March 2017

Image processing and understanding based on graph similarity testing: algorithm design and software development

Jieqi Kang
University of Massachusetts - Amherst

IMAGE PROCESSING AND UNDERSTANDING BASED ON GRAPH SIMILARITY TESTING: ALGORITHM DESIGN AND SOFTWARE DEVELOPMENT

A Dissertation Presented

by

JIEQI KANG

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2017

Electrical and Computer Engineering

IMAGE PROCESSING AND UNDERSTANDING BASED ON GRAPH SIMILARITY TESTING: ALGORITHM DESIGN AND SOFTWARE DEVELOPMENT

A Dissertation Presented

by

JIEQI KANG

Approved as to style and content by:

Weibo Gong, Chair

Patrick Kelly, Member

Don Towsley, Member

Christopher V. Hollot, Department Chair
Electrical and Computer Engineering

DEDICATION

To my family.

ACKNOWLEDGMENTS

I would like to express my deep gratitude to my advisor, Professor Weibo Gong, for his continuous support, guidance and assistance over the years. I greatly appreciate his patience, flexibility and genuine care for me and my family. My sincere gratitude is extended to the remaining members of my dissertation committee, Professor Patrick Kelly and Professor Don Towsley, for their helpful suggestions and advices to my research. I also want to thank my wife and my colleague Shan Lu, for the love and support she has given me. In addition, I would like to acknowledge my gratefulness to my colleagues and friends, Yan Cai, Sheng xiao, Xiaolin Xu, Shuo Li, Cong Wang, Chang Liu, Fabricio Murai and James Atwood for their friendship and contributive discussions. Finally, I want to thank my parents for their love, support and belief in me.

ABSTRACT

IMAGE PROCESSING AND UNDERSTANDING BASED ON GRAPH SIMILARITY TESTING: ALGORITHM DESIGN AND SOFTWARE DEVELOPMENT

FEBRUARY 2017

JIEQI KANG

B.E., UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Weibo Gong

Image processing and understanding is a key task in the human visual system. Among all related topics, content based image retrieval and classification is the most typical and important problem. Successful image retrieval/classification models require an effective fundamental step of image representation and feature extraction. While traditional methods are not capable of capturing all structural information on the image, using graph to represent the image is not only biologically plausible but also has certain advantages.

Graphs have been widely used in image related applications. Traditional graph-based image analysis models include pixel-based graph-cut techniques for image segmentation, low-level and high-level image feature extraction based on graph statistics and other related approaches which utilize the idea of graph similarity testing. To compare the images through their graph representations, a graph similarity testing

algorithm is essential. Most of the existing graph similarity measurement tools are not designed for generic tasks such as image classification and retrieval, and some other models are either not scalable or not always effective. Graph spectral theory is a powerful analytical tool for capturing and representing structural information of the graph, but to use it on image understanding remains a challenge.

In this dissertation, we focus on developing fast and effective image analysis models based on the spectral graph theory and other graph related mathematical tools. We first propose a fast graph similarity testing method based on the idea of the heat content and the mathematical theory of diffusion over manifolds. We then demonstrate the ability of our similarity testing model by comparing random graphs and power law graphs. Based on our graph analysis model, we develop a graph-based image representation and understanding framework. We propose the image heat content feature at first and then discuss several approaches to further improve the model. The first component in our improved framework is a novel graph generation model. The proposed model greatly reduces the size of the traditional pixel-based image graph representation and is shown to still be effective in representing an image. Meanwhile, we propose and discuss several low-level and high-level image features based on spectral graph information, including oscillatory image heat content, weighted eigenvalues and weighted heat content spectrum. Experiments show that the proposed models are invariant to non-structural changes on images and perform well in standard image classification benchmarks. Furthermore, our image features are robust to small distortions and changes of viewpoint. The model is also capable of capturing important image structural information on the image and performs well alone or in combination with other traditional techniques. We then introduce two real world software development projects using graph-based image processing techniques in this dissertation. Finally, we discuss the pros, cons and the intuition of our proposed model

by demonstrating the properties of the proposed image feature and the correlation between different image features.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS . . . . . **v**

ABSTRACT . . . . . **vi**

LIST OF TABLES . . . . . **xii**

LIST OF FIGURES . . . . . **xiii**

CHAPTER

1. INTRODUCTION **1**

1.1 Motivation 1
1.2 Related work: Traditional CBIR/CBIC and feature extraction 2
1.3 Related work: Why use a graph? 5
1.4 The basic idea: testing similarity between graphs 7
1.5 Outlines of this dissertation 8

2. GRAPH SIMILARITY TESTING **11**

2.1 Related work: graph similarity testing 11
2.2 Graph similarity testing using heat content 16
2.3 Graph similarity testing example: E-R vs B-A 21
2.4 Summary 25

3. THE IMAGE HEAT CONTENT FEATURE **27**

3.1 Introduction 27
3.2 From image to graph 27
3.3 The image heat content feature 32

3.3.1 Heat equation and heat content 32
3.3.2 Heat content estimation based on matrix multiplication 33
3.3.3 Heat content estimation based on random walk 35
3.3.4 Connections to continuous domain heat diffusion 39

3.3.5 Multi-scale image heat content 41

3.4 Image retrieval and classification based on the image heat content 41

3.4.1 Affine-transformed images 43
3.4.2 Facial images with pose variations 43
3.4.3 Natural image retrieval 46
3.4.4 Speech and spectrogram retrieval 47
3.4.5 Hand written digits classification experiment 50

3.5 Summation 52

4. IMAGE UNDERSTANDING BASED ON SPECTRAL GRAPH INFORMATION 54

4.1 How to improve the heat content feature? 54
4.2 Graph generation of large images 55

4.2.1 Graph generation by corner detection 55
4.2.2 Graph generation by nodes merging 59
4.2.3 Graph generation based on edge detection 64

4.3 Improved image representation model based on spectral graph information 64

4.3.1 Oscillatory image heat content 65
4.3.2 Weighted eigenvalues 70
4.3.3 Weighted heat content spectrum 70

4.4 Experimental results 76

4.4.1 Oscillatory heat content 77
4.4.2 Coil-100 color image retrieval 78
4.4.3 Outex-10 texture classification 79
4.4.4 Hand written digits classification 79

4.5 Summary 80

5. GRAPH-BASED IMAGE PROCESSING AND UNDERSTANDING SOFTWARE DEVELOPMENT 81

5.1 introduction 81
5.2 Fast image retrieval/classification desktop application based on image heat content 81

5.2.1 Image retrieval module 82
5.2.2 Image comparison module 83

5.2.3 Test Examples 84

5.3 Real-time graph-matching based image marker detection Android application 86

5.3.1 Background introduction 86
5.3.2 Software design 88
5.3.3 Implementation Details 92
5.3.4 Test Examples 95

6. DISCUSSION 102

6.1 The illustration of eigenvectors in the Laplacian spectrum 102
6.2 Correlation analysis of image features 103
6.3 Limitations of our model 105
6.4 Contribution of this dissertation 106

BIBLIOGRAPHY 108

LIST OF TABLES

Table **Page**

3.1 Maximum distances of different poses for one person 45

3.2 Average distances between person 1 to other people 45

3.3 Classification error rate (k-NN classifier) 52

3.4 Classification error rate (logistic/SVM classifier) 52

4.1 Eigenvalues and eigenvectors of the graph's Laplacian 68

4.2 Classification rates of COIL100 78

4.3 Classification error rate 80

5.1 Detection Range vs Scanning Speed 101

LIST OF FIGURES

Figure **Page**

1.1 The general CBIR/CBIC problem 3

1.2 The standard solution model for CBIR/CBIC problem 4

2.1 Which two are more similar? 12

2.2 The intuition of similarity testing using heat content 21

2.3 Heat content comparison between power law graphs and E-R random graphs (red line: power law graphs; blue line: E-R random graphs) 23

2.4 Time derivative of the Heat Content curves for power law graphs and E-R random graphs (left: red line: Power Law graphs; blue line: E-R random graphs) 23

2.5 The Laplacian spectrum distribution of one power law graph and one random graph with mean degree 20 24

2.6 Eigenvalues and corresponding α values 25

3.1 Pixel-based graph representation 30

3.2 Detailed illustration of pixel-based graph representation. Red lines(weight>50), Yellow lines(weight>20), Green lines(weight>10), Blue lines(weight>2, edges with weight less than 2 are not shown in this figure) 30

3.3 Basic image retrieval algorithm based on heat content feature 39

3.4 Local image heat content estimation 42

3.5 Affine transformed fish image examples and corresponding image heat content curves. In the graph generation process, $f(I_i, I_j) = |I_i - I_j|$, d is the square of distance, $\epsilon_1 = \epsilon_2 = 1$. In the Monte Carlo estimation, $\delta = 0.1$ and the number of initial random walkers per vertex is 1. 44

3.6 Human facial images with pose variations 45

3.7 Image database 46

3.8 The distance matrix of all images 47

3.9 Similar pairs in the database 47

3.10 Spectrogram examples of the normal and slower samples 49

3.11 Ratio of durations between the test samples and reference samples 49

3.12 Similarity matrix of spectrograms 50

3.13 Hand-written digit image examples in MNIST database 51

4.1 Delaunay network generation of the image 56

4.2 Fish Contour Database 57

4.3 Sample fishes and networks after the transform 58

4.4 Heat content curves of 7 types of fish contours on Delauney graphs 58

4.5 Super pixel segmentation of the image 60

4.6 Pre-segmentation of image 62

4.7 Images and their corresponding power law degree distributions 63

4.8 Degree distributions and weight spectrums (First row: original graph. Second row: power law graph. Left: CCDF. Right: weight spectrum.) 63

4.9 Asymmetric graph with edges in E_1 67

4.10 The 14 heat content components of the directed graph 69

4.11 The curves under different du/dv corrections 69

4.12 Eigenvalue distribution descriptor of the image 71

4.13 Degree distribution of graphs with power law and lognormal degree distributions 72

4.14 The heat content spectrums of power law graphs 74

4.15 $\alpha 1$ and the network's mean degree 75

4.16 Sorted adjacency matrices (first row) and weight spectrums (second row) of graphs with different structure. 75

4.17 Heat content spectrums of graphs combined by two models. The color spectrums are for original graphs and the black spectrums are for combined graphs. 76

