BOSTON UNIVERSITY
GRADUATE SCHOOL OF ARTS AND SCIENCES

Dissertation

SPACE-VARIANT COMPUTER VISION: A GRAPH-THEORETIC APPROACH

by

LEO JOHN GRADY
B.S., University of Vermont, 1999

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
2004

© Copyright by
LEO JOHN GRADY
2003

Approved by

First Reader

Eric L. Schwartz, PhD
Professor of Cognitive and Neural Systems, Electrical and Computer Engineering, and Anatomy and Neurobiology

Second Reader

Michael Cohen, PhD
Associate Professor of Cognitive and Neural Systems and Computer Science

Third Reader

Ennio Mingolla, PhD
Professor of Cognitive and Neural Systems and Psychology

Acknowledgments

I'd like to thank a lot of people for making all this happen. I want to thank Eric for letting me pursue my interests and giving me guidance, Amy for believing in me, Susan for her advice and support, Jon for hours and hours and hours of discussion, the other Schwartz lab Kommandos for their enthusiasm and wisdom, Jay for commiseration, the rest of CNS for creating a great environment and, of course, my mother and father for their encouragement along this path.

SPACE-VARIANT COMPUTER VISION: A GRAPH-THEORETIC APPROACH

(Order No.)

LEO JOHN GRADY

Boston University Graduate School of Arts and Sciences, 2004

Major Professor: Eric L. Schwartz, Professor of Cognitive and Neural Systems,
Electrical, Computer, and Systems Engineering,
and Anatomy and Neurobiology

Abstract

Space-variant sampling of visual input is ubiquitous in the higher vertebrate brain, because a large input space may be processed with high peak precision without requiring an unacceptably large brain mass. Space-variant sampling has been studied in computer vision for decades. A major obstacle to exploiting this architecture in machines, and understanding its role in biology, is the lack of algorithms that generalize beyond regular samplings. Most image processing algorithms implicitly assume a Cartesian grid underlying the sensor. This thesis generalizes image processing to a sensor architecture described by an arbitrary graph. This data structure separates the sensor topology, expressed by the graph edge structure, from its geometry, represented by coordinates of the vertex set.

The combinatorial Laplacian of the sensor graph is a key operator underlying a series of novel image processing algorithms. First, a new graph partitioning algorithm for segmentation is presented that heuristically minimizes the ratio of the perimeter of the partition border and the area of the partitions, under a suitable definition of graph-theoretic area. This approach produces high quality image segmentations.

Interpolation of missing data on graphs is developed, using a combinatorial version of the Dirichlet Problem, i.e., minimizing the average gradients of the interpolated values while maintaining fixed boundary conditions. This leads to the solution of the Laplace Equation, which represents the steady-state of the diffusion process for stated boundary conditions. Results compare favorably to both isotropic and anisotropic diffusion for filling-in of missing data.

A pyramid graph is defined by connecting vertical and horizontal levels of the Laplacian pyramid data structure. The isoperimetric algorithm, run on the graph pyramid, yields an improved segmentation at little extra computational cost. Finally, a small-world graph topology is employed by randomly introducing a few new edges to the image graph. This results in a large speed-up in computation time, with identical final results.

The algorithms developed in this thesis do not require that the data associated with the graph are embedded in two-dimensions or even have a metric structure. Therefore, this approach to generalized image processing may find wider application in other areas of discrete data processing.

Contents

1 Introduction **1**

2 Graph-based machine vision **8**

2.1 Introduction 8

2.2 Mathematical background 9

2.2.1 Adjacency matrix 14

2.2.2 Indicator vectors 15

2.3 Implementation 16

2.4 Data structures 17

2.4.1 Topological information 17

2.4.2 Nodal information 17

2.4.3 Structs 18

2.5 Generating graphs 18

2.5.1 Biological datasets 19

2.6 Importing images 21

2.7 Visualization 24

2.8 Processing 29

2.8.1 Interpolation 30

2.8.2 Filtering 32

2.8.3 Edge finding 38

2.8.4 Segmentation 40

2.9 Miscellaneous functions 40

2.10 Conclusion 42

3 Isoperimetric graph partitioning 44

3.1 Introduction 44

3.2 Isoperimetric algorithm 45

3.2.1 Circuit analogy 48

3.2.2 Alternative derivation of algorithm 49

3.2.3 Choosing a ground 52

3.2.4 Solving the System of Equations 52

3.2.5 Converting the solution to a partition 53

3.2.6 Summary 54

3.2.7 Time Complexity 54

3.3 Some formal properties of the algorithm 54

3.3.1 Connectivity 54

3.3.2 Spanning trees 55

3.3.3 Fully connected graphs 56

3.4 Review of previous work 56

3.4.1 Spectral Partitioning 56

3.4.2 Geometric Partitioning 58

3.4.3 Geometric-Spectral Partitioning 59

3.4.4 Inertial Partitioning 59

3.4.5 Coordinate Partitioning 60

3.5 Comparison of algorithms 60

3.5.1 Planar graphs 61

3.5.2 Nonplanar graphs 61

3.5.3 Three-dimensional graphs 62

3.5.4 Special graphs 64

3.6 Conclusion 67

4 Isoperimetric image segmentation 70

4.1 Introduction 70

4.1.1 Motivation for using graph-theoretic approaches in image processing 70
4.2 Modified isoperimetric algorithm 72
4.2.1 Algorithmic details 74
4.3 Relationship to other graph partitioning methods 76
4.3.1 Isoperimetric 79
4.3.2 Spectral 79
4.3.3 Normalized Cuts 80
4.3.4 Sensitivity to a weight 81
4.4 Applications 81
4.4.1 Clustering applied to examples used by Zahn 81
4.4.2 Methods of image segmentation 83
4.4.3 Completion 84
4.4.4 Segmentation of natural images 85
4.4.5 Stability 87
4.5 Conclusion 92

5 Small world 93

5.1 Introduction 93
5.2 Convergence of Krylov subspace methods 94
5.2.1 Small worlds 95
5.3 Results 97
5.4 Conclusion 99

6 Pyramid-based image segmentation 100

6.1 Introduction 100
6.2 Connected Pyramid Architecture 101
6.3 Results 103
6.3.1 Speed 103
6.3.2 Segmentation quality 103

6.4 Conclusion 106

7 Graph interpolation **108**

7.1 Introduction 108

7.2 Dirichlet problem 109

7.2.1 Definitions 109

7.2.2 Interpolation 110

7.2.3 Combinatorial formulation: Differential operators on graphs 111

7.3 Results 113

7.3.1 Space-variant (foveal) images 113

7.3.2 Anisotropic interpolation 114

7.4 Conclusion 117

8 Conclusion **120**

8.1 Conclusion 120

8.1.1 Further directions in biological mimicry 120

8.1.2 Further directions in analysis 121

A Appendices **123**

A.1 Function list 123

A.2 Demo scripts 125

A.3 Standardized variable names 127

A.4 Included retinal topographies 129

Curriculum Vitae **143**

List of Tables

2.1 Analogy between vector calculus and combinatorial operators 13
2.2 Analogy between vector calculus and combinatorial equations 13

3.1 Comparison of planar graph partitions using median cut 61
3.2 Comparison of planar graph partitions using ratio cut 62
3.3 Comparison of random graph partitions using median cut 62
3.4 Comparison of random graph partitions using ratio cut 63
3.5 Comparison of 3D graph partitions using median cut 63
3.6 Comparison of 3D graph partitions using ratio cut 64
3.7 Specifications of special graphs 65
3.8 Comparison of special graph partitions using median cut 65
3.9 Comparison of special graph partitions using ratio cut 65
3.10 Comparison of difficult graph partitions using median cut 66
3.11 Comparison of difficult graph partitions using ratio cut 66

List of Figures

1·1 Example of retinal topographies 3

2·1 Locally connected graph topologies 20

2·2 Conversion of retinal topography to a graph 22

2·3 Illustration of modified Elliptical Weighted Average filters 24

2·4 Image importing 25

2·5 Multiple foveations 26

2·6 Visualization of a space-variant image 28

2·7 Interpolation for graph drawing 31

2·8 Space-variant image filtering 34

2·9 Coordinate filtering 35

2·10 Diffusion for image filtering 36

2·11 Interpolation as image filtering 37

2·12 Space-variant edge detection 39

2·13 Space-variant segmentation 41

3·1 Graph and equivalent circuit 48

3·2 Comparison of partitioning on a “roach” graph 67

4·1 Dumbbell graph 75

4·2 Effect of a changing ground point 77

4·3 Gestalt point clustering example 82

4·4 Parameter sweep image 84

4·5 Parameter sweep of the segmentation algorithm 85

4·6 Ground point as an “attention” operator 86

4·7 Segmentation of natural images 88
4·8 Quantitative stability of image segmentation with respect to noise 90
4·9 Qualitative stability of image segmentation with respect to noise 91

5·1 Small world graphs 96
5·2 Convergence of conjugate gradients as additional random edges are added . 97
5·3 Effect of small world topology on image segmentation 98

6·1 Topology of a connected pyramid 102
6·2 Effect of additional pyramid layers on conjugate gradient convergence . . . 104
6·3 Comparison of pyramid and lattice based segmentation to blurred images . 105
6·4 Effects of pyramid architecture on segmentation results 106

7·1 Equivalent circuit for combinatorial interpolation 113
7·2 Isotropic space-variant interpolation 115
7·3 Anisotropic space-variant interpolation 116
7·4 Anisotropic interpolation of an image from a very sparse sampling 118
7·5 Anisotropic interpolation of an image based on spatially varying samples . . 119

List of Abbreviations

2D	2-Dimensional
3D	3-Dimensional
ARPACK	Arnoldi Package
B&W	Black and White
CRT	Cathode Ray Tube
ECT	Exponential Chirp Transform
FTP	File Transfer Protocol
GHz	Gigahertz
Hz	Hertz
K	Kilobytes
MATLAB	Matrix Laboratory
MEX	MATLAB Executable
MG	Maximum degree Ground
MnG	Minimum degree Ground
Ncuts	Normalized Cuts
PDF	Probability Density Function
RAM	Random Access Memory
RG	Random Ground
RGB	Red Green Blue
VLSI	Very Large Scale Integration

Chapter 1

Introduction

For various reasons, many researchers have in the past been interested in freeing themselves of the constraints of a uniformly sampled, Cartesian data representation for images. Motivations include a dependence on nonuniform sensors, feature extraction, data reduction, or a desire to process even in the absence of some samples (or regions). Another group of researchers have been interested in modeling biological sensory systems or applying the architecture to computer vision tasks.

Space-variant sampling of visual space is ubiquitous in the higher vertebrate visual system (Hughes, 1977; Schwartz, 1994), as seen in the comparison of retinal ganglion densities of different species shown in Figure 1·1. In computer vision, this architecture is of interest because it facilitates real-time vision applications due to a large (albeit lossy) reduction in space-complexity (Rojer and Schwartz, 1990), and because it represents a prototype for adaptive sampling in a more general setting. In a biological context, primate visual sampling has been demonstrated to be strongly space variant (Schwartz, 1977), possessing a single high resolution area (i.e., the fovea) with resolution falling off linearly toward the periphery. Many non-primate species possess an even more exotic visual architecture. Several bird species have multiple foveas (Collin, 1999), and elephants have a magnified representation in the region of their trunk to facilitate "eye-trunk" coordination (Stone and Halasz, 1989). Computer vision systems in which the architecture and spatial sampling is adaptively tailored to the specific problem domain may well follow this design path. Thus, it is of importance to develop a universal approach to visual representation which is not implicitly dependent on a regular Cartesian grid. Representations of image data on graph-theoretic structures provide one such route to a universal sampling and topology for visual sensing, since it separates the topological (via connectivity) from the geometric (via sampling arrangement of visual space) aspects of the sensor.

The main reason for employing a space-variant architecture is to process with dramatically lower bandwidth while retaining a high resolution in part of the visual scene. However, the difference in visual sampling across species suggests that there is a relationship between features of the sampling regime and the animal's "visual ecology". An example of such a relationship is the belief that the "horizontal streak" seen in many animals (e.g., the rabbit in Figure 1·1) is helpful to species that live in open (i.e., non-occlusive) visual environments (Hughes, 1977). This belief has been supported by the strong correlation between species with less occlusive visual ecologies and those possessing a horizontal streak. Uncovering relationships of this nature help the engineer of a computer vision system design an architecture that is optimized to match the design constraints for the visual ecology of the artificial system. It is necessary that a data structure exists with sampling-independent (i.e., generalized) computer vision algorithms in order for the designer to be free to craft the visual sampling to the purpose of the system.

Work on the sampling of visual space employed by the upper primate visual system may be divided into two sections:

1. *Properties of the mapping*

   Traditional methods of space variant processing have focused largely on the *mapping* from a regular Cartesian grid to an alternate space (Casasent and Psaltis, 1976; Bonmassar and Schwartz, 1997). Therefore, the primary interest has been on the properties of the transform and not the underlying data structure. In addition to the considerable data reduction, the discretized log-polar transform of a Cartesian grid has a number of properties that make it interesting to researchers in computer vision. Casasent and Psaltis (1976) discovered that objects in the fovea which are scaled, rotated or translated have a roughly invariant spectral signature. Their method is to employ the log-polar transform of an image, apply the discrete Fourier transform,

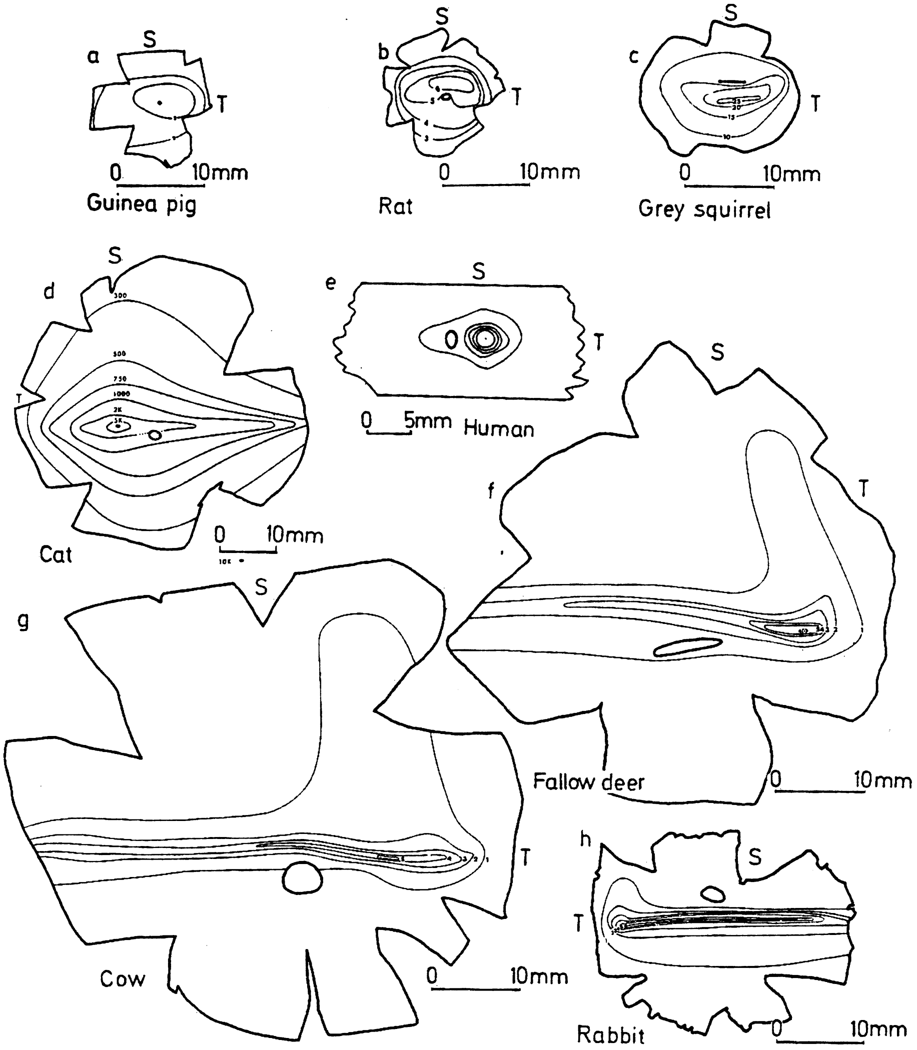


Figure 1·1: Isodensity lines for retinal ganglion cell distribution. Reprinted from Hughes (1977), with permission.

and find the log-polar transform of the spectrum (i.e., a Mellin transform). This invariance may be used for object recognition (see Lin and Wu (2001) for a recent application of this idea). The drawback to this idea is that the properties of scale and rotation invariance only apply if the fovea is directed to the center of the object. Furthermore, a translated object against a static background will not have the same spectrum as a translated object against a translated background, which means that the utility of this approach is limited to simple cases of an object moving against a uniform background (or uniformly translated background). Another approach to exploiting the structure of the log-polar mapping is that of Bonmassar and Schwartz (1997), who developed a specialized Fourier transform called the "exponential chirp transform" (ECT). The ECT finds the Fourier transform of an image in log-polar format and uses the structure of the logarithmic function to obtain an even faster algorithm than what would be obtained simply by the data reduction of an ordinary log-polar transform.

2. *Image Processing in a Space-Variant Domain*

The notion of a "connectivity graph" was introduced by Wallace et al. (1994) to allow for image processing on a foveal sensor. This notion is introduced specifically to model the sampling of the macaque retina (Schwartz, 1977). However, standard computer vision tasks (e.g., edge finding, filtering) were not developed for this structure. Neither visualization of the connectivity graph nor how one simulates a space-variant sensor, given access to an acquisition device based on a Cartesian sensor array, is addressed. Furthermore, the generalization of the connectivity graph to other biological sampling schemes was not discussed. More recent approaches to space-variant vision appear to have abandoned this idea (e.g., Lin and Wu, 2001). Chen (2001) used a mesh-based representation to perform image processing. His data structure does not allow for an arbitrary topology (i.e., the graph must be planar and polygonal). Furthermore, his operators are discretized continuum operators, instead of

> combinatorial operators (see Section 2.2 for a discussion). Significantly, Chen proposed the use of a method from computer graphics (Heckbert, 1989) for *resampling* an image to allow the determination of pixel values for an arbitrary sampling based on a given image with Cartesian sampling. Unfortunately, this method only applies to resamplings for which the Jacobian is known. For the nearly log-polar mapping known to exist in macaque (Schwartz, 1977), this resampling works well. However, the method for resampling does not apply for most species, since the distribution of ganglion cell density has not been formally characterized. Chapter 2 details a set of software tools that provides algorithms and data structures intended to facilitate computer vision on arbitrary visual sampling arrangements, even if a space-variant sensor is unavailable.

Since Zahn's classic paper (Zahn, 1971), graph processing algorithms have become increasingly popular in the context of computer vision (e.g., Wu and Leahy, 1993; Perona and Freeman, 1998; Shi and Malik, 2000; Sarkar and Soundararajan, 2000; Wang and Siskund, 2003). Typically, pixels are associated with the nodes of a graph and edges are derived from a 4- or 8-connected lattice topology. Some authors have also chosen to associate higher level features with nodes (Sarkar and Soundararajan, 2000; Perona and Freeman, 1998). For purposes of importing images to space-variant architectures, we adopt the conventional view that each node corresponds to a pixel.

Graph-theoretic algorithms often translate naturally to the proposed space-variant architecture. Unfortunately, algorithms that employ convolution (or correlation) implicitly assume a shift-invariant topology. Although shift-invariance may be a characteristic of the topology for a locally connected lattice, a locally connected space-variant sensor array (e.g., obtained by connecting to K-nearest-neighbors) will typically result in a shift-*variant* topology. Therefore, a reconstruction of computer vision algorithms for space-variant architectures requires the use of additional theory to generalize these algorithms.

Data acquired from sensors may be viewed as samples of an exterior, continuous world, about which conclusions must be drawn from the limited information given by the samples.

An alternate approach is to view the sensor data itself as the object about which conclusions must be drawn. For reasons that will become clear in Section 2.2, the former view of sensor data will be referred to as the **sampling paradigm** and the second as the **combinatorial paradigm**.

The difference between these paradigms may appear to be purely academic, since the primary output of many computer vision tasks (e.g., face detection) make no comment on whether the result pertains to the pixels or the "real-world". However, some algorithms do operate under an implicit paradigm. For example, shape analysis (Loncaric, 1998; Zhang et al., 1999) typically adopts the approach of the sampling paradigm, while morphological analysis (Soille, 1999) employs the combinatorial paradigm. One practical difference between these two approaches is that algorithms developed to output statements about the continuous world should *improve performance* with increasing samples, while algorithms developed to output statements about the pixels should *decrease performance* due to the increased processing required. The difference between these two approaches is amplified when the sensor arrangement is space-variant, since the sampling theory is not as well developed for nonuniform samples (Unser, 2000), despite the fact that some authors have adopted the sampling viewpoint (Bonmassar and Schwartz, 1997). Furthermore, since a typical motivation for employing a space-variant architecture is the ability to employ a small number of pixels (while maintaining a high peak resolution), algorithms developed in the sampling paradigm are expected to decrease accuracy while combinatorial algorithms are expected to significantly increase in speed.

The approach taken in this thesis to the combinatorial paradigm for space-variant computer vision is through combinatorial analogs of vector calculus, since operators such as the gradient (Roberts, 1965) and Laplacian (Marr and Hildreth, 1980) play such a prominent role in computer vision. Before developing new combinatorial space-variant algorithms, the next chapter begins by reviewing the mathematical analogies between operators from vector calculus and combinatorial operators. The remainder of the chapter is devoted to the implementation of a $^{\text{TM}}$MATLAB toolbox intended to provide a software environment for combinatorial computer vision. This toolbox contains functions to execute the original algorithms developed in later chapters, combinatorial algorithms which were developed by other researchers, and standard vector calculus-based computer vision algorithms that were translated to combinatorial algorithms. The toolbox is publicly available at `http://eslab.bu.edu/software/graphanalysis`, along with scripts that generate all of the figures and tables in this thesis. Chapter 3 introduces an original graph partitioning algorithm, called **isoperimetric partitioning**, and compares it to other state-of-the-art algorithms. Chapter 4 applies a modified, recursive version of the isoperimetric algorithm to problems in unsupervised data clustering and image segmentation. Chapter 5 proposes the use of a "small world" topology in order to increase the speed of algorithms that make use of iterative Krylov subspace methods (e.g., conjugate gradients), and demonstrates the effect that this topology has on the segmentations obtained with the isoperimetric algorithm. Chapter 6 introduces the use of an image pyramid considered as a single graph, to which the isoperimetric algorithm is applied. Despite the additional nodes used in a pyramid structure, the topology of the pyramid significantly mitigates the additional cost when iterative Krylov subspace methods are used in the image analysis. Finally, in Chapter 7, the question of how to interpolate data on a graph is addressed, and applications to low-level computer vision are developed.

Chapter 2

Graph-based machine vision

2.1 Introduction

In order to perform computer vision in the graph-theoretic, combinatorial, space-variant paradigm outlined in the last chapter, a more generalized notion of standard vector calculus operators must be generated, and a software environment must be developed. The purpose of this chapter is to establish notation, review the analogies between vector calculus and combinatorial methods, and outline the software, called the Graph Analysis Toolbox, we have established for performing graph-theoretic computer vision.

The Graph Analysis Toolbox follows in the tradition of Wallace et al. (1994) and Chen (2001) by providing methods to perform computer vision on graph-based architectures. However, since data processing on graphs appears in other fields such as computer graphics (Taubin, 1995), 3D surface flattening (Wandell et al., 2000) and data clustering (Jain et al., 1999), we hope that this toolbox will find a larger audience. Specifically, there are three problems that we hope to address with this toolbox:

1. *Importing images*

   Since space-variant sensors (Sandini et al., 2000, 1989) are rare, tools must be developed for transferring image data acquired with a standard, Cartesian sensor array to a desired space-variant sampling arrangement. We use the phrase **importing an image** to refer to the process of transferring image data from a Cartesian sensor array to a space-variant arrangement. Our strategy is to extend the resampling work of Heckbert (1989) by removing the requirement that the resampling is performed by a known differentiable function.

2. *Visualization*

   Additional tools are required in order to visualize space-variant images on a standard raster CRT display. A Voronoi cell and interpolation method are provided to allow visualization of image data on graphs.

3. *Processing*

   At the core of the Graph Analysis Toolbox are methods for performing image processing on data associated with each node in a graph. Some of the methods represent original work, while others are collected from the literature. The theory underpinning much of this work was developed by Roth (1955), Branin (1966) and classic research in circuit theory (Kirchhoff, 1847; Weyl, 1923).

First we will introduce the basic mathematics and notation used in this thesis for developing and applying combinatorial algorithms. A discussion of the data structures and implementational approach to these issues will then be addressed. The remainder of this chapter describes and demonstrates the various tools for importing, visualizing and processing data on graphs.

2.2 Mathematical background

Most of the early work on the algebraic properties of graphs was done in the context of linear circuit theory. This section is essentially a short review of Branin's exposition on the algebraic-topological basis for analogy between graphs and vector calculus (Branin, 1966).

A **graph** is a pair $G = (V, E)$ with vertices (nodes) $v \in V$ and edges $e \in E \subseteq V \times V$. An edge, e, spanning two vertices, v_i and v_j, is denoted by e_{ij}. Let $n = |V|$ and $m = |E|$ where $|\cdot|$ denotes cardinality. A **weighted graph** has a value (typically nonnegative and real) assigned to each edge called a **weight**. The weight of edge e_{ij}, is denoted by $w(e_{ij})$ or w_{ij}. Since weighted graphs are more general than unweighted graphs (i.e., $w(e_{ij}) = 1$ for all $e_{ij} \in E$ in the unweighted case), we will develop all our results for weighted graphs.

Define the **degree** of a vertex v_i, denoted d_i, as

$$d_i = \sum_{e_{ij}} w(e_{ij}) \ \ \forall \ e_{ij} \in E. \tag{2.1}$$

A graph may be defined from a linear electrical circuit by identifying the wire between circuit components with the node set, and the components bridging nodes (i.e., branches) as the edge set with weights equal to the admittance of each component (or the conductance, in the case of resistors) (Branin, 1966). In this way, every linear circuit has an equivalent graph and *vice versa.* The explicit connection between circuits, graphs and algebraic topology was made in Roth's fundamental paper (Roth, 1955). Roth showed that Kirchhoff's Current Law corresponds to a homology sequence in topology, while Kirchhoff's Voltage Law corresponds to a cohomology sequence. Roth then proposed Ohm's Law as a bridge between the sequences. Largely adopting the notation of Strang (1986), we may write the fundamental equations of circuit theory in matrix form.

Define the $m \times n$ **edge-node incidence matrix** as

$$A_{e_{ij}v_k} = \begin{cases} +1 & \text{if } i = k, \\ -1 & \text{if } j = k, \\ 0 & \text{otherwise,} \end{cases} \tag{2.2}$$

for every vertex v_k and edge e_{ij}, where e_{ij} has been arbitrarily assigned an orientation. The notation $A_{e_{ij}v_k}$ is used to indicate that the rows of A are indexed by edge e_{ij} and the columns of A are indexed by node v_k.

Define the $m \times m$ **constitutive matrix**, C, as the diagonal matrix with the weights of each edge along the diagonal.

The three main laws governing circuit theory may be written as

$$A^T y = f \qquad \text{(Kirchhoff's Current Law)}, \tag{2.3a}$$

$$Cp = y \qquad \text{(Ohm's Law)}, \tag{2.3b}$$

$$p = Ax \qquad \text{(Kirchhoff's Voltage Law)}, \tag{2.3c}$$

where f represents current sources at the nodes, p is the potential drop (voltage) across a branch, x is the potential at a node and y is the current through a branch.

By viewing the incidence matrix as a linear operator, it may be seen that application of that operator to a set of numbers assigned to each node induces a related set of numbers on the edges. Kirchhoff's Voltage Law is an example of this operation, in which electric potentials at each node are converted to voltages across edges by application of the incidence matrix. In a similar manner, the application of the operator A^T to a set of numbers on the edge set of a graph induces a related set of values defined on the node set. Kirchhoff's Current Law is an example of this operation, since the application of A^T to the currents through each branch yields the values of the current sources at each node. Application of C may be viewed as bridging the voltages and currents defined on each edge.

When an edge is added to a tree, the unique closed path so formed is called a **loop**. The set of loops, Q, formed by the addition of edges to a tree consists of elements, q_i, such that $q_i \in Q$. Note that $|Q| = |E| - |V| + 1$ (a variation of the Euler formula), since the number of edges in a tree of a connected graph is $|V| - 1$ (Biggs, 1974). Define the **loop-edge incidence matrix**

$$K_{m_k e_{ij}} = \begin{cases} +1 & \text{if } e_{ij} \text{ is crossed positively in a clockwise traversal of } q_k, \\ -1 & \text{if } e_{ij} \text{ is crossed negatively in a clockwise traversal of } q_k, \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

Similar to the edge-node incidence matrix, the application of the loop-edge incidence matrix to a set of numbers defined on the edges returns a related set of values defined on each loop.

Branin (1966) identified the A, A^T and K operators with the familiar gradient, divergence and curl operators from vector calculus. This analogy holds for familiar identities such as $A^T K^T = 0$ (i.e., the divergence of the curl is zero) and allows definition of other operators, such as the **Laplacian**, $L = A^T A$. The generalization of the Laplace operator to the Laplace-Beltrami operator (Warner, 1983) fits well with this analogy, where $L = A^T C A$, with the matrix C representing the metric information. As a matrix, the Laplacian may be derived directly from knowledge of V and E by letting

$$L_{v_i v_j} = \begin{cases} d_i & \text{if } i = j, \\ -w(e_{ij}) & \text{if } e_{ij} \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{2.5}$$

The notation $L_{v_i v_j}$ is used to indicate that the matrix L is being indexed by vertices v_i and v_j. This matrix is also known as the *admittance matrix* in circuit theory, and a good review of its properties is given by Merris (1994). More than one representation of the combinatorial Laplacian operator has been developed, depending on the choice of metric and normalization (Dodziuk, 1984; Dodziuk and Kendall, 1986; Mohar, 1988; Chung, 1997). However, unless otherwise noted, the above formulation will be referred to as the Laplacian. A summary of the analogies between operators in vector calculus and graph theory is given in Table 2.1, while additional relationships are listed in Table 2.2.

There is both a conceptual and practical difference between the graph-theoretic analog of a concept from mathematical physics and a discretization of the standard continuum representation of that same concept. Consider solving Poisson's equation (Courant and Hilbert, 1989a) on a continuous domain with a digital computer (e.g., using finite elements). The objective of such a solution would be that the values assumed at any point in the domain could be determined, not just those points used in the calculation. In contrast, solving Poisson's equation on a graph, $Lx = f$, returns values *only for the node set.* Furthermore, the number of nodes and the graph topology (i.e., edge set) directly affects

Operator	Vector calculus	Combinatorial
Gradient	∇	A
Divergence	$\nabla\cdot$	A^T
Curl	$\nabla \times \nabla$	K
Laplacian	$\nabla \cdot \nabla$	$A^T A$
Beltrami	$\nabla C \cdot \nabla$	$A^T C A$

Table 2.1: Correspondence between continuum differential operators and combinatorial differential operators on graphs. C represents a constitutive matrix relating flux to flow, e.g., a conductivity tensor, a diffusion tensor, a thermal conductivity, a stress-strain tensor, or, in the context of differential geometry, a metric tensor. A is the edge-node incidence matrix of the graph representing the topology of the problem and K is the loop-edge incidence matrix of the graph.

Equation	Continuum	Combinatorial
KVL	$\nabla V = E$	$Ax = p$
KCL	$\nabla \cdot J = \frac{d\rho}{dt}$	$A^T y = f$
Ohm's Law	$\sigma^{-1} E = J$	$Cp = y$
Dirichlet Integral	$\frac{1}{2}\int_\Omega \|\nabla u\|^2 d\Omega$	$\frac{1}{2} x^T A^T C A x$

Table 2.2: Correspondence between continuum differential equations and combinatorial differential equations on graphs. Kirchhoff's current law is a quasi-static ($\frac{\partial B}{\partial t} = 0$) approximation to Maxwell's Equation $\nabla \times E = \frac{\partial B}{\partial t}$. Kirchhoff's voltage law follows from the definition of electric field as the gradient of potential. Ohm's Law is a constitutive (phenomenological) law asserting a presumed linear dependence between voltage and current.

the solution to Poisson's equation. In contrast, a major design goal of a discretization procedure is that the solution for points in the domain are *invariant* to changes in the meshing.

Conceptually, one can understand the difference between solving for the charge distribution at the nodes of a planar electrical circuit given initially charged capacitors (i.e., the combinatorial diffusion equation of Perona and Malik (1990)), and solving for the values taken at discrete samples of a planar conductive material with an initial heat distribution (i.e., the discretized continuous diffusion equation). Therefore, the solution of a continuous problem by use of a digital computer is referred to here as a **discrete** approach, while the solution of a problem using the graph-theoretic analogies shown above is referred to as a **combinatorial** approach.

All functions in this toolbox operate under the graph-theoretic, combinatorial paradigm. By this we mean that the operators we are concerned with are represented by matrices and the quantities of interest are defined by vectors associating values to nodes, edges, or meshes. Typically, the values associated with nodes are image values (e.g., grayscale, RGB color channels) or coordinate values, while those associated with edges and loops are dependent on the nodal values (e.g., the result of applying the gradient operator).

2.2.1 Adjacency matrix

Another fundamental matrix in graph theory is the $n \times n$ adjacency matrix defined as

$$W_{v_i v_j} = \begin{cases} w(e_{ij}) & \text{if } e_{ij} \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

The adjacency matrix has also appeared in applications (e.g., Perona and Freeman (1998)), and its spectral properties have been thoroughly analyzed (Cvetković et al., 1995). By noting that the adjacency matrix, Laplacian matrix and edge-node incidence matrix all completely specify the topology of a graph, it is not surprising that these matrices are related. Specifically, it is well known (Biggs, 1974) that $D - W = L = A^T C A$, where D is the diagonal matrix with $D_{ii} = d_i$.

2.2.2 Indicator vectors

Another important type of value associated with nodes, edges and loops is an **indicator vector**. Indicator vectors are used to indicate membership of a node, edge or loop in a set. A common goal (Pothen et al., 1990; Shi and Malik, 2000) is to determine which components of a graph belong to a set (e.g., which pixels belong to a segment). In this scenario, the indicator vector represents the solution. Another important use of an indicator vector is to perform set operations (e.g., union, intersection) or determine properties of the set (e.g., cardinality, boundary nodes). We develop this section in the context of a nodal indicator vector. Indicator vectors on the set of edges or loops follow an identical development. For a set of nodes, $S \subset V$ an indicator column vector, x, may be defined as

$$x_i = \begin{cases} 0 & \text{if } v_i \notin S, \\ 1 & \text{if } v_i \in S. \end{cases} \tag{2.7}$$

For two sets, S_1 and S_2, with corresponding indicator vectors x_1 and x_2, the usual set operations may be performed with

$$|S_1| = x_1^T x_1 \qquad \text{(Set cardinality)}, \tag{2.8}$$

$$S_1 \cap S_2 = x_1 \wedge x_2 \qquad \text{(Set intersection)}, \tag{2.9}$$

$$S_1 \cup S_2 = x_1 \vee x_2 \qquad \text{(Set union)}, \tag{2.10}$$

where $\wedge$ and $\vee$ denote the logical (binary) "and", "or" operations.

The matrices W and L may be used to determine useful properties of a vertex set, S, through operations with its indicator vector, x, in the following manner

$$\frac{1}{2} x^T W x = \sum_{e_{ij}, v_i \in S, v_j \in S} w(e_{ij}) \qquad \text{(Sum of the weights internal to } S\text{)}, \tag{2.11}$$

$$x^T L x = \sum_{e_{ij}, v_i \in S, v_j \in \overline{S}} w(e_{ij}) \qquad \text{(Sum of the weights on the boundary of } S\text{)}, \tag{2.12}$$

where $\overline{S}$ indicates the set complement of S.

2.3 Implementation

We chose to implement the toolbox in $^{\text{TM}}$MATLAB for several reasons:

1. *Numerical linear algebra* is at the core of the combinatorial approach to space-variant vision outlined above. Since $^{\text{TM}}$MATLAB is well equipped with a numerical linear algebra package and a sparse matrix package (Gilbert et al., 1992), $^{\text{TM}}$MATLAB is a natural environment for the toolbox.

2. *Rapid prototyping* of new algorithms is facilitated by the extensive set of tools available in $^{\text{TM}}$MATLAB. Since this toolbox is intended for a research audience, the ability to rapidly prototype new algorithms is essential.

3. *Visualization* of space-variant images associated with graphs is a major design objective of the Graph Analysis Toolbox. $^{\text{TM}}$MATLAB provides an excellent ability to visualize data.

However, $^{\text{TM}}$MATLAB also has several drawbacks:

1. *^{TM}MATLAB is proprietary* and licenses for the standard package and additional toolboxes may cost hundreds to thousands of dollars. This fact limits the accessibility of the Graph Analysis Toolbox.

2. *Speed* of computation in $^{\text{TM}}$MATLAB can be very slow for certain types of operations (e.g., code loops). Although it is possible to use the MEX functionality of $^{\text{TM}}$MATLAB to speed up some of this computation, the portability of the code suffers. Fortunately for the Graph Analysis Toolbox, the numerical linear algebra package in $^{\text{TM}}$MATLAB is relatively fast.

