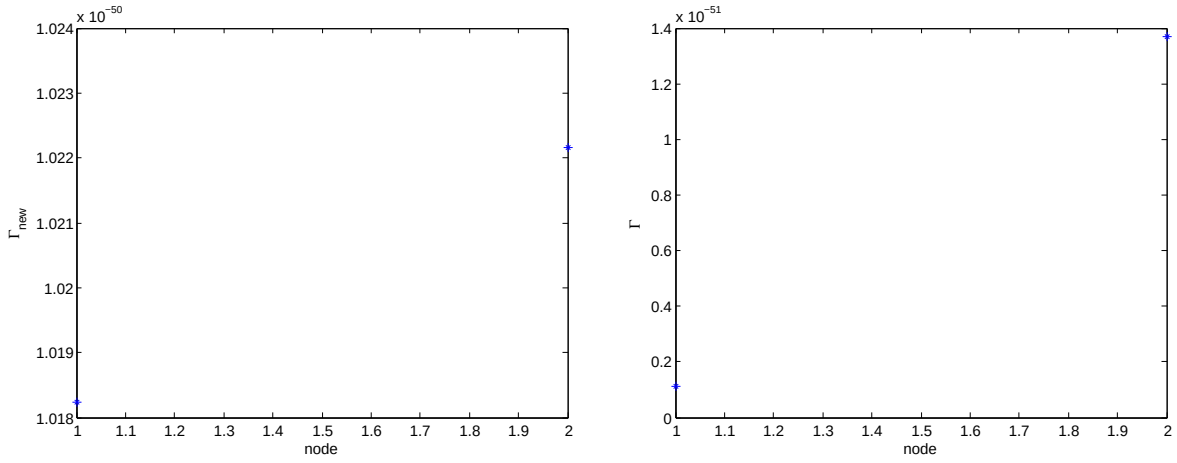


(a) Scale-invariant Measure



(b) Original Saliency Measure

Figure C.6: Sorted saliency measure values for the segmentation of Figure C.1 (level 10).



(a) Scale-invariant Measure (b) Original Saliency Measure

Figure C.7: Sorted saliency measure values for the segmentation of Figure C.1 (level 11).

nodes. From Figures C.3 to C.7, we observe that the Γ values exhibit more gradual changes between nodes, which can become an issue when determining the saliency tolerance parameter γ. The Γ_{new} values for the scale-invariant measure display more abrupt changes between nodes, especially between salient and non-salient nodes. This is clearly evident in Figure C.5.

For the scale-invariant saliency measure, the non salient values are more consistent than the original measure, in fact we could have used a saliency tolerance $\gamma = 0.6$ for the scale-invariant measure and we would have still been able to segment the image. For the original measure, there is no clear saliency tolerance threshold by simply observing the plots in Figures C.3 to C.7. Although we were able to eventually find a tolerance that worked, it was not a trivial task. With some prior knowledge of the range in values that the saliency measure takes on, experimentation becomes much easier and determining a sufficient saliency tolerance parameter γ is simplified. This is just one experiment of many that were conducted to observe the behaviour of the saliency measures, and for each

instance the same behaviour was exhibited.

References

[1] Commission internationale de l'eclairage. `http://www.cie.co.at/`. 17, 73

[2] Geoeye. `http://www.geoeye.com/`. 52, 53

[3] Giovanni Adorni, Alberto Broggi, Gianni Conte, V. D'Andrea, and Claudio Sansoè. High-level and low-level computer vision: Towards an integrated approach. In *AI*IA: Proceedings of the 2nd Congress of the Italian Association for Artificial Intelligence on Trends in Artificial Intelligence*, pages 322–331, 1991. 2

[4] D. Ballard and C. Brown. *Computer Vision*. Prentice-Hall, 1982. 1, 2, 5, 7, 8

[5] Asker Bazf, Asker M. Bazen, and Sabih H. Gerez. Segmentation of fingerprint images. In *ProRISC 2001 Workshop on Circuits, Systems and Signal Processing*, pages 276–280, 2001. 3

[6] Maged K. Boulos and Larry Robinson. Web GIS in practice vii: stereoscopic 3-d solutions for online maps and virtual globes. *International Journal of Health Geographics*, 8(1):59, 2009. 51

[7] Achi Brandt. Algebraic multigrid theory: The symmetric case. *Appl. Math. Comput.*, 19(1-4):23–56, 1986. 18

[8] William L. Briggs, Van Emden Henson, and Steve F. McCormick. *A multigrid tutorial (2nd ed.)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000. 12, 18, 19, 20, 22

[9] J. J. Corso, E. Sharon, S. Dube, S. El-Saden, U. Sinha, and A. Yuille. Efficient Multilevel Brain Tumor Segmentation with Integrated Bayesian Model Classification. *IEEE Transactions on Medical Imaging*, 27(5):629–640, 2008. 3

[10] Agns Desolneux, Lionel Moisan, and Jean-Michel Morel. *From Gestalt Theory to Image Analysis: A Probabilistic Approach*. Springer, 2007. 7

[11] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000. 30

[12] Perry Durning, Stephen R. Polis, Eric G. Frost, and John V. Kaiser. Integrated use of remote sensing and gis for mineral exploration. Technical report, La Cuesta International, Inc., SDSU, NASA, 1998. 51

[13] Robert D. Falgout. An introduction to algebraic multigrid. *Computing in Science and Engg.*, 8(6):24–33, 2006. 18, 19, 20