4.18 Seven categories of similar images from COREL dataset 77

4.19 Heat Content 78

4.20 Oscillatory HC 78

4.21 Weighted heat content spectrum 79

5.1 Images in retrieve dataset 82

5.2 Images retrieve module 83

5.3 Images comparison module 84

5.4 Retrieval example 1 85

5.5 Retrieval example 2 85

5.6 Real Google image search result 86

5.7 Simulated Google image search result 87

5.8 Retrieval based on our algorithm 87

5.9 Color distribution of marker variations in real situations 89

5.10 Sampling process in the detection algorithm 91

5.11 Sub-graph matching algorithm for marker detection 92

5.12 Overall structure of the application 93

5.13 Notification of Sliding the cell phone 94

5.14 Pattern Update User Interface 95

5.15 Maximum Detection Range Example 1 ($\approx$ 20 meters) 96

5.16 Maximum Detection Range Example 2 ($\approx$ 15 meters) 96

5.17 Maximum Detection Angle Example 1 97

5.18 Maximum Detection Angle Example 2 97

5.19 Normal lighting condition Example 98

5.20 Dim lighting condition Example 98

5.21 Tag Detection Under Sophisticated Background 99

5.22 Distraction Background with Different Multi-Color Tag 99

5.23 Input and Detect New Color Pattern 100

5.24 Example for Scanning Time 101

6.1 Image illustrations of the eigenvectors 104

6.2 Correlation between different image features. (PF1: Oscillatory heat content, PF2: Weighted heat content spectrum) 105

CHAPTER 1

INTRODUCTION

1.1 Motivation

One of the greatest masterpiece of evolution is the creation of the human visual system (HVS). Images, the most important source of information input for humans, are processed and analyzed every second when our eyes are open. Millions years of evolution made the HVS extremely fast, effective and light weight. The development of modern science and engineering has already unraveled many of mysteries in such a marvelous complicated system. However, creating a computer vision system to achieve the performance of the HVS still remains a dream. Although researchers have accumulated a great amount of knowledge about the biology behind the HVS, the story behind the intelligence remains a mystery to us.

The HVS is a combination of hardware and software. State-of-the-art technological breakthroughs have tremendously narrowed the gap between machines and our eyes at least on the hardware side. Advanced cameras such as the Canon 5D mark III can produce very high resolution vivid color images and videos. Some cameras can capture high resolution images that are even sharper than the human retina can receive. However, the advancement of hardware does little to help solve the major challenge of understanding the HVS: the software of the visual system. How does the visual cortex work to understand the input image? Although in recent years, deep neural networks such as the convolutional neural network [41,44] and deep belief networks [27] have shown considerable potential in catching up to the performance of the HVS, the human visual system is still far better than any of the best artificial

computer vision systems. We have not even mentioned the embarrassed comparison of power consumption between the extremely lightweight HVS and the big clusters of workstations and GPUs yet. Reaching human-level performance of processing and understanding images remains an active and difficult problem for a long time.

The general computer vision problem can be tackled by dividing it into small sub-problems. One of the most well-defined sub-problem, which also can be seen as a trophy in the general machine learning and image understanding community, is the human-like fast image retrieval and classification. Finding visually similar images and assigning a label to an input image is considered to be one of the most important applications of any vision system. In this dissertation, in spite of the existence of a vast literature, algorithms, models in such field, instead of modifying current models to improve the performance a little bit, we focus on providing some new ideas and methods to this long-existed difficult problem.

1.2 Related work: Traditional CBIR/CBIC and feature extraction

The most typical problem of image retrieval and classification is content based image retrieval/classification, or CBIR/CBIC problem. By "content", we mean that we only look at the image itself. We do not use any other information attached to the image such as tags, locations and exif information to help the analysis. The information that we use will be the pixels on the image. Our visual system can only take the image as input. Even if the attached information proves to be useful, we can always handle it later. Figure 1.1 illustrates the definition and the goal of the general image retrieval/classification problem. The images are from the COREL dataset. [73]

Fast content based image retrieval/classification has been a subject of intensive research for many years [15, 16, 45, 51, 56, 58, 66, 71]. The task of a typical image retrieval problem is to find similar images in the database according to the query

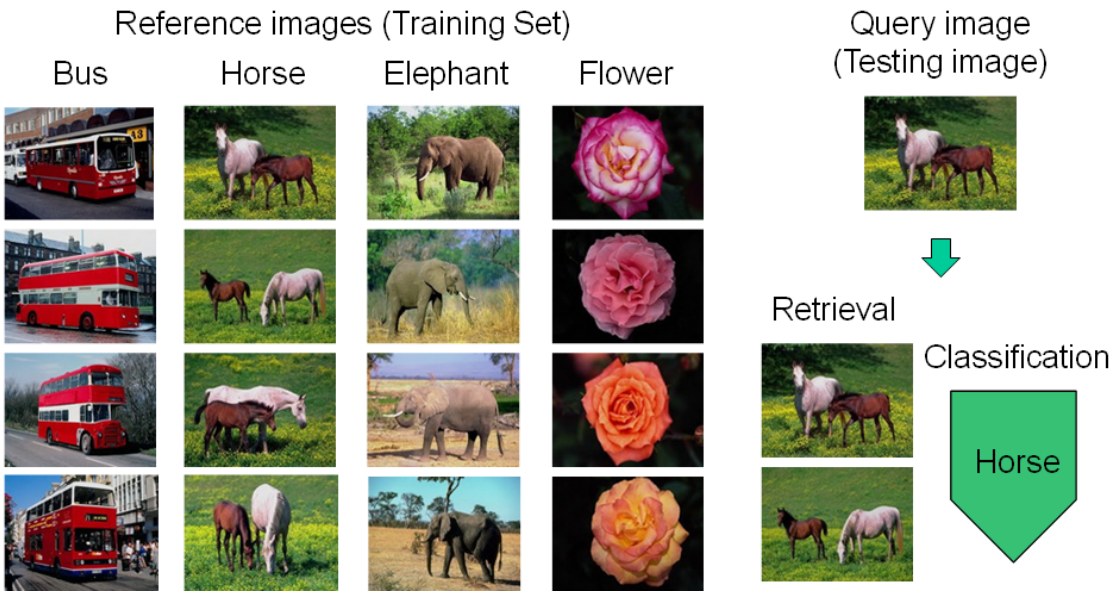


Figure 1.1. The general CBIR/CBIC problem

image. For the image classification problem, the target would be to assign a label in the dataset to the query image. Generally, solving CBIR/CBIC problems includes requiring solving a set of sub-tasks. Among all the traditional models, the most standard solution includes three major steps, which are the image pre-processing, feature selection and design of machine learning algorithms or classifiers. Figure 1.2 shows the general structure of this standard solution to the CBIR/CBIC problem.

In the standard model, the critical initial step in most image retrieval/classification systems is the extraction of appropriate image features. Normally the input of the image is high dimensional and cannot be directly used as an input to the learning systems. A feature extraction procedure for the image can help to capture important information on the image as well as to greatly reduce the dimension of the input data, which generates appropriate feature vector for classifiers at the same time. Researchers have designed a variety of image features based on intensity, color, shape and texture statistics or descriptors. Recent surveys [15, 17, 30, 58] have briefly introduced or evaluated many of the low-level or high-level image features appearing

in the state of the art CBIR/CBIC literature [36, 77]. These features have all been derived from reasoning and/or intuition about useful information for classification. However, no feature can completely capture all of the essential structural information in the image. Among most models, the "bag of words" model, which means to use a combination of features, is still the mainstream of the traditional system.

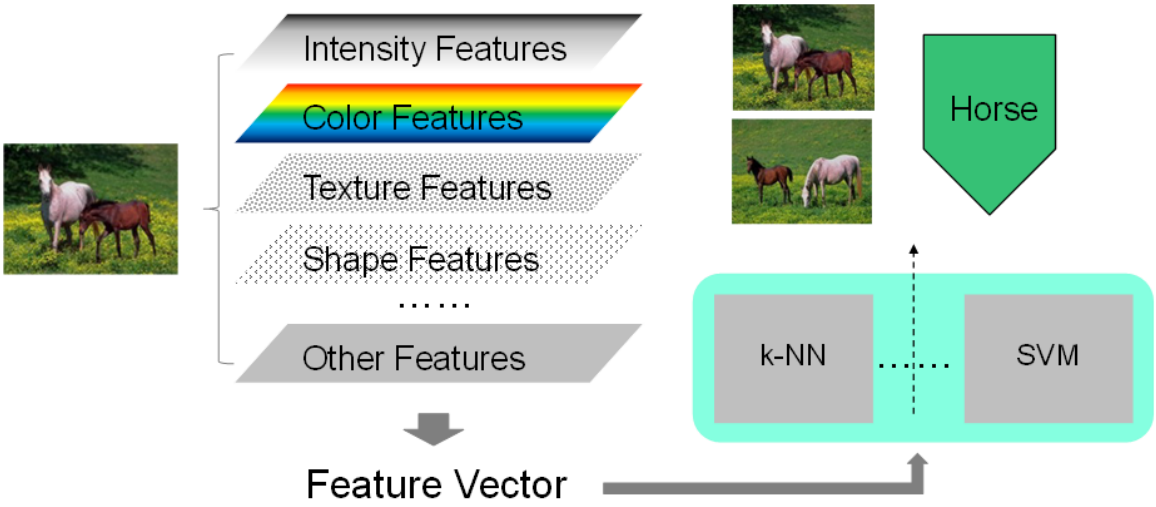


Figure 1.2. The standard solution model for CBIR/CBIC problem

One thing worth mentioning is the new development in the deep learning models. In recent years, the idea of deep learning and deep neural network models has become increasingly popular in machine learning [43, 63]. The philosophy behind this type of approach is completely different compared to the traditional approach. In deep learning models, hand designed feature-extraction is no longer necessary. Instead, the learning process automatically generates feature extraction for the system. There are certain advantages and disadvantages of such a philosophy. We will discuss the pros, cons and the comparison to our model in the last chapter.

The second major step is learning algorithm selection, or the choice of an appropriate classifier for the input features. Widely used methods include logistic regression, support vector machine, k-nearest-neighbor and other machine learning techniques.

A great volume of literature has focused on the selection of learning algorithms or on ways to improve the standard models to be more suitable for specific image problems. In this dissertation, we directly utilize some existing traditional learning models, because they have proven to be very effective as long as the extracted features are good.

In the dissertation, we will generally focus on the first step of the CBIR/CBIC model, which is to effectively generate a low-dimensional representation of and extract features from the input image. This step is more important than any other step in the model. In machine learning, there is a saying:"Garbage in, garbage out". If the image features or the low-dimensional representation vectors are not good enough, even the best learning algorithm cannot do much to improve the retrieval/classification result. As a matter of fact, feature extraction is not only the essential component in human visual system, but also the most important part in standard machine learning models.