A full listing of the functions in the Graph Analysis Toolbox is given in Appendix A.1 and a list of demos is given in Appendix A.2. For reasons of consistency, ease of readability and agreement with publications, the same variable names were used to refer to the same variables across functions and demos. A listing of standardized variable names used in the Graph Analysis Toolbox is given by Appendix A.3.

2.4 Data structures

There are different ways of representing a graph on a computer (e.g., lists, matrices). The choice of representation is often dependent on the particular application. The guiding principle in defining data structures for the Graph Analysis Toolbox is that functions should exist for switching between different representations and that information which may not be bound together should not be forced together (e.g., in a `struct`). Since $^{\text{TM}}$MATLAB uses a pass-by-value system, this latter principle is especially important. Consequently, there is no all-purpose `struct` that contains all possible information about a graph. A full listing of standardized variable names is given in Appendix A.3.

2.4.1 Topological information

The most fundamental description of a graph is the topology, since none of the matrix representations may be defined without it. The most space efficient representation of the graph topology is given by a list, `edges`, that contains pairs of integers indicating nodes joined by an edge. However, the matrices A, L and W may be generated from the edge list with the functions `incidence.m`, `laplacian.m` and `adjacency.m`, respectively. The function `adjtoedges.m` allows conversion from a adjacency matrix representation of topology to an edge set.

2.4.2 Nodal information

Numbers associated with nodes in the Graph Analysis Toolbox typically have two separate meanings: coordinates and image values. Aside from the semantics, there is no difference in the way in which these values may be represented or processed. However, since one may be interested in keeping notation of these quantities separate, two different N-dimensional lists are kept to refer to these quantities. The list called `points` refers to coordinate values, while the list `vals` typically refers to image values (e.g., RGB, grayscale).

2.4.3 Structs

Two quantities are kept in $^{\text{TM}}$MATLAB `structs`, since their component values are never used separately. The information necessary to perform importing of a graph is kept in a `struct` called `imgGraph`. Voronoi visualization information (see below) is kept in a struct called `voronoiStruct`.

2.5 Generating graphs

When building a graph from a sensor array (or simulated sensor array), it is common to assign the value of each sensor to a node. However, the choice of connectivity (i.e., edge set) is much less clear. Typically, one wants the nodes to be locally connected. Two functions are provided for locally connecting a point set in arbitrary dimensions, `knn.m` and `triangulatepoints.m`. An N-dimensional Delaunay triangulation is implemented by `triangulatepoints.m` and K-nearest-neighbors is implemented by `knn.m`. For 2-dimensional points, `triangulatepoints.m` calls the MEX version of Shewchuck's `triangle.c` (Shewchuk, 1996), if installed. Recent interest in the use of "small-world" networks (Watts and Strogatz, 1998; Watts, 1999; Strogatz, 2001) prompts inclusion of the function `addrandedges.m` to randomly add a specified number of edges to the graph. The effect of using `addrandedges.m` is to dramatically decrease the diameter of the graph (Watts and Strogatz, 1998) while only minimally increasing the number of edges.

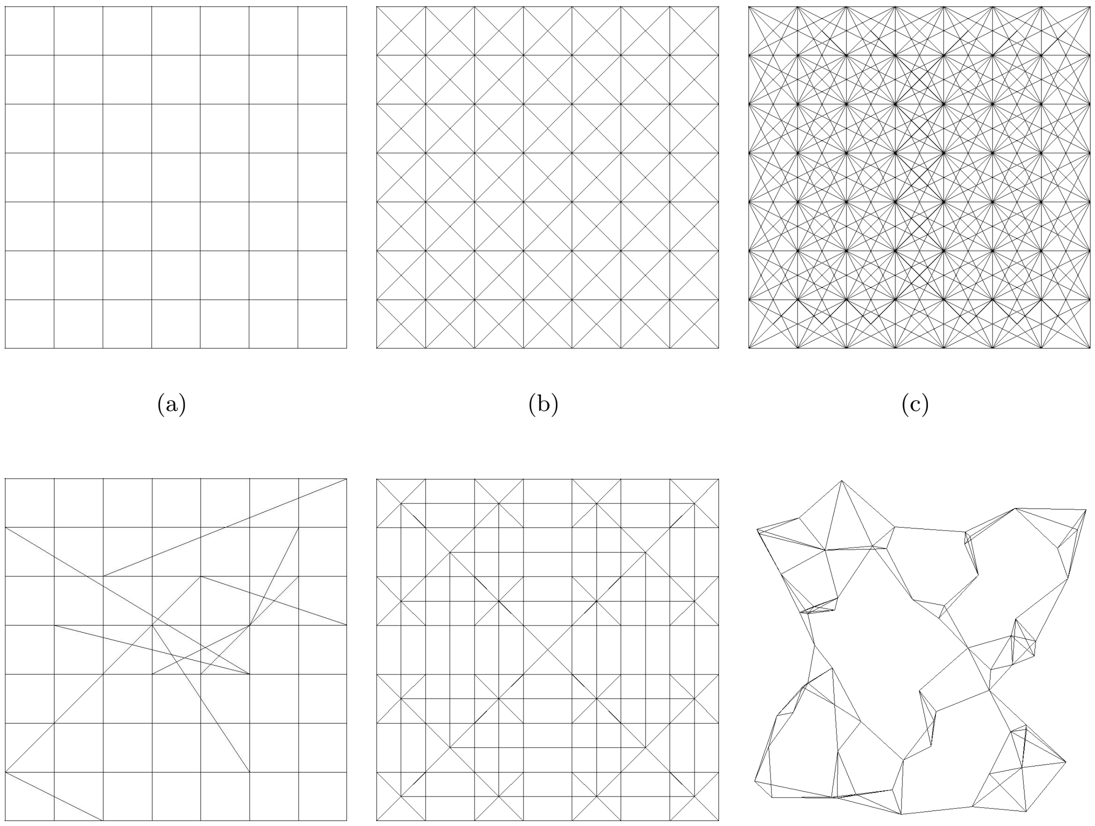
Due to the prominence of the Cartesian lattice in traditional computer vision, the function `lattice.m` generates a lattice with three different topologies: 4-connected, 8-connected or a radially connected topology (as in Shi and Malik (2000)).

The use of a multi-scale image representation to enhance image analysis algorithms has a long history dating back to Burt et al. (1981) and Witkin (1983). Typically (Wright and Acton, 1997; Chen and Acton, 1998; Acton, 1996; Comer and Delp, 1995; Pachai, 1998), a multi-resolution representation is employed both for speed and robustness against noise by performing the analysis at the coarsest level and projecting the solution back to the original image. Multiresolution approaches to general graphs have also been proposed (Barnard and Simon, 1994; Eppstein, 1992). The Graph Analysis Toolbox includes one function `latticepyramid.m` that draws on this literature by returning a pyramid-shaped graph with 3-dimensional coordinates, such that every node at a higher level is connected to four (non-overlapping) nodes on the lower level. Although the pyramid graph is returned as a unit (i.e., a single node and edge set), an index is also returned indicating the level of each node and a list of its four children on the lower level. Some of the graphs described in this section are displayed in Figure 2·1.

Finally, the function `roach.m` generates the "roach" graph of Guattery and Miller (1998) for purposes of testing.

2.5.1 Biological datasets

Included in the extended (demo) release of the Graph Analysis Toolbox is a precomputed (i.e., saved in `.mat` files) set of filters corresponding to the visual sampling associated with 22 different species. Following the retinal diagrams in Hughes (1977) of ganglion cell isodensity lines (see Figure 1·1), topographical maps of the retinal distribution of other species have been published. Specifically, we have included node sets (and filters) corresponding to the visual sampling (as determined by ganglion cell counts) for baboon (Whitteridge, 1965), beagle (Peichl, 1992), bottlenosed dolphin (Mass and Supin, 1995), cat (Hughes, 1975), cheetah (Hughes, 1977), cow (Hughes, 1977), deep-sea bass (Collin and Partridge, 1996), deer (Hughes, 1977), German shepherd (Peichl, 1992), harlequin tusk fish (Collin and Pettigrew, 1988), labrador (Hughes, 1977), pig (Hughes, 1977), pigeon (Whitteridge, 1965), plains kangaroo (Hughes, 1974), rabbit (Hughes, 1971), sacred kingfisher (Moroney and Pettigrew, 1987), tree kangaroo (Hughes, 1974), two-toed sloth (Costa et al., 1989), squirrel (Hughes, 1977), wolf (Peichl, 1992) and yellow-finned trevally (Collin, 1999). A sampling reflecting the macaque retina is also provided by using the retinotopic function



(a) (b) (c)

(d) (e) (f)

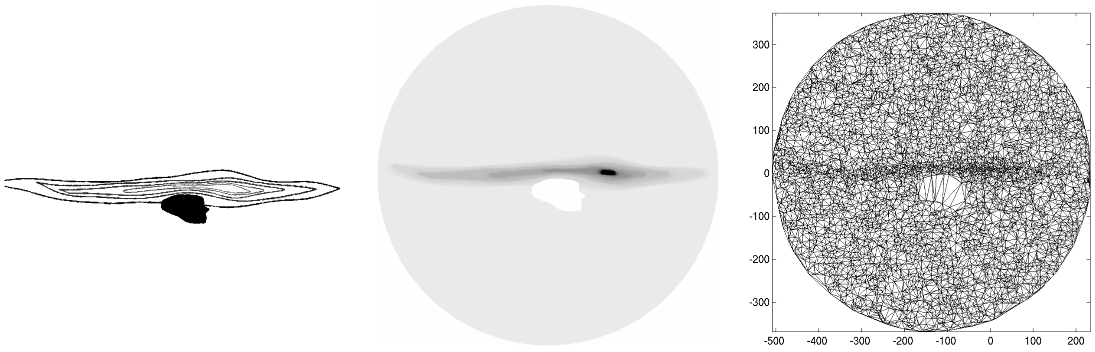
Figure 2·1: Examples of the structured graphs (i.e., node coordinates and edge sets) generated by functions in the Graph Analysis Toolbox. (a) A 4-connected lattice. (b) An 8-connected lattice. (c) A radially connected lattice. (d) A 4-connected "small world" lattice. (e) A flattened 4-connected pyramid lattice. (f) A K-nearest-neighbors graph with a randomly generated set of coordinates.

$w = \log(z + a)$ of Schwartz (1977). Generating sampling points according to this distribution is accomplished by the function `logz.m`.

Graphs were generated from topographic maps by interpolating the contours across the extent of the topographic image, treating this interpolation as a probability density function (PDF), sampling a predetermined number of points and scaling the coordinates of the points to fit the desired input image size. In the case of the precomputed filters included with the Demos package, the desired input image size was 256×256 ($65,536$ pixels) and the number of samples was $6,400$. The function `contour2graph.m` accepts contour images where the background is white, the contours are shades of gray proportional to the values of the isodensity lines and the blind spot (or pecten, in the case of Aves) is colored red and returns an interpolated image (i.e., the PDF) by solving the corresponding Dirichlet problem using the technique of Chapter 7. Sampling points from the PDF is accomplished with the function `pdf2graph.m`. The process of converting a contour to a graph is shown in Figure 2·2. For purposes of comparison, Appendix A.4 illustrates the PDFs of all the species included in the Graph Analysis Toolbox.

2.6 Importing images

The problem of importing an image is to transfer an image taken with a conventional lattice sensor array to a space-variant graph (usually with many fewer nodes than pixels in the original) with minimal aliasing. The approach proposed by Chen (2001) was to treat the problem of importing as a *resampling* problem by calculating appropriate filters (Heckbert, 1989) to apply to a neighborhood of pixels in the original image in order to output a value for each node. The method of Heckbert (1989) for defining filters requires the definition of a differential resampling function that inputs points in the original sampling and outputs points in the new sampling. This formulation was sufficient for Chen (2001), since he was using the $w = \log(z + a)$ mapping given by Schwartz (1977) to model macaque retinotopy. However, we would like to make use of the visual sampling strategies of other species, for which no retinotopic function has been proposed.



(a) (b) (c)

Figure 2·2: Demonstration of conversion from a retinal topography to a graph. The example chosen here is based on the retinal topography of the cheetah (Hughes, 1977). (a) Isodensity contours of the cheetah retinal topography. Darker contours represent a lower ganglion cell density, while lighter contours represent a higher ganglion cell density. The blind spot is shown in black for publication, although the function `contour2graph.m` requires it to be colored red. (b) The interpolated and normalized probability density function determined from the contours (see text for details). Darker areas correspond to areas of greater probability. (c) The graph obtained by sampling 6, 400 nodes from the distribution in (b) and connecting with a Delaunay triangulation. Note that the fovea is not at the center of the image (i.e., the contours are taken from the left eye).

Since space-variant sampling arrangements typically have some areas of high acuity and some areas of low acuity, they require an active vision system to allow the high acuity regions to sample different regions of a visual scene. By analogy with the high acuity foveal pit found in most vertebrate vision systems, we refer to the region of highest acuity as the **fovea** and the process of addressing the high acuity area to a region of an image as **foveation**. In order to simulate the active vision aspects of a space-variant visual sampling regime, we would like to be able to specify a point in a large image for the system to fixate on. Therefore, a further design criteria for an importing procedure is that we want the importing procedure to be relative to its own internal coordinate system that may be "aimed" at different areas of a large image. Our method for accomplishing this is to give the nodes coordinates such that the fovea is at the origin and each unit represents one pixel in a standard raster image. The advantage of this design is that the filters may be precomputed for a graph relative to the origin and simply shifted to different areas of the image, resulting in fast on-line importation. The drawback to this design is that the extent of the "visual field" must be fixed prior to computing the filters.

Heckbert's Elliptical Weighted Average filters are ellipses computed for each new sample such that the axes of the ellipse lay along the eigenvectors of the Jacobian matrix and the weights for each point in the ellipse are given by an elliptical Gaussian function. In other words, the image value assigned to each resampled point is the weighted sum of image values on the original points lying within the computed ellipse. Our approach to computing the ellipses for the resampled points is to perform a least-squares fit of an ellipse to the Voronoi cell of each node and compute Gaussian weights. The ellipses computed for a small randomly generated set of Gaussian distributed points in the plane are seen in Figure 2·3.

The filters for a point set are stored in a $^{\text{TM}}$MATLAB `struct` named `imgGraph`. An `imgGraph` has three fields: `pntMap`, `breakpoints` and `filtWeights`. In order for the importation of images to be fast in $^{\text{TM}}$MATLAB the three fields are used to avoid code loops. The `breakpoints` field contains a list of indices to `pntMap` and `filtWeights` that indicate the start and end of a block of pixels or weights corresponding to each new node.

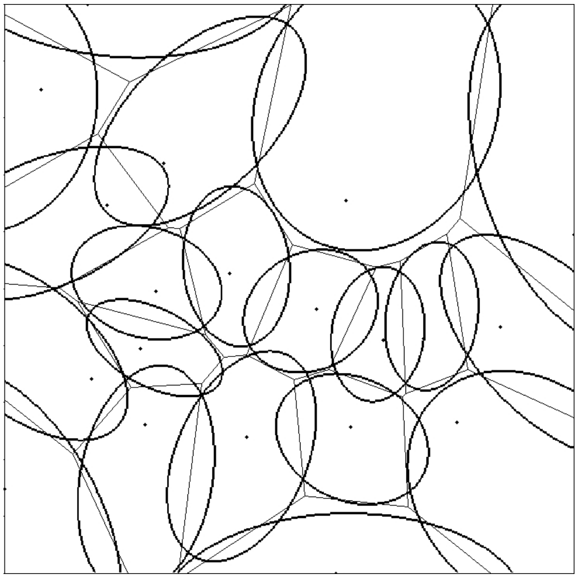
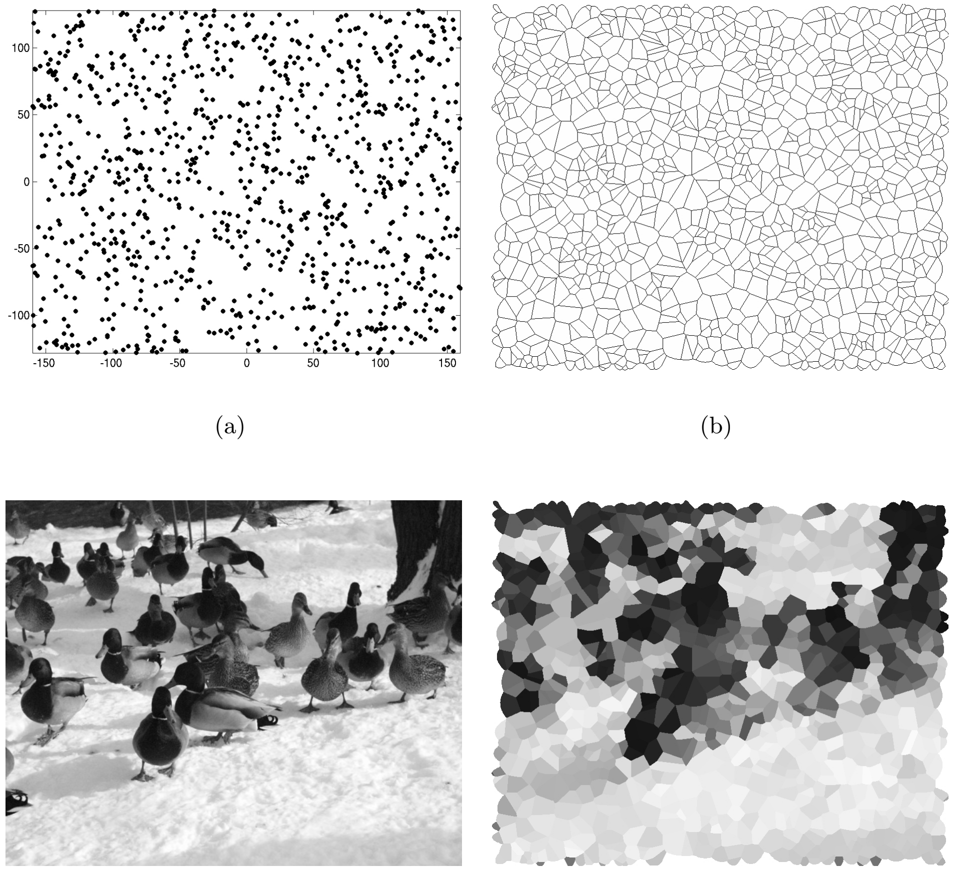


Figure 2·3: Voronoi cells for a point set and the corresponding ellipses fit with least squares error used to generate the Elliptical Weighted Average filters of Heckbert (1989).

The `pntMap` field is a set of two vectors containing the x- and y-coordinates for the nodes corresponding to those pixels that lie within the ellipse for each point. The `filtWeights` field contains the weight to be applied to each pixel in `pntMap`. Therefore, the size of `breakpoints` is the same as the cardinality of the node set, while the size of `pntMap` and `filtWeights` are larger than the node set (since more than one pixel typically maps to each node), but equal to each other. Figure 2·4 demonstrates the importing of an image onto a set of nodes that were randomly distributed in the plane with a uniform probability. Figure 2·5 demonstrates the simulated active vision system by importing a large image at multiple fixation points.

2.7 Visualization

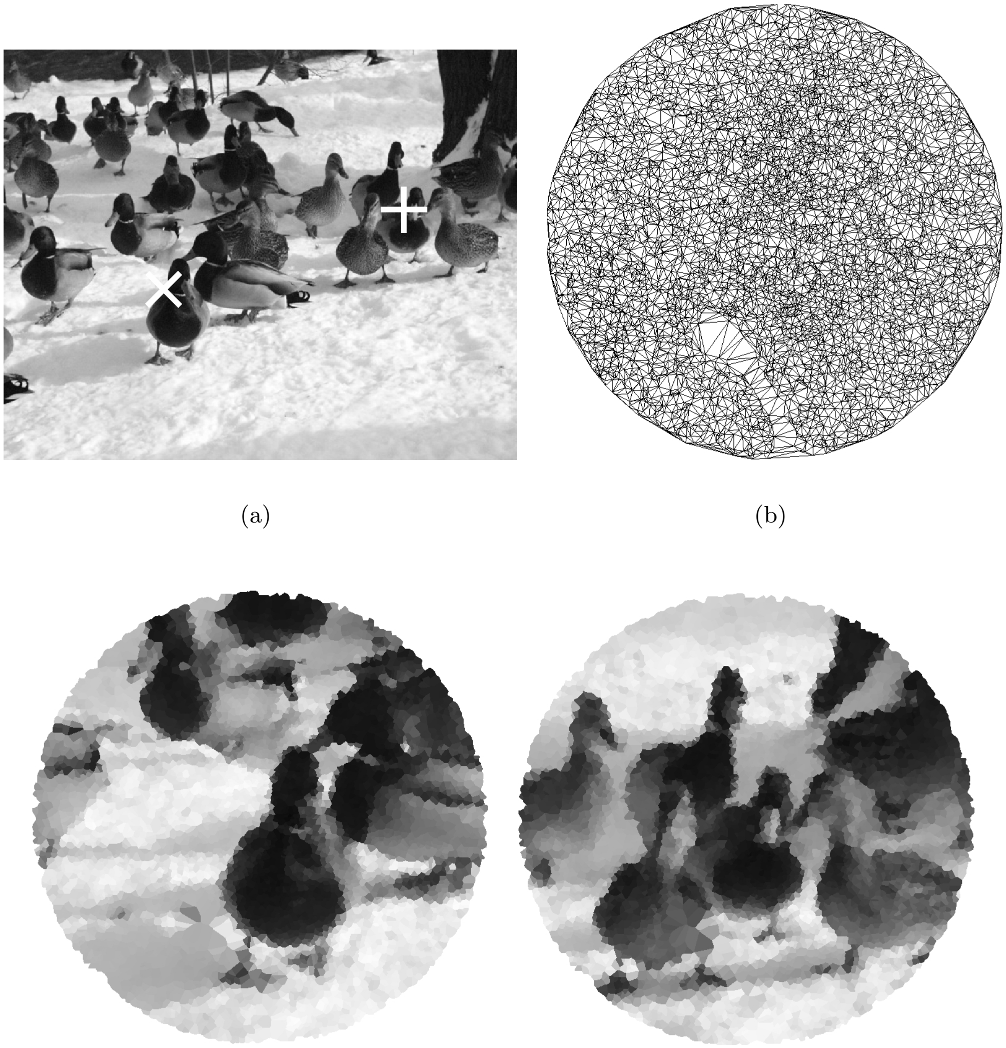
Visualization of arbitrary, nonuniformly sampled 2D data is a difficult issue. Typical "stick-and-ball" representations of graphs poorly convey the content of an image associated with the nodes. Two methods have been implemented for visualizing an image on an arbitrary architecture.



(a) (b)

(c) (d)

Figure 2·4: Demonstration of importing an image onto a set of points distributed randomly in the plane with a uniform distribution. Note that the resolution of the sampling is almost two orders of magnitude smaller than the resolution of the original. (a) Nodes randomly placed in the plane with a uniform distribution. (b) Voronoi cells for the node set upon which the filters were generated. (c) The original raster image: `ESLab0043.jpg`. (d) The image imported onto the node set.



(a) (b)

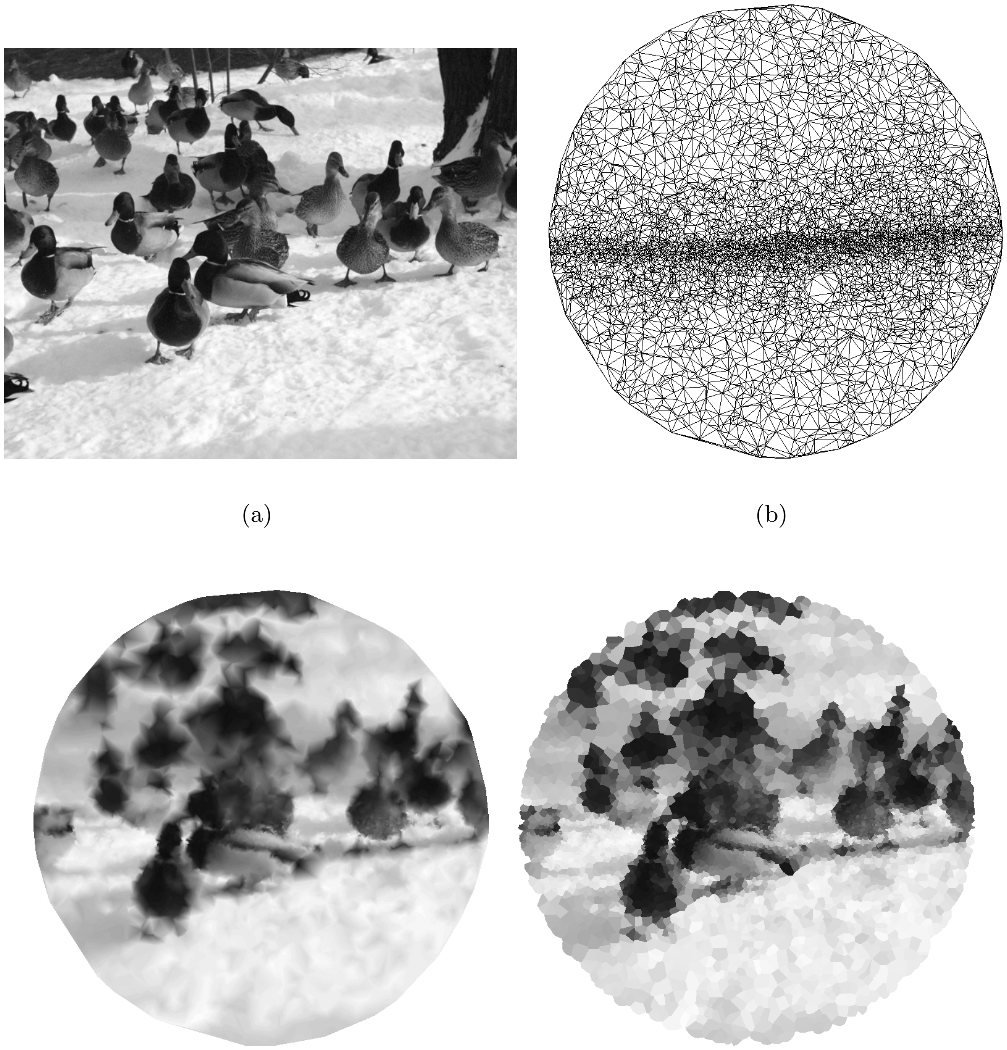
(c) (d)

Figure 2·5: Demonstration of importing a large image onto a smaller graph at different points (i.e., multiple fixation points). The example chosen here is based on the retinal topography of the pigeon (Whitteridge, 1965). (a) Original image: `ESLab0043.jpg`. The two fixation points are marked with a white '×' and a white '+'. (b) The graph corresponding to the retinal topography of the pigeon. (c) Result of importing the image in (a) at the point marked with a white '×'. (d) Result of importing the image in (a) at the point marked with a white '+'.

The first of these methods interpolates image values at a node across the *faces* of a graph. If the graph is planar, the interior faces will be polygons, with image data at each point on the polygon. If the graph is nonplanar, a Delaunay triangulation of the points may be found quickly for purposes of the visualization. However, it should be noted that not all graphs will have faces that produce a good image (e.g., if the nodes were collinear). A common "shading" technique from computer graphics is to perform a bilinear interpolation across vertices (i.e., Gourand shading). Applying this technique to the internal faces provides a smooth change across the image. Unfortunately, the polygonal faces can introduce artifacts into the visualization that degrade the quality. The function `showmesh.m` generates a display in this manner.

A second visualization method is implemented in the Graph Analysis Toolbox that uses the Voronoi diagram of the vertex set. This technique simply assigns each Voronoi cell the color (or grayscale value) of its corresponding vertex. The visualization benefits from its independence of the planar and internal faces requirements of the previous technique. However, the visualization can sometimes look blocky. Furthermore, since the boundary nodes have Voronoi cells that extend to infinity, a dense set of phantom nodes is used to make finite, appropriately sized cells for the boundary nodes. The Voronoi information for a graph is stored in a $^{\text{TM}}$MATLAB `struct` called `voronoiStruct`. A `voronoiStruct` is produced from a node set by the function `voronoicells.m` and consists of the three fields `pts`, `faces` and `index`. The field `voronoiStruct.pts` contains coordinates for the vertices of the Voronoi cells for the node set. Faces intended for use by `patch.m` are contained in `voronoiStruct.faces` and an index referencing nodes with Voronoi cells contained inside the convex hull is given by `voronoiStruct.index`.

The two visualization techniques are illustrated in Figure 2·6 for a grayscale image. Despite the smoothness given by the face interpolation method of visualization, the Voronoi cells method offers a better notion of the structure of the image distribution on the nodes, and we therefore prefer it for visualization of space-variant images in the remainder of this document.



(a) (b)

(c) (d)

Figure 2·6: Comparison of the two visualization techniques implemented in the Graph Analysis Toolbox. (a) Original image: `ESLab0043.jpg`. (b) The graph corresponding to the retinal topography of the plains kangaroo (Hughes, 1974). (c) Visualization of the space-variant image performed by interpolating across the Delaunay triangles of the graph implemented in `showmesh.m`. (d) Visualization of the space-variant image performed by assigning a uniform grayscale value to the Voronoi cells of the node set, implemented in `showvoronoi.m`.

It is important to distinguish between *sampling aliasing* and *visual aliasing.* Sampling aliasing refers to the inadequacy of the local sampling density to satisfy the image frequency (as detailed above). On the contrary, visual aliasing refers to the displeasing visual artifacts induced as a result of the visualization technique. Sampling aliasing is minimized by the precomputation of Heckbert's resampling filters. Visual aliasing, however, depends on the visualization method employed and in no way reflects an inadequacy of the data obtained nor its internal representation.

2.8 Processing

Processing data on graphs is a recurring theme that extends beyond space-variant vision systems. There are four main types of processing implemented in the Graph Analysis Toolbox: interpolation, filtering, edge finding and segmentation.

Isotropic and anisotropic versions of all the processing methods exist in the sense that the edge weights all take unity value in the isotropic case and the weights assume different values in the isotropic case. For example, use of uniform or nonuniform weights in building the Laplacian matrix, is the difference between the isotropic diffusion of Koenderink (1984) and the anisotropic diffusion of Perona and Malik (1990). Therefore, the theme is to encode data information (e.g., intensity changes for images, distances for a point set) in the structure of edge weights, and then build the operator in accordance with the weights. This procedure provides the difference between isotropic or anisotropic diffusion, interpolation, filtering and edge finding, as well as affording the structure necessary for some segmentation algorithms (e.g., Shi and Malik, 2000).

The functions used to build the important matrix operators listed above are `incidence.m`, `laplacian.m` and `adjacency.m`. The function `incidence.m` generates the edge-node incidence matrix, `laplacian.m` generates the Laplacian matrix and `adjacency.m` generates the adjacency matrix. In order to allow the user to choose between isotropic and anisotropic operators, all of the operator generating functions allow specification of edge weights in order to produce anisotropic operators, but default to generating isotropic operators if weights are not specified.

The important task of generating a weight set is handled by the function `makeweights.m`. Define the vector of data changes, c_{ij}, as the Euclidean distance between the fields (e.g., coordinates, image RGB channels, image grayscale, etc.) on nodes v_i and v_j. For example, if we represent grayscale intensities defined on each node with vector b, then $c = Ab$. If the fields are nodal coordinates in the plane, then c represents the Euclidean distance in the plane.

Although we typically treat coordinates as any other data field (e.g., to filter, interpolate, etc.), in the context of space-variant vision we may want to treat spatial differences between the nodes separately from image-derived differences. Accordingly, associated with the data and coordinates is a separate parameter, β^1 and β^2 we call **scale**. Setting either parameter to zero nullifies the effects of the corresponding data or coordinates. A common weighting function is implemented in `makeweights.m` and given by

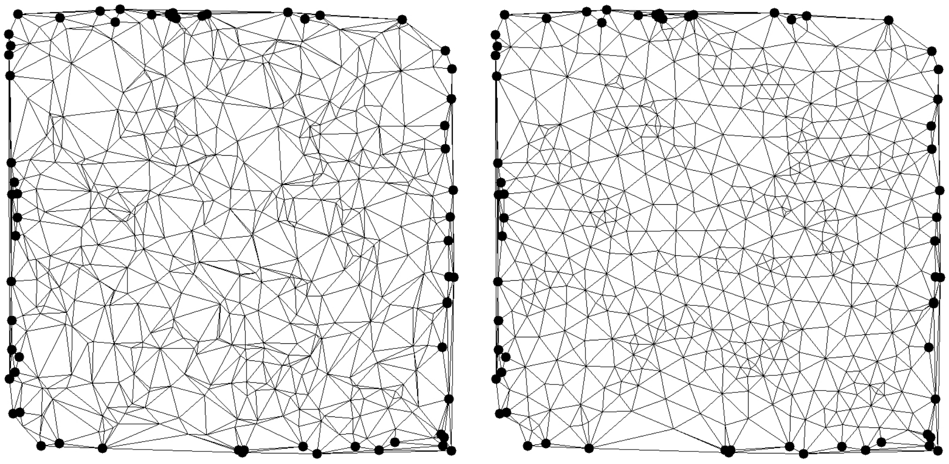
$$w_{ij} = \exp\left(-|\beta^1 c^1_{ij} + \beta^2 c^2_{ij}|\right). \tag{2.13}$$

Therefore, `makeweights.m` optionally accepts both data values and coordinate values for the node set and generates a corresponding set of weights. In order to make one choice of parameters (i.e., β^1, β^2) applicable to a wide range of data sets, we have found it helpful to normalize the vectors c^1 and c^2.

2.8.1 Interpolation

The method of interpolation implemented in the Graph Analysis Toolbox is to solve the combinatorial Laplace equation with Dirichlet boundary conditions given by the known values, developed in Chapter 7.

The function `dirichletboundary.m` inputs an index of boundary nodes and their values and solves the combinatorial Dirichlet problem for a graph with arbitrary connectivity, producing a combinatorial harmonic function, x, such that $Lx = 0$ subject to boundary conditions. Examples of using `dirichletboundary.m` to perform isotropic and anisotropic



(a) (b)

Figure 2·7: An example of using the interpolation method for graph drawing. (a) A graph was created by randomly generating the coordinates with a uniform distribution and connecting the points with a 2D Delaunay triangulation. The black dots represent the extremal points chosen to represent the boundary (i.e., to have their coordinates fixed). (b) The graph generated by interpolating the coordinates of the interior nodes.

interpolation on missing image data are reserved until Chapter 7. However, graph drawing is another task in which `dirichletboundary.m` is useful. By treating the extremal nodes as boundary nodes, the coordinates of the interior nodes may be interpolated in order to produce a more regular representation in the sense that each interior node is placed at the average of its neighbors (by the mean value theorem). This usage of `dirichletboundary.m` is displayed in Figure 2·7.

2.8.2 Filtering

In the context of surface fairing, Taubin has already taken a combinatorial approach to filtering on a graph (Taubin, 1995; Taubin et al., 1996). Taubin treats the coordinates of a vertex set of a 3D mesh as a signal for which low-pass filters may be designed in order to smooth a noisy surface. The signal processing treatment in Taubin's work follows standard signal processing approaches, except that Taubin wants to apply the same techniques to an arbitrary, shift-variant, topology. Generally the eigenfunctions of the Laplacian operator define the surface harmonics (Courant and Hilbert, 1989a). In the combinatorial setting, the Laplacian operator is represented by the Laplacian matrix, although Taubin chooses a combinatorial representation of the Laplacian operator that is different than the definition given in (2.5). For shift-invariant topologies, the Laplacian matrix is circulant and the complex exponential basis vectors (functions) are the eigenvectors (Strang, 1986). In the general case of a shift-variant topology, the Laplacian is not circulant, requiring a different set of (usually unknown) eigenvectors in order to perform signal filtering. Taubin's method of filtering circumvents the need to compute the eigenvectors explicitly in order to modify the spectral coefficients of an input signal (e.g., the coordinates of a graph or an image on a graph). The function `filtergraph.m` implements Taubin's $\lambda - \mu$ filtering technique as well as standard mean filtering. Both the mean filter and the $\lambda - \mu$ filter are low-pass filters. However, a high pass filter may be generated by subtracting the low-pass filtered signal from the original. A band-pass filter may be generated by using the difference of two low-pass filters. Figure 2·8 demonstrates image filtering and Figure 2·9 demonstrates coordinate filtering.