[14] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, 2004. 3

[15] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002. 3, 8, 17, 52, 73

[16] Meirav Galun, Eitan Sharon, Ronen Basri, and Achi Brandt. Texture segmentation by multiscale aggregation of filter responses and shape elements. *IEEE International Conference on Computer Vision*, 1:716, 2003. 8, 12, 22, 24, 26, 30

[17] Ya'ara Goldschmidt. *Fast Multiscale Methodsfor data mining*. PhD thesis, Weizmann Institute of science, 2007. 8

[18] Dahai Guo, Arthur Weeks, Harold Klee, and Xuedong Yan. A framework for high-resolution geo-specific road database creation based on image processing techniques for driving simulation. *Adv. Eng. Softw.*, 40(9):820–829, 2009. 52

[19] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1992. 8, 27, 29

[20] Ramesh Jain, Rangachar Kasturi, and Brian G. Schunck. *Machine vision*. McGraw-Hill, Inc., New York, NY, USA, 1995. 8, 9, 10, 11

[21] Jean-Michel Jolion and Azriel Rosenfeld. *A Pyramid Framework for Early Vision: Multiresolutional Computer Vision*. Kluwer Academic Publishers, Norwell, MA, USA, 1994. 4, 5

[22] F.P. Kressler, M. Franzen, and K. Steinnocher. Segmentation based classification of aerial images and its potential to support the update of existing land use data bases. In *International Society for Photogrammetry and Remote Sensing (ISPRS) Hannover Workshop*, 2005. 52

[23] Dan Kushnir, Meirav Galun, and Achi Brandt. Fast multiscale clustering and manifold identification. *Pattern Recogn.*, 39(10):1876–1891, 2006. 8

[24] Jitendra Malik, Jianbo Shi, Serge Belongie, and Thomas K. Leung. Grouping in the normalized cut framework. In *Shape, Contour and Grouping in Computer Vision*, pages 155–164, 1999. 11, 12

[25] David Marr. *Vision*. W.H. Freeman and Company, 1982. 2

[26] Ian Overington. *Computer Vision: A Unified, Biologically-Inspired Approach*. Elsevier Science Inc., New York, NY, USA, 1992. 5

[27] Nikhil R. Pal and Sankar K. Pal. A review on image segmentation techniques. *Pattern Recognition*, 26(9):1277–1294, 1993. 8

[28] J. R. Parker. *Practical computer vision using C*. John Wiley & Sons, Inc., New York, NY, USA, 1993. 10, 11

[29] Dzung L. Pham, Chenyang Xu, and Jerry L. Prince. Current methods in medical image segmentation. *Annual Review of Biomedical Engineering*, 2:315–337, 2000. 3

[30] Erik Reinhard, Erum Arif Khan, Ahmet Oguz Akyz, and Garrett M. Johnson. *Color Imaging: Fundamentals and Applications*. A. K. Peters, Ltd., Natick, MA, USA, 2008. 74, 75

[31] John C. Russ. *Image Processing Handbook, Second Edition*. CRC Press, Inc., Boca Raton, FL, USA, 1995. 9

[32] Eitan Sharon, Achi Brandt, and Ronen Basri. Fast multiscale image segmentation. In *Computer Vision and Pattern Recognition*, pages 1070–1077, 2000. 8, 12, 22, 24, 26

[33] Eitan Sharon, Achi Brandt, and Ronen Basri. Segmentation and boundary detection using multiscale intensity measurements. In *Computer Vision and Pattern Recognition. Volume I.*, pages 469–476, 2001. 3, 4, 8, 22, 24, 26, 28, 29

[34] Eitan Sharon, Meirav Galun, Dahlia Sharon, Ronen Basri, and Achi Brandt. Hierarchy and adaptivity in segmenting visual scenes. *Nature*, 442(7104):810–813, August 2006. ix, 8, 13, 15

[35] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. 11, 12, 18, 23

[36] K. Sobottka and I. Pitas. Looking for faces and facial features in color images. In *PRIA: Advances in Mathematical Theory and Applications*, 1996. 3

[37] Shutong Tse, Laura Bradbury, Justin W.L. Wan, Haig Djambazian, Robert Sladek, and Thomas Hudson. A combined watershed and level set method for segmentation of brightfield cell images. In *Proceedings of SPIE Symposium on Medical Imaging: Image Processing*, 2009. 10

[38] Shimon Ullman. *High-Level Vision: Object Recognition and Visual Cognition*. The MIT Press, 1996. 2

[39] Steven H. Ullman. *Visual Perception: A clinical orientation*. Appleton and Lange, 1994. 17, 52

[40] Scott E. Umbaugh. *Computer Vision and Image Processing: A Practical Approach Using CVIPtools*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1997. 1

[41] Deborah Walters. Computer vision analysis of boundary images. In K.N Leibovic, editor, *Science of vision*. 1990. 1, 2

[42] Yiting Wang, Xinliang Li, Liqiang Zhang, and Wuming Zhang. Automatic road extraction of urban area from high spatial resolution remotely sensed imagery. In *International Society for Photogrammetry and Remote Sensing (ISPRS) Congress Beijing*, 2008. 52, 61

[43] E. Sharon R. Basri A. Brandt Y. Goldschmidt, M. Galun. Fast multilevel clustering. Technical report mcs05-09, Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science, 2005. 8

[44] Xiaolong Zheng, Yangsheng Wang, and Xuying Zhao. Fingerprint image segmentation using active contour model. In *ICIG '07: Proceedings of the Fourth International Conference on Image and Graphics*, pages 437–441, 2007. 3