The key question in this dissertation is to design an image feature or an image representation model to analyze and encode the image effectively and efficiently. Our model has the following properties:

(1) The image feature should capture all important information on the image, including low-level local image features and high-level structural information;

(2) The result should be effective for the subsequent learning process;

(3) The algorithm should be efficient enough for real-time processing of high-resolution images;

(4) The model should have a connection to the real human visual system.

1.3 Related work: Why use a graph?

Among the vast literatures exploring topics related to image feature extraction, a recent trend is the use of graphs in image representation and understanding tasks.

Some models have already been shown to be effective in a variety of image related applications [3, 18, 24, 25, 54, 64, 72].

Using a graph to help our feature extraction and image understanding is natural. As a matter of fact, all the texts, images, memories, theories and even human faces are stored in the brain represented by the neurons with their connections. Neural science has already proved that the connection structure of neurons is the basic component for long-term memory [7,11]. Some recent literature [8,9,47,61] also suggested graphs as the form that organizes information in the brain. We can simplify neurons with their connections to be a graph contains nodes and edges and we believe that all the information should be captured by the graph structure. One evidence for this graph-representation of information is that some neural science experiments have shown that the same area of brain (a number of neurons) is capable of handling both tasting and language processing tasks [8]. That is to say, human intelligence and the human vision system are built upon the neurons with their connections in the visual cortex, which forms a complex network [9, 61, 68]. This discovery is profound evidence that the structure of the connections of neurons, not the neurons themselves, really matter in the brain when we talk about the intelligence, or at least to some extent. Thus, we believe that a graph-representation for images (and of course, for other information) is not only probable, but also essential.

The idea of graph-based image processing and understanding is not entirely new [54]. In [19] and [24], the authors treat every pixel in the image as a vertex in a graph and connect the neighboring pixels according to the intensity differences. This type of model is widely used in graph based image segmentation literature [18, 24, 64, 72]. In these models, the problem of image segmentation is transformed to a graph-cut problem and ideas such as min-cut or max-flow can be used in the segmentation models. Graph based image segmentation now becomes a rich source of literature and a very successful area of applying graph in image processing and understanding.

Meanwhile, besides the image segmentation problem, the graph representation of image is also widely applied in other image understanding tasks. In [3,25], researchers build graphs based on descriptors such as the edge location and segmented region for generating image signatures. Other applications include image retrieval, clustering and classification based on graph statistics [3,4,14,25,54]. In [22,23] the author points out that Monte-Carlo simulations of diffusion can be helpful in testing the similarity of complex graphs, which would essentially help the analysis of graphs generated by images. Graph based image representation also allows the possibility of applying some well-studied techniques in graph theory to the image related applications. Theories and methods like the graph-cuts theory, similarity testing [40, 53] and the spectral graph theory [13] can then be used as new tools in image related problems.

1.4 The basic idea: testing similarity between graphs

It is generally accepted that images are represented by neuronal networks in the brain but the structure and the signal processing in such neuronal networks are not clear. Our basic motivation of the algorithm development is guided by the following observations in the behavior of the human (and many animal) brains:

(1) Quickness of the recognition with slow neurons;

(2) Most brain neurons have multiple input (dendrite branches) with only one output (axon);

(3) Slight rotation, stretching or siding of an image does not alter recognition;

(4) The brain consumes very little energy performing tasks comparing to the current computer algorithms.

If graphs are mathematical representations of images then it follows that an efficient graph similarity-finding algorithm is at the core of image retrieval and classification. Similarity-finding is the key in intelligence and fast comparison of objects

is a daily task in every corner of the human brain. The idea leads us to propose the following guidelines for the development of fast image representation and feature extraction model.

(1) Images are represented by network connectivity, or graphs;

(2) Image representations should be invariant against the rotation/permutation changes of the input signal representations;

(3) Image representations could check similarity almost instantaneously;

(4) The spatial/temporal signal conversion mechanisms should be plausible for emerging from randomness.

In this work we motivate our image processing and understanding algorithm based on a high level of analogue to the processes in the brain. We abstract the neuronal networks, regarding them as weighted graphs, and we use random walks over the graph to model the effect of neuronal spikes as the initial idea. In doing so we disregard many details and hope to grasp the fundamental mechanisms of fast image retrieval and classification.

1.5 Outlines of this dissertation

This dissertation includes six chapters. Chapter 1 is the introduction of the dissertation. In this chapter, we first discuss the background, the motivation and propose the idea of graph-based image processing and graph similarity testing as a tool of analyzing images. We also briefly introduce the traditional image retrieval/classification models and graph-based image understanding models in modern image analysis.

In Chapter 2, we propose a fast graph similarity testing algorithm based on the asymptotic behavior of the graph heat content. In Section 2.1, we introduce the idea

of graph similarity testing. In Section 2.2 we propose the heat content generation algorithm for complex network. In Section 2.3, we show a preliminary experiment result of graph similarity testing based on the graph heat content. Section 2.4 summarize the proposed heat content methord.

In Chapter 3, we propose the image heat content feature as a feature extraction for image understanding tasks. In Section 3.2, we propose a framework of the graph generation based on the image. We then propose and discuss in detail the fast feature extraction algorithm based on the heat content method proposed in section 3.3. In Section 3.4, five experiments are shown to illustrate the performance of the image heat content feature. Section 3.5 is a brief summation of this chapter.

In Chapter 4, we discuss the possibility of improving the image heat content feature. In Section 4.2, we introduce several approaches to drastically reduce the size of the traditional pixel-based graph representation. In Section 4.3, we propose a novel feature extraction model based on the spectral graph information. Experiments in Section 4.4 show that our improved features perform better than the original heat content feature and is a more effective supplement to further improve the performance of traditional feature-based classification. Section 4.5 is a summation of the improved model proposed in this chapter.

In Chapter 5, we describe two software development projects of fast image processing and understanding. In Section 5.2, we propose the design and the result of a desktop image retrieval/classification application based on the image heat content model. We also use the result of the Google Image Search to illustrate the ability of our algorithm. In Section 5.3, we briefly introduce an Android application of fast real-time image processing and analysis based on a grid-graph representation of the image.

The last chapter is the discussion and conclusion chapter. We discuss the pros and cons of our proposed model and introduce a few potential future directions of research.

CHAPTER 2

GRAPH SIMILARITY TESTING

2.1 Related work: graph similarity testing

The foundation of our work is based on the graph representation of the image. In order to find the similarity between images, naturally one can ask a simple question: how do we compare two graphs? The question is not trivial at all. In fact, graph similarity is not a well-defined problem which has a close-form solution. Consider we have two graphs, and we know they represent two different objects, which can be images and other interesting things. Like everything we have ever met, we care about what is "same" or "different". Are they similar or not? How similar they are? These questions not only are the consequences of logic, but also have application values in many areas if we use graph-base techniques. In fact, many applications require a quantitative measurement of the similarity between two graphs. With this similarity measurement, we can apply classification techniques on the graph-based training and testing samples without huge effort.

Unfortunately, the idea of "graph similarity" is a complex concept and varies from different applications. There is no single definition that satisfying all kinds of problems. For example, consider three graphs as Figure 2.1 shows, which two graphs are more similar? The answer might be not as obvious as we think. In some situations, the number of vertices is very important since it roughly represent the scale of the graph, and that will lead to a higher similarity score for graph (a) and (b). However, intuitively we can easily observe the graph (b) and (c) have two clusters while graph (a) only has one. Can we also say that graph (b) and (c) should be considered more

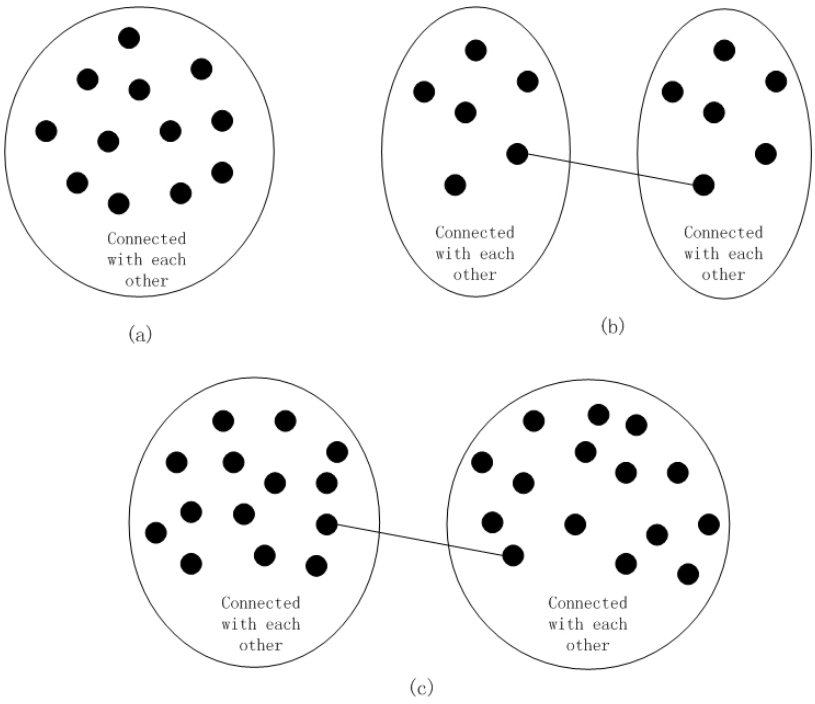


Figure 2.1. Which two are more similar?

similar in this perspective? We believe that the answer is yes if we care about the topological structure of networks. We can immediately notice that if we delete or destroy any node in (a), the rest of the nodes are still connected, but if we delete the hub node in (b) and (c), the network will be disconnected which means the whole system is down in this case. As we can see, the concept of "graph similarity" should be defined by the specific application case by case.

Graph similarity testing is useful in many different areas. The demanding of appropriate approach rises as the graph sizes are increasing extremely fast in diverse areas, such as social networks (like Facebook, Twitter, etc.), Web graphs (like Google), knowledge networks (like Wikipedia), etc. As we can expect, there are many existing traditional methods. In [40], the authors summarized the traditional methods into three categories: graph isomorphism, feature extraction and iterative methods.

One of the most typical and popular definition of graph similarity is derived from graph matching problem. From middle of last century, researchers developed tremen-

dous methods to find exact matching of two graphs. Exact graph matching means that the mapping between the nodes of the two graphs must be edge-preserving. Namely that if two nodes in the first graph are linked by an edge, they are mapped to two nodes in the second graph that are linked by an edge as well. In fact, if two graphs has a perfect match, they can be treated as same graph, and that is the start point of one definition of traditional graph similarity.