The spectrum of the Laplacian matrix has been thoroughly investigated (Mohar, 1991; Chung, 1997; Merris, 1994; Anderson and Morley, 1971; Fiedler, 1975a). It is well known that the eigenvalues of the Laplacian matrix are nonnegative and ordered such that the smallest eigenvalue corresponds to the lowest frequency harmonic (i.e., the DC component) and the largest eigenvalue corresponds to the highest frequency harmonic. This view of the spectral characteristics of the Laplacian matrix predicts its use as an edge detector since, as an operator, it will clearly have the effect of a high-pass filter. Another implication of the spectral properties of the Laplacian matrix is that an iteration on the signal x of the form

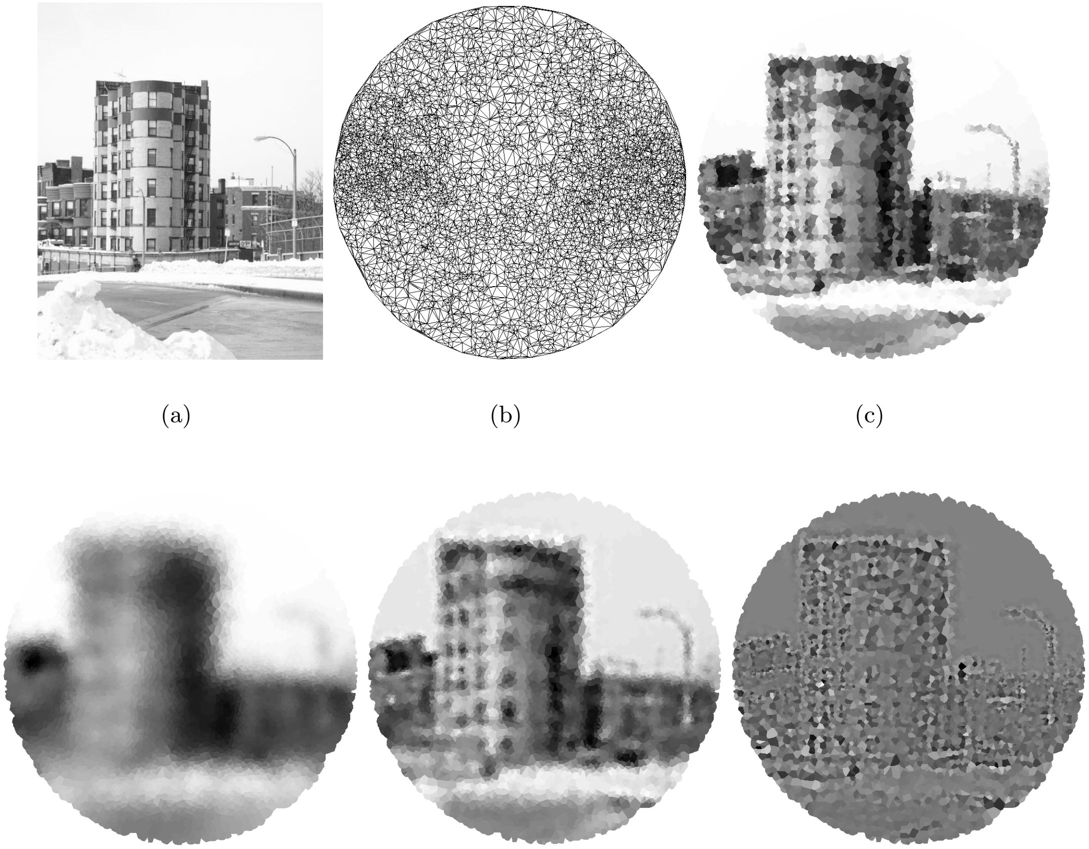
$$x_1 = x_0 - \alpha L x_0, \tag{2.14}$$

will have the effect of creating a low-pass signal, since a high-pass form of the signal is subtracted from the original. Since equation (2.14) represents one iteration of the combinatorial diffusion equation

$$\frac{dx}{dt} = Lx, \tag{2.15}$$

both isotropic (Koenderink, 1984) and anisotropic (Perona and Malik, 1990) diffusion may be viewed as a low-pass filter. The function `diffusion.m` performs diffusion on a signal. Figure 2·10 shows an example of isotropic and anisotropic diffusion on an image.

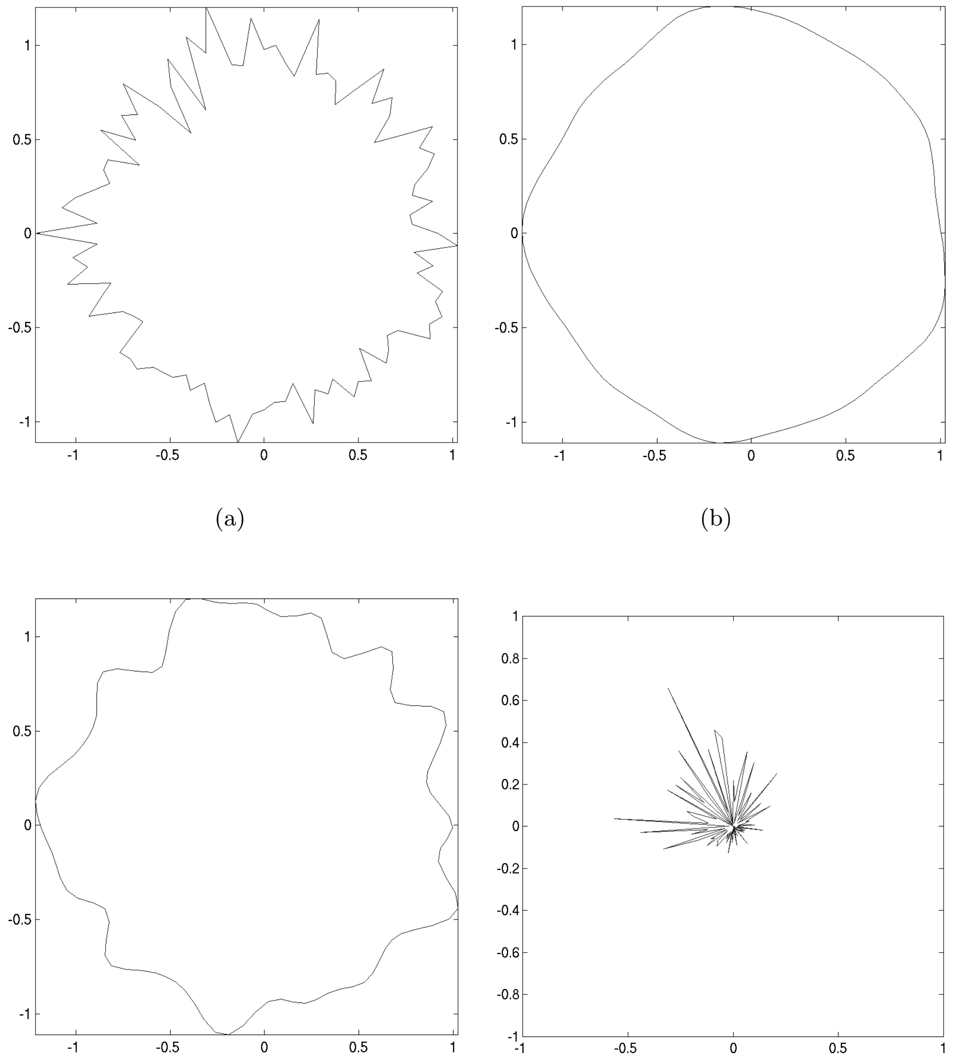
The combinatorial Dirichlet problem method of interpolation presented in Chapter 7 may also be viewed as a low-pass filter, since it requires the solution to a system of equations corresponding to the Laplace equation, constrained by Dirichlet boundary conditions. A solution to a system of equations $Lx = b$ may be viewed (if not computed) as $x = L^{-1}b$. The inverse of a matrix retains the same eigenvectors as the original, but the corresponding eigenvalues of the inverse matrix are the reciprocal of the eigenvalues of the original. Therefore, the solution to the constrained Laplace equation may be viewed as a low-pass filter since the lowest frequencies of the inverse of the (constrained) Laplacian matrix will correspond to the largest eigenvalues and *vice versa.* The relationship of the solution to a



(a) (b) (c)

(d) (e) (f)

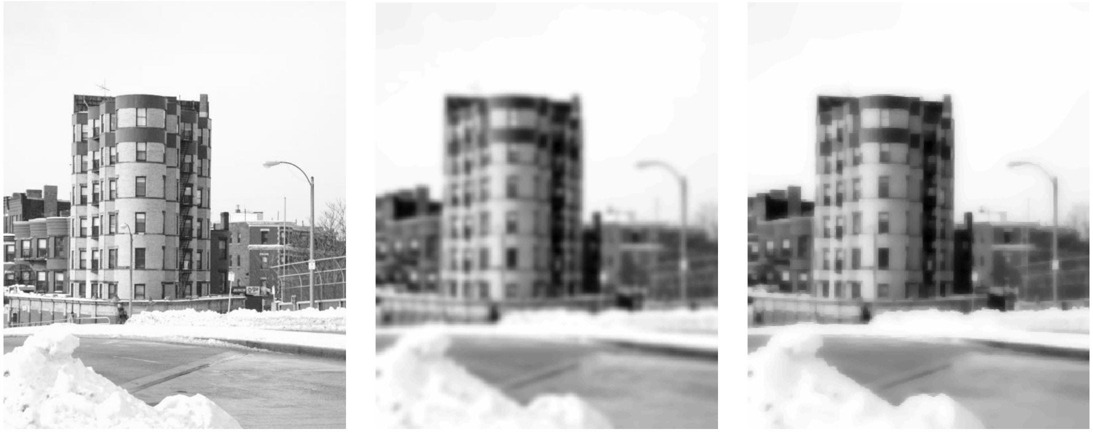
Figure 2·8: Filtering image data on a space-variant image. (a) The original image: `ESLab0059.jpg`. (b) A space-variant graph patterned after the retinal genglion cell distribution of the bottlenosed dolphin (Mass and Supin, 1995). (c) The imported image, before any processing. (d) Result of the mean filter applied to the image in (c). (e) The low-pass $\lambda-\mu$ filter (Taubin, 1995) applied to the image in (c). (f) A high-pass filter of (c), produced by differencing the low-pass signal of (e) with the original.



(a) (b)

(c) (d)

Figure 2·9: Filtering coordinate data on a ring graph. (a) A noisy ring graph produced by adding radial Gaussian distributed random to nodes arranged in a perfect circle. (b) The effect of applying the mean filter to the coordinates of the graph in (a). (c) The low-pass $\lambda - \mu$ filter (Taubin, 1995) applied to the coordinates of the graph in (a). (d) A high-pass filter of the coordinates in (c), produced by differencing the low-pass signal of (c) with the original.

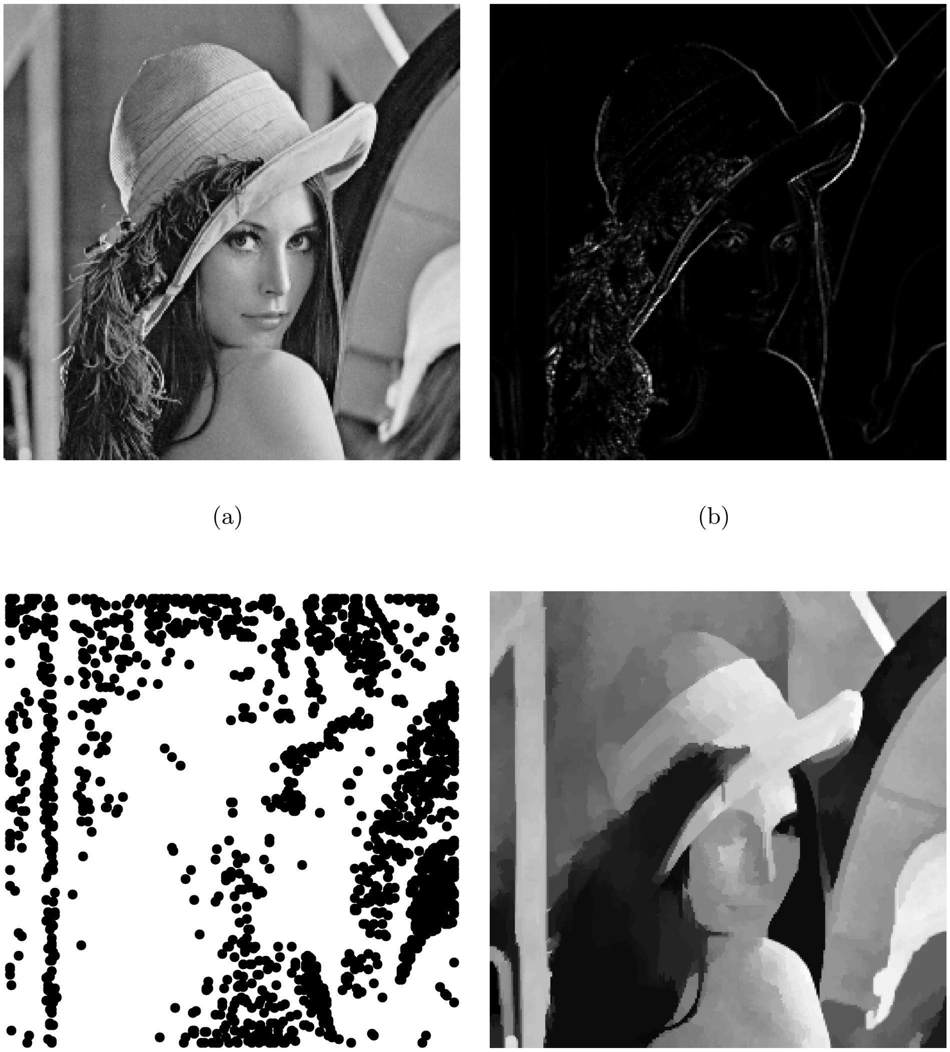


(a) (b) (c)

Figure 2·10: Diffusion filtering on an image. (a) The original image: `ESLab0059.jpg`. (b) The effect of performing isotropic diffusion on the 4-connected lattice representing image (a). (c) The effect of performing anisotropic diffusion on the 4-connected lattice representing image (a).

constrained Laplace equation to low-pass filtering and steady state diffusion with boundary conditions (see Doyle and Snell, 1984) justifies its inclusion as a filtering method.

Figure 2·11 demonstrates anisotropic interpolation applied to low-pass filtering (i.e., smoothing) an image. To generate Figure 2·11, a 4-connected lattice was generated with the weight function of (2.13), based on the Lena image. Samples were chosen from relatively uniform areas by computing the sum of the edge gradients incident on each node. All nodes with gradient sums below a threshold were selected as sample nodes to have their values fixed. The remaining nodes were anisotropically interpolated, given the fixed set. One can see that sharp boundaries are maintained, due to the encoding of image information with weights. Areas of the image with high variability (e.g., the feathers) are smoothed considerably since very few samples were taken, while areas with initially low variability remain uniform.



(a)

(b)

(c)

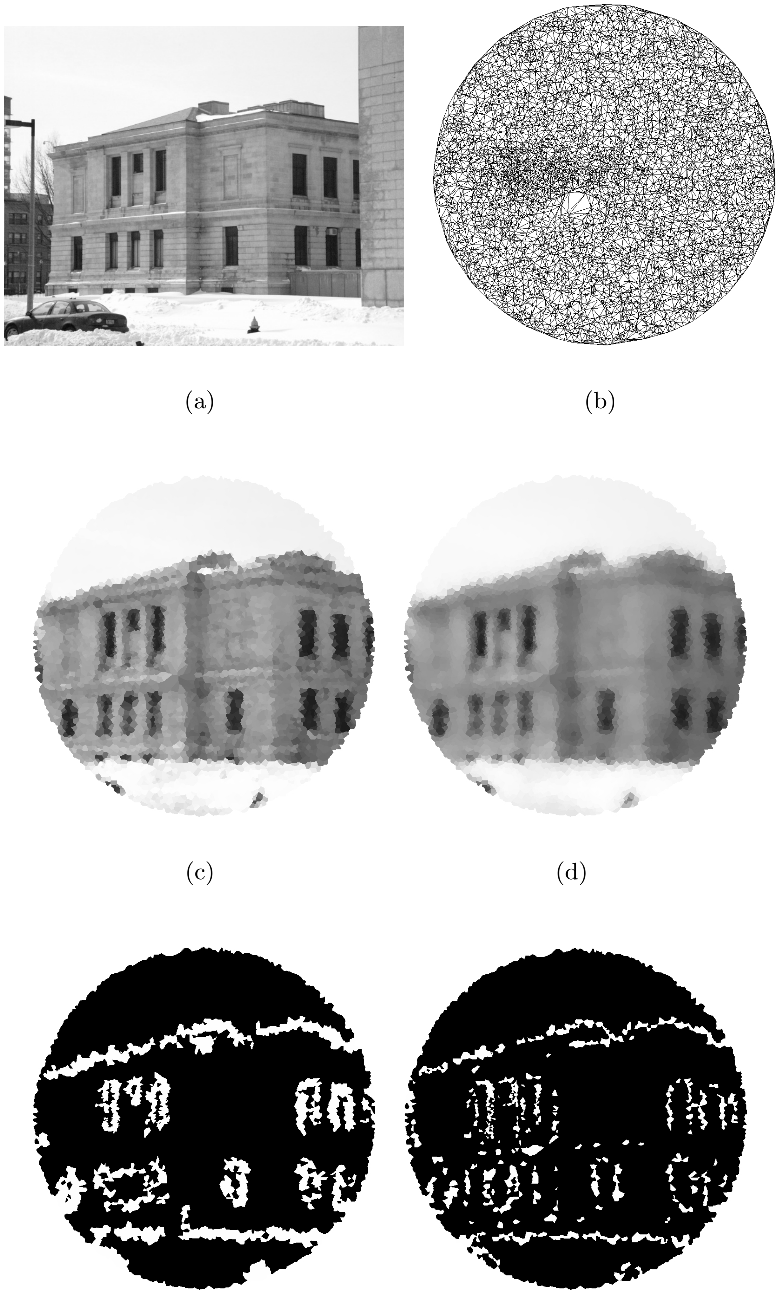
(d)

Figure 2·11: Anisotropic interpolation used to low-pass filter an image. (a) Original Lena image. (b) Magnitude of summed image gradients. (c) Samples taken from lowest magnitude points. (d) Anisotropically interpolated image.

2.8.3 Edge finding

Edge detection is a common goal of low-level computer vision. Common edge detection approaches (Jain, 1989) make use of gradient (Roberts, 1965) or Laplacian (Marr and Hildreth, 1980) operators. An interesting feature of these operators is that although they both operate on values associated with the node set, the combinatorial gradient operator (the edge-node incidence matrix) returns values on the *edge set* while the combinatorial Laplacian returns values on the *node set*. This difference is analogous to 3D vector calculus in which the application of both the gradient and Laplacian operators to a scalar field results in a scalar field for the Laplacian and a vector field for the gradient. Standard gradient operators used for edge detection are applied to pixels and return values on pixels (Roberts, 1965; Prewitt, 1970; Davis, 1975), causing the output of a typical gradient or Laplacian-based edge detection algorithm to be a set of edge-pixels.

The function `findedges.m` keeps with tradition by returning a set of edge pixels, regardless of whether a combinatorial gradient or Laplacian-based edge detection scheme is chosen. In the context of space-variant edge detection, we have found that better edge detection results are obtained by using an anisotropic edge operator, where the weights are based on the Euclidean distance of the point coordinates. The reason for this is that sensors (nodes) that are located more distant from each other are more likely to have an intensity change that crosses threshold, even if the continuous light distribution varies smoothly across a single object. Weighting the edge operator by distance compensates for this effect. Figure 2·12 demonstrates gradient and Laplacian-based edge detection on a space-variant image. In order to reduce noise but preserve edges, anisotropic diffusion was performed before the edge operator was applied. Note that different nodes correspond to different sized Voronoi regions in the visualization, due to the space-variance, which results in varying edge width.



(a) (b)

(c) (d)

(e) (f)

Figure 2·12: Edge detection of image data in a space-variant image. (a) The original image: `ESLab0032.jpg`. (b) A space-variant graph patterned after the retinal ganglion cell distribution of the labrador (Hughes, 1977). (c) The imported image, before any processing. (d) Result of anisotropic diffusion preprocessing used to sharpen edges and blur noise. (e) Result of gradient-based edge detection. (f) Result of Laplacian-based edge detection.

2.8.4 Segmentation

The use of graph theory for data clustering and image segmentation may be traced to the work of Zahn on Gestalt clustering (Zahn, 1971). By framing the segmentation problem in the context of graph partitioning, Wu and Leahy (1993) developed the minimum cut algorithm. Graph partitioning approaches to segmentation has led to several other algorithms (Shi and Malik, 2000; Perona and Freeman, 1998; Wang and Siskund, 2003; Sarkar and Soundararajan, 2000). A new method for applying graph partitioning to image segmentation is presented in Chapter 4, and termed the **isoperimetric algorithm**. The function `partitiongraph.m` performs a graph bipartition using the isoperimetric algorithm, normalized cuts (Shi and Malik, 2000) or spectral partitioning (Pothen et al., 1990). Converting a graph bipartitioning algorithm to a complete segmentation may be accomplished by recursively applying the bipartitioning algorithm to each new segment and stopping the recursion when a specified metric of partition quality fails to be satisfied. The function `recursivepartition.m` recursively applies `partitiongraph.m` and returns integer labels of each node such that nodes sharing the same label are considered to be in the same partition. Since one often wants to apply these algorithms to standard Cartesian images, the functions `imgsegment.m` and `imgsegpyr.m` input standard images and return segmentations by using an underlying lattice or pyramid topology, respectively. The function `isosolve.m` performs the calculations for computing the potentials for a single application of the isoperimetric algorithm of Chapter 4. An example of applying the segmentation algorithm to a space-variant image is given in Figure 2·13.

2.9 Miscellaneous functions

This section details the minor functions included in the Graph Analysis Toolbox used to manipulate and visualize data.

Three functions used to perform minor graph and matrix manipulation are `adjtoedges.m`, `circulant.m` and `removeisolated.m`. The function `adjtoedges.m` converts an adjacency matrix to an edge set. Since shift-invariant graphs correspond to circulant adjacency and

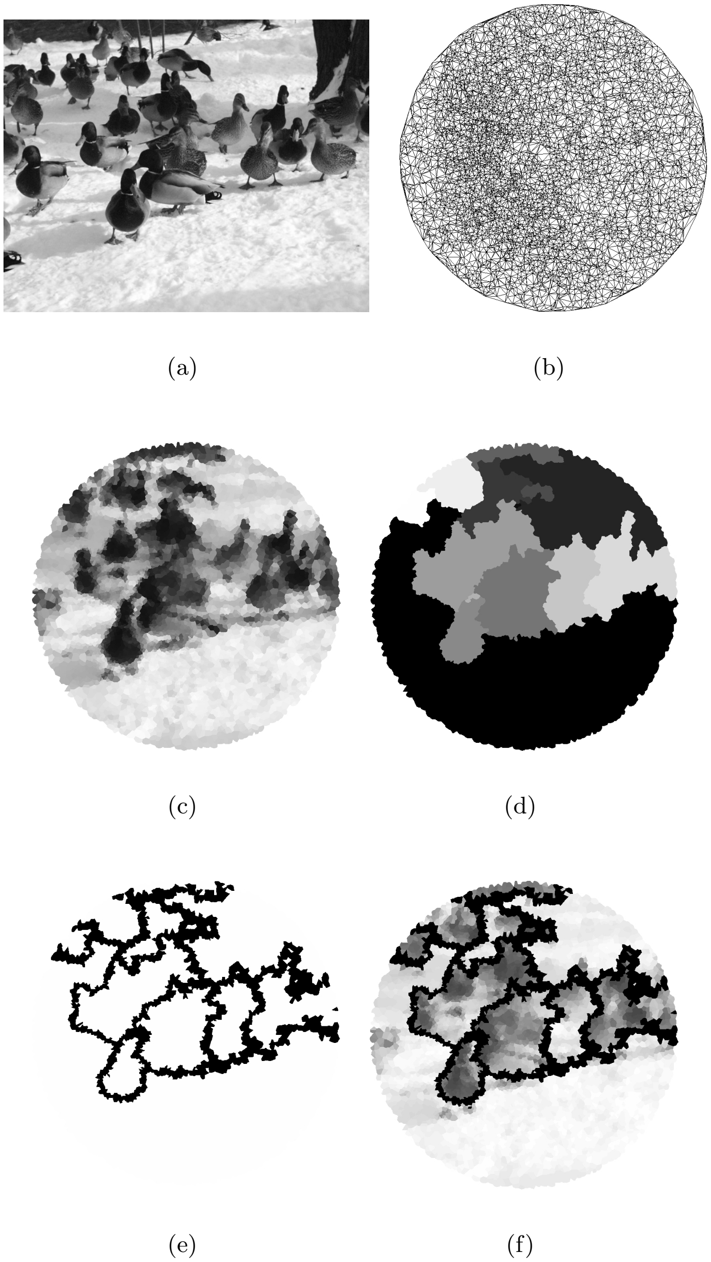


Figure 2·13: Segmentation of image data in a space-variant image. (a) The original image: `ESLab0043.jpg`. (b) A space-variant graph patterned after the retinal ganglion cell density of the two-toed sloth (Costa et al., 1989). (c) The imported image, before any processing. (d) Segment labels obtained using the isoperimetric algorithm ($\beta^1 = 95, \beta^2 = 15, \text{stop} = 1 \times 10^{-5}$). (e) Segmentation result displayed as the outline of segments against a white background. (f) Segmentation result displayed as black outlines against the faded image.

Laplacian matrices, the function `circulant.m` is used to generate a sparse circulant matrix from one row that details the topology of a single node (which suffices to define the topology of the whole graph, since the graph is shift-invariant). The function `removeisolated.m` is used to remove any nodes which are not connected to any other node.

The functions `binarysearch.m`, `equalize.m` and `normalize.m` provide additional data processing. The function `binarysearch.m` implements a divide-and-conquer search algorithm of a sorted vector for the closest value of a given input. Normalization of the columns of a matrix to a specified range (defaulting to [0,1]) is accomplished by the function `normalize.m`. Histogram equalization of a vector is performed by the function `equalize.m`.

Four functions are available to aid in visualization of a segmentation and the production of segmentation results for publication. Three representations of a segmentation are returned by `segoutput.m` and `segoutputSV.m`: integer node labels, black outlines of segments against a white background and red outlines of segments superimposed on the original image. The special case of Cartesian images is handled by `segoutput.m`, while the general case of space-variant images is handled by `segoutputSV.m`. Since one may want to use the red-outlined segmentations returned by `segoutput.m` and `segoutputSV.m` in publication, the functions `colorseg2bwseg.m` and `colorseg2bwsegSV.m` convert the red-outlined segmentation to a black-outlined segmentation superimposed on a faded out copy of the original (so as not to confuse the segmentation lines with image features). Cartesian and space-variant segmentations are handled by `colorseg2bwseg.m` and `colorseg2bwsegSV.m` respectively.

Since there is a frequent need in the Graph Analysis Toolbox for vectorizing an RGB image, the function `rgbimg2vals.m` exists for this purpose.

2.10 Conclusion

The Graph Analysis Toolbox was developed to provide tools for space-variant computer vision that are independent of sampling regime or a chosen topology. Graphs comprise the primary data structure, and combinatorial methods are the tools used to process the data. Since graph-theoretic data structures appear in other disciplines with problems of analysis similar to computer vision (e.g., segmentation, edge finding), we hope that the tools developed here may find a wider audience.

Despite the ubiquity of space-variant sensors in biological systems, it is rare to find artificial space-variant acquisition devices or displays. For this reason, the Graph Analysis Toolbox allows the simulation of a space-variant sensor, by providing tools to transfer images acquired with a Cartesian sensor array to an arbitrary space-variant representation. Likewise, tools are also available to allow the display of space-variant images on a standard CRT monitor. In addition to simulation of a space-variant sensor, the Graph Analysis Toolbox provides the simulation of an active vision system that allows the simulated space-variant architecture to be directed to different points in a larger image.

The mathematical tools developed by Roth and Branin provide a foundation upon which to build the combinatorial approach to space-variant computer vision taken in this thesis. Development of both the isoperimetric segmentation algorithm of Chapter 4 and the interpolation method of Chapter 7 have their basis in these mathematics. Furthermore, the software tools comprising the Graph Analysis Toolbox are used to generate all the figures and tables in this thesis.

Chapter 3

Isoperimetric graph partitioning

3.1 Introduction

The graph partitioning problem is to choose subsets of the vertex set of a graph such that the sets share a minimal number of spanning edges while satisfying a specified cardinality constraint. Graph partitioning appears in parallel processing (Simon, 1991), solving sparse linear systems (Pothen et al., 1992), VLSI circuit design (Alpert and Kahng, 1995) and image segmentation (Shi and Malik, 2000; Wu and Leahy, 1993), among others.

Methods of graph partitioning take different forms, depending on the number of partitions required, whether or not the nodes have coordinates, and the cardinality constraints of the sets. In this chapter, we use the term **partition** to refer to the assignment of each node in the vertex set into two (not necessarily equal) parts. We propose an algorithm termed **isoperimetric partitioning**, since it is derived and motivated by the isoperimetric constant defined by Cheeger for continuous manifolds Cheeger (1970). The isoperimetric algorithm most closely resembles spectral partitioning in its use and ability to create hybrids with other algorithms (e.g., multilevel spectral partitioning (Karypis and Kumar, 1998), geometric spectral partitioning (Chan et al., 1994)), but requires the solution to a large, sparse system of equations, rather than solving the eigenvector problem for a large, sparse matrix. This leads to improved speed and numerical stability. In this chapter we present the isoperimetric algorithm, prove some of its properties, review other graph partitioning methods and compare partitioning algorithms on randomly generated weighted and unweighted planar, nonplanar and three-dimensional graphs as well as a few specialized graphs.

3.2 Isoperimetric algorithm

Graph partitioning has been strongly influenced by properties of a combinatorial formulation of the classic isoperimetric problem: *For a fixed area, find the shape with minimum perimeter* (Chung, 1997). The approach to graph partitioning presented here is a heuristic for finding a region with minimum perimeter for a fixed area.

Cheeger defined (Cheeger, 1970) the **isoperimetric constant** h of a manifold as

$$h = \inf_S \frac{|\partial S|}{\mathrm{Vol}_S}, \tag{3.1}$$

where S is a region in the manifold, Vol_S denotes the volume of region S, $|\partial S|$ is the area of the boundary of region S, and h is the infimum of the ratio over all possible S. For a compact manifold, $\mathrm{Vol}_S \leq \frac{1}{2}\mathrm{Vol}_{\mathrm{Total}}$, and for a noncompact manifold, $\mathrm{Vol}_S < \infty$ (see Mohar (1989, 1988)).

For a graph, G, the **isoperimetric number** (Mohar, 1989), h_G is

$$h_G = \inf_S \frac{|\partial S|}{\mathrm{Vol}_S}, \tag{3.2}$$

where $S \subset V$ and

$$\mathrm{Vol}_S \leq \frac{1}{2}\mathrm{Vol}_V. \tag{3.3}$$

In finite graphs, the infimum in (3.2) becomes a minimum. The boundary of a set, S, is defined as $\partial S = \{e_{ij} | i \in S, j \in \overline{S}\}$ and on a weighted graph

$$|\partial S| = \sum_{e_{ij} \in \partial S} w(e_{ij}). \tag{3.4}$$

Although different definitions of volume have been proposed for a set containing nodes (c.f., Dodziuk (1984); Dodziuk and Kendall (1986); Mohar (1988)), the standard definition in the context of graph partitioning is

$$\mathrm{Vol}_S = |S|. \tag{3.5}$$

For a given set of nodes, S, we term the ratio of its boundary to its volume as the **isoperimetric ratio** and denote it by $h(S)$. The **isoperimetric sets** for a graph, G, are any S and $\overline{S}$ for which $h(S) = h_G$. The specification of a set satisfying (3.3), together with its complement may be considered as a *partition* and therefore we use the term interchangeably with the specification of a set satisfying (3.3). A good partition is defined to be one with a low isoperimetric ratio (i.e., the optimal partition consists of the isoperimetric sets themselves). Therefore, our goal is minimize $|\partial S|$ for a constant Vol_S. Unfortunately, finding isoperimetric sets is an NP-hard problem (Mohar, 1989), so our algorithm is a heuristic for finding a set with a low isoperimetric ratio that runs in polynomial time.

Define an indicator vector, x, that takes a binary value at each node

$$x_i = \begin{cases} 0 & \text{if } v_i \in S, \\ 1 & \text{if } v_i \in \overline{S}. \end{cases} \tag{3.6}$$

A specification of x may also defines a partition. By definition (2.5) of the Laplacian matrix, L,

$$|\partial S| = x^T L x, \tag{3.7}$$

and $\mathrm{Vol}_S = x^T r$ where r denotes the vector of all ones. Maximizing the volume of S subject to $\mathrm{Vol}_S = k$ for some constant $k \leq \frac{1}{2}\mathrm{Vol}_V$ may be done by asserting the constraint

$$x^T r = k. \tag{3.8}$$

In terms of the indicator vector, the isoperimetric number of a graph (3.2) is given by

$$h_G = \min_x \frac{x^T L x}{x^T r}. \tag{3.9}$$

Given an indicator vector, x, then $h(x)$ is used to represent the isoperimetric ratio associated with that partition.

The constrained optimization of the isoperimetric ratio is made into a free variation via the introduction of a Lagrange multiplier Λ (Arfken and Weber, 1985) and relaxation of the binary definition of x to take nonnegative real values. Therefore, solving for an optimal partition may be accomplished by minimizing the function

$$F(x) = x^T L x - \Lambda(x^T r - k). \tag{3.10}$$

Since L is positive semi-definite (see Biggs (1974); Fiedler (1986)) and $x^T r$ is nonnegative, $F(x)$ will be at a minimum for any critical point. Differentiating $F(x)$ with respect to x yields

$$\frac{dF(x)}{dx} = 2Lx - \Lambda r \tag{3.11}$$

Thus, the problem of finding the minimal x (minimal partition) reduces to solving the linear system

$$2Lx = \Lambda r. \tag{3.12}$$

Henceforth, we ignore the scalar multiplier 2 and the scalar Λ since, as we will see later, we are only concerned with the relative values of the solution, rather than the actual values.

The matrix L is singular: all rows and columns sum to zero (i.e., the vector r spans its nullspace), so finding a unique solution to equation (3.12) requires an additional constraint.

We assume that the graph is connected, since the optimal partitions are clearly each connected component (i.e., $h(x) = h_G = 0$) if the graph is disconnected. A linear time breath-first search may be performed to check for connectivity of the graph. Note that in general, a graph with c connected components will correspond to a matrix L with rank $(n-c)$ (Biggs, 1974). If we arbitrarily designate a node, v_g to include in S (i.e., fix $x_g = 0$), this is reflected in (3.12) by removing the gth row and column of L, and the gth row of x and r such that

$$L_0 x_0 = r_0. \tag{3.13}$$

where L_0 indicates the Laplacian with a row/column removed, x_0 is the vector x with the corresponding removed entry and r_0 is r with the removed row. Removal of a column of the incidence matrix A, yields a matrix A_0 referred to as the **reduced incidence matrix** (Foulds, 1992), and $A_0^T C A_0 = L_0$. Note that (3.13) is a nonsingular system of equations.

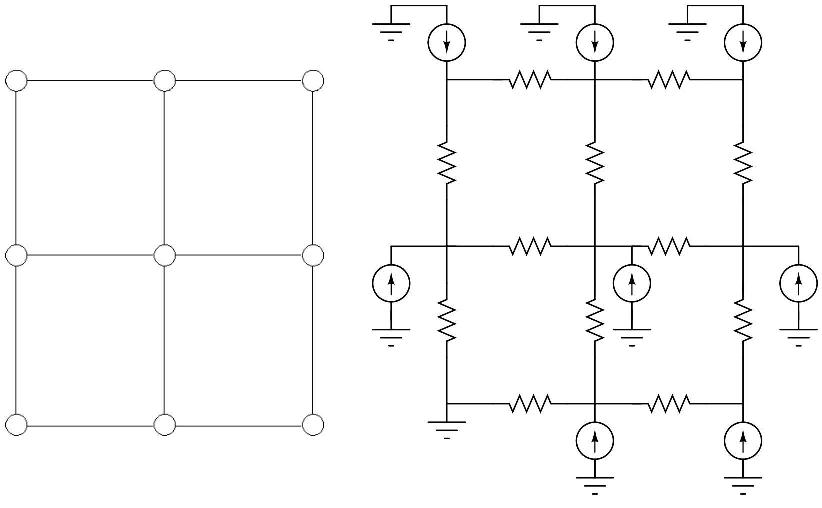


Figure 3·1: An example of a simple graph (a), and its equivalent circuit (b). Solving (3.13) (using the node in the lower left as ground) for the graph depicted in (a) is equivalent to connecting (b) and measuring the potential values at each node.

3.2.1 Circuit analogy

The three fundamental equations of circuit theory (2.3) may be combined into the linear system

$$A_0^T C A_0 x = L_0 x = f, \tag{3.14}$$

since $A^T C A = L$ (Biggs, 1974).

In other words, the solution to equation (3.13) in the isoperimetric algorithm is provided by the steady state of a circuit where each edge has a conductance equal to the edge weight and each node is supplied by a unit current source. The potentials that are established on the nodes of this circuit are exactly those which are being solved for in equation (3.13). An example of this equivalent circuit is displayed in Figure 3·1. By analogy then, we refer to the node, v_g, for which we set $x_g = 0$ as the **ground node**. Likewise, the solution, x_i, obtained from equation (3.13) at node v_i will be referred to as the **potential** for node v_i. The need for fixing an $x_g = 0$ to constrain equation (3.12) may be seen not only from the necessity of grounding a circuit powered only by current sources in order to find unique potentials, but also from the need to provide a boundary condition in order to find a solution to Laplace's equation, of which (3.12) is a combinatorial analog. In our case, the "boundary condition" is that the grounded node is fixed to zero.

One final remark on the circuit analogy to (3.13) follows from recalling Maxwell's principle of least dissipation of power: A circuit with minimal power dissipation provides a solution to Kirchhoff's current and voltage laws (Maxwell, 1991a). Explicitly, solving (3.13) for x is equivalent to solving the dual equation for $y = CAx$. The power of the equivalent circuit is $P = I^2R = y^TC^{-1}y$ subject to the constraint from Kirchhoff's law that $A^Ty = f$. Therefore, the y found by $y = CAx$ also minimizes the above expression for y (Strang, 1986; Baak, 1998). Thus, our approach to minimizing the combinatorial isoperimetric ratio is identical to minimizing the power of the equivalent electrical circuit with the specified current sources and ground (Strang, 1986).

3.2.2 Alternative derivation of algorithm

Suppose we wish to partition a graph by trying to find all the "bottleneck" edges that separate a reference node, v_g, from all the other nodes and set the partitions as those on either side of the bottleneck edges. Define a **path vector** as an $m \times 1$ vector with ± 1 in the entries for edges along the path (according to orientation of the edge). Define a **usage vector** as an $m \times 1$ vector whose entries give the number of times the corresponding edge appears in a path from a particular node to ground. We may propose the following procedure for finding the "bottleneck" edges in a graph and converting this identification into a partition:

1. Choose a ground point, v_g.
2. For each other node, find all paths from that node to ground.
3. For each of these paths, build the corresponding path vector.
4. Sum up the set of path vectors to form a usage vector for this node, and normalize by

the number of paths (to allow comparison with other nodes that may have a different number of paths).

5. Sum the usage vectors across all non-ground nodes to get a global usage vector.
6. Treat the global usage vector as a set of "potential drops" across edges and find the least-squares solution for the potentials.
7. Cut the graph based on the node potentials.

Although this may appear computationally expensive, we show in this section that this idea leads us to the same equations derived above for the isoperimetric problem.

An undirected graph is simply assigned arbitrary orientations to the edges. Orientation of the edges should not be confused with the notion of a **directed graph**. A directed graph only permits flow between nodes in accordance with the orientation of the edges. In contrast, the arbitrary orientation assigned above still permits flow in either direction, but changes the sign of the flow depending on whether or not the flow is in accordance with the orientation.

Each row of the edge-node incidence matrix of (2.2), A, sums to zero (since each edge has a single initial node and a single terminal node), meaning that the column space is not full rank. This issue is avoided (for a connected graph) by removing one column of A, forming the reduced incidence matrix, A_0 (Foulds, 1992).

For each non-ground node, v_i, find the set of all paths, T_i, from that node to ground. Formally, a path vector, t_i, is defined

$$t_k = \begin{cases} \pm 1 & \text{if } e_{ij} \text{ is in the path to ground,} \\ 0 & \text{else.} \end{cases} \tag{3.15}$$

For s_i number of paths from node v_i to ground, $T_i = \{t_1, t_2, \ldots, t_{s_i}\}$. Formally define the usage vector as

$$u_i = \frac{1}{s_i} \sum_{t \in T_i} t. \tag{3.16}$$

Now, v_i, sum the individual usage vectors, u_i, to produce the global usage vector, p,

$$p = \sum_i u_i. \tag{3.17}$$

In order to solve for the least-squares solution to the potentials, x, given the "potential drops" of p, one solves the equation

$$A_0 x = p. \tag{3.18}$$

Now, one may multiply both sides by the matrix A_0^T to yield

$$\begin{aligned} A_0^T A_0 x &= A_0^T p, \\ A_0^T A_0 x &= \sum_i A_0^T u_i, \\ A_0^T A_0 x &= \frac{1}{s_i} \sum_i \sum_{t \in T_i} A_0^T t. \end{aligned}$$

However, since t is a path from node v_i to ground, the result of $A_0^T t$ is a vector, q, with the value

$$q_k = \begin{cases} 1 & \text{if } k = i, \\ 0 & \text{else.} \end{cases} \tag{3.19}$$

One way to interpret this result is by considering the matrix A^T as the discrete divergence (Rockafellar, 1984). The divergence of a path from a node v_i to ground yields one source (i.e., node v_i) and one sink (i.e., node v_g), but the sink is obscured since we are using the reduced incidence matrix (Foulds, 1992).

It may be seen that

$$\begin{aligned} A_0^T A_0 x &= \frac{1}{s_i} \sum_i \sum_{t \in T_i} A_0^T t, \\ A_0^T A_0 x &= \frac{1}{s_i} \sum_i \sum_{t \in T_i} q, \\ A_0^T A_0 x &= \frac{1}{s_i} \sum_i s_i q, \\ A_0^T A_0 x &= r_0, \end{aligned} \tag{3.20}$$

where r_0 denotes the reduced vector of all ones. Now, it simply remains to be noted that the left hand side $A_0^T A_0 = L_0$ (Biggs, 1974; Strang, 1986) (with uniform weights) to see the relationship between this section and the original derivation. In other words, equation (3.20) is the same as the original derivation, given by equation (3.13).

If a weighted graph is used instead, the potential drop across each edge is inversely biased by its weight (i.e., smaller weights induce a larger potential drop). In other words, build a diagonal $m \times m$ matrix, C that has the edge weights on the diagonal. Then, modify equation (3.18) to be $CA_0x = p$. Now, since $A_0^T C A_0 = L_0$ for arbitrary weights, then the result of (3.20) is the same as (3.13).

3.2.3 Choosing a ground

Although the choice of a ground node may roughly correspond to an "attention" point in the context of image segmentation (see Chapter 4), there are several possible strategies for choosing the ground node in the general case. Anderson and Morley (1971) proved that the spectral radius of L, $\rho(L)$, satisfies $\rho(L) \leq 2d_{\max}$, suggesting that grounding the node of maximum degree may have the most beneficial effect on the conditioning of equation (3.13). In the comparison section of this chapter, we employ three grounding strategies. We ground the node with maximum degree, the node with minimum degree and choose the best of three random nodes.

3.2.4 Solving the System of Equations

Solving (3.13) is the computational core of the algorithm. It requires the solution to a large sparse system of symmetric equations where the number of nonzero entries in L will equal $2m$.

Methods for solving a system of equation fall generally into two categories: direct and iterative methods (Golub and Van Loan, 1996; Hackbusch, 1994; Fiedler, 1986). The former are generally based on Gaussian elimination with partial pivoting while for the latter, the method of conjugate gradients is arguably the best approach. Iterative procedures have the advantage that a partial answer may be obtained at intermediate stages of the solution by specifying a limit on the number of iterations allowed. This feature allows one to trade speed for accuracy in finding a solution. An additional feature of using the method of conjugate gradients to solve equation (3.13) is that conjugate gradients is efficiently parallelized (Dongarra et al., 1991; Gremban, 1996). In this work, we used the sparse matrix package in $^{\text{TM}}$MATLAB (Gilbert et al., 1992) to find direct solutions.

3.2.5 Converting the solution to a partition

The binary definition of x was extended to the real numbers in order to solve (3.13). Therefore, in order to convert the solution, x, to a partition, a subsequent step must be applied (as with spectral partitioning). Conversion of a potential vector to a partition may be accomplished using a threshold. A **cut value** is a value, α, such that $S = \{v_i | x_i \leq \alpha\}$ and $\overline{S} = \{v_i | x_i > \alpha\}$. The partitioning of S and $\overline{S}$ in this way may be referred to as a **cut**. This thresholding operation creates a partition from the potential vector, x. Note that since a connected graph corresponds to an L_0 that is an M-matrix (Fiedler, 1986), and is therefore monotone, $L_0^{-1} \geq 0$. This result then implies that $x_0 = L_0^{-1} d_0 \geq 0$.

Employing the terminology of Spielman and Teng (1996), the standard approaches to cutting the indicator vector in spectral partitioning are to cut based on the median value (the **median cut**) or to choose a threshold such that the resulting partitions have the lowest available isoperimetric ratio (the **ratio cut**). The ratio cut method will clearly produce partitions with a lower isoperimetric ratio than the median cut. Unfortunately, because of the required sorting of x, the ratio cut method requires $\mathcal{O}(n \log(n))$ operations (assuming a bounded degree). The median cut method runs in $\mathcal{O}(n)$ time, but forces the algorithm to produce equal sized partitions, even if a better cut could be realized elsewhere. Despite the required sorting operation for the ratio cut, the operation is still very inexpensive relative to the solution of equation (3.13) for the range of n we focus on (typically a few hundred thousand nodes).

3.2.6 Summary

Therefore, the isoperimetric algorithm for partitioning a graph may be summarized by

1. Choose a ground node, v_g.
2. Solve (3.13).
3. Cut x using the method of choice to obtain S and $\overline{S}$.

3.2.7 Time Complexity

Running time depends mainly on the solution to equation (3.13). A sparse matrix-vector operation depends on the number of nonzero values, which is, in this case, $\mathcal{O}(m)$. If we may assume a constant number of iterations required for the convergence of the conjugate gradients method, the time complexity of solving (3.13) is $\mathcal{O}(m)$. Cutting the potential vector with the ratio cut requires a $\mathcal{O}(n \log(n))$ sort. Combined, the time complexity is $\mathcal{O}(m + n \log n)$. In the case of graphs with bounded degree, then $m \leq n d_{\max}$ and the time complexity reduces to $\mathcal{O}(n \log(n))$.

3.3 Some formal properties of the algorithm

3.3.1 Connectivity

In this section, we prove that the partition containing the grounded node (i.e., the set S) must be connected, regardless of how a threshold (i.e., cut) is chosen. The strategy for establishing this will be to show that every node has a path to ground with a monotonically decreasing potential.

Proposition 1. *If the set of vertices, V, is connected then, for any α, the subgraph with vertex set $N \subseteq V$ defined by $N = \{v_i \in V | x_i < \alpha\}$ is connected when x_0 satisfies $L_0 x_0 = f_0$ for any $f_0 \geq 0$.*

This proposition follows directly from proof of the following

Lemma 1. *For every node, v_i, there exists a path to the ground node, v_g, defined by $P_i = \{v_i, v^1, v^2, \ldots, v_g\}$ such that $x_i \geq x^1 \geq x^2 \geq \ldots \geq 0$, when $L_0 x_0 = f_0$ for any $f_0 \geq 0$.*

Proof. By equation (3.13) each non-grounded node assumes a potential

$$x_i = \frac{1}{d_i} \sum_{e_{ij} \in E} x_j + \frac{f_i}{d_i}, \tag{3.21}$$

i.e., the potential of each non-grounded node is equal to a nonnegative constant added to the (weighted) average potential of its neighbors. Note that (3.21) is a combinatorial formulation of the Mean Value Theorem (Ahlfors, 1966) in the presence of sources.

For any connected subset, $S \subseteq V, v_g \notin S$, denote the set of nodes on the boundary of S as $S_b \subset V$, such that $S_b = \{v_i | \, e_{ij} \in E, \, \exists \, v_j \in S, \, v_i \notin S\}$.

Now, either

1. $v_g \in S_b$, or
2. $\exists \, v_i \in S_b$, such that $x_i \leq \min x_j, \, \forall \, v_j \in S$ by (3.21), since the graph is connected.

By induction, every node has a path to ground with a monotonically decreasing potential, (i.e., start with $S = \{v_i\}$ and add nodes with a nondecreasing potential until ground is reached). □

3.3.2 Spanning trees

Since a tree will have a unique path from any node to ground, Lemma 1 guarantees that the nodes in this path will have a nonincreasing potential. However, since a tree is a special case of a graph (e.g., the reduced incidence matrix is square), there is an alternate derivation of this result. A theorem by Branin (1963) shows that for node v_k, edge e_{ij} and ground v_g, the inverse of the reduced incidence matrix for a spanning tree, $B = A_0^{-1}$, is

$$B_{v_k e_{ij}} = \begin{cases} +1 & \text{if } e_{ij} \text{ is positively traversed in the path from } v_k \text{ to } v_g, \\ -1 & \text{if } e_{ij} \text{ is negatively traversed in the path from } v_k \text{ to } v_g, \\ 0 & \text{otherwise.} \end{cases} \tag{3.22}$$

Therefore,

$$L_0^{-1} = (A_0^T C A_0)^{-1} = B^T C^{-1} B. \tag{3.23}$$

Each value of L_0^{-1} may therefore be interpreted as the sum of the reciprocal weights of shared edges along the unique path to v_g between nodes v_i and v_j i.e., the shared *distance* of the unique paths from v_i and v_j to v_g in the metric interpretation.

It follows that the potential values taken by x_0 in $x_0 = L_0^{-1} f_0$ are monotonically increasing along the path from v_g to any other node for nonnegative f_0 and C.

3.3.3 Fully connected graphs

The isoperimetric algorithm will produce a uniform solution to equation (3.13) when applied to fully connected graphs with uniform weights. Any set with cardinality equal to half the cardinality of the vertex set is a solution to the isoperimetric problem for a fully connected graph with uniform weights. For a uniform edge weight, $w(e_{ij}) = \kappa$ for all $e_{ij} \in E$, the solution, x_0, to equation (3.13) will be $x_i = 1/\kappa$ for all $v_i \in V$. The use of the median or ratio cut method will choose half of the nodes arbitrarily. Although it should be pointed out that using a median or ratio cut to partition a vector of randomly assigned potentials will also produce equal sized (in the case optimal) partitions, the solution to equation (3.13) is unique for a specified ground (in contrast to spectral partitioning, which has $n-1$ solutions) and explicitly gives no preference to a node since all the potentials are equal.

3.4 Review of previous work

Since the isoperimetric graph partitioning algorithm is a *global* partitioning algorithm (i.e., it inputs a graph and outputs a partition (Preis and Diekmann, 1996)), this section will only review other global graph partitioning algorithms.

3.4.1 Spectral Partitioning

Building on the early work of Fiedler (1975a,b, 1973), Alon and Milman (1985); Alon (1986) and Cheeger (1970) who demonstrated the relationship between the second smallest eigenvalue of the Laplacian matrix (the **Fiedler value**) for a graph and its isoperimetric number, spectral partitioning was one of the first graph partitioning algorithms to be successful (Donath and Hoffman, 1972; Pothen et al., 1990). The algorithm partitions a graph by finding the eigenvector corresponding to the Fiedler value, termed the **Fiedler vector** and cutting the graph based on the value in the Fiedler vector associated with each node. Physically, the Fiedler vector corresponds to the second harmonic of a surface. Like isoperimetric partitioning, the output of spectral partitioning is a set of values assigned to each node, which allows a cut to be a perfect bisection by choosing a zero threshold (the median cut) or by choosing the threshold that generates a partition with the best isoperimetric ratio (the ratio cut). The flexibility of spectral partitioning allows it to be used as a part of hybridized graph partitioning algorithms, such as geometric-spectral partitioning (Chan et al., 1994) and multilevel schemes (Karypis and Kumar, 1998).

Spectral partitioning attempts to minimize the isoperimetric ratio of a partition by solving

$$Lz = \lambda z \tag{3.24}$$

with L defined as above and λ representing the Fiedler value. Since the vector of all ones, r, is an eigenvector corresponding to the smallest eigenvalue (zero) of L, the goal is to find the eigenvector associated with the second smallest eigenvalue of L. Requiring $z^T r = 0$ and $z^T z = n$ may be viewed as additional constraints employed in the derivation of spectral partitioning to circumvent the singularity of L (for an explicit formulation of spectral partitioning from this viewpoint, see Hu and Blake (1994)). Therefore, one way of viewing the difference between the isoperimetric and the spectral methods is in the choice of the additional constraint that regularizes the singular nature of the Laplacian L.

In the context of spectral partitioning, the indicator vector z is usually defined as

$$z_i = \begin{cases} -1 & \text{if } v_i \in \overline{S}, \\ +1 & \text{if } v_i \in S, \end{cases} \tag{3.25}$$

such that z is orthogonal to r for an equal sized partition. The two definitions of the indicator vector (equations (3.6) and (3.25)) are related through $x = \frac{1}{2}(z + r)$. Since r is in the nullspace of L, the definitions are equivalent up to a scaling.

Simple, unhyrbridized, unilevel spectral partitioning lags behind modern multilevel algorithms (e.g., Karypis and Kumar (1998)) in terms of partition quality. However, compared against other global graph partitioning algorithms, spectral partitioning still performs well. The major drawbacks of spectral partitioning are its speed and numerical stability. Even using the Lanczos algorithm (Golub and Van Loan, 1996) to find the Fiedler vector for a sparse matrix, spectral partitioning is still much slower than many other partitioning algorithms. Furthermore, the Lanczos algorithm becomes unstable as the Fiedler value approaches its neighboring eigenvalues (see Hendrickson and Leland (1995); Golub and Van Loan (1996) for discussion of this problem). In fact, the eigenvector problem becomes fully degenerate if the Fiedler value assumes algebraic multiplicity greater than one. For example, consider finding the Fiedler vector of a fully connected graph, for which the Fiedler values has algebraic multiplicity equal to $n - 1$. This situation could allow the Lanczos algorithm to converge to any vector in the subspace spanned by the eigenvectors corresponding to the Fiedler value. Finally, it has been pointed out by Guattery and Miller (1998) that classes of graphs exists for which spectral partitioning will produce consistently poor partitions.

3.4.2 Geometric Partitioning

Geometric partitioning (Gilbert et al., 1998) is only defined for graphs with nodal coordinates specified (e.g., finite-element meshes). Furthermore, the geometric partitioning algorithm assumes that the nodes are locally connected and computes a good *spatial* separator (i.e., ignoring topology altogether). Building on theoretical results of Miller et al. (1998, 1993), geometric partitioning works by stereographically projecting nodes from the plane to the Riemann sphere, conformally mapping the nodes on the sphere such that a special point (called the **centerpoint**) is at the origin and then randomly choosing any great circle on the Riemann sphere to divide the points into two equal halves. In practice, several great circles are randomly chosen and the best one is used as the output.

Although geometric partitioning is computationally inexpensive (albeit using multiple trials) and produces good partitions, the major drawback of the geometric partitioning algorithm is its inapplicability to graphs without coordinates or to graphs that are not locally connected (e.g., nonplanar graphs).

3.4.3 Geometric-Spectral Partitioning

Geometric-spectral partitioning (Chan et al., 1994) combines elements of both spectral partitioning and geometric partitioning. By finding the eigenvectors corresponding to both the second and third smallest eigenvalues, one may treat the nodes as having **spectral coordinates** by viewing the values associated with each node in the two eigenvectors as two-dimensional coordinates in the plane. Applying geometric partitioning to the spectral coordinates of the nodes instead of actual coordinates (if available) heuristically gives better partitions than either spectral or geometric partitioning alone.

Geometric-spectral partitioning represents a more general algorithm than straightforward geometric partitioning since it applies to more general graphs (i.e., graphs without coordinates) and because it takes advantage of topological information (in computing the spectral coordinates). However, geometric-spectral partitioning is more computationally expensive than spectral partitioning since it requires the computation of two eigenvectors instead of one. Numerically, the algorithm is also prone to more numerical problems than spectral partitioning, since the Lanczos algorithm has increased error as more "interior" eigenvectors are computed (Golub and Van Loan, 1996), and because the spectral coordinates are not unique if either the second or the third eigenvalue of L has an algebraic multiplicity greater than one.

3.4.4 Inertial Partitioning

Like geometric partitioning, the inertial method (Hendrickson and Leland, 1995) requires coordinates for each node and ignores topological considerations. Treating each node as a point mass embedded in the plane, the principle inertial axis is computed. Nodes on opposite sides of a hyperplane orthogonal to this axis are considered to be in different partitions. This method is fast and, for planar graphs, produces reasonable partitions.

3.4.5 Coordinate Partitioning

A partition of the vertex set may be achieved by placing the nodes on opposite sides of a hyperplane orthogonal to a coordinate axis into different partitions. The coordinate partitioning algorithm performs this partitioning for each coordinate axis and chooses the best partition. Like geometric and inertial partitioning, coordinate partitioning requires coordinate information for the node set. This method is very fast, but tends to produce poor partitions.

3.5 Comparison of algorithms

Three versions of the isoperimetric algorithm are compared with the algorithms reviewed above. The three versions are obtained from three grounding strategies: grounding the maximum degree node, the minimum degree node and a random node. Grounding strategies are denoted by "MG" for maximum degree ground, "MnG" for minimum degree ground and "RG" for random ground. For each random ground partition, three randomly chosen grounds were tried and the best one chosen. In order to give a notion of how "breakable" the graphs in question are, a purely random partition is also included. The geometric, geometric-spectral, inertial and coordinate partitions were all obtained through the MESHPART software package for $^{\text{TM}}$MATLAB (Gilbert et al., 1998).

The algorithms were tested on weighted and unweighted planar, nonplanar and three-dimensional graphs, as well as some special graphs. All algorithms were compared by considering the number of edges spanning the two equally sized partitions produced by the algorithm (i.e., the median cut, for the isoperimetric and spectral algorithms). The spectral and isoperimetric algorithms were additionally compared in terms of the isoperimetric ratio by employing the ratio cut to find the best partition (i.e., not necessarily partitions of equal size). As noted by Schreiber and Martin (1999), we found that the distribution of spanning edges and $h(x)$ resembled Gaussian distributions. For this reason, we only report the mean and variance of each.

3.5.1 Planar graphs

Planar graphs were generated by uniformly sampling one thousand points from a two-dimensional unit square and connecting them with a Delaunay triangulation. One thousand such graphs were generated for both the weighted and unweighted trials. In the unweighted trial, weights were randomly assigned to each edge from a Gaussian distribution with a variance of $10,000$ and mean adjusted such that all weights were positive. Results for the median cut comparison are found in Table 3.1 and for the ratio cut comparison are found in Table 3.2.

	Unweighted graphs		Weighted graphs	
Algorithm	Mean Cut	Variance Cut	Mean Cut	Variance Cut
Iso MG	95.7	59.3	3.26×10^6	42.9
Iso RG	93.5	38.3	3.08×10^6	8.47
Iso MnG	92.4	40	3.3×10^6	8.32
Spectral	86.6	20.7	2.94×10^6	7.86
Geometric	87.4	8.71	3.05×10^6	4.57
Geo-Spec	82.5	4.06	2.9×10^6	8.2
Inertial	97.2	70.2	3.36×10^6	16.3
Coordinate	94.2	49.1	3.23×10^6	13.6
Random	$1,490$	$1,120$	5.07×10^7	$1,090$

Table 3.1: Comparison of the algorithms on $1,000$ randomly generated planar graphs produced by uniformly sampling $1,000$ two-dimensional points in the unit square and connecting via a Delaunay triangulation. The leftmost two columns represent the mean and variance of $|\partial S|$ for equal sized partitions produced by each algorithm when applied to unweighted planar graphs. The rightmost two columns represent the same quantities for weighted planar graphs.

3.5.2 Nonplanar graphs

Purely random graphs (typically nonplanar) were generated by connecting each pair of $1,000$ nodes with a 1% probability. One thousand such graphs were generated for both the weighted and unweighted trials. In the unweighted trial, weights were randomly assigned to each edge from a Gaussian distribution with a variance of $10,000$ and mean adjusted such

	Unweighted graphs		Weighted graphs	
Algorithm	Mean $h(x)$	Variance $h(x)$	Mean $h(x)$	Variance $h(x)$
Iso MG	0.185	2.0×10^{-4}	6.21×10^{3}	1.6×10^{6}
Iso RG	0.177	9.64×10^{-5}	5.71×10^{3}	1.73×10^{5}
Iso MnG	0.181	$1.2 \times 10-4$	6.36×10^{3}	1.96×10^{5}
Spectral	0.171	8.41×10^{-5}	5.66×10^{3}	2.59×10^{5}

Table 3.2: Comparison of the isoperimetric and spectral algorithms on $1,000$ randomly generated planar graphs produced by uniformly sampling $1,000$ two-dimensional points in the unit square and connecting via a Delaunay triangulation. The leftmost two columns represent the mean and variance of $h(x)$ obtained by applying the ratio cut to the output of each algorithm when applied to unweighted planar graphs. The rightmost two columns represent the same quantities for weighted planar graphs.

that all weights were positive. Since coordinate information was meaningless, only those algorithms which made use of topological information were compared (i.e., the isoperimetric, spectral and geometric-spectral algorithms). Results for the median cut comparison are found in Table 3.3 and for the ratio cut comparison in Table 3.4.

	Unweighted graphs		Weighted graphs	
Algorithm	Mean Cut	Variance Cut	Mean Cut	Variance Cut
Isoperimetric MG	4.46×10^{3}	6.84×10^{3}	1.7×10^{8}	2.24×10^{4}
Isoperimetric RG	4.14×10^{3}	5.86×10^{3}	1.6×10^{8}	1.86×10^{4}
Isoperimetric MnG	4.16×10^{3}	4.65×10^{3}	1.6×10^{8}	1.86×10^{4}
Spectral	4.06×10^{3}	5.98×10^{3}	1.57×10^{8}	1.77×10^{4}
Geometric-Spectral	3.98×10^{3}	3.93×10^{3}	1.54×10^{8}	1.69×10^{4}
Random	4.97×10^{3}	4.88×10^{3}	1.92×10^{8}	2.64×10^{4}

Table 3.3: Comparison of the algorithms on $1,000$ random graphs (symmetric) produced by connecting each edge with a 1% probability. Partitioning algorithms that rely on coordinate information were not included, since they are not applicable to this problem. The leftmost two columns represent the mean and variance of $|\partial S|$ for an equal sized bipartition produced by each algorithm for unweighted random graphs. The rightmost two columns represent the same quantities for weighted random graphs.

3.5.3 Three-dimensional graphs

Modeling and computer graphics applications frequently require the use of locally connected points in a three dimensional space. One thousand graphs were generated by uniformly

	Unweighted graphs		Weighted graphs	
Algorithm	Mean $h(x)$	Variance $h(x)$	Mean $h(x)$	Variance $h(x)$
Isoperimetric MG	4.73	0.712	1.78×10^5	1.25×10^9
Isoperimetric RG	4.27	0.441	1.57×10^5	8.45×10^8
Isoperimetric MnG	5.17	0.633	1.92×10^5	1.15×10^9
Spectral	5.6	1.86	2.17×10^5	3.22×10^9

Table 3.4: Comparison of the algorithms on 1,000 random graphs (symmetric) produced by connecting each edge with a 1% probability. The leftmost two columns represent the mean and variance of $h(x)$ for a partition generated using the ratio cut for random graphs. The rightmost two columns represent the same quantities for weighted random graphs.

sampling 1,000 points in the unit cube and connecting them via the three-dimensional Delaunay. Both weighted and unweighted graphs were generated. In the unweighted trial, weights were randomly assigned to each edge from a Gaussian distribution with a variance of 10,000 and mean adjusted such that all weights were positive. Results for the median cut comparison are found in Table 3.5 and for the ratio cut comparison in Table 3.6.

	Unweighted graphs		Weighted graphs	
Algorithm	Mean Cut	Variance Cut	Mean Cut	Variance Cut
Isoperimetric MG	3.04×10^3	6×10^4	1.27×10^8	1.87×10^4
Isoperimetric RG	3.09×10^3	2.84×10^4	1.29×10^8	1.24×10^4
Isoperimetric MnG	3.25×10^3	3.23×10^4	1.36×10^8	1.54×10^4
Spectral	3.01×10^3	3.44×10^4	1.26×10^8	1.38×10^4
Geometric	2.82×10^3	7.48×10^3	1.19×10^8	8.25×10^3
Geometric-Spectral	2.74×10^3	1.15×10^4	1.15×10^8	8.8×10^3
Inertial	3.2×10^3	3.81×10^4	1.34×10^8	1.56×10^4
Coordinate	2.8×10^3	7.86×10^3	1.17×10^8	8.13×10^3
Random	1.9×10^4	6.66×10^4	7.99×10^8	3.18×10^5

Table 3.5: Comparison of the algorithms on 1,000 randomly generated three-dimensional graphs produced by uniformly sampling 1,000 three-dimensional points in the unit cube and connecting via a three-dimensional Delaunay. The leftmost two columns represent the mean and variance of $|\partial S|$ for an equal sized bipartition produced by each algorithm for unweighted planar graphs. The rightmost two columns represent the same quantities for weighted planar graphs.

	Unweighted graphs		Weighted graphs	
Algorithm	Mean $h(x)$	Variance $h(x)$	Mean $h(x)$	Variance $h(x)$
Isoperimetric MG	6.06	0.261	2.53×10^5	6.91×10^8
Isoperimetric RG	6.12	0.105	2.55×10^5	4.89×10^8
Isoperimetric MnG	6.5	0.134	2.71×10^5	5.5×10^8
Spectral	5.99	0.142	2.51×10^5	5.77×10^8

Table 3.6: Comparison of the isoperimetric and spectral algorithms on $1,000$ randomly generated three-dimensional graphs produced by uniformly sampling $1,000$ three-dimensional points in the unit cube and connecting via a three-dimensional Delaunay. The leftmost two columns represent the mean and variance of $h(x)$ obtained using the ratio cut for each algorithm on unweighted planar graphs. The rightmost two columns represent the same quantities for weighted planar graphs.

3.5.4 Special graphs

In addition to the randomly generated graphs used above to benchmark the isoperimetric algorithm, we also applied the set of algorithms to two-dimensional graphs taken from applications (see Gilbert et al. (1998); Chan et al. (1994); Cao et al. (1996) for other uses of these graphs). The meshes were obtained through the FTP site of John Gilbert and the Xerox Corporation at `ftp.parc.xerox.com` from the file `/pub/gilbert/meshpart.uu`. The list of graphs used is given in Table 3.7. The results using the median cut comparison are found in Table 3.8 and for the ratio cut comparison are found in Table 3.9. Random partitions were not included here, since only one trial was performed (i.e., there was no randomness in the graph generation).

The set of algorithms were also applied to three graph families of theoretical interest, for which standard algorithms are known to produce poor partitions. The first of these is the "roach" graph of Guattery and Miller (1998) with the total length of the roach ranging from 10 to 50 nodes long (i.e., 20 to 100 nodes total). The family of roach graphs is known to result in poor partitions when spectral partitioning is employed with the median cut. For a roach with an equal number of "body" and "antennae" segments, the spectral algorithm will always produce a partition with $|\partial S| = \Theta(n)$ (where $\Theta(\cdot)$ is the function of Knuth (1976)) instead of the constant cut set of two edges obtained by cutting the antennae from

Mesh Name	Nodes	Edges
Eppstein	547	1556
Tapir	1024	2846
“Crack”	136	354
Airfoil1	4253	12289
Airfoil2	4720	13722
Airfoil3	15606	45878
Spiral	1200	3191
Triangle	5050	14850

Table 3.7: Information about the special graphs used to benchmark the algorithms on. All graphs were obtained from the ftp site of John Gilbert and the Xerox Corporation at `ftp.parc.xerox.com` in file `/pub/gilbert/meshpart.uu`

.

Algorithm	Epp	Tapir	Tri	Air1	Air2	Air3	Crack	Spiral
Iso MG	24.8	51	150	137	111	255	200	9
Iso RG	20.3	34	152	100	111	194	158	9
Iso MnG	25.6	51	150	107	199	446	149	9
Spectral	26.6	58	152	132	117	194	157	9
Geometric	22	37	152	98	102	152	190	65
Geom-Spec	21.8	25	144	93	113	157	146	9
Inertial	23.8	49	142	94	209	245	268	83
Coordinate	23	55	142	94	172	230	231	59

Table 3.8: The number of edges cut by the equal-sized partitions output by the various algorithms.

Algorithm	Epp	Tapir	Tri	Air1	Air2	Air3	Crack	Spiral
Iso MG	0.0906	0.0474	0.0588	0.0351	0.0425	0.0268	0.0758	0.0108
Iso RG	0.0657	0.0262	0.0592	0.0347	0.0443	0.0275	0.0613	0.0108
Iso MnG	0.0936	0.0262	0.0588	0.037	0.0705	0.0477	0.0576	0.0108
Spectral	0.0809	0.0262	0.0598	0.0342	0.0425	0.02	0.059	0.0108

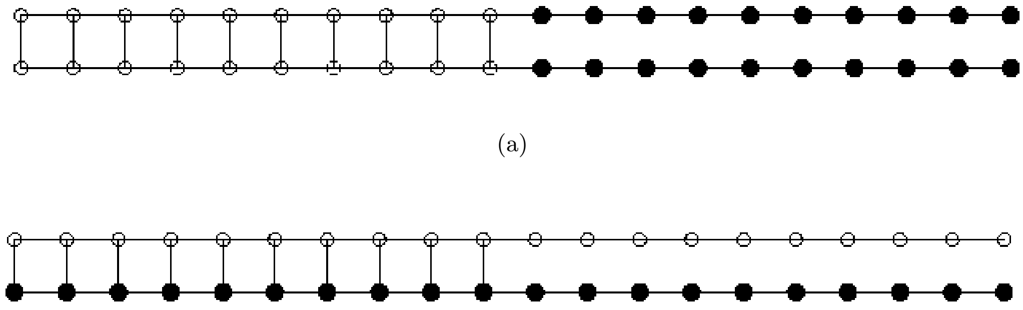
Table 3.9: The minimum isoperimetric ratio obtained by applying the ratio cut method to the output of the isoperimetric and spectral partitioning algorithms.

Algorithm	Mean Cut Roach	Mean Cut TreeXPath	Mean Cut “Badmesh”
Iso MG	2	27.1	2.98
Iso RG	2.44	27.4	3.17
Iso MnG	3.34	27.6	3.63
Spectral	14.8	36.3	12.1
Geometric	2	26.6	449
Geom-Spec	2	27	2.98
Inertial	2	28.7	513
Coordinate	2	26.2	513
Random	36.8	505	$1,050$

Table 3.10: The mean number of edges cut by the equal-sized partitions output by the various algorithms over a parameter range for each family of graphs (see text for details). The three graph families here (roach, treeXPath and “badmesh”) are of theoretical interest in that they are known to produce poor results for different classes of partitioning algorithms.

Algorithm	Mean $h(x)$ Roach	Mean $h(x)$ TreeXPath	Mean $h(x)$ “Badmesh”
Iso MG	0.0815	0.128	0.0297
Iso RG	0.0798	0.128	0.0295
Iso MnG	0.0796	0.127	0.0294
Spectral	0.0796	0.128	0.0294

Table 3.11: The mean isoperimetric ratio obtained using a ratio cut on the output of the partitioning algorithms when applied to three graph families of theoretical interest. Means are calculated over a range of parameters defining the graph family (see text for details).



(a)

(b)

Figure 3·2: The "roach" graph ($n = 20$) illustrated here is a member of a family of graphs for which spectral partitioning is known to fail to produce a partition with low isoperimetric ratio. Uniform weights were used for both algorithms. (a) Solution using isoperimetric algorithm. Ratio = 0.1. (b) Solution using spectral algorithm. Ratio = 0.5

the body. Spielman and Teng (1996) demonstrated that the spectral approach may be made to correctly partition the roach graph if additional processing is performed. The partitions obtained from the spectral and isoperimetric algorithms when applied to the roach graph are compared in Figure 3·2. The second graph family of theoretical interest, referred to as "TreeXPath", was also proposed by Guattery and Miller (1998), and is known to result in poor partitions when spectral partitioning is used with the median cut. For purposes of benchmarking here, the cardinality of the three-dimensional point set was varied between 50 and $1,000$ nodes. The final graph family of theoretical interest is the so-called "badmesh" of Cao et al. (1996) for which no quality straight-line separator exists. Badmeshes were generated with node sets varying between 200 and $4,000$, with a constant ratio of $\frac{4}{5}$ between shell sizes. Mean values across the nodal range are reported in Table 3.10 for partitions obtained with the median cut and Table 3.11 using the ratio cut.

3.6 Conclusion

We have presented a new algorithm for partitioning graphs. The results of our comparison with other global partitioning algorithms demonstrates that the isoperimetric algorithm produces similar quality outputs on the graph partitioning problem in terms of the mean and variance of the partition quality (i.e., either the number of edges cut or $h(x)$). The primary advantages of the isoperimetric algorithm are the ability to incorporate topological and weight information in finding a partition (i.e., coordinates are not required), the computational stability of always having a unique solution and the ability to find a partition of optimal size (i.e., the ratio cut). Spectral partitioning has similar characteristics, but requires the solution to the eigenvector problem instead of the solution to a sparse, symmetric, positive definite system of equations required by the isoperimetric algorithm. In our experience, this difference translates to a speed advantage for the isoperimetric algorithm of more than an order of magnitude over the spectral algorithm. Additionally the spectral algorithm suffers from a degeneracy not present in the isoperimetric algorithm for graphs with a Fiedler value corresponding to multiple eigenvectors. Furthermore, the isoperimetric algorithm finds good partitions for families of graphs which are known to cause poor results when certain graph partitioning schemes are applied. For example, the spectral partitioning algorithm is known to produce poor partitions when applied to the“roach” and “TreeXPath” graphs, while any algorithm that relies on coordinate information (e.g., geometric, inertial) produces poor partitions when applied to the “badmesh” family of graphs. In contrast, the isoperimetric algorithm does not suffer from the problems associated with these families. The isoperimetric algorithm also has an exact circuit analogy, which leaves open the possibility for analog computation e.g., with VLSI. Finally, a proof was given which guarantees that the grounded partition will always be connected.

Choosing the best partition resulting from a constant number of random grounds appears to be the most consistent method for finding a quality partition without increasing the time complexity of the algorithm. However, the partitions resulting from choosing the node of maximum degree do not appear, on average, to be drastically different than the best of three random grounds approach used to produce the above results. Therefore, if one wants to reduce computations by grounding only one node, the suggested heuristic of grounding the node of maximum degree appears to be reasonable.

In the next chapter, a modified, recursive version of the isoperimetric algorithm is applied to the problem of data clustering and image segmentation.

Chapter 4

Isoperimetric image segmentation

4.1 Introduction

The application of graph-theoretic methods to spatial pattern analysis has a long history, including the pioneering work of Zahn (1971) on minimal spanning tree clustering, the development of connectivity graph algorithms for space-variant sensors by Wallace et al. (1994), and the seminal work on image segmentation, termed "Ncuts" by Shi and Malik (2000). One reason for this interest is that the segmentation quality of Ncuts is very good. However, there are several other important advantages of graph-based sensor strategies, which we will now address.