Generally, given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the graph exact matching methods called graph isomorphism requires to find a one-to-one mapping $f : V1 \mapsto V2$ such that $(u, v) \in E_1$ if $(f(u), f(v)) \in E2$ with $|V1| = |V2|$. And if a strict correspondence among the nodes or the edges of the two graphs can not be found, namely no isomorphism can be expected between both graphs, the graph matching task can be revised to find the best matching between them, which can be viewed as an inexact graph matching problem.

Inexact graph matching can lead to edit operations on graph, and this idea can be used to define the similarity of graphs based on the effort needed to make the graphs identical. This is the basic idea for the concept of graph edit distance (GED) [62]. Since there are many approaches to edit one graph into another, the cost of the least expensive sequence of edit operations that are needed to transform one graph into another is defined as graph edit distance. The task is transforming the current graph into a target graph and all edit operations are performed on the current graph. A cost function is defined for each operation and the cost for this edit operation sequence is sum of costs for all operations in the sequence. The formal definition of graph edit distance can be described as: Let $g_1 = (V_1, E_1, \mu_1, v_1)$ be the source and $g_2 = (V_2, E_2, \mu_2, v_2)$ the target graph. The graph edit distance between g_1 and g_2 is defined by

$$d(g_1, g_2) = \min_{(e_1,\ldots,e_k)\in\Upsilon(g_1,g_2)} \sum_{i=1}^{k} c(e_i), \tag{2.1}$$

where $\Upsilon(g_1, g_2)$ denotes the set of edit paths transforming g_1 into g_2, and c denotes the cost function measuring the strength $c(e_i)$ of edit operation e_i. The following algorithm shows a typical graph edit distance algorithm [21].

Algorithm 1 Graph edit distance algorithm

INPUT: Non-empty graph $g_1 = (V_1, E_1, \mu_1, v_1)$ and $g_2 = (V_2, E_2, \mu_2, v_2)$.
OUTPUT: A minimum cost edit path from g_1 to g_2
Initialize $OPEN$ to the empty set
For each vertex $\omega \in V_2$, insert the substitution $\{u_1 \to \omega\}$ into $OPEN$
Insert the deletion $\{u_1 \to \epsilon\}$ into $OPEN$
loop
  Remove p_{min}=arg $\min_{p \in OPEN}\{g(p) + h(p)\}$ from $OPEN$
  if p_{min} is a complete edit path **then**
    Return p_{min} as the solution
  else
    Let $p_{min} = \{u_1 \to v_{i_1}, \ldots, u_k \to v_{i_k}\}$
    if $k < |V_1|$ **then**
      For each $\omega \in V_2 \setminus \{v_{i_1}, \ldots, v_{i_k}\}$, insert $p_{min} \cup \{u_{k+1} \to \omega\}$ into $OPEN$
      Insert $p_{min} \cup \{u_{k+1} \to \epsilon\}$ into $OPEN$
    else
      $p_{min} \cup \bigcup_{\omega \in V_2 \setminus \{v_{i_1}, \ldots, v_{i_k}\}} \{\epsilon \to \omega\}$ into $OPEN$
    end if
  end if
end loop

If our target graphs are not with labels, the problem will become more complicated. The algorithm of this graph edit distance is usually carried out by tree search type approaches. The algorithm explores the space of all possible mappings of the nodes and edges of the first graph to the nodes and edges of the second graph. In application a widely used algorithm is based on the A* algorithm [26] which is a best-first search algorithm.

Unfortunately, to find a global optimal of the edit distance problem is NP-complete even for planar graphs [39]. There are a great number of literatures proposing different types of fast algorithms such as [59], gives suboptimal solutions. However, algorithms which are based on search technique, hardly can be applied in large scale graphs. Therefore, researchers proposed another type of algorithms based on feature extrac-

tion. The key idea behind these methods is that similar graphs probably share certain properties, such as degree distribution, diameter, spectra, etc. After extracting these features, some similarity measure is applied in order to compute the similarity between the graphs. Some researchers combine the idea with the previous described edit difference, which provides a synthesis of ideas from spectral graph-theory and structural pattern recognition. Robles-Kelly and Hancock [60] use a graph spectral seriation method based on the leading eigenvector of the adjacency matrix to convert graphs into strings, and match the resulting string representations by finding the path on the edit lattice whose cost is minimum. These methods are powerful and scale well, as they map the graphs to several statistics that are much smaller in size than the graphs. However, depending on the statistics that are chosen, it is possible to get results that are not intuitive. For instance, it is possible to get high similarity between two graphs that have very different node set size, which is not always desirable.

One group of graph similarity measure algorithm is labelled as iterative methods. The motivation of the iterative methods is that two nodes are similar if their neighborhoods are also similar. In each iteration, the nodes exchange similarity scores and this process ends when convergence is achieved. In this category, there are some successful algorithms. First, we have the similarity flooding algorithm by Melnik et al. [50] This algorithm focus on the matching problem, that is, it attempts to find the correspondence between the nodes of two given graphs. What is interesting about the paper is the way that the algorithm is evaluated: humans check whether the matchings are correct, and the accuracy of the algorithms is computed based on the number of adaptations that have to be done in the solutions in order to get the right ones. Another successful algorithm is SimRank [35], which measures the self-similarity of a graph, ie. it assesses the similarities between all pairs of nodes in one graph. The algorithm computes iteratively all pairs similarity scores, by propagating similarity scores in the A^2 matrix, where A is the adjacency matrix of the graph; the process

ends when convergence is achieved. Furthermore, a recursive method related to graph similarity and matching is the algorithm proposed by Zager and Verghese et al. [75]. This method introduces the idea of coupling the similarity scores of nodes and edges in order to compute the similarity between two graphs. In this work, the node correspondence is unknown and the proposed algorithm computes the similarity between all pairs of nodes, as well as all pairs of edges, in order to find the mapping between the nodes in the graph. Bayati et al. [5] proposed two approximate sparse graph matching algorithms using message passing algorithms. Specifically, they formalized the problem of finding the correspondence between the nodes of two given graphs as an integer quadratic problem and solved it using a Belief Propagation (BP) approach.

The major problem of the graph isomorphism and iterative methods is scalability. Feature extraction methods are more like the method that we will propose. But sometimes, the features can be partial and inaccurate, like eigenvalues. For example, in [49], the authors proved that two isospectral nonisometric planar graphs can be distinguished by using the heat content method, even they share the same sets of eigenvalues. In [53], the authors discussed methods for similarity testings in directed web graphs (although we are only interested in assessing similarities of symmetric graphs in our current work), like vertex ranking, sequence similarity, signature similarity, etc. However, like the proposed algorithm in [40], most of the methods discussed need knowing the nodes correspondence. In the following section, we will propose a new graph similarity testing algorithm based on the asymptotic behavior of the heat content.

2.2 Graph similarity testing using heat content

The asymptotic behavior of the heat content has been used as a tool to understand the structure of a manifold [6, 55] or a graph [48, 49], etc.. Heat content is determined by the solution to the heat equation associated to the corresponding Laplacian oper-

ators. It describes the heat diffusion with time on the structure with a given initial heat distribution. In [**?**], the authors showed that the convergence of a random graph Laplacian to manifold Laplacian under certain conditions. The convergence of Laplacian operator shows close connections between heat diffusion on manifold and on graph. One advantage of the heat content method is that, the asymptotic behavior of the heat amount as $t \to 0^+$ makes the heat content curves separate from each other at the very beginning part if the structures are different. This property enables us to generate fast speed graph distinguishing algorithm, combining with our simulation method. In [49], the authors gave an example to use the heat content to distinguish isospectral planar graphs. Our work is the first attempt to apply the heat content methods to distinguish complex networks, which involves work about graph similarity testing and multivariate distribution matching. In [22, 23] the author points out that Monte-Carlo simulations of diffusion can be effective in testing the similarity of complex graphs and that this may have implications on concept abstraction mechanisms in human and animal intelligence. Heat content can be estimated using a random walk on graph. In [46], the authors show that a random walk based similarity algorithm is effective in differentiating large graphs with the same heavy tail degree distribution.

Given a graph $G = (V, E)$ with vertex set V and edge set $E \subseteq V \times V$. The adjacency matrix $A = [a_{vu}]$ is defined as follows:

$$a_{uv} = \begin{cases} w_{uv} & \text{if } (u, v) \in E \\ 0 & \text{otherwise} \end{cases},$$

w_{vu} is the edge weight from u to v. The degree matrix is $D = diag[d_u]$, a diagonal matrix with

$$d_u = \sum_v a_{vu}.$$

Graph Laplacian, a matrix representation of a graph, is defined as

$$L = D - A.$$

For undirected graph with $w_{uv} = w_{vu}$, the adjacency matrix A is symmetric and the degree matrix is diagonal, so the Laplacian matrix L is symmetric and diagonalizable. The normalized Laplacian is defined as

$$\mathcal{L} = D^{-1/2} L D^{-1/2}.$$

$\mathcal{L}$ is also diagonalizable. However, the random walk Laplacian, defined as $L_r = LD^{-1}$, is unsymmetric. The relationship between $\mathcal{L}$ and L_r is as follows:

$$L_r = D^{1/2} \mathcal{L} D^{-1/2}.$$

Now, let $\lambda_1 \leq \lambda_2 \leq \ldots, \leq \lambda_n$ be the eigenvalues of $\mathcal{L}$ and $\phi_i, i = 1, \ldots, n$ be the corresponding eigenvectors. With $\Lambda = diag[\lambda_i]$ and $\Phi = [\phi_1, \ldots, \phi_n]$, we can diagonalize $\mathcal{L}$ to be

$$\mathcal{L} = \Phi \Lambda \Phi^{-1}.$$

Meanwhile, we can get

$$L_r = (D^{1/2}\Phi)\Lambda(\Phi^{-1}D^{-1/2}) = (D^{1/2}\Phi)\Lambda(D^{1/2}\Phi)^{-1}. \tag{2.2}$$

Equation 2.2 proves that, L_r is also diagonalizable. L_r and $\mathcal{L}$ share the same group of eigenvalues, but the corresponding eigenvectors are different. $\mathcal{L}$ is the Laplacian operator used in the heat equation. With the relationship between $\mathcal{L}$ and L_r, we propose a random walk simulation method in the later section.