4.1.1 Motivation for using graph-theoretic approaches in image processing

There are at least four distinct reasons to employ graph theoretic approaches to image segmentation:

1. *Local-global interactions* are well expressed by graph theoretic algorithms. Zahn (1971) used a minimal spanning tree on a weighted graph to illustrate Gestalt clustering methods. The term "Gestalt" derives from early theories of visual psychology which attempted to relate local and global features of visual stimuli in terms of "rules" which may be best described as simple variational principles (e.g., "best completion", etc.). Zahn's results were impressive for the time, coming at the very beginnings of modern image processing and clustering. The central reason for this success, we believe, is that the minimal spanning tree defines a minimizing principle (e.g., a tree of minimal edge weights) which respects the global structure of the problem set, allow-

ing a simple local rule (e.g., cut the graph at peak values of local density measure (Zahn, 1971)) to effectively cluster a set of feature vectors. Many graph-theoretic approaches involve the use of global and local information, as will be made explicit later in the present paper. As Zahn originally pointed out, the important notion of Gestalt in image processing—the relationship of the whole to the part—seems to be an important ingredient in both biological and machine image processing.

2. *New algorithms* for image processing may be crafted from the large corpus of well-explored algorithms which have been developed by graph theorists. For example, spectral graph partitioning was developed to aid in design automation of computers (Donath and Hoffman, 1972) and has become the basis for the Ncuts algorithm (Shi and Malik, 2000). Similarly, graph theoretic methods for solving lumped, Ohmic electrical circuits based on Kirchhoff's voltage and current law (Weyl, 1923; Roth, 1955; Branin, 1966; Strang, 1986), form the basis for the method proposed in Chapter 3 for solving the isoperimetric problem.

3. *Adaptive sampling* and space-variant vision require a "connectivity graph" approach to allow image processing on sensor architectures with space-variant visual sampling and the loss of the shift-invariant property of a lattice (Wallace et al., 1994). Space-variant architectures have been intensively investigated for application to computer vision for several decades (Wallace et al., 1994; Sandini et al., 1989) partly because they offer extraordinary data compression. A sensor (Sandini et al., 1989) employing a complex logarithmic visual sampling function can provide equivalent peak resolution to the workspace size of a constant resolution sensor with 10,000 times the pixel count (Rojer and Schwartz, 1990). However, even simple space-variant architectures provide significant challenges with regard to sensor topology and anti-aliasing. The connectivity graph was proposed by Wallace et al. (1994) as a general algorithmic framework for processing the data output from space-variant sensors. This architecture describes sensor pixels by a specific neighborhood connectivity as well as

geometric position. Using this approach, a variety of image processing algorithms were defined on an arbitrary pixel architecture, including spatial filtering via the graph Laplacian.

4. *New architectures for image processing* may be defined that generalize the traditional Cartesian design. The basic sampling scheme of visual sensor design has changed little since its origins in the 1930's. Vision sensors are generally based on fixed size pixels with fixed rate clocks (i.e., they are space-invariant and synchronous). The space-invariant pixel design, as noted above, can lead to a huge inefficiency when compared to a spatially adaptive sensor (e.g., a foveal architecture). A similar issue holds for temporal sampling, indicating that biological systems once again provide a counterexample to current engineering practice. Retinal sensors are not synchronous, but are based on an asynchronous "integrate and fire" temporal design. They integrate the locally available intensity, and fire when a fixed threshold is achieved. It is evident that this "just in time" strategy for temporal sampling will improve the average response speed of the ensemble of sensors. Presumably, a slight difference in response speed may translate to a significant difference in survival value. Just as in the spatial case, the temporal domain can (and does, in animals) exploit an adaptive, variable sampling strategy. In a computational context, this suggests the use of graph-theoretic data structures, rather than pixels and clocks. In turn, the flexible data structures based on graphs, which are familiar in computer graphics, have been relatively unexplored in computer vision. The structure and algorithms of graph theory provide a natural language for space-time adaptive sensors.

4.2 Modified isoperimetric algorithm

In Chapter 3 it was mentioned that different definitions of combinatorial volume exist. The Ncuts algorithm for image segmentation may be viewed as spectral partitioning with a different notion of volume, and consequent Laplacian matrix (compare the combinatorial Laplacian derived in Dodziuk (1984) and Dodziuk and Kendall (1986)). In fact, Shi and Malik (2000) state that spectral partitioning (called the "average cut") generates poor image segmentations compared with Ncuts, suggesting that the notion of combinatorial volume employed by Dodziuk and Kendall (1986) is more appropriate for image segmentation. In our experience, application of the isoperimetric algorithm with the notion of volume (Dodziuk, 1984) used in Chapter 3 produces lower quality image segmentations than a modified version of the isoperimetric algorithm that uses volume defined for a set of nodes, S, by Dodziuk and Kendall (1986)

$$\mathrm{Vol}_S = \sum_i d_i \;\; \forall v_i \in S. \tag{4.1}$$

The use of (4.1) to define volume, rather than (3.5), may be more appropriate for image segmentation, since a region of nodes with uniform intensity will be more likely to be a single object and register a higher volume, than a region of varying intensity that contains the same number of nodes. In other words, an algorithm attempting to minimize the isoperimetric ratio will be biased toward partitioning regions of uniform intensity rather than regions that simply include a large number of nodes. For this reason, we use the isoperimetric algorithm derived in Chapter 3 with a modified notion of volume to perform data clustering and image segmentation.

Employing the notion of volume in (4.1), gives an isoperimetric ratio for an indicator vector, x, as

$$h(x) = \frac{x^T L x}{x^T d}. \tag{4.2}$$

By substituting (4.1) for (3.5) and following the derivation for the isoperimetric partitioning algorithm, the new equation used to solve for the potentials (as opposed to (3.13) is

$$L_0 x_0 = d_0, \tag{4.3}$$

where d_0 is the vector of degrees, d, with the entry corresponding to d_g removed. In the derivation of the isoperimetric algorithm from bottleneck paths given in Chapter 3, (4.3) may be interpreted as representing the same procedure, where each node is weighted by its degree. Furthermore, the same circuit analogy exist for the modified isoperimetric algorithm as that outlined in Chapter 3, except that the unity current sources are replaced by current sources equaling the degree of each node.

Solving (3.13) for x_0 yields a real-valued solution that may be converted into a partition by setting a threshold, using the same methods as in Chapter 3. In order to generate a clustering or segmentation with more than two parts, the algorithm may be recursively applied to each partition separately, generating subpartitions and stopping the recursion if the isoperimetric ratio of the cut fails to meet a predetermined threshold. We term this predetermined threshold the **stop parameter** and note that since $0 \leq h(x) \leq 1$, the stop parameter should be in the interval $(0, 1)$. Since lower values of $h(x)$ correspond to more desirable partitions, a stringent value for the stop criteria is small, while a large value permits lower quality partitions (as measured by the isoperimetric ratio).

4.2.1 Algorithmic details

Choosing edge weights

In order to apply the isoperimetric algorithm to partition a graph, the position values (for data clustering) or the image values (for image segmentation) must be encoded on the graph via edge weights. Define the vector of data changes, c_{ij}, as the Euclidean distance between the fields (e.g., coordinates, image RGB channels, image grayscale, etc.) on nodes v_i and v_j. For example, if we represent grayscale intensities defined on each node with vector b, then $c = Ab$. We employ the weighting function of (2.13). For a standard, Cartesian, 4-connected lattice the geometric distance between each pair of nodes is equal, and therefore we set $\beta^2 = 0$. In order to make one choice of β^1 applicable to a wide range of data sets, we have found it helpful to normalize the vector c.

Ground point

We will demonstrate that, in the image processing context, the ground point may be viewed from an attentional standpoint. In Chapter 3 it was determined that the best of three

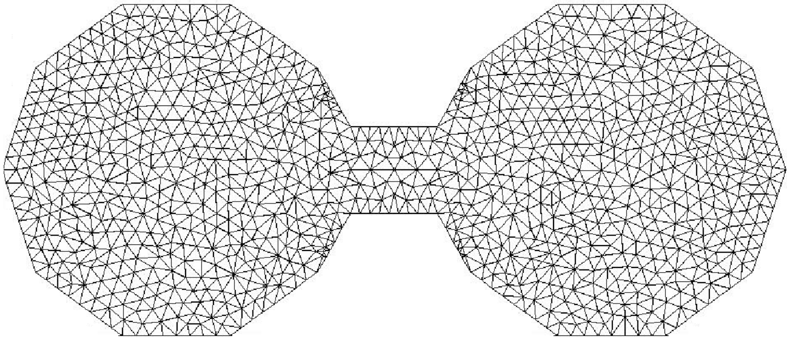


Figure 4·1: Dumbbell graph with uniform weights

random grounds consistently yielded a cut with lower isoperimetric ratio than choosing to ground either the node of maximum degree or the node of minimum degree. However, grounding the node of maximum degree appeared to be a good heuristic if the additional cost of choosing the best cut of three random grounds was not acceptable. For purposes of generating the figures in this chapter, the node of maximum degree was chosen as ground, unless stated otherwise.

Figure 4·2 illustrates the effect of choosing different ground points on the solution to (4.3). We have employed the dumbbell shape in Figure 4·1, discussed in Cheeger (1970) on the relationship of the isoperimetric constant and the eigenvalues of continuous manifolds. The left column (i.e., (a), (c), (e), and (g) in Figure 4·2) shows the potentials, x, solved for using (3.13). The brightest node on the graph represents the ground point. For the rest of the nodes, bright nodes are closer to ground (i.e., have lower potentials) and dark nodes are further from ground. The right column (i.e., (b), (d), (f), and (h) in Figure 4·2) shows the post-threshold function where the ratio cut method has been employed. The top two rows indicate random ground points and the bottom two represent pathological ground points. Of the two pathological cases, the third row example (i.e., (e) and (f) in Figure 4·2) uses a ground in the exact center of the neck, while the last takes ground to be one node over from the center. Although the grounding in the exact center produces a partition that does not resemble the known ideal partition, grounding one node over produces a partition that is nearly the same as the ideal, as shown in the fourth row example (i.e., (g) and (h) in

Figure 4·2). This illustrates that the solution is largely independent of the ground point, except in the pathological case where the ground is on the ideal cut. Moreover, it is clear that choosing a ground point in the interior of the balls is better than choosing a point on the neck, which corresponds in some sense to our above rule of choosing the point with maximum degree since a node of high degree will be in the "interior" of a region, or in an area of uniform intensity in the context of image processing.

Summary of the algorithm

Applying the isoperimetric algorithm to data clustering or image segmentation may be described in the following steps:

1. Find weights for all edges using equation (2.13).
2. Build the L matrix (2.5) and d vector.
3. Choose the node of largest degree as the ground point, v_g, and determine L_0 and d_0 by eliminating the row/column corresponding to v_g.
4. Solve (4.3) for x_0.
5. Threshold the potentials x at the value that gives partitions corresponding to the lowest isoperimetric ratio.
6. Continue recursion on each segment until the isoperimetric ratio of the subpartitions is larger than the stop parameter.

4.3 Relationship to other graph partitioning methods

The relationship between the isoperimetric algorithm and spectral partitioning was explored in Chapter 3. It was demonstrated that the isoperimetric algorithm produces good partitions for some families of graphs for which spectral partitioning is known to fail. The Ncuts algorithm of Shi and Malik (2000) is essentially the spectral partitioning algorithm,

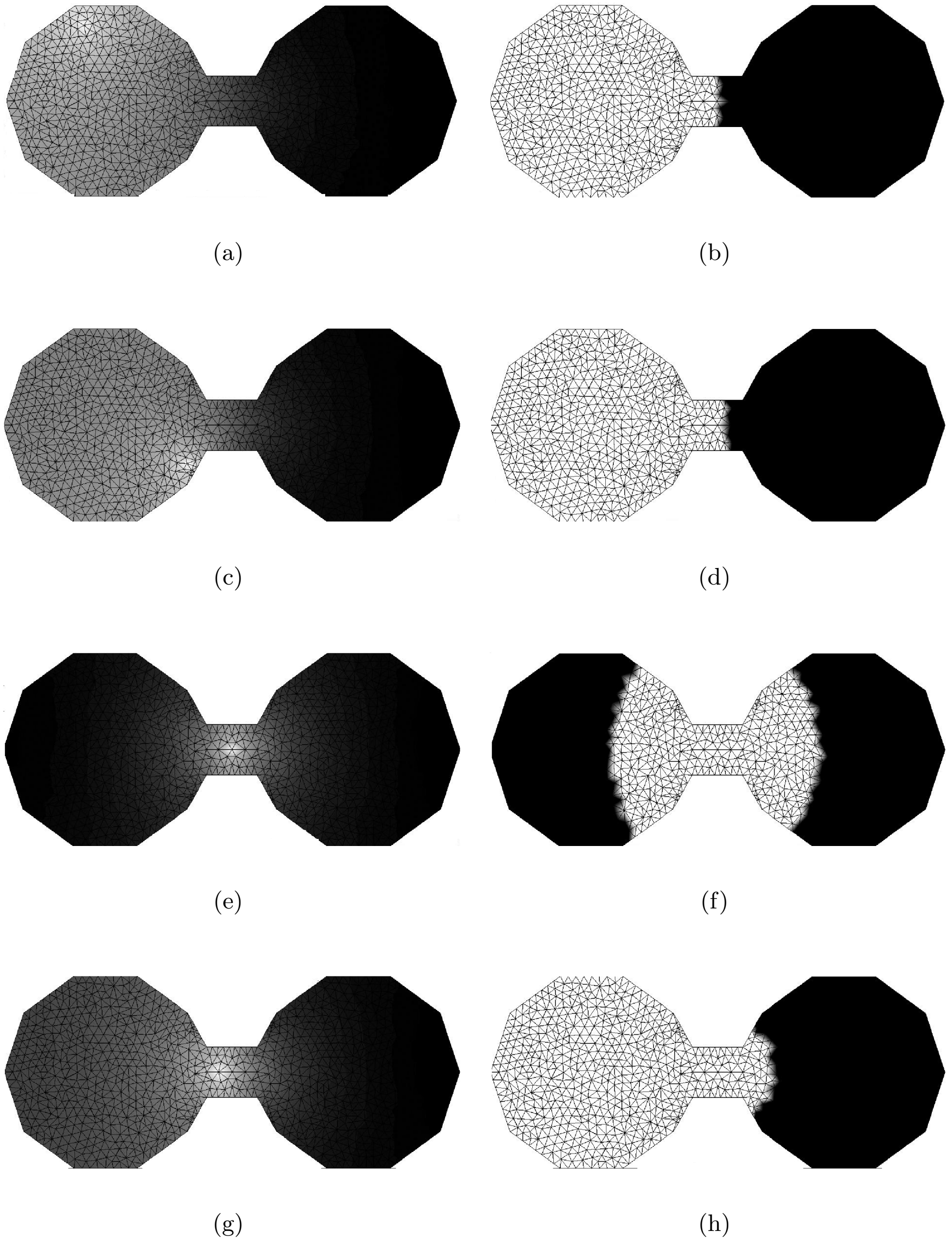


Figure 4·2: An example of the effects on the solution with different choices of ground point for a problem with a well-known optimal partition. The left column shows the potential function (brightest point is ground) for several choices of ground while the right column shows thresholded partitions. Uniform weights ($\beta^2 = 0$) were employed.

except that the authors implicitly choose the metric of Dodziuk and Kendall (1986) to define a combinatorial Laplacian matrix rather than the metric of Dodziuk (1984) typically used to define the Laplacian in spectral partitioning. Specifically, the Ncuts algorithm requires the solution of

$$D^{-\frac{1}{2}}LD^{-\frac{1}{2}}z = \lambda z \tag{4.4}$$

where $D = \text{diag}(d)$. Therefore, although the spectral and Ncuts algorithms produce different results when applied to a specific graph, they share many theoretical properties.

Eigenvector methods of graph partitioning have recently become popular (Perona and Freeman, 1998; Sarkar and Soundararajan, 2000; Wang and Siskund, 2003), employing an eigenvector of the Laplacian or adjacency matrix to perform the cut. In contrast, the isoperimetric method requires the solution of a sparse linear system rather than the solution to the eigenvalue problem required by these methods. The Lanczos algorithm provides an excellent method for approximating the eigenvectors corresponding to the smallest or largest eigenvalues of a matrix with a time complexity comparable to the conjugate gradient method of solving a sparse system of linear equations (Golub and Van Loan, 1996). However, the eigenvector problem is less stable to minor perturbations of the matrix than the solution to a system of linear equations, if the desired eigenvector corresponds to an eigenvalue that is very close to other eigenvalues (see Golub and Van Loan (1996)). In fact, for graphs in which the Fiedler value has algebraic multiplicity greater than one the eigenvector problem becomes completely degenerate and the Lanczos algorithm may converge to any vector in the subspace spanned by the Fiedler *vectors* (if it converges at all). A square lattice with uniform weights is an example of a graph for which the Fiedler value has algebraic multiplicity greater than unity, as is the fully connected graph with uniform weights. Kuijaars (2000) raises additional concerns about the Lanczos method.

Previous work in network theory allows for a straightforward analysis of the sensitivity of the isoperimetric, spectral, and normalized cuts algorithms. Here we specifically examine the sensitivity to the edge weights for these three algorithms. Sensitivity to a single, general, parameter, s, is developed in this section. Sensitivity computation for many parameters (e.g., all the weights in a graph) may be obtained efficiently using the adjoint method (Vlach and Singhal, 1994).

4.3.1 Isoperimetric

Given the vector of degrees, d, the Laplacian matrix, L, and the reduced Laplacian matrix L_0, the isoperimetric algorithm requires the solution to

$$L_0 x_0 = d_0. \tag{4.5}$$

The sensitivity of the solution to equation (4.5) with respect to a parameter, s, may be determined from

$$L_0 \frac{\partial x_0}{\partial s} = -\frac{\partial L_0}{\partial s} x_0 + \frac{\partial d_0}{\partial s}. \tag{4.6}$$

Since L_0, x_0 are known for a given solution to equation (4.5) and $\frac{\partial L_0}{\partial s}$ may be determined analytically, $\frac{\partial x_0}{\partial s}$ may be solved for as a system of linear equations (since L_0 is nonsingular) in order to yield the derivative at a point x_0.

4.3.2 Spectral

The spectral method solves the equation

$$Lx = \lambda_2 x, \tag{4.7}$$

where λ_2 is the Fiedler value. The sensitivity of the solution to (4.7) to a parameter s is more complicated, but proceeds in a similar fashion from the equation

$$\frac{\partial L}{\partial s} x + L \frac{\partial x}{\partial s} = \frac{\partial \lambda_2}{\partial s} x + \lambda_2 \frac{\partial x}{\partial s}. \tag{4.8}$$

The term $\frac{\partial \lambda_2}{\partial s}$ may be calculated from the Rayleigh quotient for λ_2 and the chain rule. The Rayleigh quotient is

$$\lambda = \frac{x^T L x}{x^T x}. \tag{4.9}$$

The chain rule determines $\frac{\partial \lambda_2}{\partial s}$ by $\frac{\partial \lambda_2}{\partial s} = \frac{\partial \lambda_2}{\partial x}\frac{\partial x}{\partial s}$. This may be solved by finding $\frac{\partial \lambda_2}{\partial x}$ from the Rayleigh quotient via

$$\frac{\partial \lambda_2}{\partial x} = 2Lx(x^T x)^{-1} - 2x^T Lx(x^T x)^{-2} x. \tag{4.10}$$

Equation (4.10) allows us to solve for $\frac{\partial \lambda_2}{\partial s}$ via equations (4.8) and (4.10)

$$\left(L - \left(\frac{\partial \lambda_2}{\partial x}^T x + \lambda_2\right) I\right) \frac{\partial x}{\partial s} = \frac{\partial L}{\partial s} x. \tag{4.11}$$

Equation (4.11) also gives a system of linear equations which may be solved for $\frac{\partial x}{\partial s}$ since all the other terms are known or may be determined analytically.

4.3.3 Normalized Cuts

The normalized cuts algorithm of Shi and Malik (2000) requires the solution to

$$D^{-\frac{1}{2}} L D^{-\frac{1}{2}} x = \lambda_2 x, \tag{4.12}$$

where D is a diagonal vector with $D_{ii} = d_i$. In a similar fashion to the above treatment on the spectral algorithm, the sensitivity of x with respect to a parameter s may be determined using the Rayleigh quotient and the chain rule.

Employing the chain rule, taking the derivative of equation (4.12) with respect to s and rearranging yields

$$\left(D^{-\frac{1}{2}} L D^{-\frac{1}{2}} - \left(\frac{\partial \lambda_2}{\partial x}^T x + \lambda_2\right) I\right) \frac{\partial x}{\partial s} = \left(2\frac{\partial D^{-\frac{1}{2}}}{\partial s} L D^{-\frac{1}{2}} + D^{-\frac{1}{2}} \frac{\partial L}{\partial s} D^{-\frac{1}{2}}\right) x. \tag{4.13}$$

Again, this is a system of linear equations for $\frac{\partial x}{\partial s}$. For Ncuts, the eigenvalue corresponds to $D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$ instead of L, so $\frac{\partial \lambda_2}{\partial x}$ must be recomputed from the Rayleigh quotient. The result of this calculation is

$$\frac{\partial \lambda_2}{\partial x} = 2D^{-\frac{1}{2}} L D^{-\frac{1}{2}} x (x^T x)^{-1} - 2x^T D^{-\frac{1}{2}} L D^{-\frac{1}{2}} x (x^T x)^{-2} x. \tag{4.14}$$

4.3.4 Sensitivity to a weight

Using the results above, it is possible to analyze the effect of a specific parameter by finding $\frac{\partial L}{\partial s}$, $\frac{\partial d}{\partial s}$ and $\frac{\partial D^{-\frac{1}{2}}}{\partial s}$ for the specific parameter in question. The value for $\frac{\partial L_0}{\partial s}$ is determined from $\frac{\partial L}{\partial s}$ simply by deleting the row and column corresponding to the grounded node. For a specific weight, w_{ij}, these quantities become

$$\left(\frac{\partial d}{\partial w_{ij}}\right)_{v_i} = \begin{cases} 1 & \text{if } e_{ij} \text{ is incident on } v_i, \\ 0 & \text{otherwise,} \end{cases} \tag{4.15}$$

and

$$\left(\frac{\partial D^{-\frac{1}{2}}}{\partial w_{ij}}\right)_{v_p v_q} = \begin{cases} -\frac{1}{2} d_p^{-\frac{3}{2}} & \text{if } p = q,\ p = i \text{ or } p = j, \\ 0 & \text{otherwise.} \end{cases} \tag{4.16}$$

The matrix $\frac{\partial L}{\partial w_{ij}}$ equals the L matrix of a graph with an edge set reduced to just $E = \{e_{ij}\}$. The degree of node v_i is specified by d_i.

Equations (4.6), (4.8) and (4.13) demonstrate that the derivative of the isoperimetric solution is never degenerate (i.e., the left hand side is always nonsingular for a connected graph), whereas the derivative of the spectral and normalized cuts solutions may be degenerate depending on the current state of the Fiedler vector and value.

4.4 Applications

4.4.1 Clustering applied to examples used by Zahn

When humans view a point cluster, certain groupings immediately emerge. The properties that define this grouping have been described by the Gestalt school of psychology . Unfortunately, these descriptions are not precisely defined and therefore finding an algorithm that can group clusters in the same way has proven very difficult. Zahn (1971) used his minimal spanning tree idea to try to capture these Gestalt clusters. To this end, he established a collection of point sets with clear cluster structure (to a human), but which are difficult for a single algorithm to group.

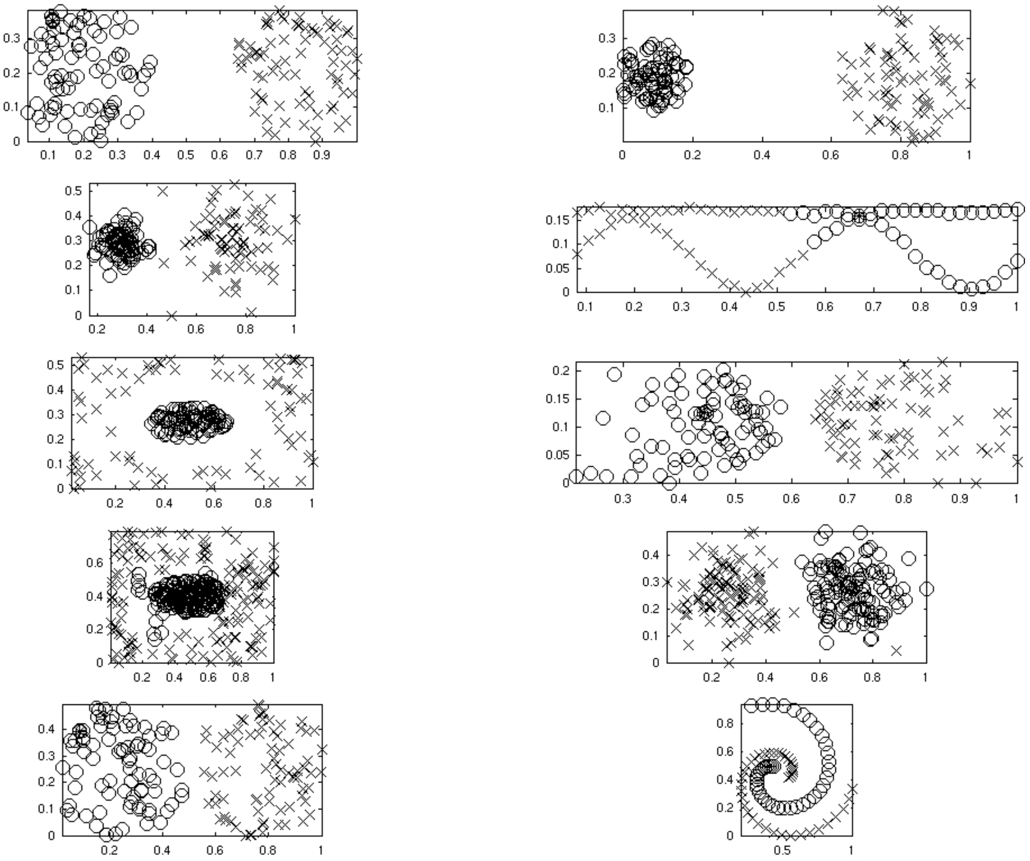


Figure 4·3: An example of partitioning the Gestalt-inspired point set challenges of Zahn using the isoperimetric algorithm. The x's and o's represent points in different partitions. $\beta^2 = 50$.

We stochastically generated point clusters to mimic the challenges Zahn issues to automatic clustering algorithms. For a set of points, it is not immediately clear how to choose which nodes are connected by edges. In order to guarantee a connected graph, but still make use of local connections, we generated an edge set from the Delaunay triangulation of the points. Edge weights were generated as a function of Euclidean geometric distance, as in (2.13).

The clusters and partitions are shown in Figure 4·3. Each partition is represented by a symbol, with the 'x's and 'o's indicating the points belonging to the same partition. Partitions were generated using the median cut on a single solution to (3.13). Ground points were chosen using the maximum degree rule discussed above.

Of these clusters, it is shown in Figure 4·3 that the algorithm performs as desired on all groups except that which requires grouping into lines. This "failure" in the Gestalt sense may be viewed as the same "success" recorded in chapter 3 against Guattery and Miller's roach example. In both cases the "lines" have a small connection to each other at one end and practically no connection at the other. In both the Gestalt and roach graphs, a better partition (from the isoperimetric standpoint) is obtained by cutting the "body" from the "antennae".

4.4.2 Methods of image segmentation

As in the case of point clustering, it is not clear, *a priori*, how to impose a graph structure on an image. Since pixels define the discrete input, a simple choice for nodes is the pixels and their values. Traditional neighborhood connectivity employs a 4-connected or 8-connected topology (Jain, 1989). Another approach, taken by Shi and Malik (2000) is to use a fully connected neighborhood within a parameterized radius from each node. We chose to use a minimal 4-connected topology, since the matrix L becomes less sparse as more edges are added to the graph. Consequently, a graph with more edges requires more time to solve (3.13). Edge weights were generated from intensity values in the case of a grayscale image or from RGB color values in the case of a color image using (2.13).

A similar measure of partition quality has been employed by other authors (e.g., (Hendrickson and Leland, 1995; Schreiber and Martin, 1999)) to develop image segmentation algorithms, but a different notion of volume (e.g., the algorithm of Hendrickson and Leland (1995) is defined only for planar graphs) and different methods for achieving good partitions under this metric of quality separate their work from ours.

For a 4-connected lattice graph (i.e., a standard image), the algorithm is controlled by only two parameters: the scale parameter β^1 of (2.13) and the stop parameter used to end the recursion. The scale affects how sensitive the algorithm is to changes in feature space (e.g., RGB, intensity), while the stop parameter determines the maximum acceptable isoperimetric ratio a partition must generate in order to accept it and continue the

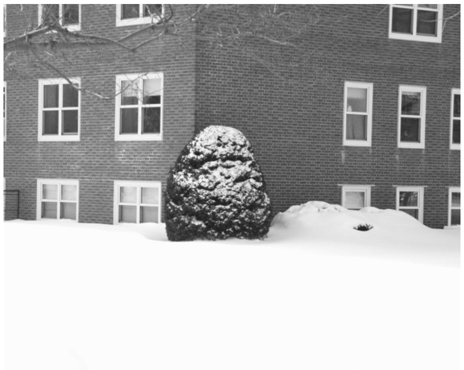


Figure 4·4: Image used to benchmark the effects of a changing scale and stop parameter.

recursion. In order to illustrate the dependence of the results on parameterization, a sweep of the two-dimensional parameter space was performed on individual natural images. An example of this parameter sweep is shown in Figure 4·5 using the natural image in Figure 4·4, with scale on the vertical and stop on the horizontal. It can be seen that the solution is similar over a broad range with respect to changes in scale and that the effect of raising the stop parameter (i.e., making more partitions admissible) is to generate a greater number of small partitions.

4.4.3 Completion

Study of the classic Kaniza illusion (Fineman, 1996) suggests that humans segment objects, based on something beyond perfectly connected edge elements.

The isoperimetric algorithm was used to segment the image in Figure 4·6, using only one level of recursion with all nodes corresponding to the black "inducers" removed. In this case, choice of the ground point is important for determining the single bipartition. If the ground point is chosen inside the illusory triangle, the resulting partition is the illusory triangle. However, if the ground is chosen outside, the triangle partition is not produced, but instead a partition that hugs the corner in which the ground is located. In this way, the ground point may be considered as representing something like an "attentional" point, since it induces a partition that favors the region of the ground point. However, note that

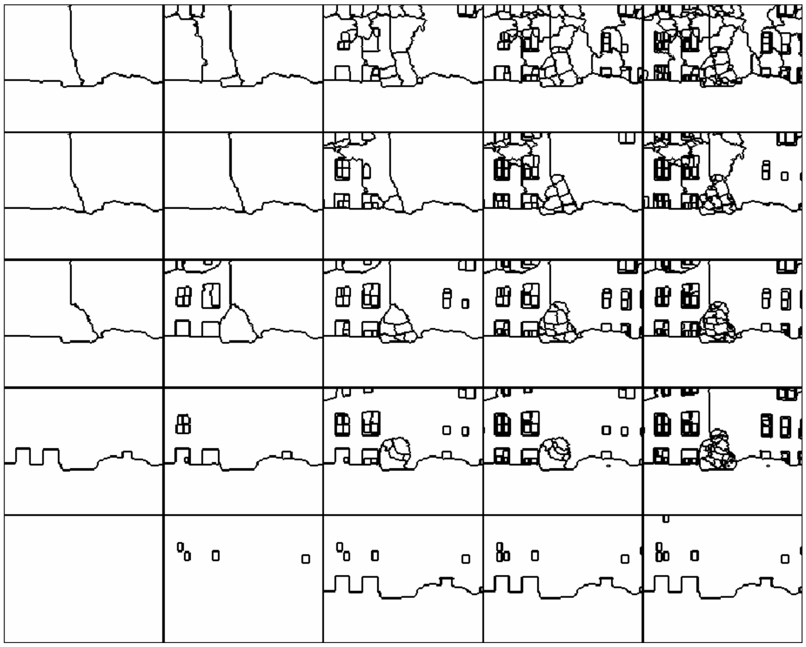
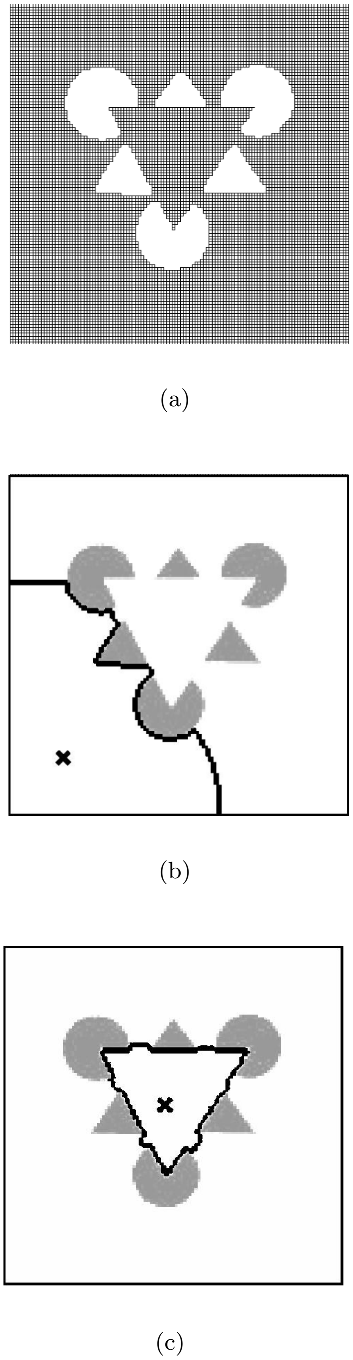


Figure 4·5: This tiled figure demonstrates the results of varying the scale (vertical) and stop (horizontal) parameters when processing the image in Figure 4·4. Scale range: 300–30, Stop range: $1 \times 10^{-5.5}$–$1 \times 10^{-4.5}$.

these partitions are compatible with each other, suggesting that the choice of ground may affect only the order in which partitions are found.

4.4.4 Segmentation of natural images

Having addressed issues regarding stability and completion, we proceed to examples of the segmentation found by the isoperimetric algorithm when applied to natural images. Examples of the segmentation found by the isoperimetric algorithm for some natural images are displayed in Figure 4·7. All results in the example segmentations were obtained using the same two parameters. It should be emphasized in comparisons of segmentations produced by the Ncuts algorithm that the authors of Ncuts make use of a more fully connected neighborhood as well as fairly sophisticated spatial filtering (e.g., oriented Gabor filters at multiple scales) in order to aid in textural segmentation. The demonstrations with the isoperimetric algorithm used a basic four-connected topology and no spatial filtering at all. Consequently, the segmentations produced by the isoperimetric algorithm should be



(a)

(b)

(c)

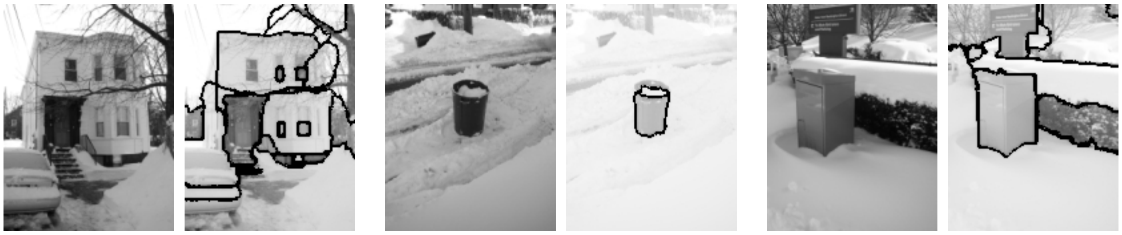
Figure 4·6: The Kaniza triangle illusion with the single bipartition outlined in black and the ground point marked with an ‘x’. (a) The graph being segmented. (b) Isoperimetric partition using a ground point in the corner. (c) Isoperimetric partition using a ground point inside the triangle. Uniform weights ($\beta^1 = 0$) were employed in both cases.

expected to perform less well on textural cues. However, for general grayscale images, it appears to perform at least as well as Ncuts, but with increased numerical stability and a speed advantage of more than one order of magnitude (based on our $^{\text{TM}}$MATLAB implementation of both algorithms). Furthermore, because of the implementation (e.g., 4-connected lattice, no spatial filtering), the isoperimetric algorithm makes use of only two parameters, compared to the four basic parameters (i.e., radius, two weighting parameters and the recursion stop criterion) required in Shi and Malik (2000).

The asymptotic (formal) time complexity of Ncuts is roughly the same as the isoperimetric algorithm. Both algorithms have an initial stage in which nodal values are computed that requires approximately $\mathcal{O}(n)$ operations (i.e., via Lanczos or conjugate gradient). Generation of the nodal values is followed in both algorithms by an identical cutting operation. Using the $^{\text{TM}}$MATLAB sparse matrix solver for the linear system required by the isoperimetric algorithm and the Lanczos method ($^{\text{TM}}$MATLAB employs ARPACK (Lehoucq et al., 1998) for this calculation) to solve the eigenvalue problem required by Ncuts, the time was compared for a 10000×10000 L matrix (i.e., a 100×100 pixel image). Since other aspects of the algorithms are the same (e.g., making weights from the image, cutting the indicator vector, etc.), and because solving for the indicator vector is the main computational hurdle, we only compare the time required to solve for the indicator vector. On a 1.4GHz AMD Athlon with 512K RAM, the time required to approximate the Fiedler vector in (4.4) was 7.1922 seconds while application of the direct solver to the isoperimetric partitioning (3.13) required 0.5863 seconds. In terms of actual computation time (using $^{\text{TM}}$MATLAB), this result means that solving the crucial equation for the isoperimetric algorithm is more than an order of magnitude faster than solving the crucial equation required by the Ncuts algorithm.