The Heat equation associated with the normalized Laplacian is

$$\begin{cases} \frac{\partial h_t}{\partial t} = -\mathcal{L}h_t \\ h_t(v,u) = 0 \text{ for } u \in \partial D \end{cases},$$

with initial condition

$$h_0(u,u) = \begin{cases} 1 & \text{if } u \in iD \\ 0 & \text{else} \end{cases},$$

∂D is the boundary set, the collection of all boundary vertices. The other vertices are called interior vertices and their collection is denoted as iD. Assume the total number of vertices is N and the number of interior vertices is m, h_t is an $N \times N$ matrix. $h_t(u,v)$ is the amount of heat flow from vertex v to vertex u at time t. All the heat flowing to the boundary vertices will be absorbed. The normalized Laplacian $\mathcal{L}$ can be partitioned into four parts:

$$\mathcal{L} = \begin{bmatrix} \mathcal{L}_{iD,iD} & \mathcal{L}_{\partial D,iD} \\ \mathcal{L}_{iD,\partial D} & \mathcal{L}_{\partial D,\partial D} \end{bmatrix}$$

We call $\mathcal{L}_{iD,iD}$ the interior Laplacian and the unique solution to the above heat equation is

$$h_t = e^{\mathcal{L}_{iD,iD}t}$$

$\mathcal{L}_{iD,iD}$ is still symmetric and diagonalizable. For convenience, still use Λ and Φ to be the eigenvalue matrix and eigenvector matrix of $\mathcal{L}_{iD,iD}$, we have

$$h_t = \Phi e^{-\Lambda t}\Phi^{-1} = \Phi e^{-\Lambda t}\Phi^T.$$

So, for each entry of h_t,

$$h_t(u,v) = \sum_{i=1}^{m} e^{-\lambda_i t}\phi_i(u)\phi_i(v) \tag{2.3}$$

Heat content $Q(t)$ is defined as the summation of all the entries in h_t, which measures the total heat remaining in the interior domain. We have

$$Q(t) = \sum_{v \in iD} \sum_{u \in iD} h_t(u, v) = \sum_{v \in iD} \sum_{u \in iD} \sum_{i=1}^{m} e^{-\lambda_i t} \phi_i(u) \phi_i(v) \tag{2.4}$$

Let $\alpha_i = \sum_{v \in iD} \sum_{u \in iD} \phi_i(u) \phi_i(v)$, we get

$$Q(t) = \sum_{i=1}^{m} \alpha_i e^{-\lambda_i t} \tag{2.5}$$

From equation 2.5, the heat content can be viewed as the summation of exponential decay functions with different rates and different weights. The rates and weights are determined by the Laplacian eigenvalues and eigenvectors separately.

According to the work in [10] and [13], eigenvalues of the normalized Laplacian for symmetric weighted graphs will be bounded between 0 and 2. Except for a dominant small eigenvalue closing to 0, the eigenvalues of a random power law graph follow the semicircle law and are close to 1. The dominant small eigenvalue has a large corresponding α value, the exponential decay component of which drops much slower and dominates the heat content curve. To emphasize more on the larger eigenvalues, we may use the time derivatives of the heat content to compare, as equations below:

$$\frac{\partial Q(t)}{\partial t} = -\sum_{i=1}^{m} \alpha_i \lambda_i e^{-\lambda_i t}$$

$$\frac{\partial^2 Q(t)}{\partial t^2} = \sum_{i=1}^{m} \alpha_i \lambda_i^2 e^{-\lambda_i t}$$

Larger eigenvalues will be given comparatively larger weights by the derivation process. Meanwhile, when $t \to 0^+$, $Q(t)|_{t \to 0^+} = \sum_i \alpha_i = 1$, while $\frac{\partial Q(t)}{\partial t}|_{t \to 0^+} = -\sum_i \alpha_i \lambda_i$ and $\frac{\partial^2 Q(t)}{\partial t^2}|_{t \to 0^+} = \sum_i \alpha_i \lambda_i^2$. So, the differences of the graphs can be seen immediately using the initial time derivatives of the heat content.

Intuitively, the idea of our graph similarity testing algorithm is shown in Figure 2.2. We can understand the algorithm by apply the idea in heat diffusion of a hot object surrounded by fixed-degree ice. Eventually the temperature of the object would be the same as the ice because of the heat diffusion process. However, we can imagine that if the shape of the object and the conductivity efficient is different, the heat diffusion pattern will be not the same neither. Thus the total heat preserved in the object over time should follows a unique pattern determined by the properties of the object itself, which is exactly the intuition behind the idea of the heat content.

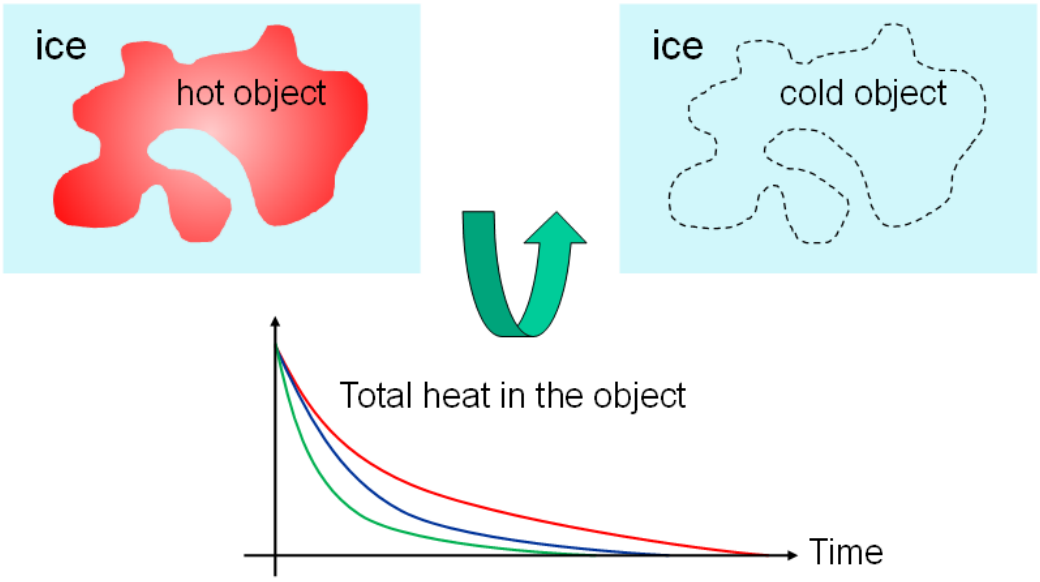


Figure 2.2. The intuition of similarity testing using heat content

2.3 Graph similarity testing example: E-R vs B-A

A Erdos-Renyi (E-R) graph $G(n, p)$ is constructed by connecting nodes randomly. Each edge is included in the graph with probability p. This model is used to generate random graphs. The Barabasi-Albert(B-A) model generate the power-law graphs with m_0 initial nodes. Each new node is connected to m existing nodes with a probability proportional to the number of links that the existing nodes already have. The degree

distribution follows $P(k) \sim k^{-3}$. Barabasi-Albert model is normally used to generate undirected power law graphs. The average degree of the graph is about $2m$.

We use Barabasi-Albert model and generate one power law graph with $m_0 = m = 10$ and the total number of nodes is $n = 5000$. We treat the edges between two vertices the same, and set the weights to be 1. The adjacency matrix of the graph generated by the above two models are symmetric. So, we can apply our method to the generated graph to get the approximated heat content curves to compare. We are interested in comparing the graphs with scale-free degree distributions to the random graphs and finding similarities of the graphs generated by one growing model with different parameters.

Two groups of graphs are generated using the B-A model and E-R model respectively. The total number of nodes are 5000. In each group, there are four graphs with mean degree varies from 20 to 50. Boundaries are selected to be the 4% smallest degree vertices. $\hat{Q}(t)$ for the 8 graphs are plotted in the Fig. 2.3.

As shown in the Figure 2.3, the heat content curves of the two groups of graphs perform differently. At the beginning part, curves for Power Law graphs drop faster, but quickly slow down at the later part. The dropping speed for the curves of random graphs are comparatively more constant during the whole process. The difference between the curves of the two kinds of graphs can be indicated more clearly if we draw the time derivative of the curves instead, as shown in Fig. 2.4. The heat content derivative at time $t \to 0^+$ for power law graphs separated from those for the E-R random graphs dramatically. And if we zoom in the derivatives for the power law graphs, as shown in the subfigure on right, we can see that the time derivatives at the beginning part are in the order of the average degrees. So, using heat content method, not only we can separate graphs with heavy tail distribution from the random graphs, we can tell apart graphs with different mean degrees generated from the same model.

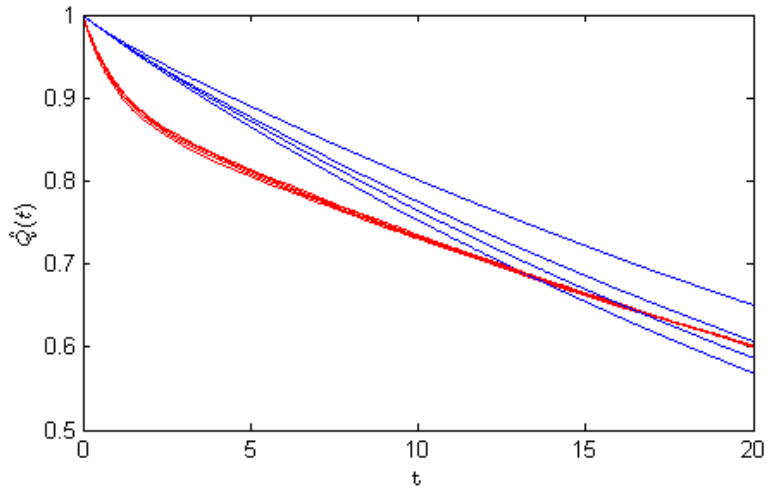


Figure 2.3. Heat content comparison between power law graphs and E-R random graphs (red line: power law graphs; blue line: E-R random graphs)

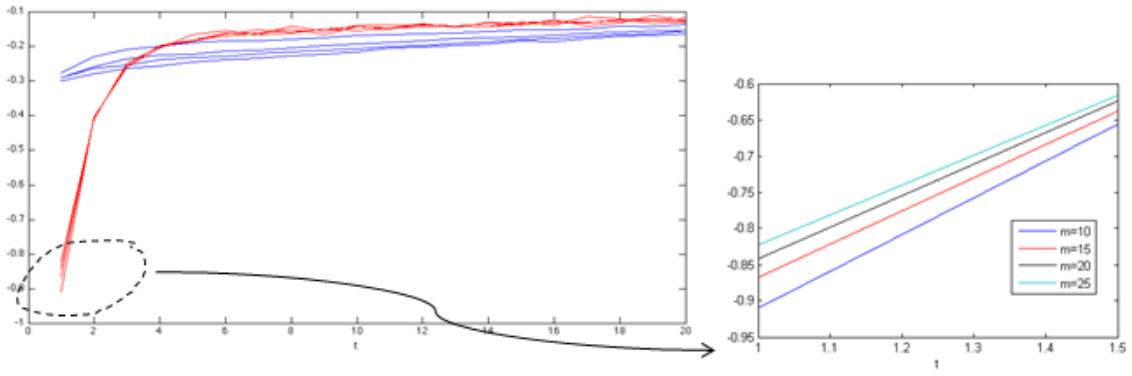


Figure 2.4. Time derivative of the Heat Content curves for power law graphs and E-R random graphs (left: red line: Power Law graphs; blue line: E-R random graphs)

To explain the difference between heat content curves of power law graphs and E-R random graphs, we study the spectrum of the two kinds of graphs first. In [13], Chung *et.al.* proved that eigenvalues of the normalized Laplacian satisfy the semicircle law under the condition that the minimum expected degree is relatively large. Both E-R random graphs and power law graphs satisfy this condition as the paper indicated;

and if two graphs has the same mean degree, the circle radius will be almost the same. As shown in the Fig. 2.5, for the two graphs we generated (a power law graph generated with B-A model and a random graph generated with E-R model with the same mean degree 20), since only a very small proportion of vertices are chosen to be boundary, the main structures are kept and the interior Laplacians still follow semicircle law.

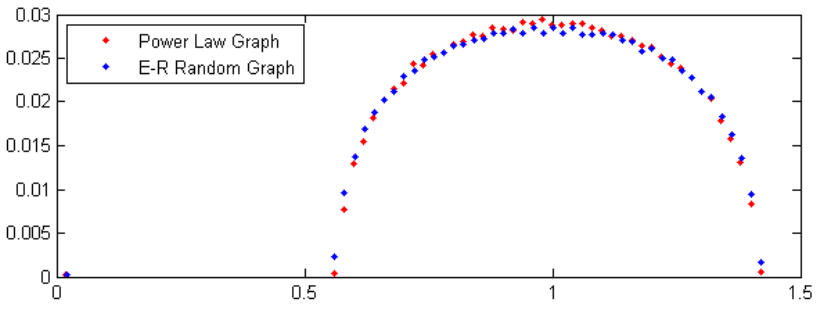
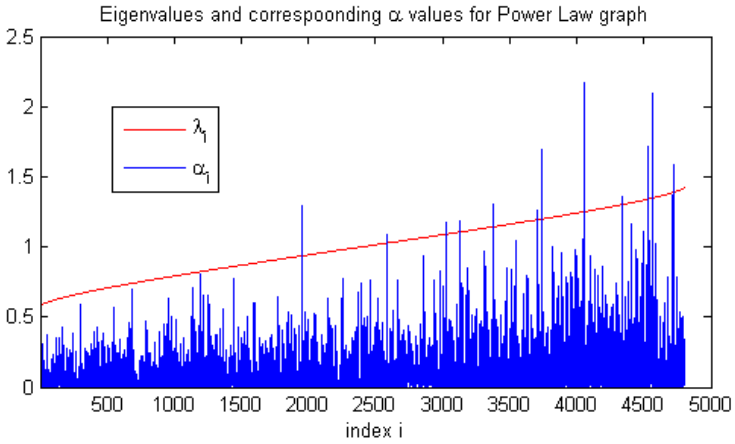


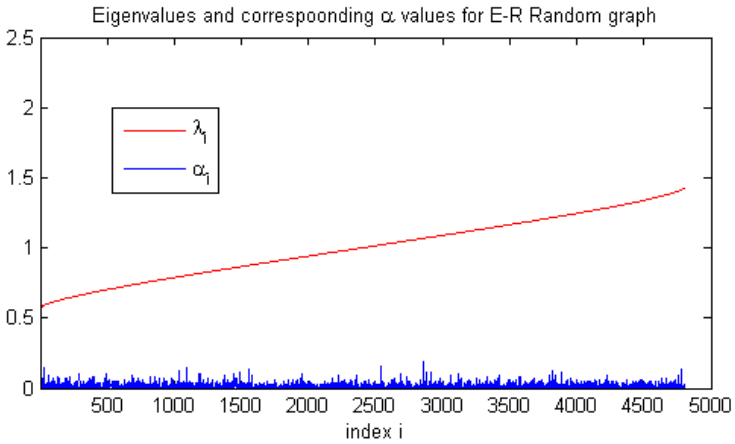
Figure 2.5. The Laplacian spectrum distribution of one power law graph and one random graph with mean degree 20

Using Laplacian spectrum only can hardly distinguish the two kinds of graph; however, according to (2.5), the values of α_i also play important roles in heat content. In Fig. 2.6, the eigenvalues (except for the smallest eigenvalue) and corresponding α values are plotted with the x-axis being the index.

As shown in Fig. 2.6, the eigenvalues of the two graphs, which are plotted in red line, are quite similar. But the weights for the power law graph are much larger than that for the E-R random graph. Exponential decay with decay constant larger than 0.5 drops much faster and will goes to 0 in less than 10 time steps. With the larger weights, the impact of the larger eigenvalues in the heat content curve are highlighted. So, heat content curve for the power law graph drops faster at the beginning part. On the other hand, for the E-R random graph, which is more homogeneous in the graph structure, the α_1 value corresponding to the smallest eigenvalue λ_1 is larger.



(a) Power Law graph



(b) E-R random graph

Figure 2.6. Eigenvalues and corresponding α values

On the other hand, the larger eigenvalues in the Laplacian spectrum have quite small weights, thus do not impact the heat content behavior much.

2.4 Summary

In this section, we proposed a heat content feature for symmetric weighted graph. We can apply the method to compare different types of networks. Graphs with heavy tail degree distribution have different heat content curves comparing to the random graphs generated by the E-R model: the dropping speed for the previous is much faster than that for the later at the very beginning part. This difference is caused by the

different α values (determined by the corresponding eigenvectors) of the Laplacian's eigenvalues (except for the smallest eigenvalue).

Comparing to the previous work, our algorithm has the following advantages in graph similarity testing. The heat content method is a kind of feature extraction method. It maps the graphs to one dimensional data to compare. The difference of two graphs can be presented at the very beginning part of the heat content curves. Meanwhile, our algorithm does not need a given nodes correspondence, which means our method can be applied into the practical problems in wider fields. At last, according to the interlacing theorem in [10], our method is robust to the minor changes in comparatively large graphs. This feature is quite useful when we are dealing with complex random graphs.

CHAPTER 3

THE IMAGE HEAT CONTENT FEATURE

3.1 Introduction

As we have discussed in the first chapter, graph structure plays an important role in representing images, texts and concepts in human intelligence. In this chapter, we formally propose the idea of using graph to help the process of image representation. We will also discuss the possibility of applying the graph similarity testing model that we have proposed in Chapter 2 in the following image understanding process and propose a new general image feature extraction model: the image heat content feature. Further more, we will show the ability of the new feature by several experiments focus on different aspects of image variations.

3.2 From image to graph

The first step in our model is to generate a graph based on an image. One of the the most apparent approach is to treat each pixel in the image as a node in the graph. In this approach, every pixel on the input image corresponds to a vertex of the graph we want to generate. Basically the input image is represented by a $M \times N$ matrix in which each entry corresponds to one pixel on the image. The value of the entry is the intensity of the related pixel which is an integral number from 0 to 255 in typical situation for gray-scale images. In the following algorithm, every pixel p_i is represented by one vertex $v_i \in V$ in our graph $G = (V, E)$.

One problem of this idea is that the nodes in the graph do not capture the intensity information of the pixels in the image. In order to code the intensity information

in the graph, the only approach that we can choose is to use edges to represent the difference of images. Consider one situation that all the pixels in an image increase the same amount of intensity at the same time, the image is essentially not changed much especially when regarding the understanding of the image only. In most of the normal situations, the relative relationship between the pixels is more important comparing to the absolute intensity values of the pixels, which is basically the motivation of our graph generative model.

We denote the image matrix by I with the intensity value of the pixel (i, j) denoted as I_{ij}. Our graph is denoted as $G(V, E, W)$ where V is the set of the graph vertex with $|V| = n = MN$, E is the set of the edges and W is the set of the weights on the edges. First, every pixel p_i is represented by one vertex $v_i \in V$ in our graph $G = (V, E)$. Then the weight of each edge $e = (v_i, v_j) \in E$ is defined by the following equation

$$A(i,j) = \begin{cases} \frac{\epsilon_1 + f(I_i, I_j)}{d(p_i, p_j)} & (I_i \neq I_j) \\ \\ \frac{\epsilon_2}{d(p_i, p_j)} & (I_i = I_j) \end{cases} \tag{3.1}$$

in which I_i, I_j are intensities of pixels p_i and p_j. $d(p_i, p_j)$ is a distance measure of pixel p_i to p_j. A monotonically increasing function $f(I_i, I_j)$ calculates the difference between the intensity of pixel p_i and p_j. We could either connect every pixels pair or only connect all the neighboring pixels for less computational complexity. The intuition behind this graph generation approach is that human vision tends to focus on high-contrast places. Our approach emphasizes the close high-difference pixel pairs by giving the corresponding graph edges larger weights.

The representation vertexes of the pixels on the frame or the natural boundaries of the image are defined as the boundary vertexes of the graph, with the collection denoted as the set V_B. We now consider the graph $G = (V, E, W)$ with a nonempty set of boundary vertexes. The adjacency matrix of the graph G is denoted as $A = [w_{v,u}]$ with $w_{v,u}$ being the weight of the edge between the vertexes u and v. The degree

matrix is $D = diag\,[d_u]$ with $d_u = \sum_v w_{vu}$. Algorithm 2 shows the pseudo code of our proposed model.

Algorithm 2 Generate a graph from an image

```
The input is grey level image Image
The output is an adjacency matrix A, a boundary set B
nc=0
for i = 1 to Image.height do
  for j = 1 to Image.width do
    Assign every pixel an index
    nc=nc+1
    I(nc)=Image(i, j)
    I(nc).h=i
    I(nc).w=j
    if Image(i, j) is a boundary node then
      nc → B
    end if
  end for
end for
for m = 1 to nc do
  for n = 1 to nc do
    Connect node m and n
    d=((I(m).h − I(n).h)^2 + (I(m).w − I(n).w)^2)
    if I_m > I_n then
      A(i, j)=(α + f(I(m), I(n)))/d
    else
      A(i, j)=γ/d
    end if
  end for
end for
return A, B
```

The intuition of this graph generation approach is stated as follows. The first reason is still from biology, which is, human vision tends to be focusing on high-contrast parts, namely "edges" of the image. Our approach emphasize the high-difference pixel pairs by giving the corresponding edges in graph higher weights. The connectivity has to be normalized by the geometric distance between the nodes. Here we just arbitrarily choose the square of the Euclidean distance. In the real application,

we eliminate links with very small weights in order to make the random walk based algorithm faster.