4.4.5 Stability

Stability of the solution for both the isoperimetric algorithm and the spectral algorithms differs considerably, as does the perturbation analysis for any solution to a system of



(a) `ESLab0002`

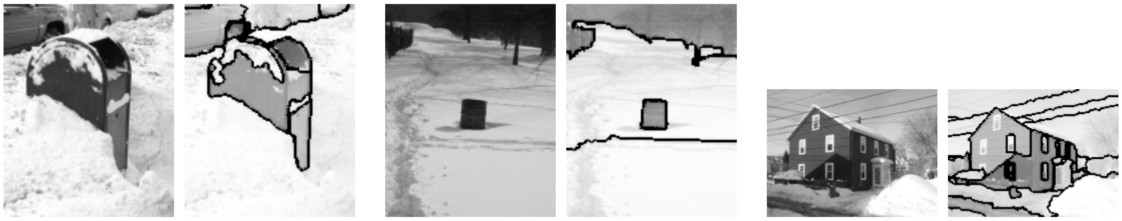
(b) `ESLab0002` segments

(c) `ESLab0005`

(d) `ESLab0005` segments

(e) `ESLab0009`

(f) `ESLab0009` segments



(g) `ESLab0015`

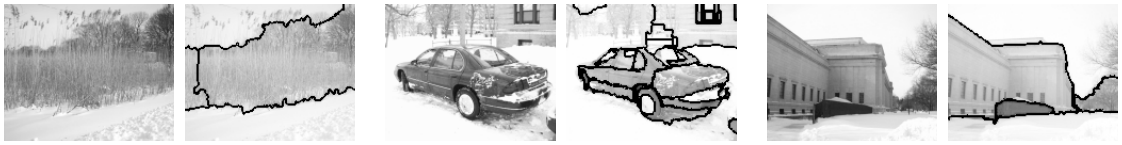
(h) `ESLab0015` segments

(i) `ESLab0021`

(j) `ESLab0021` segments

(k) `ESLab0007`

(l) `ESLab0007` segments



(m) `ESLab0024`

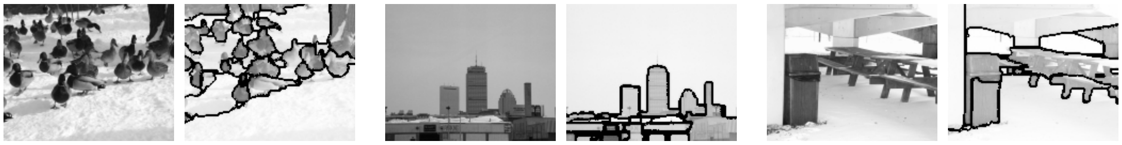
(n) `ESLab0024` segments

(o) `ESLab0027`

(p) `ESLab0027` segments

(q) `ESLab0033`

(r) `ESLab0033` segments



(s) `ESLab0043`

(t) `ESLab0043` segments

(u) `ESLab0052`

(v) `ESLab0052` segments

(w) `ESLab0054`

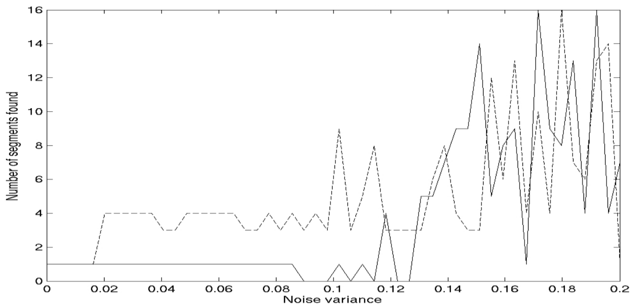
(x) `ESLab0054` segments

Figure 4·7: Examples of segmentations produced by the isoperimetric algorithm using the same parameters ($\beta^1 = 95$, stop $= 10^{-5}$). More segmentation results from the same database may by found at `http://eslab.bu.edu/publications/grady2003isoperimetric/`. Images may be obtained from `http://eslab.bu.edu/resources/imageDB/imageDB.php` equations versus the solution to the eigenvector problem (Golub and Van Loan, 1996). Differentiating equations (3.13) and (4.4) with respect to an edge weight reveals that the derivative of the solution to the spectral (3.24) and Ncuts (4.4) equations is highly dependent on the current Fiedler value, even taking degenerate solutions for some values (see Appendix B). By contrast, the derivative of the isoperimetric solution has no poles. Instability in spectral methods due to algebraic multiplicity of the Fiedler value is a common problem in implementation of these algorithms (see Hendrickson and Leland (1995)). This analysis suggests that the Ncuts algorithm may be more unstable to minor changes in an image than the isoperimetric algorithm.

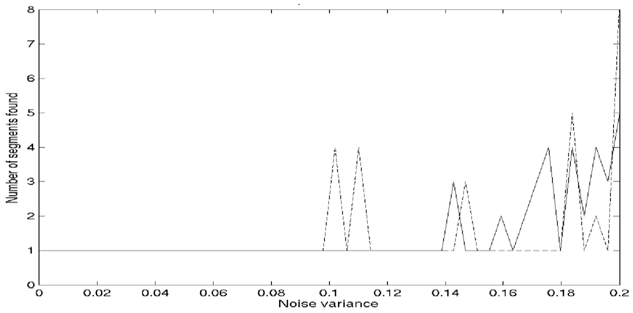
The stability of Ncuts (our implementation) and the isoperimetric algorithm is compared by the resistance of their output to noise via two different comparisons. First, each algorithm was applied to an artificial image of a white circle on a black background, using a 4-connected lattice topology. Increasing amounts of additive, multiplicative and shot noise were applied, and the number of segments output by each algorithm was recorded. Results of this comparison are recorded in Figure 4·8.

In order to visually compare the result of the segmentation algorithms applied to progressively noisier images, the isoperimetric and Ncuts algorithms were applied to a relatively simple natural image of red blood cells. The isoperimetric algorithm operated on a 4-connected lattice, while Ncuts was applied to an 8-connected lattice, since we had difficulty finding parameters that would cause Ncuts to give a good segmentation of the original image, if a 4-connected lattice was used.

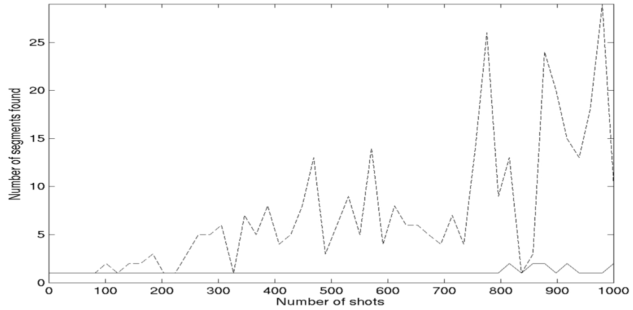
In both comparisons, additive, multiplicative, and shot noise were used to test the sensitivity of the two algorithms to noise. The additive noise was zero mean Gaussian noise with variance ranging from 1–20% of the brightest luminance. Multiplicative noise was introduced by multiplying each pixel by a unit mean Gaussian variable with the same variance range as above. Shot noise was added to the image by randomly selecting pixels that were fixed to white. The number of "shots" ranged from 10 to 1,000. The above discussion of stability is illustrated by the comparison in Figure 4·9. Although additive noise heavily



(a) Additive noise



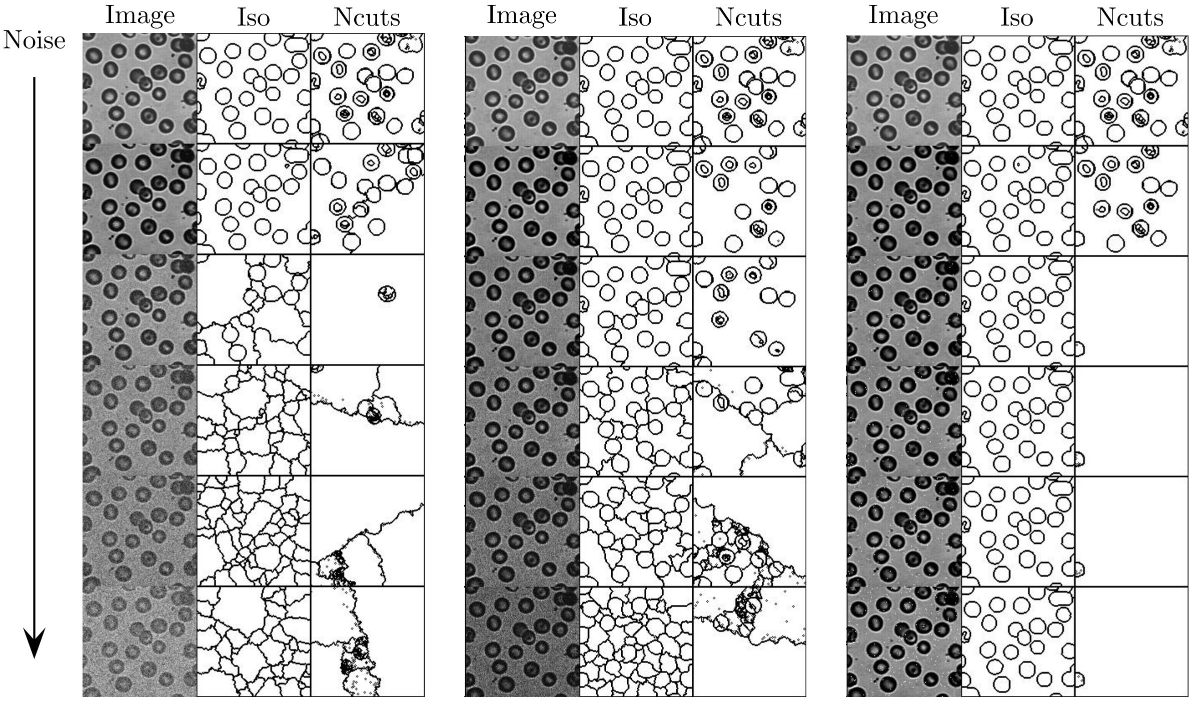
(b) Multiplicative noise



(c) Shot noise

Figure 4·8: Stability analysis relative to additive, multiplicative and shot noise for an artificial image of a white circle on a black background. The x-axis represents an increasing noise variance for the additive and multiplicative noise, and an increasing number of "shots" for the shot noise. The y-axis indicated the number of segments found by each algorithm. The solid line represents the results of the isoperimetric algorithm and the dashed line represents the results of the Ncuts algorithm. The underlying graph topology was the four-connected lattice with $\beta^1 = 95$ for the isoperimetric algorithm and $\beta^1 = 35$, Ncuts stop criterion $= 10^{-2}$ (relative to the Ncuts criterion) and isoperimetric stop criterion $= 10^{-5}$.

degrades the solution found by both algorithms, multiplicative noise appears to slightly degrade the solution found by Ncuts while the introduction of shot noise significantly affects the convergence of Ncuts to any solution. The solution found by the isoperimetric algorithm appeared resilient to both multiplicative and shot noise.



(a) Additive noise (b) Multiplicative noise (c) Shot noise

Figure 4·9: Stability analysis relative to additive, multiplicative and shot noise. Each row represents an increasing amount of noise of the appropriate type. The top row in each subfigure is the segmentation found for the `blood1.tif` image packaged with $^{\text{TM}}$MATLAB (i.e., zero noise). Each figure is divided into three columns representing the image with noise, isoperimetric segmentation and Ncuts segmentation from left to right respectively. The underlying graph topology was the four-connected lattice with $\beta^1 = 95$, Ncuts stop criterion $= 10^{-6}$ (relative to the Ncuts criterion) and isoperimetric stop criterion $= 10^{-8}$. (a) Additive noise. (b) Multiplicative noise. (c) Shot noise.

4.5 Conclusion

We have applied the isoperimetric algorithm for graph partitioning to the data clustering and image segmentation problem. The algorithm was compared with Ncuts to demonstrate that it is simpler, faster, and more stable, while providing visually comparable results with less pre-processing.

Since the graph representation is not tied to any notion of dimension, the algorithm applies equally to graph-based problems in N-dimensions as it does to problems in two dimensions. Suggestions for future work are applications to segmentation in space-variant architectures, supervised or unsupervised learning, 3-dimensional segmentation of mesh-based objects, and the segmentation/clustering of other areas that can be naturally modeled with graphs.

Chapter 5

Small world

5.1 Introduction

Traditional solution methods to partial differential equations (e.g., finite differences, finite elements) typically culminate in the solution to a large, sparse system of linear equations. The sparsity pattern of the matrix corresponds directly to the topology of the sampling grid. In situations where the goal is to model physical systems (e.g., heat flow, electrostatic fields), the choice of topology is limited to a 4- or 8-connected grid (in 2D). However, many problem domains fail to have a clear topology, yet require the solution to a large, sparse system of equations. Examples include unsupervised clustering of points in space or the collection of pixels in image processing. The arbitrary character of data topology has generally resulted in the choice of locally connected data (e.g., Delaunay triangulation, K-nearest-neighbors), although more exotic local topologies are sometimes used (e.g., Shi and Malik (2000)). In Chapter 4, a simple, 4-connected topology was employed for image analysis, largely because fewer edges results in a shorter execution time (i.e., the matrix L is more sparse).

Given that some situations allow for the choice of topology, one may ask: What is the best topology for the problem? Obviously, the answer to this question will depend heavily on the problem itself and solution being sought. Regardless of the specifics of the problem, a faster solution to the system of linear equations is frequently desirable. We demonstrate here that it is possible to significantly accelerate the convergence of standard iterative methods for the solution to a system of linear equations with only a tiny additional cost per iteration. This idea holds for any Krylov subspace method, which includes the popular conjugate gradients and Lanczos methods (Golub and Van Loan, 1996).

5.2 Convergence of Krylov subspace methods

Iterative, Krylov subspace algorithms are the methods of choice for solving both the system of linear equations (conjugate gradients) and the eigenvector problem (Lanczos method) (Golub and Van Loan, 1996). Define a Krylov subspace of an $n \times n$ matrix A, an $n \times 1$ vector x_0 and an integer k as

$$K(A; x_0; k) = \text{span}(x_0, Ax_0, A^2x_0, \ldots, A^{(k-1)}x_0). \tag{5.1}$$

The solution found at each iteration, i, of conjugate gradients is the solution to $Ax = b$ projected onto the Krylov subspace $K(A, Ax_0 - b, i)$ (Dongarra et al., 1991).

Since square matrices are isomorphic to graphs, a graph theory interpretation of matrix operations often lends fresh insight into the dynamics (Gilbert, 1994). From the standpoint of graph theory, each iteration of a Krylov subspace method propagates information one edge. For example, if x_0 represents an impulse function, then that impulse will have spread only k edges after k iterations. This analogy allows for the interpretation of Krylov subspace iterative methods as a mixing process (Gremban, 1996).

In fact, this analogy can be made explicit by considering the solution to a diffusion process over a graph (e.g., discrete lattice) with discrete time steps. For the graph Laplacian matrix (Merris, 1994), L, and current state, x_i, the discrete diffusion equation may be written

$$x_{i+1} = x_i + \Delta t L x_i, \tag{5.2}$$

meaning that each iteration x_i is the sum of a polynomial in L multiplied by the vector representing the initial state x_0. In other words, x_i is a vector in the subspace $K(L, x_0, i-1)$.

It is this analogy between conjugate gradients and a mixing process which led to the observation that the rate of convergence will be a function of graph diameter (Gremban et al., 1995). In other words, since a Krylov subspace iterative method only spreads information along one edge with each iteration, the algorithm cannot converge until the information has spread to all nodes in the graph. Therefore, the minimum number of iterations is the length of the longest optimal path between any two nodes (i.e., the graph diameter). Clearly, the above statement assumes that the initial guess does not happen to be correct.

5.2.1 Small worlds

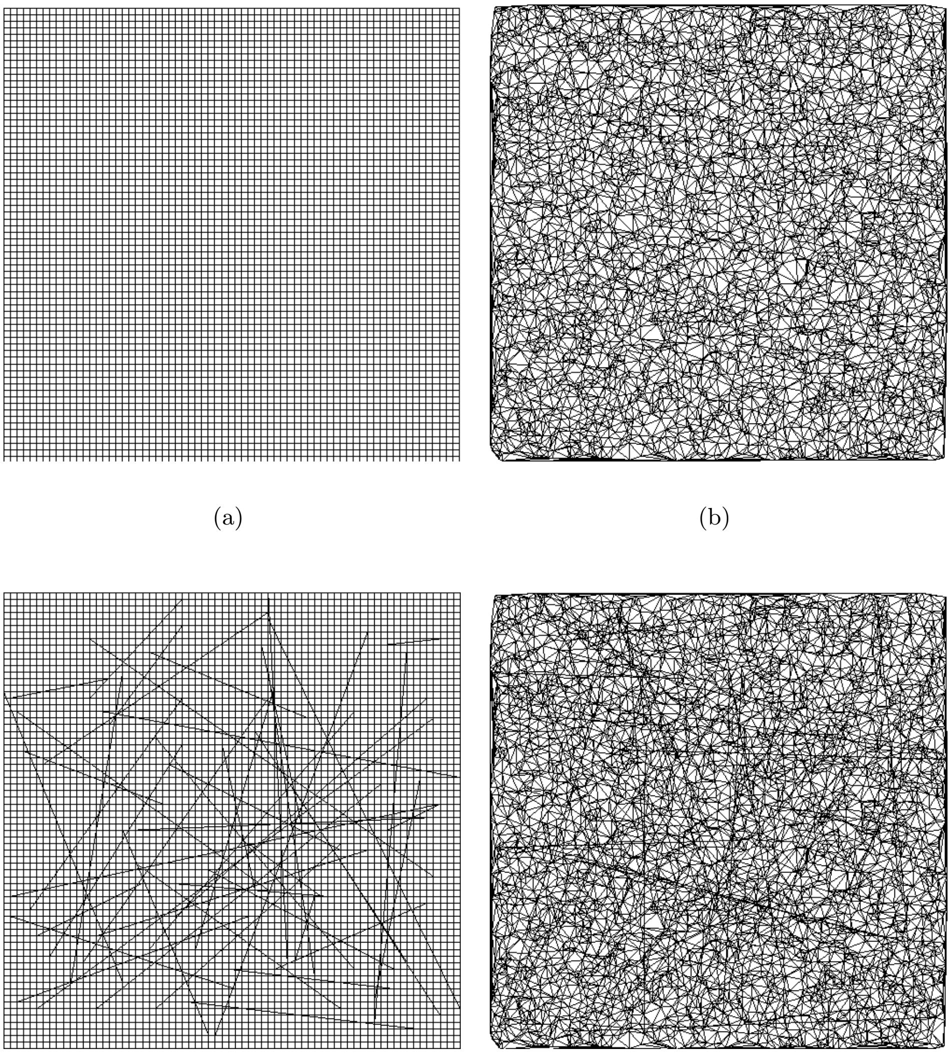
In their landmark paper, Watts and Strogatz (1998) define what they term a “small world” topology based on the six degrees of separation or small world phenomenon found in social networks. The main property of a small world network is that it is locally connected but has a small graph diameter.

Watts and Strogatz demonstrate that a graph with these properties may be obtained by “interpolating” between a typical, locally connected graph and the random graphs first defined by Erdös and Renyi (1960, 1959). Most remarkably, Watts (1999) demonstrates that a locally connected graph may be made into a small world graph (i.e., given a small diameter) with the addition of only a tiny number of random edges. Figure 5·1 shows a lattice substrate and a Delaunay triangulation substrate with a small random edges added.

Our proposal for increasing the convergence rate of Krylov subspace methods in situations with a negotiable topology is therefore to choose a locally connected “substrate” topology (e.g., lattice, nearest-neighbor) and add in a tiny number of random edges. Based on the intuition above, since the graph diameter is dramatically decreased by the addition of these new edges, the convergence rate of the iterative method should substantially decrease. Furthermore, the computational increase per iteration should be negligible, since only a tiny number of additional edges were added. The next section attempts to address the following issues:

1. What is the effect of adding a few random edges on convergence?

2. What is the effect of adding a few random edges on the number of computations?



(a) (b)

(c) (d)

Figure 5·1: (a) Lattice substrate. (b) Delaunay triangulation substrate. (c) Small world graph built on a lattice substrate. (d) Small world graph built on a Delaunay triangulation substrate.

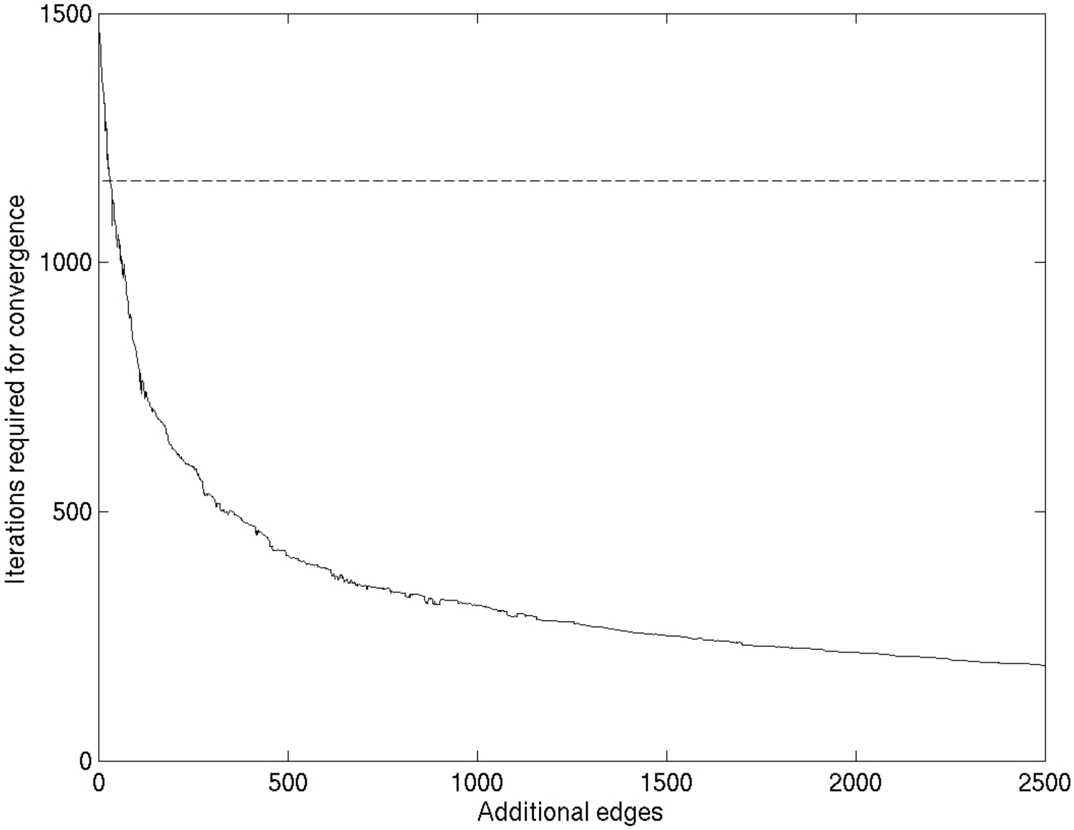
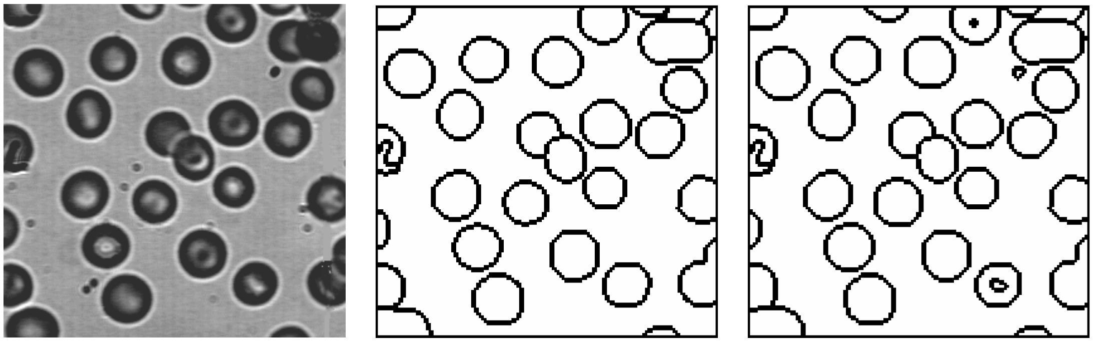


Figure 5·2: Iterations required to converge on a solution for the isoperimetric algorithm with conjugate gradients as the number of random edges added increases for a 256 × 256 4-connected lattice with uniform weights. The dashed line represents the number of iterations required for convergence with the unaltered 4-connected topology.

3. How does the addition of these random edges affect the solution?

5.3 Results

Several problems require the solution to a system of linear equations on a graph. Among these are the Dirichlet problem on a graph (Doyle and Snell, 1984) and the isoperimetric algorithm of Chapter 3. For purposes of the following examples, we demonstrate the convergence of the isoperimetric algorithm with a uniformly weighted 4-connected lattice substrate. Figure 5·2 demonstrates that the number of iterations required to obtain convergence drops dramatically with the addition of only a tiny number of random edges.



(a) (b) (c)

Figure 5·3: (a) Original (input) image. (b) Segmentation obtained with unaltered 4-connected topology ($\beta^1 = 95$, `stop`$= 10^{-5}$). (c) Segmentation obtained with the addition of 200 random edges ($\beta^1 = 95$, `stop`$= 10^{-5}$).

Recall that the number of `multiply` operations per iteration in a Krylov subspace method is equal to the number of nonzeros in the matrix. In the case of a 4-connected lattice, the number of nonzeros, p, is $p \leq 8n$. By contrast, the fully connected graph corresponds to a matrix with n^2 nonzero elements. Every random edge added incurs an additional 2 nonzero elements (due to symmetry). Therefore, the amount of computation required (i.e., number of `multiply` operations) per iteration using a small world graph with a few extra edges is essentially the same as the computation required to process on the substrate graph.

Perturbation theory allows for the study of the effect on the solution to an $Ax = b$ equation when the matrix A is perturbed (Golub and Van Loan, 1996). Since the solution, x, clearly changes with a change in A (i.e., a change in topology), it is useful to examine the effect of adding random edges on the solution. For purposes of applying the isoperimetric algorithm of Chapter 4 to an image, the effect of a significant number of edges (as regards the number of iterations required for convergence) is shown in Figure 5·3 to have a minimal effect on the final solution.

Fully connected graphs cause the solution at each node for an $Ax = b$ problem to depend heavily on every other node, since each node is connected to all other nodes. For some problems (e.g., unsupervised clustering of points in feature space) this “global” approach appears to be a good choice. However, the tremendous costs involved in sorting and processing the large, full, matrix corresponding to a fully connected graph often outstrips the resources of a desktop computer, even when the number of nodes is small. Watts (1999) argues that a small world graph has many of the same properties of a fully connected graph, based on the fact that the small diameter of the graph allows more global information sharing. Therefore, generating a small world graph through the addition of random edges may not only reduce the required computations, but may also produce a solution that is close to the potentially desirable solution obtained by using a fully connected topology.

5.4 Conclusion

The purpose of this chapter is to introduce the idea of employing a small world topology to increase the convergence rate of Krylov subspace iterative algorithms by adding a small number of random edges. We have found that the increase in speed with the addition of random edges is almost an order of magnitude greater than the speed required to converge on the unaltered substrate. Although problems that are intended to model the physical world do not allow for arbitrary choices of topology, it is frequently the case that data is given at spatial locations (e.g., pixels in images) without any indication of how a topology should be defined. In these cases, we have found that the addition of a small number of random edges to the substrate topology increases the convergence rate of an iterative Krylov subspace algorithm by a significant amount.

Chapter 6

Pyramid-based image segmentation

6.1 Introduction

The use of a multi-scale image representation to enhance image analysis algorithms has a long history dating back to Witkin (1983). Typically (Wright and Acton, 1997; Chen and Acton, 1998; Acton, 1996; Comer and Delp, 1995; Pachai, 1998), a multi-resolution representation is employed both for speed and robustness against noise by performing the analysis at the coarsest level and projecting the solution back to the original image. This approach is very efficient, since the coarse levels in the pyramid are substantially smaller than the original. The robustness of an algorithm to noise is also increased substantially, since the filtering required to obtain images at coarser resolutions dampens or eliminates noise altogether.

The approach taken in this chapter is to employ a pyramid structure in a different manner. Viewing the pyramid as a graph with fixed topology, we ignore the distinction between levels and simply treat the pyramid as a graph structure to which image values are attached. Applying the isoperimetric segmentation algorithm of Chapter 4 to the pyramid, we demonstrate improved quality segmentations in terms of detecting objects with blurred boundaries and consequent edge localization for image segmentation. Although the use of a pyramid architecture incurs the additional cost of more nodes, we demonstrate that this fact is mitigated or entirely compensated by the fact that the graph diameter is dramatically decreased.

Part of the motivation for the approach comes from the architecture of biological vision systems, which appear to have multiple representations of the visual field at progressively coarser resolutions. Furthermore, the use of graph-based algorithms allows us to analyze image data on space-variant, multi-resolution vision architectures.

6.2 Connected Pyramid Architecture

Traditional pyramid methods of image analysis employ a Gaussian or Laplacian filter (Burt and Adelson, 1983) to generate new levels. After reaching an arbitrary coarse resolution, the image is analyzed using a standard segmentation algorithm, such as autoregressive models (Comer and Delp, 1995), diffusion (Acton, 1996), watersheds (Wright and Acton, 1997) or adaptive thresholding (Pachai, 1998). The solution on the coarse level is then projected down and refined until the original, fine resolution, image is achieved. Other pyramidal approaches employ multi-level region growing (Koster, 1997), adaptively linking nodes between levels (Rehrauer et al., 1998), or using the levels to build a feature vector that may be analyzed by a standard clustering algorithm (Rezaee, 2000).

Our approach begins, like those above, by building a pyramid of progressively coarser images and linking them with the original in a standard quadtree topology. We term this a **connected pyramid**. Once the structure is built, we differ from the above approaches by ignoring the distinction between levels and applying graph processing algorithms directly to the structure. This approach means that segments may exist between multiple levels.

In order to perform graph-based image processing, the connections within layers are made explicit. Taking the within layer topology to be the standard 4/8-connected or a radially connected topology (Shi and Malik, 2000) results in the three layer connected pyramids in figure 6·1.

Since we want the algorithm to be sufficiently general such that shift-invariant spaces are not assumed (e.g., for convolution), the values at upper levels are determined from their children's mean or median value. Although we lose some properties of the Gaussian filter (e.g., preserving average grayscale, etc), this general framework admits trees built from space-variant image representations. Other approaches to transferring image values from lower levels to higher levels might include the combinatorial diffusion operator.

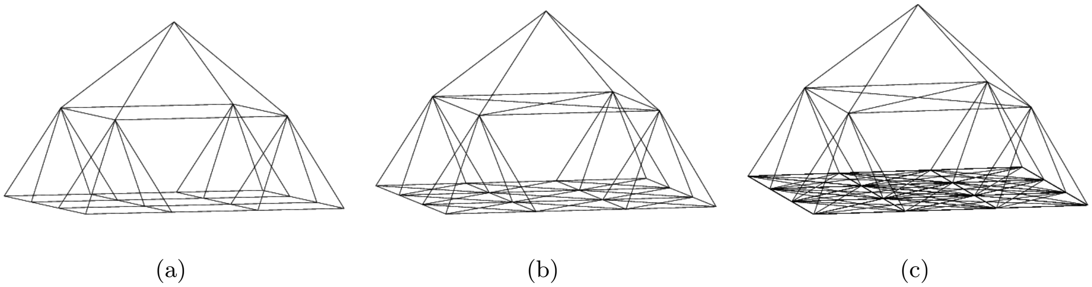

(a) (b) (c)

Figure 6·1: Topology of the connected pyramid graph with 4-connected (a), 8-connected (b), and radius = 5 connected (c) within level connections.

Although any graph-based image analysis algorithm could be applied to the connected pyramid (e.g., Shi and Malik (2000); Perona and Freeman (1998); Sarkar and Soundararajan (2000); Wang and Siskund (2003)), we choose to apply the isoperimetric algorithm of Chapter 4 for purposes of the current work.

Since (4.3) is a large, sparse, set of linear equations, it is most efficiently solved using the conjugate gradients algorithm (Golub and Van Loan, 1996). As reviewed in Chapter 5, conjugate gradients (or any Krylov subspace iterative method) may be viewed as a *mixing process*, meaning that convergence is a function of graph diameter (Gilbert, 1994; Gremban, 1996; Gremban et al., 1995). The graph diameter in an $n \times n$ Cartesian lattice is $2n$, while the addition of each new level causes the graph to have half the diameter of the previous level, to a minimum diameter of $2\log_2(n)$ for a full quadtree pyramid. Therefore, despite the fact that the addition of new levels requires the solution of (4.3) for more nodes (to a limit of $\frac{4}{3}(n \times n)$), the graph diameter decreases dramatically with the addition of new levels, meaning that conjugate gradients should converge faster. In the results section, the effect of decreasing graph diameter is shown to almost entirely compensate for the additional nodes in terms of computational efficiency. In summary, the quality of the pyramidal isoperimetric segmentation is improved while incurring a negligible additional computational cost.

6.3 Results

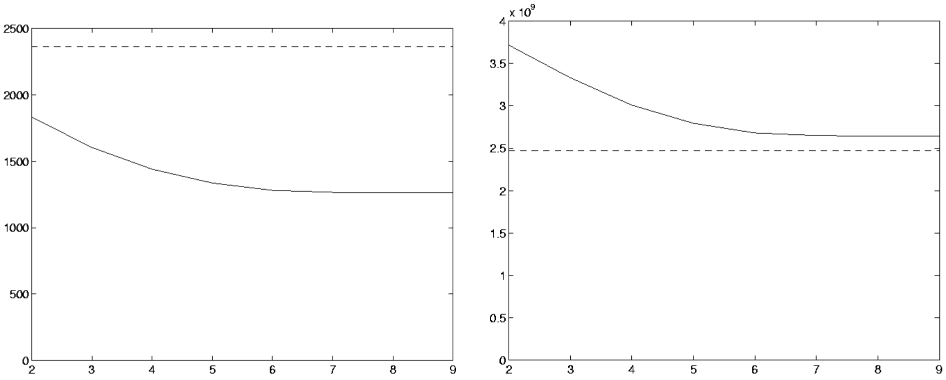
6.3.1 Speed

In order to determine the mitigating effect of decreased graph diameter on the solution to (4.3), we varied the number of levels used in a 512×512 lattice with uniform weights and measured the number of iterations required for convergence of the conjugate gradients method. However, this measure can be misleading since the number of computations per iteration increases as the cardinality of the node and edge sets increases. In order to capture the computational efficiency of conjugate gradients in solving (4.3) on a lattice and a pyramid, the number of `multiply` operations required to solve (4.3) was also calculated. Figure 6·2 demonstrates that the number of iterations required for convergence decreases significantly as new levels are incorporated into the graph, such that the number of iterations required for convergence for a full pyramid is slightly greater than half that required for a lattice. The computational effect of reducing the number of iterations required for convergence is also displayed in Figure 6·2, demonstrating that the improved segmentations obtained from a pyramid architecture incur less than 7% additional computations. This result represents significant improvement over the additional computations of 33% expected by a an algorithm that is linear in the number of nodes.

6.3.2 Segmentation quality

Due to the additional levels in a connected pyramid, more global information is used by the isoperimetric algorithm in determining good partitions. This additional global information generates improved localization of blurred boundaries, resulting in higher quality edge detection. Since there is an increased number of edges in a segment boundary, as a result of the multiresolution representation, the stop parameter of Chapter 4 must be increased. In other words, since the isoperimetric ratio of the segments in the pyramid representation is expected to be larger (since the segments have boundaries that exist in multiple levels), the threshold for an acceptable partition must be raised slightly.

We employed the weighting function of (2.13), with $\beta^2 = 0$. Since this weighting



(a)

(b)

Figure 6·2: (a) Number of iterations required for convergence of conjugate gradients for (4.3) on a 512×512 lattice as the number of levels in the pyramid are added. Dashed line represents the number of iterations required to converge for a simple (non-pyramid) lattice. (b) Total number of `multiply` operations required to perform conjugate gradients as the number of levels increases. Dashed line represents the number of `multiply` operations required for a simple (non-pyramid) lattice.

function is nonlinear, a ramp edge will result in a series of medium sized weights instead of the tiny weight that typically signifies an edge. However, in a pyramid architecture, the burred edges remain sharp on some level of the pyramid, allowing the algorithm to properly make use of them in determining a segmentation. In Figure 6·3 we blurred an image of blood cells and compared the results of the connected pyramid segmentation with the lattice segmentation using the isoperimetric algorithm. Despite the fact that each level in the connected pyramid is locally connected (a 4-connected lattice in this case), the blurred edges are detectable because the boundaries remain sharp on the upper levels of the pyramid.