The following example illustrate the graph generation model which is shown in Figure 3.1. The graph representation of this fish image is in the right side, and we can take the natural boundary of the object as the boundary nodes in the graph. However, it's hard to illustrate the edge weights in this small figure. A more precise graph of a compressed is shown in Figure 3.2, in which different level of weights corresponds to colors and widths of lines.

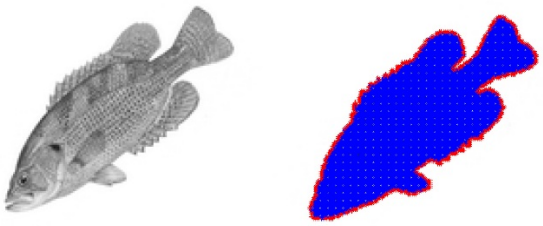


Figure 3.1. Pixel-based graph representation

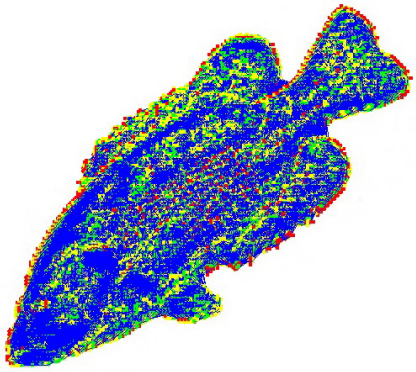


Figure 3.2. Detailed illustration of pixel-based graph representation. Red lines(weight>50), Yellow lines(weight>20), Green lines(weight>10), Blue lines(weight>2, edges with weight less than 2 are not shown in this figure)

A simplified model would possibly speed up the graph generation process, which is actually a "lazy" version of the previous model. The major difference is that instead connecting all the nodes pairs first, only the four-neighboring nodes pairs are

connected. Consequently, the weight of each directed edge $e = (v_i, v_j) \in E$ (from p_i to p_j) can be defined by the following equation

$$w(i,j) = \begin{cases} \epsilon + f(I_i, I_j) & (I_i \geq I_j) \\ \epsilon & (I_i < I_j) \end{cases} \tag{3.2}$$

in which I_i, I_j are intensity of four-neighboring pixel p_i and p_j, similarly, a monotonically increasing function $f(I_i, I_j)$ calculate the difference between the intensity value of pixel p_i and p_j.

Although this graph, which is an approximation of the original graph, is less-connected, all the information is still stored in the connection weights combinations. However, this approach may lead to negative effect on the global behavior of the random walk on the graph, we still consider it as a reasonable compromise between running speed and potential precision improvement. Furthermore, by introducing localized behavior of the random walk and lower-resolution compressed graph, the disadvantages of this approach could be reduced to certain level. Another side effect of this approach is that this grid like graph happens to be planar graph, which is exactly the type of graph discussed in [48].

For the graph generation process, the time complexity for an image with resolution $M \times N$ is $O(M^2N^2)$ in the worst case because we need to compute every link weight between two arbitrary chosen nodes for MN as the total number of nodes in the representation graph of the given image. And meanwhile, the stored weight matrix will consume $O(M^2N^2)$ memory space. However, for a grid graph, the time and space complexity tremendously reduce to be both $O(MN)$ since the maximum number of links for each nodes is fixed to be four. Another property for grid graph is that we can simply prove that in this grid graph, if any one pixel's intensity is given or the average intensity is known, the rest pixel intensities can be automatically calculated

one by one from the neighbor nodes consequently. This property means that nearly all the information of the original image is preserved in this grid graph.

3.3 The image heat content feature

After we generate a graph for the image, what's next? As we have discussed in the first chapter, One obvious idea should naturally come into our mind, that is, the similarity measurement of graphs could be the only way that two images (or ideas, etc) are compared. Because similarity measure is the fundamental element of intelligence, to effectively measure the similarity of graphs (with very fast speed) turns to be the major challenge if we accepted the graph based representation of images, which is essentially the very task in image retrieval – to find visually similar images in very fast speed. As we have discussed in Chapter 2, traditional graph similarity measurement is NP-hard and is impossible to solve when the target graphs has more than a couple dozens of nodes. The approach we want, on the other hand, should be very fast on large scale graphs, robust to little changes of the graph, and comparable to the visual similarity of the images.

3.3.1 Heat equation and heat content

The proposed image feature is based on a fast feature extraction algorithm for complex networks. As we proposed in last chapter, we utilize the heat content concept to design the image feature based on the graph representation. We briefly introduce the heat content concept first in this subsection. For a symmetric image graph representation G, let $D = diag[d_u]$ be the diagonal degree matrix. The normalized graph Laplacian [13] of the graph G is defined as $\mathcal{L} = D^{-1/2}LD^{-1/2}$, where $L = D - A$. Suppose that vertex set V is divided into $V = iD \cup \partial D$, where iD is the interior nodes and ∂D is the boundary. Then the following heat equation describes the heat-flow dynamics on the graph:

$$\begin{cases} \frac{\partial h_t}{\partial t} = -\mathcal{L}h_t \\ h_t(u,v) = 0 \text{ for } u \in \partial D, \end{cases} \tag{3.3}$$

with the initial condition $h_0(u,u) = 1$ if $u \in iD$. Suppose $\Lambda = diag[\lambda_i]$ is the diagonal eigenvalue matrix and Φ is the eigenvector matrix of the interior part of $\mathcal{L}$. The solution to the heat equation is $H_t = e^{-\mathcal{L}t} = \Phi e^{-\Lambda t}\Phi^T$ and for each entry of H_t, we have $H_t(u,v) = \sum_{i=1}^{|iD|} e^{-\lambda_i t}\phi_i(u)\phi_i(v)$, where ϕ_i is the ith column vector in Φ. The heat content $Q(t)$ is defined as

$$Q(t) = \sum H_t(u,v) = \sum_{uv}\sum_{i=1}^{|iD|} e^{-\lambda_i t}\phi_i(u)\phi_i(v). \tag{3.4}$$

A discrete-time heat content vector with a given length could be seen as a type of feature of the corresponding image. Research has already proved that small changes in the graph will not affect the Laplacian spectrum too much [10]. Thus the heat content feature should be invariant to perturbations and small distortions of the original image.

Directly finding the eigenvalues and the eigenvectors of the graph is very time consuming. We have to estimate the heat content in a more affordable way, especially when image-generated graphs typically have tens of thousand of nodes. One approach to estimate the heat content is to use matrix multiplication estimation algorithm. We could also approximate the continuous heat content by the summation of a discrete time random walk on graph.

3.3.2 Heat content estimation based on matrix multiplication

Assume that the transition probability is proportional to the weights of the edges, then random walkers on graph G move from vertex u to neighbor vertex v with

probability w_{vu}/d_u. Define the transition matrix $M = AD^{-1}$ and the lazy random walk transition matrix is defined as follows:

$$M_L = (1-\delta)I + \delta M,$$

which means the random walkers move to one of the neighbor vertex with probability δ, or stay at the current state with probability $1-\delta$. Now, for any time $t = k\delta$, we have

$$P_t = M_L^k P_0 = [1 - \frac{t}{k}L_r]^k P_0 \xrightarrow{k\to\infty,\ \delta\to 0} e^{-L_r t}P_0 \tag{3.5}$$

P_0 is the initial amount of random walker distributions. So, we have $M_L^k \xrightarrow{k\to\infty,\ \delta\to 0} e^{-L_r t}$. With equation 2.2, we can prove each entry in matrix M_L^k converging to the following:

$$M_L^k(u,v) \xrightarrow{k\to\infty,\ \delta\to 0} \sum_1^m e^{-\lambda_i t}\phi_i(u)\phi_i(v)\sqrt{\frac{d_u}{d_v}},$$

comparing to equation 2.3, multiply $\sqrt{\frac{d_v}{d_u}}$ on both sides, we have

$$M_L^k(u,v)\sqrt{\frac{d_v}{d_u}} \xrightarrow{k\to\infty,\ \delta\to 0} h_t(u,v)$$

$M_L^k(u,v)$ measures the proportion of random walkers initiated at vertex v and end up at vertex u in k steps of lazy random walk. The term $\sqrt{\frac{d_v}{d_u}}$ is a correction term makes the value converges to the value of $h_t(u,v)$. Now, with equation 3.4, we get the approximation for $Q(t)$:

$$\hat{Q}(t) = \sum_{v\in iD}\sum_{u\in iD} M_L^k(u,v)\sqrt{\frac{d_v}{d_u}} \tag{3.6}$$

Computing the matrix multiplication, we can get the approximation of $M_L^k(u,v)$, for each pair of nodes $u, v \in iD$. Then, multiply with the correction term and add up the terms, we get the estimated heat content value at time $t = \delta k$. Thus, the

approximated heat content curve can be achieved consequently. The major advantage of our algorithm is, eigenvalue and eigenvector computing can be avoided. So, no matter how large the graph is, it is possible to get an approximated heat content.

3.3.3 Heat content estimation based on random walk

We can also use a Monte Carlo method to estimate the heat content. Suppose we have already generated a graph $G = (V, E)$ from an image I with the boundary B defined as a set of all the nodes corresponding to pixels on the natural boundary of the image. The major steps of our Monte Carlo heat content estimation algorithm are:

(1) According to the weight matrix A, compute the degree of each node D and the corresponding random walk matrix P.

(2) At the initial time, give every vertex the same number k of random walkers to simulate a uniform initial condition. The initial size of random walkers on the graph is $s = nk$.

(3) In each step, every random walker stays at its current vertex v_i with probability δ $(0 \leq \delta \leq 1)$ and with probability $1-\delta$ goes out. The probability of going to neighbor v_j should equal to p_{ji} in the stochastic random walk matrix P. If $v_j \in B$, for which B represent all the boundary vertices, then the random walker is deleted.

(4) After every step, a heat content value is calculated by $\hat{Q}(t) = \sum_{i=1}^{s} 1 \times (\frac{d_u}{d_v})_i^{1/2}$ in which d_u and d_v are the degree of origin node u and current node v of the ith random walker.

(5) Running the algorithm for T/δ steps to get the heat content $\hat{Q}(t)$ from 0 to T.