That the connected pyramid based isoperimetric algorithm makes better use of blurred edges suggests that the quality of natural image segmentation will be increased. In Figure 6·4 the lattice-based and pyramid-based isoperimetric segmentations are compared for several natural images. One can see that difficult edges are better localized with the

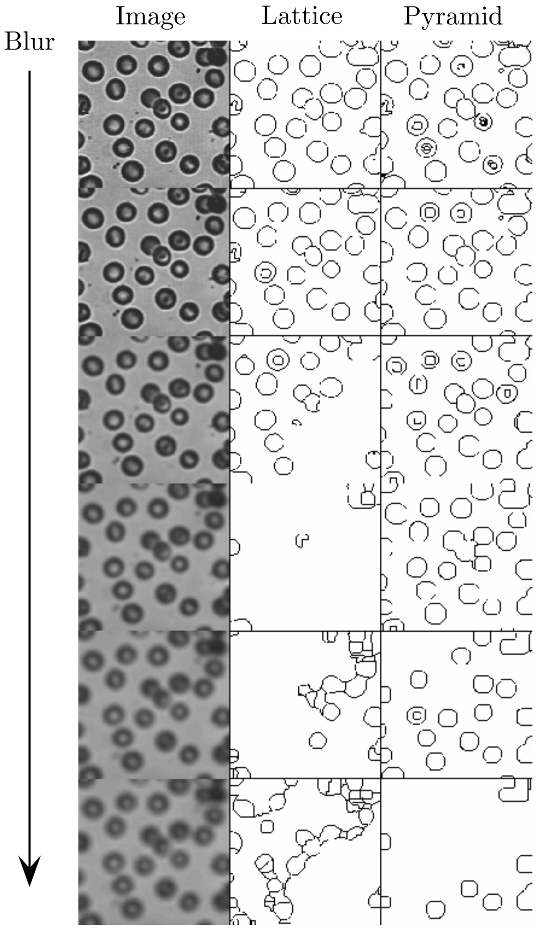
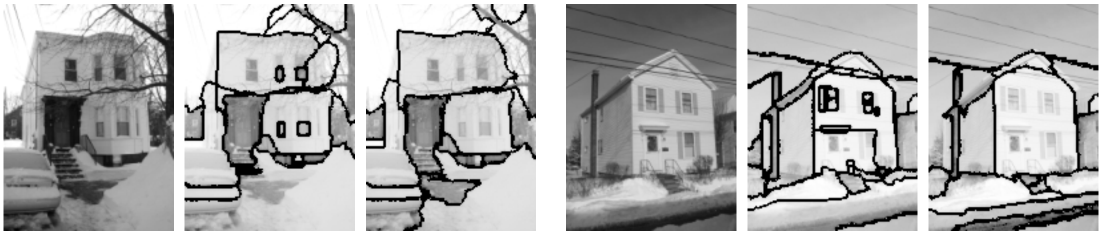
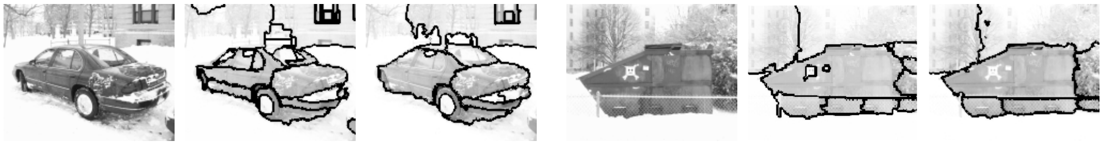


Figure 6·3: Comparison of segmentations produced by lattice-based and pyramid-based isoperimetric algorithm in response to increased blur. Left: Image with increased variance Gaussian kernel (1–7 pixel variance). Middle: Lattice-based segmentation ($\beta^1 = 95$, $\texttt{stop} = 1.0 \times 10^{-5}$). Right: Pyramid-based segmentation ($\beta^1 = 180$, $\texttt{stop} = 2.0 \times 10^{-5}$).



(a) `ESLab0002` (b) Lattice (c) Pyramid (d) `ESLab0002` (e) Lattice (f) Pyramid



(g) `ESLab0002` (h) Lattice (i) Pyramid (j) `ESLab0002` (k) Lattice (l) Pyramid

Figure 6·4: Comparison of images segmented with pyramid ($\beta^1 = 180$, `stop`$= 10^{-5}$) and lattice ($\beta^1 = 95$, `stop`$= 10^{-5}$) based isoperimetric algorithms. More examples of the segmentations produced by the pyramid-based algorithm may be found at `http://eslab.bu.edu/publications/2003/grady2003connected/`.

pyramid-based algorithm.

6.4 Conclusion

Segmentation algorithms that input an arbitrary graph with image values attached to the nodes and return a set of node labels require no modification to enable them to operate on a connected pyramid. The multiresolution image representation results in enhanced segmentations, on account of an increased resilience to blurred edges and noise. Since the isoperimetric algorithm uses a conjugate gradient solver and the graph diameter of the pyramid is much smaller, the enhanced segmentations require very little additional cost, despite a larger node set. Therefore, the use of a connected pyramid to perform isoperimetric segmentation results in higher quality segmentations at only marginally higher computational costs. As a side effect, segmentations at lower resolutions are also obtained, which might allow an object recognition system to operate at different resolutions as the result of one pass of the segmentation algorithm. The preceding suggests that the isoperimetric algorithm should be used with the connected pyramid whenever a fast conjugate gradient solver is available.

Chapter 7

Graph interpolation

7.1 Introduction

Uniformly sampled images are conventionally represented via 4-connected or 8-connected (Cartesian) grids. However space-variant images require a more flexible image topology. In this thesis, we have attempted to further develop the tools necessary for performing space-variant machine vision on a graph-theoretic structure.

This chapter addresses the problem of how to interpolate nodal data on a graph, and then demonstrates applications to image processing. An algorithm is presented that allows interpolation from known values on the nodes of a graph to missing data in such a way that the interpolated values are "smooth". The method is to solve the combinatorial Laplace equation with Dirichlet boundary conditions given by the known values. A solution to the combinatorial Laplace equation has several desirable properties in the context of an interpolation method (see below). Both isotropic and anisotropic interpolation are handled similarly. Furthermore, use of the algorithm is independent of the dimension in which a graph is embedded. Combinatorial differential operators corresponding to the vector calculus operators Div and Grad are used to develop combinatorial versions of the Laplace and Laplace-Beltrami operators. This homology between continuum and combinatorial (graph) algorithms is well known in the literature of circuit theory, mechanical engineering, and related areas in which discretizations of partial differential equations play a central role (Strang, 1986). The solution to the Laplace equation is analogous to solving an equivalent electrical circuit. The solution to problems of this type, as first noted by Maxwell (1991a,b), represents a minimal power dissipation state in the electrical circuit formulation, as shown by Dirichlet's Principle (Courant, 1950; Morse and Feshbach, 1953). An application of these ideas to isotropic and anisotropic image interpolation is presented, and a brief discussion of the relation of this work to anisotropic diffusion is outlined.

7.2 Dirichlet problem

Solving the Laplace equation in order to "fill-in" missing values has been described in the context of digital elevation models (Burrough, 1986; Wood and Fisher, 1993), image editing (Elder and Goldberg, 2001), and is even used by the $^{\mathrm{TM}}$MATLAB function `roifill.m` to fill in regions of missing data in images. What is new about the present work is the generalization of this interpolation concept to arbitrary geometries, topologies and metrics, i.e., to an image representation based on an arbitrary graph rather than on the familiar uniform raster.

7.2.1 Definitions

The **Dirichlet integral** may be defined as

$$D[u] = \frac{1}{2} \int_{\Omega} |\nabla u|^2 d\Omega, \tag{7.1}$$

for a field u and region Ω (Courant and Hilbert, 1989b). This integral arises in many physical situations, including heat transfer, electrostatics and random walks.

A **harmonic function** is a function that satisfies the **Laplace equation**

$$\nabla^2 u = 0. \tag{7.2}$$

The problem of finding a harmonic function subject to its boundary values is called the **Dirichlet problem**. The harmonic function that satisfies the boundary conditions minimizes the Dirichlet integral, since the Laplace equation is the Euler-Lagrange equation for the Dirichlet integral (Morse and Feshbach, 1953). In a graph setting, points for which there exist a fixed value (e.g., data nodes) are termed **boundary points**. The set of boundary points provides a Dirichlet boundary condition. Points for which the values are not fixed (e.g., missing data) are termed **interior points**.

7.2.2 Interpolation

Solutions to the Laplace equation with specified boundary conditions are harmonic functions, by definition. Finding a harmonic function that satisfies the boundary conditions may be viewed as a method for finding values on the interior of the volume that interpolate between the boundary values in the "smoothest" possible fashion (Courant and Hilbert, 1989b). In this section, we discuss the properties of harmonic functions that make them useful for interpolation, defining smoothness in terms of extremal solutions to the Dirichlet integral.

From a physical standpoint, one may think of a heat source with a fixed temperature at the center of a copper plate and a second heat source with fixed temperature on the boundary of the copper plate. The temperature values taken by the plate at every point are those assumed by a harmonic function subject to the internal and external boundaries imposed by the heat sources. In this analogy, the temperatures measured on the inside of the copper plate may be viewed as smoothly interpolated between the temperature on the internal heat source and the external heat source. The internal and external heat sources are considered to be *boundary* points, while points on the copper plate for which temperature values are found are *interior* points.

Three characteristics of harmonic functions are attractive qualities for generating a "smooth" interpolation.

1. The *mean value theorem* states that the value at each point in the interior (i.e., not a boundary point) is the average value of its neighbors (Ahlfors, 1966).

2. The *maximum principle* follows from the mean value theorem. It states that harmonic functions may not take values on interior points that are greater (or less) than the values taken on the boundary (Ahlfors, 1966).

3. The Dirichlet integral is minimized by harmonic functions (Courant, 1950). This means that the integral of the gradient magnitudes for the system will be minimized, subject to fixed boundary conditions.

7.2.3 Combinatorial formulation: Differential operators on graphs

Using the combinatorial formulations of continuum operators developed in Chapter 2, we can determine how to solve for the harmonic function that interpolates values on free ("interior") nodes between values on fixed ("boundary") nodes.

A combinatorial formulation of the Dirichlet integral (7.1) is

$$D[u] = \frac{1}{2}(Au)^T C(Au) = \frac{1}{2}u^T L u \tag{7.3}$$

and a combinatorial harmonic is a function u that minimizes (7.3). Since L is positive semi-definite, the only critical points of $D[u]$ will be minima.

If we want to fix the values of boundary nodes and compute the interpolated values across interior nodes, we may assume without loss of generality that the nodes in L and u are ordered such that boundary nodes are first and interior nodes are second. Therefore, we may decompose (7.3) into

$$D[u_i] = \frac{1}{2}\begin{bmatrix} u_b^T u_i^T \end{bmatrix} \begin{bmatrix} L_b & R \\ R^T & L_i \end{bmatrix} \begin{bmatrix} u_b \\ u_i \end{bmatrix} = u_b^T L_b u_b + 2u_i^T R^T u_b + u_i^T L_i u_i. \tag{7.4}$$

where u_b and u_i correspond to the potentials of the boundary and interior nodes respectively. Differentiating $D[u_i]$ with respect to u_i and finding the critical point, yields

$$L_i u_i = -R^T u_b, \tag{7.5}$$

which is a system of linear equations with $|u_i|$ unknowns. If the graph is connected, or if every connected component contains a boundary node, then (7.5) will be nonsingular (Biggs, 1974). Although various methods exist for solving a system of linear equations (Golub and Van Loan, 1996; Hackbusch, 1994), the conjugate gradient method is arguably the best in terms of speed and parallelization (Dongarra et al., 1991). Conjugate gradients requires one sparse matrix multiply per iteration, which is bounded above by $d_{\max}s$, where $d_{\max}$ is the maximum degree of an interior node and s is the cardinality of the set of interior nodes. Assuming a constant number of iterations are required for convergence and that the maximum degree is independent of the number of nodes (e.g., a 4-connected lattice), the time complexity of the algorithm is $\mathcal{O}(s)$.

Combinatorial harmonic functions arise in a wide variety of applications, playing a central role in systems of springs (Strang, 1986), the stress and strain of connected beams (Strang, 1986), Markov chains (Doyle and Snell, 1984) and electrical circuits (Doyle and Snell, 1984). As an example, we will examine the application domain of electrical circuits. The other contexts are essentially identical, differing mainly in language and physical meaning of the respective equations (see Strang (1986) for a full discussion). The electrical metaphor, however, is of greater interest in the present context since there is some chance that a VLSI implementation of these methods is possible in terms of the equivalent circuits presented here.

Using the formulation of the fundamental equations of circuit theory (2.3) developed in Chapter 2, we may write the power, P, associated with a circuit as

$$P = \frac{1}{2} y^T C^{-1} y = \frac{1}{2} x^T L x. \tag{7.6}$$

A comparison of (7.3) and (7.6) demonstrates that the set of electric potentials at the nodes of a circuit is a discrete harmonic function, i.e., those nodes with a fixed potential due to voltage sources or grounding are the *boundary* nodes, the nodes without a fixed potential are the *interior* nodes. Furthermore, the interior nodes assume potentials that minimize (7.3) (see Doyle and Snell (1984) for extensive discussion of electrical networks, random walks and the Dirichlet integral). If one were to build a circuit with the same topology as a graph, with appropriate voltage sources to encode the boundary values and resistors to encode the weights, the physical solution (i.e., a minimum energy solution) to the interpolation problem would be exactly equal to the nodal potentials of every interior

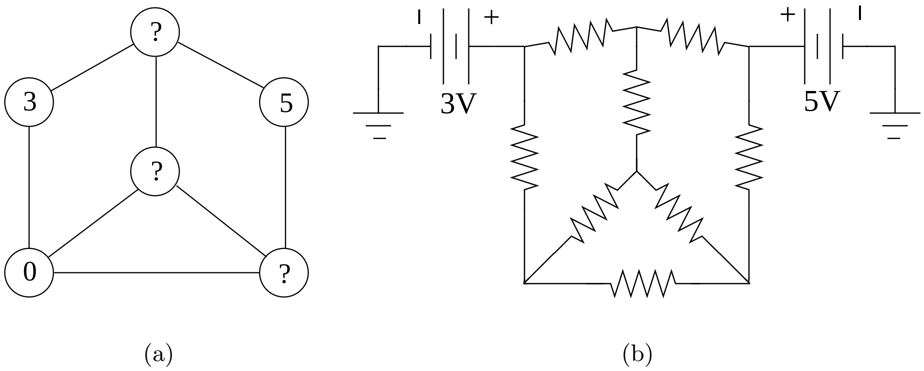


Figure 7·1: Interpolating on a graph with a harmonic is equivalent to setting voltage sources (and grounds) at some nodes and reading off the potentials at nodes which are not fixed. (a) A graph with known values on some nodes and unknown values (indicated with a '?') on other nodes. (b) The equivalent circuit that would produce potentials on the nodes equal to those found by the interpolation method.

node. Figure 7·1 illustrates the circuit corresponding to a graph interpolation problem.

7.3 Results

In this section we demonstrate the interpolation algorithm in the context of image processing.

7.3.1 Space-variant (foveal) images

In order to demonstrate the use of the interpolation algorithm on an arbitrary graph, we imported the Lena image to a graph patterned after the space-variant sampling of the macaque foveal visual system (Schwartz, 1977; Wallace et al., 1994).

Here, we have removed image data in a circular region and performed the interpolation obtained via (7.5) to fill in the lost values. No weighting was used to compensate for the changing length (if embedded in a Euclidean plane) of the edges. In other words, the interpolation was *isotropic* in the sense that every edge had unit length (corresponding to unit resistors in the circuit analogy). The results are displayed in Figure 7·2. One can see that the region of the graph for which image values were removed take values that smoothly interpolate between the dark and light regions. However, since no image information is encoded into the structure (i.e., uniform weights), the interpolation algorithm simply fills in the region with a smooth solution. In the next section, it will be shown that encoding image information in the weights and performing an anisotropic interpolation provides a solution that resembles the missing (original) values more than the isotropically interpolated solution.

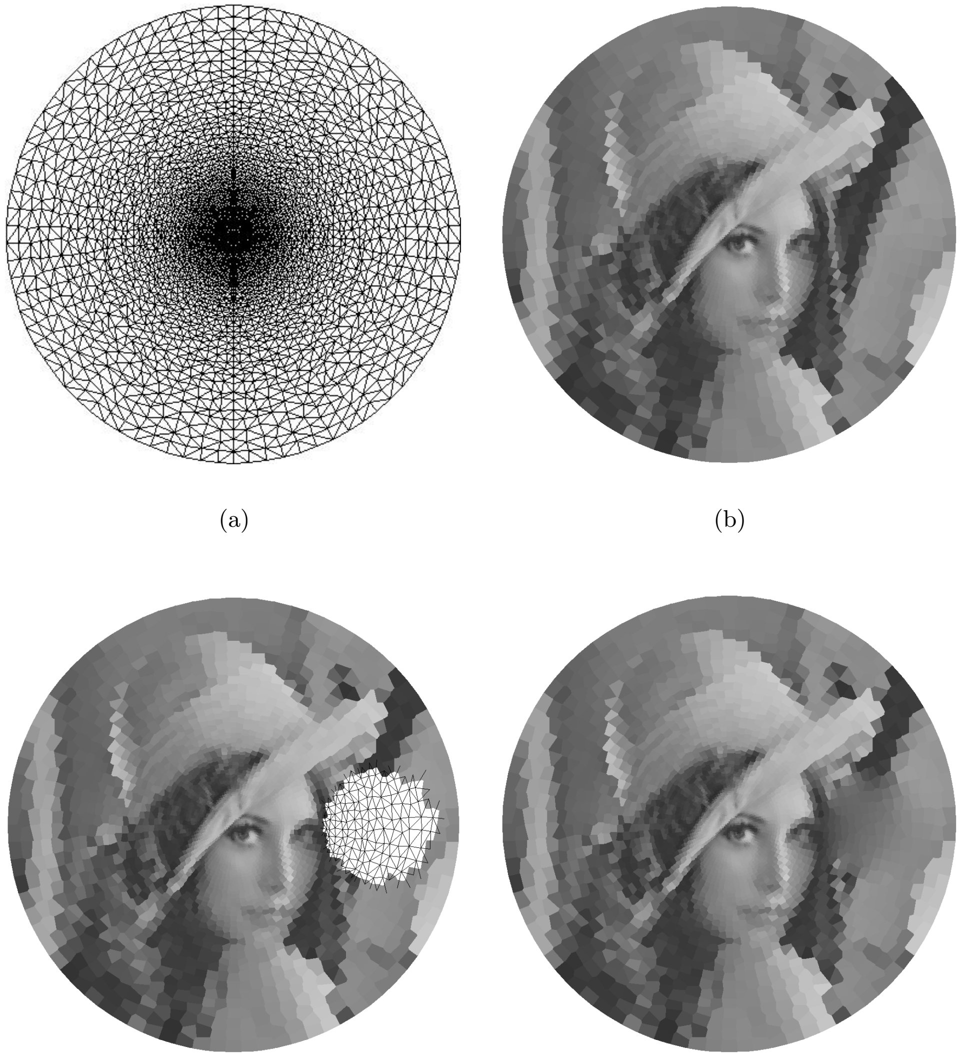
7.3.2 Anisotropic interpolation

Anisotropic interpolation may be thought of as weighted interpolation or as finding the potentials in a resistive network in which the resistor values are nonuniform. It is possible to return to the missing data situation of Figure 7·2 and perform *anisotropic* interpolation using weights derived from the image values (acquired before the data was removed). We employed a Gaussian weighting function of (2.13).

Weighting the space-variant mesh in accordance with (2.13) allows for a more accurate reconstruction of the missing data values, as seen in Figure 7·3.

Building a weighted (i.e., anisotropic) graph for an image using (2.13) allows for a smoothed reconstruction of the original image via anisotropic interpolation from the sampling of a small number of points. These reconstructed images resemble those produced by anisotropic diffusion methods. This is because the solution to the Laplace equation is the steady state of the diffusion equation with specified boundary conditions (Doyle and Snell, 1984). The primary difference between diffusion-based methods of image enhancement and those presented here is that diffusion methods approach zero (or constant) when run for infinite time, since Dirichlet boundary conditions are usually not specified in diffusion approaches to image processing. Because of this, diffusion methods (both isotropic and anisotropic) require a stopping condition, while the present method solves directly for a time-independent solution.

This relationship may be seen even more clearly by comparing the equivalent circuit for our interpolation algorithm and the equivalent circuit presented for anisotropic diffusion



(a)

(b)

(c)

(d)

Figure 7·2: Interpolation of image data on a foveal mesh from which a hole has been cut out. (a) Underlying foveal graph structure. (b) The Lena image imported onto the foveal structure. (c) Foveal image with a hole arbitrarily cut out of it. Underlying graph structure is shown inside the hole. (d) Foveal image with interpolated data in the hole.

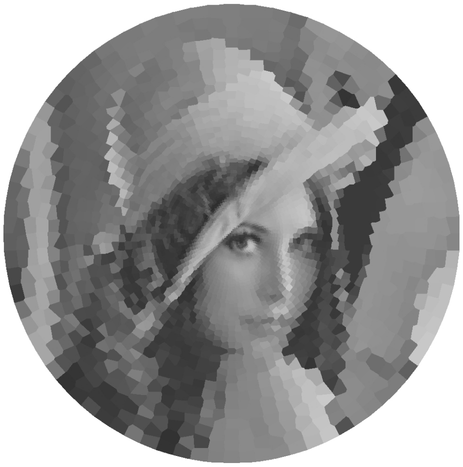


Figure 7·3: Anisotropic interpolation of image data on a foveal mesh with the same hole as in Figure 7·2 has been cut out. Weights were determined using $\beta^1 = 95$ (see text for details)

by Perona and Malik (1990). If one replaced the voltage source at every node in our circuit with an appropriately charged capacitor, then the Perona-Malik equivalent circuit would be obtained exactly. Insofar as similar results are produced for image enhancement tasks with (steady state) anisotropic diffusion and the present method, two advantages of anisotropic interpolation present themselves over diffusion. The first of these is that the solution to the Laplace equation is a steady state solution, while the solution to the diffusion equation depends on time. Therefore, we have no need to iterate and, thus, we circumvent the need to choose a stopping point for the diffusion. Secondly, we can smooth less or smooth more in different areas of the image by decreasing the sampling density in areas where we desire more smoothing and increasing it in areas where we desire less smoothing.

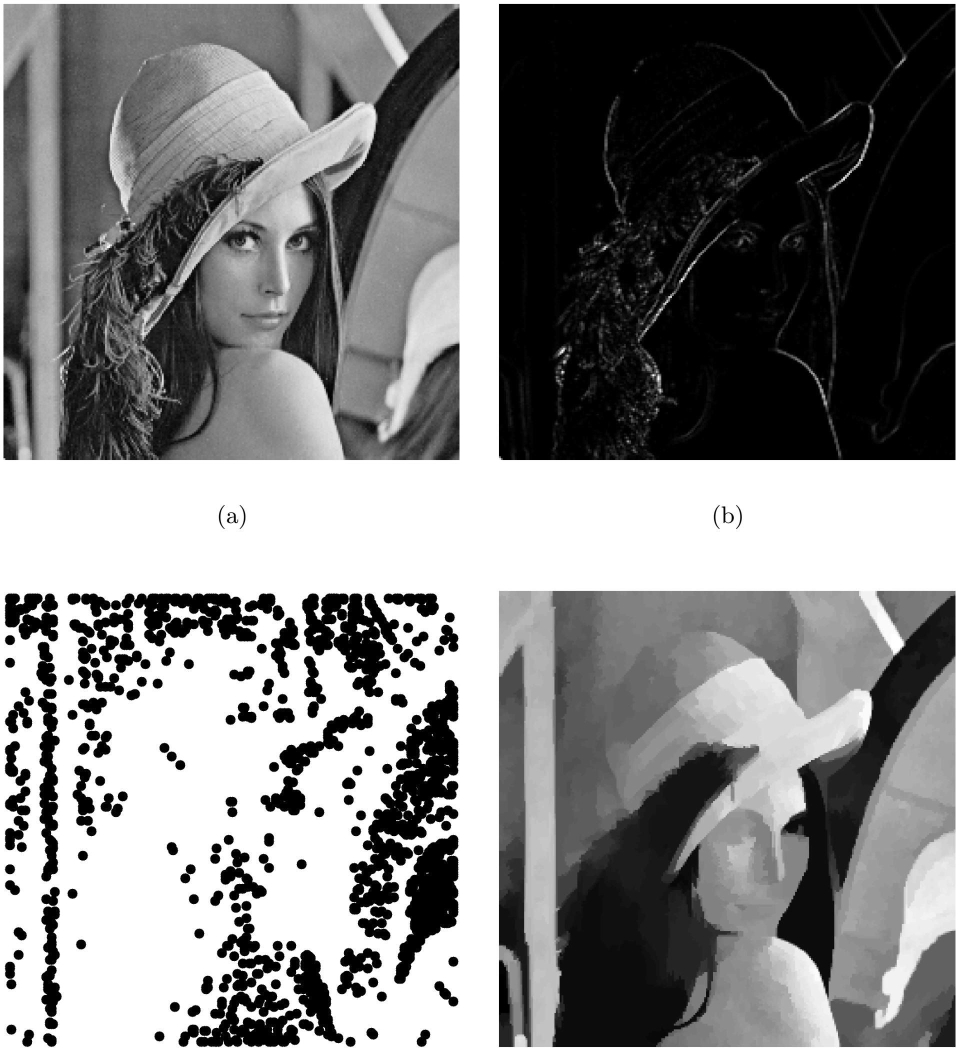
Figure 7·4 demonstrates results that are visually comparable to anisotropic diffusion applied to the same image. To generate Figure 7·4, a 4-connected lattice was generated with weights obtained from (2.13) based on the Lena image. Samples were chosen from relatively uniform areas by computing the square root of the sum of the edge gradients incident on each node. All nodes with a value below a threshold were selected as sample nodes to have their values fixed. The remaining nodes were anisotropically interpolated, given the fixed set. One can see that sharp boundaries are maintained, due to the encoding of image information with weights. Areas of the image with high variability (e.g., the feathers) are smoothed considerably since very few samples were taken, while areas with initially low variability remain uniform.

Of course, it is possible to interpolate by a variety of sampling strategies. Figure 7·5 illustrates the results of different structured sampling schemes, as well as the flexibility to smooth more or smooth less in different areas of the image. Using the same weights as in Figure 7·4, but a different choice of samples, it is possible to keep the center of the image true to the original while diffusing out the background or *vice versa*.

7.4 Conclusion

We have posed the question of how to interpolate nodal values on a graph and proposed a solution based on solving the combinatorial Dirichlet problem. This interpolation method has desirable properties as a result of the mean value theorem and the maximum principle. Furthermore, the method naturally incorporates a metric into the interpolation if anisotropic interpolation is desired. Finally, a circuit analogy was presented which both affords additional intuition into the process as well as holding open the possibility for a VLSI implementation.

Applications of this method to image processing demonstrate its use for filling in missing values in a space-variant image and in the anisotropic smoothing of Cartesian images. Further applications include a smoothing operator for multiresolution reconstruction of graph-based pyramids or three-dimensional interpolation for surfaces. Graphs are general structures that may arise in three dimensions for the purpose of computer graphics (Taubin, 1995) or in an arbitrary number of dimensions for data clustering (Jain et al., 1999). Since this interpolation method depends only on the topology of the structure and not any information about the dimensionality of the space in which it is embedded, one may interpolate on graph structures existing in arbitrary dimensions possessing an arbitrary metric.



(a)

(b)

(c)

(d)

Figure 7·4: Anisotropic interpolation of an image based on very sparse sampling. (a) Original Lena image. (b) Magnitude of image gradient. (c) Samples taken from lowest magnitude points. (d) Anisotropically interpolated image. $\beta^1 = 95$

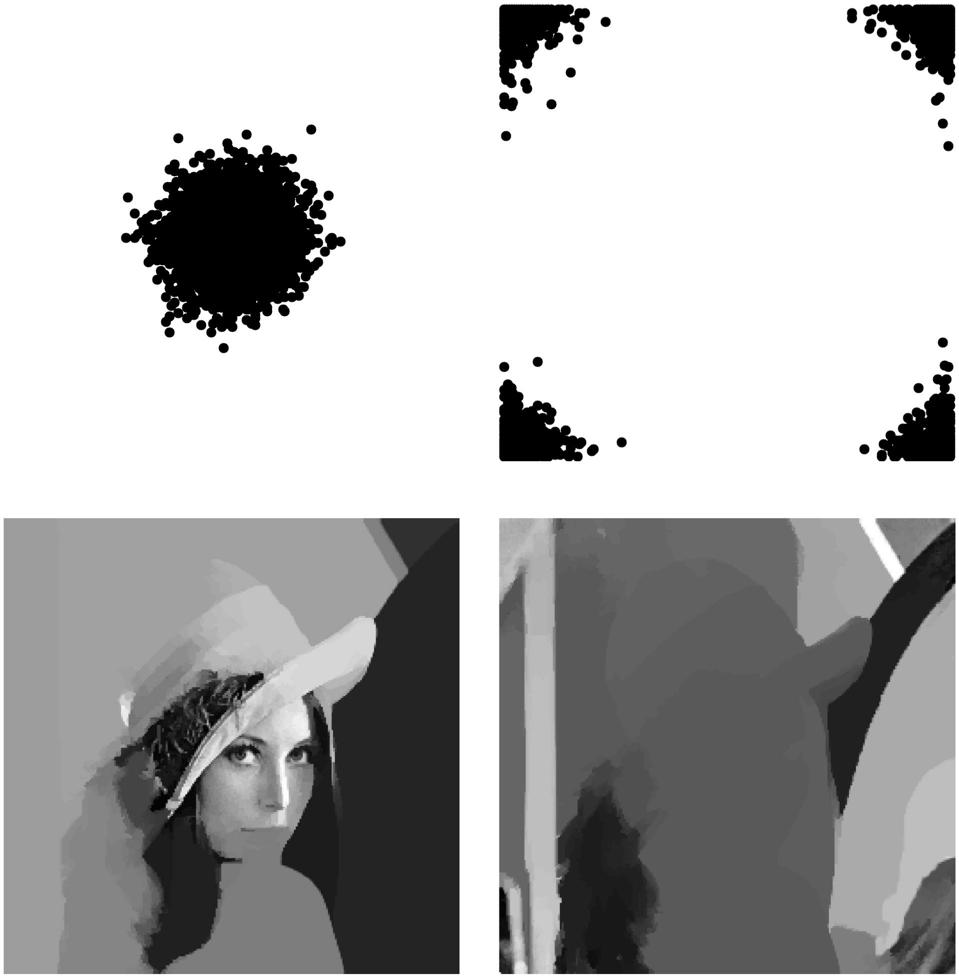


Figure 7·5: Spatially nonuniform sampling allows for more "diffusion" in some areas over others. This figure demonstrates the effects of two different spatial sampling regimes on the anisotropic interpolation of the Lena image. $\beta^1 = 95$

Chapter 8

Conclusion

8.1 Conclusion

If computer vision is going to mimic the space-variant sampling of visual space employed by biological vision systems, then a principled approach to developing algorithms and data structures must be adopted. In this thesis, we have proposed the use of combinatorial algorithms to process data on a graph-theoretic structure in order to perform space-variant computer vision. This approach benefits by allowing the architect of an active vision system to incorporate the space-variant sampling most appropriate to the purpose of the system. From the standpoint of image analysis, the analogies to vector calculus of the combinatorial approach allow the translation of many calculus-based computer vision algorithms to the space-variant setting. Finally, the precise analogy between circuit theory and combinatorial operators allows for the possibility of high-speed analog VLSI implementation of the data structures and algorithms. Beyond space-variant computer vision, further development of combinatorial processing algorithms may have application to such diverse fields as computer graphics, parallel processing, neural networks, and numerical linear algebra.

8.1.1 Further directions in biological mimicry

Graph topology plays an essential role in determining both the output of an algorithm and its computational characteristics (e.g., Chapters 5 and 6). Graphs and biological neuronal networks are frequently identified with each other, as in the neural network literature. In this setting, information carried by ganglion cells is treated as representing nodal information. As in the image processing setting, the identification of an edge set is much clear. Typically, the neural network literature employs a feedforward connectivity to deeper layers of nodes (Bishop, 1995). However, the combinatorial algorithms proposed here require, and are highly dependent on the choice of, lateral connections. Therefore, to pursue a more biological interpretation of the work here, a thorough investigation of lateral connections in neural systems would have to be investigated.

Hughes (1977) discusses various proposals that account for the sampling of visual space employed by different biological vision systems. For example, the “horizontal streak” is found in many species that live in open (i.e., non-occlusive) environments, such as rabbits, great cats and ungulates. It has been proposed (see Hughes (1977) for a history of this suggestion) that species living in open environments possess a horizontal streak because most objects of interest in their visual field will converge to the horizon. Therefore, a heavy sampling of the region around the horizon confers a selective advantage. However, it has been difficult to make such an argument precise, quantify the difference between alternate sampling strategies or to reverse engineer a new sampling strategy for an artificial system that will be operating in a well-characterized visual environment. The software given in the Graph Analysis Toolbox for importing images to different visual sampling structures allows one to begin to approach these issues by offering a way to compare the performance of image analysis algorithms using different sampling strategies on the same set of images. This allows both a comparison of existing sampling strategies on image analysis, as well as allowing the designer of an artificial vision system to try new sampling strategies and compare their performance on the tasks of interest. Additionally, researchers in the organization of retinal topography may employ this system to perform hypothesis testing of different proposals, such as the usefulness of a horizontal streak. The discipline of spatial statistics (Cliff and Ord, 1981; Diggle, 1983; Upton and Fingleton, 1985) includes tools for performing hypothesis tests of spatial autocorrelation, given explicitly on graphs.

8.1.2 Further directions in analysis

Mathematical morphology and image algebra (Ritter and Wilson, 1996) explicitly adopt a set-theoretic, algebraic and topological approach to computer vision. Since combinatorial algorithms have a similar focus, the body of theory already developed in the mathematical morphology literature could probably be exploited to yield further advances in the approach to space-variant vision outlined in this thesis.

Compression of data on a graph is an interesting area of unexplored territory. The generalization of spectral analysis to shift-variant graphs Taubin (1995); Taubin et al. (1996) suggests that compression of signals on a graph-theoretic architecture might follow the development of traditional, lattice-based, signal compression. However, since compression techniques rely on a knowledge of the statistics of natural images (Field, 1987; Ruderman and Bialek, 1994), optimization of a compression technique would have to involve a similar study on the statistics of images with a given space-variant architecture.

Finally, nodes need not contain pixel information, but may rather represent image regions, objects or other features (Perona and Freeman, 1998). Since an architecture that used nodes to represent image regions would have many fewer nodes than a straightforward identification of nodes with pixels, many of the same justifications for space-variant image processing (e.g., speed) would apply to such a representation. Therefore, a promising direction for application of the data structures and algorithms outlined in this thesis would be to identify characteristic graph-based representations of images that could be coupled with the analysis presented here to yield fast, quality results.

Appendix A

Appendices

A.1 Function list

This appendix is a reproduction of the `Contents.m` file included in the Graph Analysis Toolbox

Support for space-variant images.	
`contour2pdf.m`	Convert a contour map to a probability density function.
`ellipsefit.m`	Fit an ellipse to a polygon with least-square error.
`findfilter.m`	Compute resampling filters for a point set.
I/O on space-variant graphs.	
`importimg.m`	Import a Cartesian (standard) image to a graph.
`showmesh.m`	Visualize 2D data (e.g., an image) on a graph by interpolating data across the faces of the nodes.
`showvoronoi.m`	Visualize 2D data (e.g., an image) on a graph by uniformly filling the Voronoi cell of each node with its value.
`voronoicells.m`	Compute Voronoi information of a graph for visualization.
Data processing on graphs.	
`diffusion.m`	Diffuse data on a graph.

`dirichletboundary.m`	Solve the combinatorial Dirichlet problem on a graph (e.g., interpolate missing data).
`filtergraph.m`	Filter data on a graph.
`findedges.m`	Detect edges in data on a graph.
`imgsegment.m`	Segment a Cartesian (standard) image using a lattice.
`imgsegpyr.m`	Segment a Cartesian (standard) image using a pyramid.
`isosolve.m`	Perform the calculations required by the isoperimetric algorithm.
`makeweights.m`	Convert nodal graph data to edge weights.
`partitiongraph.m`	Segment data on an arbitrary graph.
`recursivepartition.m`	Recursively segment data on an arbitrary graph.

Generating new graphs.

`addrandedges.m`	Add random edges to “small worldify” a graph.
`latticepyramid.m`	Generate a connected pyramid from a Cartesian lattice.
`knn.m`	Connect nodes to their nearest neighbors.
`lattice.m`	Generate a Cartesian lattice with varying connectivity.
`logz.m`	Generate a point set using the $w = \log(z + a)$ function describing the macaque retinotopic map.
`roach.m`	Generate the “roach” graph of Guattery and Miller.
`triangulatepoints.m`	Compute an triangulated edge set for an input node set.

Graph matrix generation.

`adjacency.m`	Generate the adjacency matrix for a node/edge set.
`incidence.m`	Generate the incidence matrix for a node/edge set.
`laplacian.m`	Generate the Laplacian matrix for a node/edge set.

Support functions.

`adjtoedges.m`	Convert an adjacency matrix to an edge list.
`binarysearch.m`	Perform a binary search of a vector.
`circulant.m`	Generate a circulant matrix (similar to `Toeplitz.m`).
`colorseg2bwseg.m`	Convert a segmentation indicated with color to a publishable (B&W) format.
`colorseg2bwsegSV.m`	Convert a space-variant segmentation indicated with color to a publishable (B&W) format.
`equalize.m`	Perform histogram equalization of a data vector.
`normalize.m`	Normalize data (columnwise) to a specified range.
`removeisolated.m`	Remove any isolated nodes in a graph.
`rgbimg2vals.m`	Vectorize an RGB image.
`segoutput.m`	Convert a segmentation labeling of a lattice to a better visualization.
`segoutputSV.m`	Convert a segmentation labeling of an arbitrary graph to a better visualization.

A.2 Demo scripts

This section provides a list of demo scripts included in the extended package. The information presented here is also included in the `Contents.m` file included in the DEMOS directory of the Graph Analysis Toolbox.

Edge finding.

`findEdgesDemo.m`	Compute edges for a Cartesian image using the

	gradient and Laplacian edge detectors.
`findEdgesDemoSV.m`	Compute edges for a space-variant image using the gradient and Laplacian edge detectors.
Graph filtering.	
`diffusionDemo.m`	Compute iso/anisotropic diffusion on a Cartesian image.
`filterCoordDemo.m`	Filter coordinate data.
`filterImageDemo.m`	Filter a space-variant image.
`interpolationFilterDemo.m`	Use anisotropic interpolation as an image filter.
Graph drawing.	
`drawGraphDemo.m`	Use isotropic interpolation to smooth a graph drawing.
Image interpolation.	
`fovealAnisotropicDemo.m`	Perform anisotropic interpolation on a missing image region.
`fovealIsotropicDemo.m`	Perform isotropic interpolation on a missing image region.
`cartesianAnisotropicDemo.m`	Perform anisotropic interpolation based on sampling different regions of the image.
Importing/Visualization.	
`buildFiltersDemo.m`	Generate importing filters for a random point set.
`contour2graphDemo.m`	Generate a graph from a retinal topography contour image.
`differentFoveationDemo.m`	Foveate on different points in a larger image.

`ellipseDisplayDemo.m`	Fit ellipses to Voronoi cells of a randomly generated point set.
`generateSVgraphsDemo.m`	Generate graphs and filters from the existing set of retinal topography images.
`importVisualizationDemo.m`	Import a Cartesian image to an `imgGraph` and visualize.

Segmentation.

`clusterPointsDemo.m`	Cluster a point set (segmentation on coordinates).
`segmentationSVDemo.m`	Segment an `imgGraph`.
`segmentationCompareDemo.m`	Compare segmentation of a Cartesian image generated by different algorithms.

Pyramids.

`pyramidSegmentationDemo.m`	Compute a segmentation using a pyramid architecture.

Graph generation.

`connectGraphDemo.m`	Computes and compares graphs with different topology and geometric arrangement.

A.3 Standardized variable names

Throughout the functions, documentation and demos, a set of standardized variable names are used. The list of variable names and their meanings is given below, and a reproduction of this list is included in the file `variableNames.txt`.

Scalars.

`Q`	Cardinality of faces set in a graph.

`K`	Dummy constant.
`N`	Cardinality of node set in a graph.
`M`	Cardinality of edge set in a graph.
`P`	Number of coordinate dimensions of a node set.
`scale`	The weighting function parameter.
`stop`	The recursion stop parameter.
`X/Y/Z`	Dimensions of an image or region (e.g., `[X Y Z]=size(img)`).

Matrices.

`W`	The $N \times N$ adjacency matrix.
`D`	The $N \times N$ diagonal matrix of node degrees.
`img`	The current image.
`L`	The $N \times N$ Laplacian matrix.
`A`	The $M \times N$ edge-node incidence matrix.

Vectors.

`d`	The $N \times 1$ vector of node degrees.
`vals`	The $N \times K$ vector of $K-$dimensional nodal values (e.g., RGB, with $K = 3$).
`weights`	The $M \times 1$ vector of edge weights.

Graph components.

`points`	An $N \times P$ list of node coordinates.
`edges`	An $M \times 2$ list of edges (containing indices to the node set).
`faces`	A $Q \times K$ list of polygonal faces with order $\leq K$.

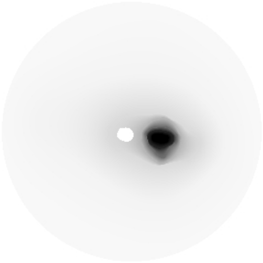
Structs.

`imgGraph`	Struct containing the filters for importing an image to a

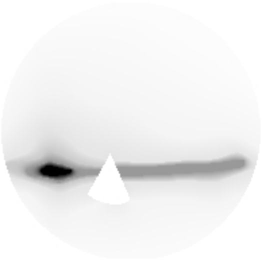
	space-variant graph.
`imgGraph.pntMap`	A $K \times 2$ list of the K points in the image plane used to filter an image for importing.
`imgGraph.breakpoints`	A $1 \times N$ list of the breakpoints in `imgGraph.pntMap` referring to the points corresponding to different nodes.
`imgGraph.filtWeights`	A $K \times 2$ list of the filter weights for each of the K points in `imgGraph.pntMap`.
`voronoiStruct`	Struct containing the information necessary to perform Voronoi visualization on a space-variant image.
`voronoiStruct.pts`	$K \times 2$ list of coordinates for the vertices of the Voronoi cells for the node set, where $K > N$.
`voronoiStruct.index`	List of nodes that are represented in the visualization (i.e., nodes with a Voronoi cell within the convex hull of the node set).
`voronoiStruct.faces`	List of faces representing the Voronoi cells, to be used by `patch.m`.

A.4 Included retinal topographies

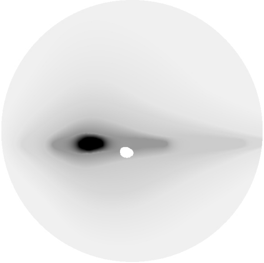
This section shows the probability density function (PDF) obtained from the retinal topographies of ganglion cell density for the species included in the Graph Analysis Toolbox. Darker areas represent higher ganglion cell density, while lighter areas represent a lower ganglion cell density. In addition to having different visual sampling arrangements, different species have varying degrees of nonuniformity in the sense that the discrepancy between the most dense and most sparse regions of ganglion cells may be $2:1$ in some species and $100:1$ in others. Since each image given below is normalized to have unity sum, the species



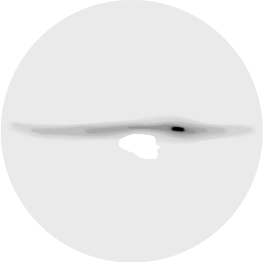
(a) Baboon (Whitteridge, 1965)



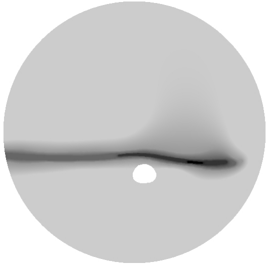
(b) Beagle (Peichl, 1992)



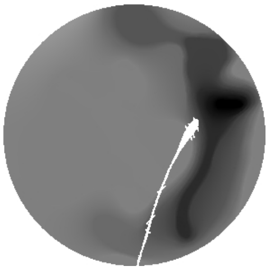
(c) Cat (Hughes, 1975)



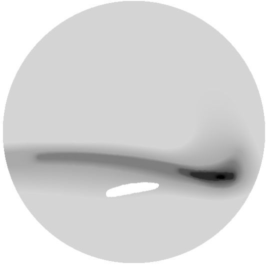
(d) Cheetah (Hughes, 1977)



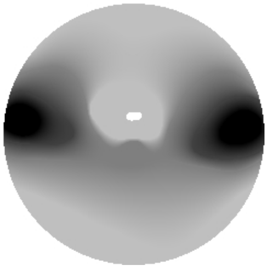
(e) Cow (Hughes, 1977)



(f) Deep-sea bass (Collin and Partridge, 1996)

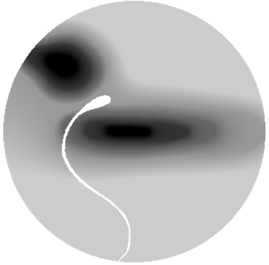


(g) Deer (Hughes, 1977)

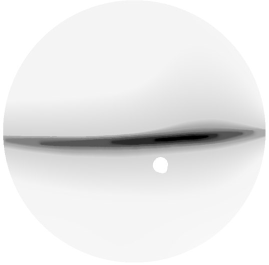


(h) Bottlenosed dolphin (Mass and Supin, 1995)

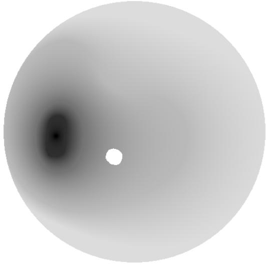
(i) German shepherd (Peichl, 1992)



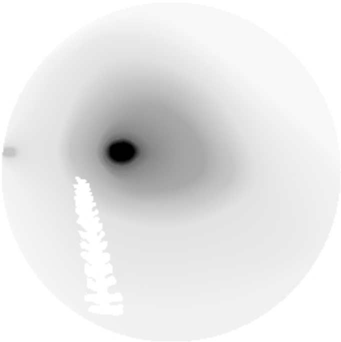
(j) Harlequin tusk fish (Collin and Pettigrew, 1988)

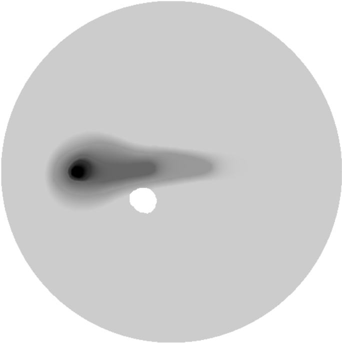


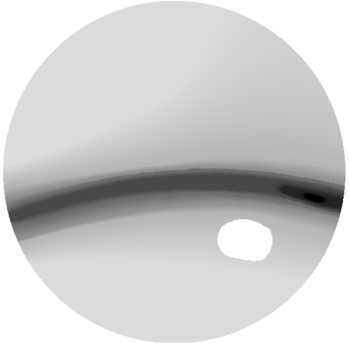
(k) Plains kangaroo (Hughes, 1974)

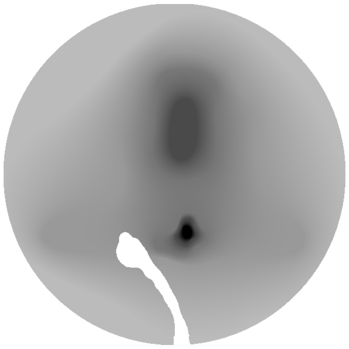


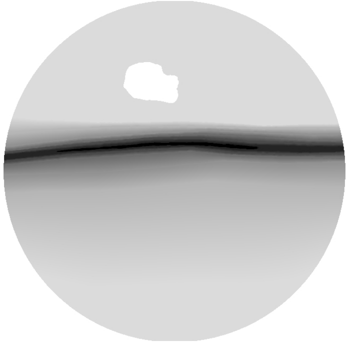
(l) Tree kangaroo (Hughes, 1974)

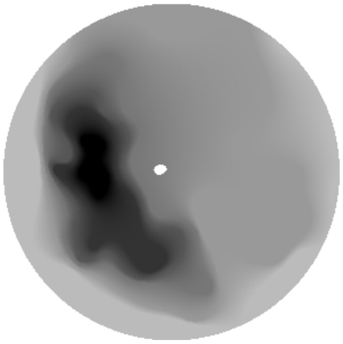

(m) Sacred kingfisher (Moroney and Pettigrew, 1987)

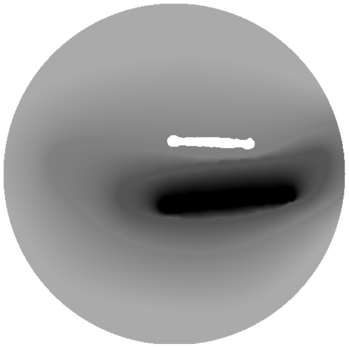

(n) Labrador (Hughes, 1977)

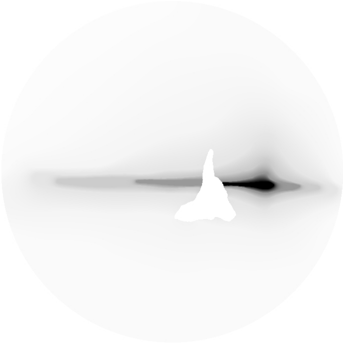

(o) Pig (Hughes, 1977)

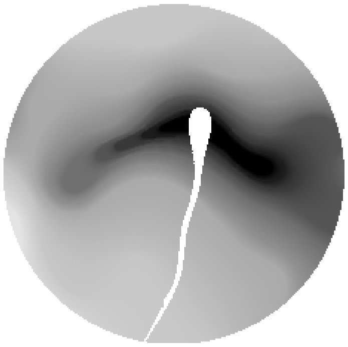

(p) Pigeon (Whitteridge, 1965)


(q) Rabbit (Hughes, 1971)


(r) Two-toed sloth (Costa et al., 1989)


(s) Squirrel (Hughes, 1977)


(t) Wolf (Peichl, 1992)


(u) Yellow-finned trevally (Collin, 1999)

with a greater ratio of dense to sparse areas are displayed as nearly white with a small dark region, while species having a lower ratio of dense to sparse regions are displayed as a more uniform gray. However, since the topographic maps for different species were studied by different researchers who stopped counting cell densities at different points in the retinal periphery, the topographies (and hence the images here) may or may not represent an actual comparison between species as regards the discrepancy between the region of highest cell density and the region of lowest cell density.

Bibliography

Acton, S. T. (1996). A pyramidal edge detector based on anisotropic diffusion. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 4, pages 2215–2218. Signal Process. Soc. IEEE, IEEE.

Ahlfors, L. (1966). *Complex Analysis.* McGraw-Hill, New York.

Alon, N. (1986). Eigenvalues and expanders. *Combinatorica*, 6:83–96. 600.

Alon, N. and Milman, V. (1985). λ_1, isoperimetric inequalities for graphs and superconcentrators. *Journal of Combinatorial Theory, Series B*, 38:73–88.

Alpert, C. J. and Kahng, A. B. (1995). Recent directions in netlist partitioning: A survey. *Integration: The VLSI Journal*, 19:1–81.

Anderson, Jr., W. N. and Morley, T. D. (1971). Eigenvalues of the Laplacian of a graph. Technical Report TR 71-45, University of Maryland.

Arfken, G. and Weber, H.-J., editors (1985). *Mathematical Methods for Physicists.* Academic Press, 3rd edition.

Baak, D. A. V. (1998). Variational alternatives of Kirchhoff's loop theorem in DC circuits. *American Journal of Physics*, 67(1):36–44.

Barnard, S. T. and Simon, H. D. (1994). A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. Technical Report RNR-92-033, NASA Ames Research Center, Moffett Field, CA.

Biggs, N. (1974). *Algebraic Graph Theory.* Number 67 in Cambridge Tracts in Mathematics. Cambridge University Press.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, England.

Bonmassar, G. and Schwartz, E. L. (1997). Space variant fourier analysis: The exponential chirp transform. *IEEE Pattern Analysis and Machine Intelligence*, 19(10):1080–1089.

Branin, Jr., F. H. (1963). The inverse of the incidence matrix of a tree and the formulation of the algebraic-first-order differential equations of an RLC network. *IEEE Transactions on Circuit Theory*, 10(4):543–544.

Branin, Jr., F. H. (1966). The algebraic-topological basis for network analogies and the vector calculus. In *Generalized Networks, Proceedings*, pages 453–491, Brooklyn, N.Y.

Burrough, P. A. (1986). *Principles of geographical information systems for land resources assessment.* Number 12 in Monographs on soil and resources survey. Clarendon Press, Oxford.

Burt, P. J. and Adelson, E. H. (1983). The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, COM-31(4):532–540.

Burt, P. J., Hong, T.-H., and Rosenfeld, A. (1981). Segmentation and estimation of image region properties through cooperative hierarchical computation. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-11(12):802–809.

Cao, F., Gilbert, J., and Teng, S.-H. (1996). Partitioning meshes with lines and planes. Technical Report CSL-96-01, Palo Alto Research Center, Xerox Corporation.

Casasent, D. and Psaltis, D. (1976). Position, rotation and scale invariant optical correlation. *Applied Optics*, 15(7):179–187.

Chan, T. F., Gilbert, J. R., and Teng, S.-H. (1994). Geometric spectral partitioning. Technical Report CSL-94-15, Palo Alto Research Center, Xerox Corporation.

Cheeger, J. (1970). A lower bound for the smallest eigenvalue of the Laplacian. In Gunning, R., editor, *Problems in Analysis*, pages 195–199. Princeton University Press, Princeton, NJ.

Chen, G.-N. (2001). *Fundamental Algorithms of Space-Variant Vision: Non-Uniform Sampling, Triangulation, and Foveal Scale-Space.* PhD thesis, Boston University.

Chen, W. and Acton, S. (1998). Morphological pyramids for multiscale edge detection. In *1998 IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 137–141. University of Arizona, University of Arizona Foundation, Arizona State University, Tucson Section of the IEEE, IEEE.

Chung, F. R. K. (1997). *Spectral Graph Theory.* Number 92 in Regional conference series in mathematics. American Mathematical Society, Providence, R.I.

Cliff, A. D. and Ord, J. K. (1981). *Spatial Processes: Models and Applications.* Pion Limited, London, England.

Collin, S. P. (1999). Behavioural ecology and retinal cell topography. In Archer, S., Djamgoz, M., Loew, E., Partridge, J., and S. Vallerga, D., editors, *Adaptive Mechanisms in the Ecology of Vision*, pages 509–535. Kluwer Academic Publishers.

Collin, S. P. and Partridge, J. C. (1996). Retinal specializations in the eyes of deep-sea teleosts. *Journal of Fish Biology*, 49 (Supplement A):157–174.

Collin, S. P. and Pettigrew, J. D. (1988). Retinal ganglion cell topography in teleosts: A comparison between nissl-stained material and retrograde labelling from the optic nerve. *The Journal of Comparative Neurology*, 276:412–422.

Comer, M. L. and Delp, E. J. (1995). Multiresolution image segmentation. In *1995 International Conference on Acoustics, Speech, and Signal Processing. Conference Proceedings*, volume 4, pages 2415–2418. Signal Processing Society of the IEEE, IEEE.

Costa, B. L. S. A., Pessoa, V. F., Bousfield, J. D., and Clarke, R. J. (1989). Ganglion cell size and distribution in the retina of the two-toed sloth (*choleopus didactylus l.*). *Brazilian Journal of Medical and Biological Research*, 22:233–236.

Courant, R. (1950). *Dirichlet's Principle*. Interscience.

Courant, R. and Hilbert, D. (1989a). *Methods of Mathematical Physics*, volume 1. John Wiley and Sons.

Courant, R. and Hilbert, D. (1989b). *Methods of Mathematical Physics*, volume 2. John Wiley and Sons.

Cvetković, D. M., Doob, M., and Sachs, H. (1995). *Spectra of Graphs: Theory and Applications*. Johann Ambrosious Barth Verlag, Heidelberg–Leipzig, Germany, 3rd revised and enlarged edition edition.

Davis, L. S. (1975). A survey of edge detection techniques. *Computer Graphics and Image Processing*, 4:248–270.

Diggle, P. J. (1983). *Statistical Analysis of Spatial Point Patterns*. Mathematics in Biology. Academic Press, New York, USA.

Dodziuk, J. (1984). Difference equations, isoperimetric inequality and the transience of certain random walks. *Transactions of the American Mathematical Soceity*, 284:787–794.

Dodziuk, J. and Kendall, W. S. (1986). Combinatorial Laplacians and the isoperimetric inequality. In Ellworthy, K. D., editor, *From local times to global geometry, control and physics*, volume 150 of *Pitman Research Notes in Mathematics Series*, pages 68–74. Longman Scientific and Technical.

Donath, W. and Hoffman, A. (1972). Algorithms for partitioning of graphs and computer logic based on eigenvectors of connection matrices. *IBM Technical Disclosure Bulletin*, 15:938–944.

Dongarra, J. J., Duff, I. S., Sorenson, D. C., and van der Vorst, H. A. (1991). *Solving Linear Systems on Vector and Shared Memory Computers*. Society for Industrial and Applied Mathematics, Philadelphia.

Doyle, P. and Snell, L. (1984). *Random walks and electric networks*. Number 22 in Carus mathematical monographs. Mathematical Association of America, Washington, D.C.

Elder, J. H. and Goldberg, R. M. (2001). Image editing in the contour domain. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3):291–296.

Eppstein, D. (1992). Sparsification–a technique for speeding up dynamic graph algorithms. In *Proceedings 33rd Annual Symposium on Foundations of Computer Science*, pages 60–69, Pittsburgh, PA. IEEE, AT&T Bell Lab., IBM, Istituto di Analisi dei Sistemi Ed Inf., Xerox, IEEE Computer Soceity Press.

Erdös, P. and Renyi, A. (1959). On random graphs. I. *Publicationes Mathematicae Debrecen*, 6:290–297.

Erdös, P. and Renyi, A. (1960). On the evolution of random graphs. *Közlemények Publications*, 5:17–61.

Fiedler, M. (1973). Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(98):298–305.

Fiedler, M. (1975a). Eigenvalues of acyclic matrices. *Czechoslovak Mathematical Journal*, 25(100):607–618.

Fiedler, M. (1975b). A property of eigenvalues of nonnegative symmetric matrices and its applications to graph theory. *Czechoslovak Mathematical Journal*, 25(100):619–633.

Fiedler, M. (1986). *Special matrices and their applications in numerical mathematics.* Martinus Nijhoff Publishers.

Field, D. J. (1987). Relations between the statistics of natural images and the response properties of cortical cells. *Journal of the Optical Society of America A (Optics and Image Science)*, 4(12):2379–2394.

Fineman, M. (1996). *The Nature of Visual Illusion.* Dover Publications.

Foulds, L. (1992). *Graph Theory Applications.* Universitext. Springer-Verlag, New York.

Gilbert, J. (1994). Predicting structure in sparse matrix computations. *SIAM Journal of Matrix Analysis and Applications*, 15(1):62–79.

Gilbert, J., Moler, C., and Schreiber, R. (1992). Sparse matrices in MATLAB: Design and implementation. *SIAM Journal on Matrix Analysis and Applications*, 13(1):333–356.

Gilbert, J. R., Miller, G. L., and Teng, S.-H. (1998). Geometric mesh partitioning: Implementation and experiments. *SIAM Journal on Scientific Computing*, 19(6):2091–2110.

Golub, G. and Van Loan, C. (1996). *Matrix Computations.* The Johns Hopkins University Press, 3rd edition.

Gremban, K. (1996). *Combinatorial preconditioners for sparse, symmetric diagonally dominant linear systems.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA.

Gremban, K., Miller, G., and Zagha, M. (1995). Performance evaluation of a new parallel preconditioner. In *Proceedings 9th International Parallel Processing Symposium*, pages 65–69, Santa Barbara, CA. IEEE Computer Society Technical Committee on Parallel Processes, IEEE Computer Society Press.

Guattery, S. and Miller, G. (1998). On the quality of spectral separators. *SIAM Journal on Matrix Analysis and Applications*, 19(3):701–719.

Hackbusch, W. (1994). *Iterative Solution of Large Sparse Systems of Equations*. Springer-Verlag.

Heckbert, P. (1989). Fundamentals of texture mapping and image warping. Master's thesis, University of California at Berkeley.

Hendrickson, B. and Leland, R. (1995). The Chaco user's guide. Technical Report SAND95-2344, Sandia National Laboratory, Albuquerque, NM.

Hu, Y. and Blake, R. (1994). Numerical experiences with partitioning of unstructured meshes. *Parallel Computing*, 20:815–829.

Hughes, A. (1971). Topographical relationships between the anatomy and physiology of the rabbit visual system. *Documents Ophthalmology*, 30:33–159.

Hughes, A. (1974). A comparison of retinal ganglion cell topography in the plains and tree kangaroo. *Journal of Physiology*, 244:61–63P.

Hughes, A. (1975). A quantitative analysis of cat retinal ganglion cell topography. *Journal of Comparative Neurology*, 163:107–128.

Hughes, A. (1977). The topography of vision in mammals of contrasting life style: Comparative optics and retinal organization. In Crescitelli, F., editor, *The Visual system in vertebrates*, volume 7 of *The Handbook of Sensory Physiology*, chapter 11, pages 613–756. Springer-Verlag, Berlin, New York. Part 5.

Jain, A. (1989). *Fundamentals of Digital Image Processing*. Prentice-Hall, Inc.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323.

Karypis, G. and Kumar, V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–393.

Kirchhoff, G. (1847). Ueber die auflösung der gleichungen, auf welche man bei der untersuchung der linearen verteilung galvanischer ströme geführt wird. *Poggendorf's Annalen der Physik Chemie*, 72:497–508.

Knuth, D. E. (1976). Big omicron and big omega and big theta. *SIGACT News*, 8(2):18–24.

Koenderink, J. (1984). The structure of images. *Biological Cybernetics*, 50:363–370.

Koster, A. S. E. e. a. (1997). Heuristic linking models in multiscale image segmentation. *Computer Vision and Image Understanding*, 65(3):382–402.

Kuijaars, A. (2000). Which eigenvalues are found by the Lanczos method? *SIAM Journal of Matrix Analysis and Applications*, 22(1):306–321.

Lehoucq, R. B., Sorenson, D. C., and Yang, C. (1998). *ARPACK User's Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods.* SIAM.

Lin, C.-Y. and Wu, Min, e. a. (2001). Rotation, scale and translation reslilient watermarking for images. *IEEE Transactions on Image Processing*, 10(5):767–782.

Loncaric, S. (1998). A survey of shape analysis techniques. *Pattern Recognition*, 31(8):983–1001.

Marr, D. and Hildreth, E. C. (1980). Theory of edge detection. *Proceedings of the Royal Soceity of London*, 207(1167):187–217.

Mass, A. M. and Supin, A. Y. (1995). Ganglion cell topography of the retina in the bottlenosed dolphin, *tursiops truncatus*. *Brain Behavior and Evolution*, 45:257–265.

Maxwell, J. C. (1991a). *A Treatise on Electricity and Magnestism*, volume 1. Dover, New York, 3rd edition.

Maxwell, J. C. (1991b). *A Treatise on Electricity and Magnestism*, volume 2. Dover, New York, 3rd edition.

Merris, R. (1994). Laplacian matrices of graphs: A survey. *Linear Algebra and its Applications*, 197,198:143–176.

Miller, G. L., Teng, S.-H., Thurston, W., and Vavasis, S. A. (1998). Geometric separators for finite-element meshes. *SIAM Journal on Scientific Computing*, 19(2):364–386.

Miller, G. L., Teng, S.-H., Thurston, W. P., and Vavasis, S. A. (1993). Automatic mesh partitioning. In George, A., Gilbert, J. R., and Liu, J. W. H., editors, *Graph theory and sparse matrix computation*, volume 56 of *The IMA volumes in mathematics and its applications*, pages 57–84. Springer-Verlag.

Mohar, B. (1988). Isoperimetric inequalities, growth and the spectrum of graphs. *Linear Algebra and its Applications*, 103:119–131.

Mohar, B. (1989). Isoperimetric numbers of graphs. *Journal of Combinatorial Theory, Series B*, 47:274–291. 609.

Mohar, B. (1991). The Laplacian spectrum of graphs. In Alavi, Y., Chartrand, G., Oellermann, O. R., and Schwenk, A. J., editors, *Graph Theory, Combinatorics, and Applications*, volume 2, pages 871–898. Wiley.

Moroney, M. K. and Pettigrew, J. D. (1987). Some observations in the visual optics of kingfishers (aves, coraciformes, alcedinidae). *Journal of Comparative Physiology A*, 160:137–149.

Morse, P. M. and Feshbach, H. (1953). *Methods of Theoretical Physics.* McGraw-Hill.

Pachai, C. e. a. (1998). A pyramidal approach for automatic segmentation of multiple sclerosis lesion in brain MRI. *Computerized Medical Imaging and Graphics*, 22:399–408.

Peichl, L. (1992). Topgraphy of ganglion cells in the dog and wolf retina. *The Journal of Comparative Neurology*, 324:603–620.

Perona, P. and Freeman, W. (1998). A factorization approach to grouping. In Burkhardt, H.; Neumann, B., editor, *Computer Vision - ECCV'98. 5th European Conference on Computer Vision. Proceedings*, volume 1, pages 655–670, Freiburg, Germany. Springer-Verlag.

Perona, P. and Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639.

Pothen, A., Simon, H., and Liou, K.-P. (1990). Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Matrix Analysis Applications*, 11(3):430–452.

Pothen, A., Simon, H., and Wang, L. (1992). Spectral nested dissection. Technical Report CS-92-01, Pennsylvania State University.

Preis, R. and Diekmann, R. (1996). The party partitioning - library user's guide - version 1.1. Technical Report TR-RSFB-96-024, University of Paderborn, Paderborn, Germany. Department of Computer Science.

Prewitt, J. M. S. (1970). Object enhancement and extraction. In Lipkin, B. S. and Rosenfeld, A., editors, *Picture Processing and Psychopictorics*, pages 75–149. Academic Press, New York.

Rehrauer, H., Seidel, K., and Datcu, M. (1998). Multiscale image segmentation with a dynamic label tree. In *IGARSS '98. Sensing and Managing the Environment. 1998 IEEE International Geoscience and Remote Sensing. Symposium Proceedings*, volume 4, pages 1772–1774. IEEE, IEEE Geoscience & Remote Sensing Society, University of Washington, NASA, NOAA, Office of Naval Research, National Space Development Agency Japan, URSI, IEEE.

Rezaee, Mahmoud Ramze, e. a. (2000). A multiresolution image segmentation technique based on pyramidal segmentation and fuzzy clustering. *IEEE Transactions on Image Processing*, 9(7):1238–1248.

Ritter, G. X. and Wilson, J. N. (1996). *Handbook of Computer Vision Algorithms in Image Algebra.* CRC Press.

Roberts, L. G. (1965). Machine perception of three-dimensional solids. In Tippet, J. T., editor, *Optical and Electro-optical Information Processing*, pages 159–197. MIT Press, Cambridge, MA.

Rockafellar, T. (1984). *Network Flows and Monotropic Optimization.* John Wiley and Sons.

Rojer, A. S. and Schwartz, E. L. (1990). Design considerations for a space-variant visual sensor with complex-logarithmic geometry. In *Proceedings. 10th International Conference on Pattern Recognition*, volume 2 of *International Conference on Pattern Recognition*, pages 278–285, Atlantic City, NJ. International Association of Pattern Recognition, IEEE Computer Society Press.

Roth, J. (1955). An application of algebraic topology to numerical analysis: On the existence of a solution to the network problem. *Proceedings of the National Academy of Science of America*, 41:518–521.

Ruderman, D. L. and Bialek, W. (1994). Statistics of natural images: scaling in the woods. *Physical Review Letters*, 73(6):814–817.

Sandini, G., Bosero, F., Bottino, F., and Ceccherini, A. (1989). The use of an anthropomorphic visual sensor for motion estimation and object tracking. In *Image Understanding and Machine Vision 1989. Technical Digest Series, Vol.14. Conference Edition*, volume 14 of *Technical Digest Series*, pages 68–72, North Falmouth, MA. Optical Society of America, AFOSR, Optical Society of America.

Sandini, G., Questa, P., Scheffer, D., and Mannucci, A. (2000). A retina-like CMOS sensor and its applications. In *IEEE Sensor Array and Multichannel Signal Processing Workshop*, Cambridge. IEEE.

Sarkar, S. and Soundararajan, P. (2000). Supervised learning of large perceptual organization: Graph spectral partitioning and learning automata. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(5):504–525.

Schreiber, G. R. and Martin, O. C. (1999). Cut size statistics of graph bisection heuristics. *SIAM Journal on Optimization*, 10(1):231–251.

Schwartz, E. L. (1977). Spatial mapping in the primate sensory projection: Analytic structure and relevance to perception. *Biological Cybernetics*, 25(4):181–194.

Schwartz, E. L. (1994). Computational studies of the spatial architecture of primate visual cortex: Columns, maps, and protomaps. In Peters, A. and Rockland, K., editors, *Primary Visual Cortex in Primates*, volume 10 of *Cerebral Cortex*. Plenum Press.

Shewchuk, J. R. (1996). Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In Lin, M. C. and Manocha, D., editors, *Applied Computational Geometry. Towards Geometric Engineering. FCRC'96 Workshop, WACG'96. Selected Papers*, pages 203–222, Philadelphia, PA. ACM, Springer-Verlag.

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

Simon, H. D. (1991). Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering*, 2:135–148.

Soille, P. (1999). *Morphological Image Analysis: Principles and Applications.* Springer-Verlag, 2nd edition.

Spielman, D. A. and Teng, S.-H. (1996). Spectral partitioning works: Planar graphs and finite element meshes. Technical Report UCB CSD-96-898, University of California, Berkeley.

Stone, J. and Halasz, P. (1989). Topography of the retina in the elephant *loxodonta africana*. *Brain Behavior and Evolution*, 34:84–95.

Strang, G. (1986). *Introduction to Applied Mathematics.* Wellesley-Cambridge Press.

Strogatz, S. H. (2001). Exploring complex networks. *Nature*, 410(6825):268–276.

Taubin, G. (1995). A signal processing approach to fair surface design. In Cook, R., editor, *Computer Graphics Proceedings. Special Interest Group in Computer Graphics (SIGGRAPH) 95*, pages 351–358, Los Angeles, CA. ACM, ACM.

Taubin, G., Zhang, T., and Golub, G. (1996). Optimal surface smoothing as filter design. Technical Report RC-20404, IBM.

Unser, M. (2000). Sampling — 50 years after shannon. *Proceedings of the IEEE*, 68(4):569–587.

Upton, G. J. G. and Fingleton, B. (1985). *Spatial Data Analysis by Example*, volume 1. Point pattern and quantitative data of *Wiley series in probability and statistics.* John Wiley & Sons.

Vlach, J. and Singhal, K. (1994). *Computer Methods for Circuit Analysis and Design.* Van Nostrand Reinhold, 2nd edition.

Wallace, R., Ong, P.-W., and Schwartz, E. (1994). Space variant image processing. *International Journal of Computer Vision*, 13(1):71–90.

Wandell, B., Chial, S., and Backus, B. T. (2000). Visualization and measurement of the cortical surface. *Journal of Cognitive Neuroscience*, 12(5):739–752.

Wang, S. and Siskund, J. M. (2003). Image segmentation with ratio cut. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(6):675–690.

Warner, F. W. (1983). *Foundations of Differentiable Manifolds and Lie Groups.* Spring-Verlag.

Watts, D. and Strogatz, S. (1998). Collective dynamics of ‘small-world’ networks. *Nature*, 393(6684):440–442.

Watts, D. J. (1999). *Small worlds: The dynamics of networks between order and randomness.* Princeton studies in complexity. Princeton University Press, Princeton, N.J.

Weyl, H. (1923). Reparticion de corriente en una red conductora. *Revista Matemática Hispano-Americans*, 5(6):153–164.

Whitteridge, D. (1965). Geometrical relations between the retina and the visual cortex. In *Mathematics and computer science in biology and medicine; proceedings of conference held by Medical Research Council in association with the health departments: Oxford, July 1964*, pages 269–276. Medical Research Council, London, H.M. Stationery Office.

Witkin, A. (1983). Scale-space filtering. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 1019–1022.

Wood, J. and Fisher, P. (1993). Assessing interpolation accuracy in elevation models. *IEEE Computer Graphics and Applications*, 13(2):48–56.

Wright, A. S. and Acton, S. T. (1997). Watershed pyramids for edge detection. In *Proceedings. International Conference on Image Processing*, volume 2, pages 578–581. IEEE Signal Processing Society, IEEE Computer Society.

Wu, Z. and Leahy, R. (1993). An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Pattern Analysis and Machine Intelligence*, 11:1101–1113. 546.

Zahn, C. (1971). Graph theoretical methods for detecting and describing Gestalt clusters. *IEEE Transactions on Computation*, 20:68–86.

Zhang, R., Tsai, P.-S., Cryer, J. E., and Shah, M. (1999). Shape-from-shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706.

Leo J. Grady

Current Affiliation:

Cognitive and Neural Systems Department
Boston University
677 Beacon Street
Boston, Massachusetts 02215

Current Address:

28 Shepard Street, Apt. #2
Brighton, MA 02135
(617) 782-7509 (home)
`lgrady@cns.bu.edu`

Date of Birth : February 10, 1977

Place of Birth : Albany, New York

Marital Status : Single

Formal Education:

University of the Vermont, Burlington, Vermont, BS in Electrical Engineering awarded in May 1999.

Boston University, Boston, Massachusetts, 1999 to present, Ph.D. program in Cognitive and Neural Systems (August 2003 Award Date)

Professional Status and Experience:

September 1999 to Present	Research Fellow Cognitive and Neural Systems Department, Boston University, Boston, Massachusetts
June 1999 to August 1999	Summer Co-op IBM Essex Junction, Vermont
June 1997 to December 1997	Research Assistant South Burlington, Vermont

Fields of Interest:

- Pattern analysis
- Signal processing
- Artificial intelligence
- Biomedical imaging
- Machine vision
- Robotics

Awards & Fellowships:

May 2001 Boston University Teaching Assistant Award for Excellence

September 1999 Presidential University Graduate Fellowship

March 1999 University of Vermont Senior Electrical Engineering Award

October 1998 Induction into Tau Beta Pi (engineering honors fraternity)

Current Research Activities:

My Ph.D. thesis at Boston University addressed the issue of space-variant machine vision with a graph-theoretic framework. New algorithms for segmentation and interpolation were developed that take advantage of this space-variant architecture, but apply to standard, space-invariant, machine vision as a special case.

References:

Eric Schwartz, Ph.D.[1]
Professor
Cognitive and Neural Systems, Electrical and Computer Engineering
Boston University
677 Beacon Street
Boston, Massachusetts 02215
(617) 353-6179
`eric@bu.edu`

Thomas Cook
IBM
1000 River Rd. Essex Junction, Vermont 05452
`tomcook@us.ibm.com`

Larry Copp
Economic and Policy Resources, Inc.
2141 Essex Rd.
Williston, Vermont 05495
(802) 878-0346
`ldc@epreconomics.com`

---

[1] Thesis Advisor