The above basic algorithm can have the following generalized formation which is could improve the classification performance in experiments and also point to some interesting conjectures in neuroscience.

Suppose we would like to emphasize more the traffic from a heavily connected vertex to a less connected vertex. Here the connectivity of a vertex v is described by the total edge weight dv. It is reasonable to insert a multiplicative factor $(d_u/d_v)^\eta, \eta > 0$ for the traffic from u to v. We call this factor the "ridge factor" since its purpose is to emphasize the connections from heavily connected regions to the less connected ones. In the matrix notations this insertion is amount to change the i step graph response expression from

$$h_i = rM^i p_0$$

to

$$h_i = rD^{-\eta} M^i D^\eta p_0.$$

Since the degree matrix D is closely related to the transition matrix $M = AD^{-1}$, at a first glance it seems that by inserting such factors we are changing the eigenvalues that govern the motion modes of the random walkers. But this is not true. Note that in multiple step transitions $D^{-\eta}$ and D^η cancels each other so our algorithm simply collect the traffic as before and then insert the factor $(dv/du)^\eta$ only once:

$$\begin{aligned} & g'_{L,i+1} \\ & \quad = \textstyle\sum_{k=1}^{n} \lambda_k^{i+1} \sum_{u \in V, v \in V} \phi_{L,k}(u) \psi_{L,k}^T(v) (dv/du)^\eta \\ & \quad = \textstyle\sum_{k=1}^{n} \alpha'_{L,k} \lambda_{L,k}^{i+1}. \end{aligned} \tag{3.7}$$

Our algorithm have the following properties:

(1) We view the random walk over the representation graph as a dynamic system and we use the initial response to distinguish graphs and thus the images they represent;

(2) In our algorithm the earlier the random walk data come in the more important they are, enabling an early decision that achieves the quickness of feature extraction;

(3) Our algorithm is intrinsically rotation invariant. It also handles stretching and posing very well without any preprocessing;

(4) Our algorithm is "embarrassingly parallel" in that it executes the same simple operations on every edge of the graph and the results collected from all the edges are simply added together to give the final output.

Algorithm 3 shows the detail of the proposed algorithm. The time complexity of this algorithm is $O(tn^2)$ if we use a full adjacency matrix to represent P, or $O(tn)$ if we use list like (since normally the number of links from one vertex to another is less than a certain threshold which can be seen as a constant if we ignore edges with weights less than a threshold), n is the size of vertices and t is the number of random walk steps. In our classification applications, the number of random walk steps is usually set to be dozens to hundreds which depend on the size of graph, how ever, this number will be more than enough if it equals to n. Thus in the worst case, we can anticipate it to be a $O(n^2)$ time algorithm. Further more, the algorithm can be parallelized using multiple machines, the random walkers are distributed to several computers and once a random walker is moved, the information is sent to a server, then the server can compute the heat content without any further effort but adding them up with a normalization. The time complexity for such a parallelized mechanism is $O(t)$.

We can directly use the image heat content in the image retrieval/classification tasks. The heat content is treated as an input feature extracted from the image and become a representation of the original image. We then use a distance measure to calculate the similarity between two heat content vectors, which represents the similarity between two images. The distance measure is set to be the Euclidean distance. Figure 3.3 shows this basic image retrieval algorithm.

Algorithm 3 Calculate image heat content from time 0 to t

The input is the graph adjacency matrix A, boundary set B, the random walkers list RW and time t
The output is a series of heat content values grf from 0 to t
$grf(0) \leftarrow$ initial condition
for $x = 1$ to t **do**
  for $i = 1$ to the size of random walkers $RW.length$ **do**
    $rand1$ = a random number in $[0, 1]$
    if $rand1 >=$ lazy probability ϵ **then**
      prepare to walk out
      $rand2$ = a random number in $[0, 1]$
      $sum = 0$
      for $j = 1$ to size of the nodes **do**
        a=$RW(i).now$
        a is the index of the node where the random walker is right now
        $P(a, j)$=$A(a, j)/degree(a)$
        sum=sum+$P(a, j)$
        if $rand2 <= sum$ **then**
          BREAK
        end if
      end for
      j is the destination node
      if $j \in B$ **then**
        set the label of the random walker to be "deleted"
        $RW(i).label$=$deleted$
      else
        $coff$=$(Degree(RW(i).initial)/Degree(RW(i).now))^{ridge\ factor}$
        $h(x) = h(x-1) + 1 \times coff$
        update the random walker's property
        $RW(i).now$=j
      end if
    end if
  end for
end for
return $hc = hc/hc(0)$

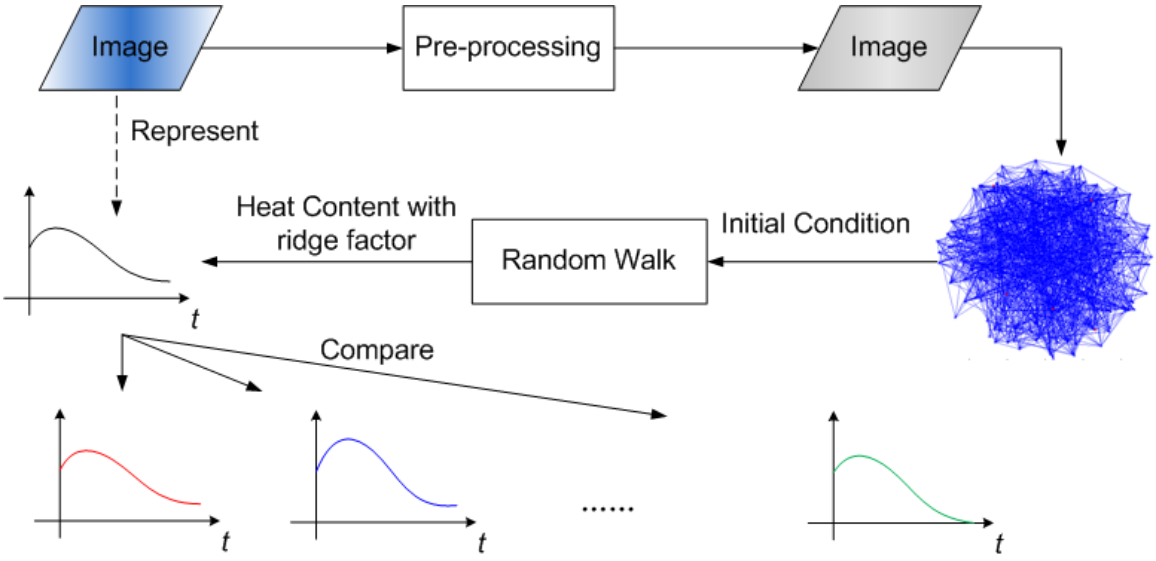


Figure 3.3. Basic image retrieval algorithm based on heat content feature

3.3.4 Connections to continuous domain heat diffusion

Let $P(t)$ be a density matrix of the random walkers at time t and $P_{ij}(t)$ describes the density of the random walkers initialized in node j and currently in node i at time t. For lazy random walk we have

$$P(t+\delta) = [(1-\delta)I + \delta M]\, P(t), \tag{3.8}$$

which is equivalent to

$$P(t+\delta) - P(t) = -\delta(I - M)P(t). \tag{3.9}$$

Multiplying $\sqrt{d_i/d_j}$ to equation 2,

$$P_{ij}(t+\delta)\sqrt{\frac{d_i}{d_j}} - P_{ij}(t)\sqrt{\frac{d_i}{d_j}} = -\delta \sum_{x=1}^{n} (I_{ix} - M_{ix})P_{xj}(t)\sqrt{\frac{d_i}{d_j}}. \tag{3.10}$$

Letting $H_{ij}(t) = P_{ij}(t)\sqrt{\frac{d_i}{d_j}}$,

$$
\begin{aligned}
H_{ij}(t+\delta) - H_{ij}(t) &= -\delta \sum_{x=1}^{n} (I_{ix} - M_{ix}) P_{xj}(t) \sqrt{\frac{d_i}{d_j}} \\
&= -\delta \sum_{x=1}^{n} (I_{ix} - M_{ix}) \sqrt{\frac{d_i}{d_x}} P_{xj}(t) \sqrt{\frac{d_x}{d_j}} \\
&= -\delta \sum_{x=1}^{n} (I_{ix} - M_{ix}) \sqrt{\frac{d_i}{d_x}} H_{xj}(t)
\end{aligned}
$$

.

Thus we have

$$
\begin{aligned}
H(t+\delta) - H(t) &= -\delta (I - D^{1/2} M D^{-1/2}) H(t) \\
&= -\delta (I - D^{-1/2} M D^{-1/2}) H(t) \\
&= -\delta \mathcal{L} H(t).
\end{aligned}
$$

The 2D matrix H can not be mapped to the original $h(u, v)$ on the plane directly. However, $h(u, v)$ can be computed by H. Suppose the (u, v) position block corresponds to node k in the grid network, $h(u, v)$ is in fact the summation of all the components in the kth row of H. We can also transform the 2D $h(u, v)$ into an 1D vector $h' = [h'_1, \cdots, h'_n]^T$ according to the node mapping relations. We have $h_{uv} = h'_k = \sum_{x=1}^{n} H_{kx}(t)$. Meanwhile,

$$
\begin{aligned}
H_{ij}(t+\delta) - H_{ij}(t) &= -\delta \sum_{x=1}^{n} \mathcal{L}_{ix} H_{xj}(t), \\
\sum_{j=1}^{n} H_{ij}(t+\delta) - \sum_{j=1}^{n} H_{ij}(t) &= -\delta \sum_{j=1}^{n} \sum_{x=1}^{n} \mathcal{L}_{ix} H_{xj}(t), \\
h'_i(t+\delta) - h'_i(t) &= -\delta \sum_{x=1}^{n} \mathcal{L}_{ix} h'_x(t), \\
\text{so we have} \qquad \frac{\partial h'}{\partial t} &= -\mathcal{L} h'.
\end{aligned}
$$

3.3.5 Multi-scale image heat content

For a more precise matching, we believe that the heat contents of local patches of the image could be compared for local similarity as well. Thus we developed the following approaches to extract more information in the same random walk process described in previous subsection. Without adding complexities to the running time, the algorithm could provide much more information than the global heat content alone.

For example, suppose an image is divided into 4 by 4 in total 16 blocks, in the random walk process, the random walkers still follow the same instruction and the total image heat content is the same as before, but the only difference is that we further more compute the local image heat content for each block, namely we only consider random walkers inside a certain block each time when we calculate the heat content for that block.

Based on this idea, we use an example to illustrate the idea of multi-scale image heat content. In the experiment, 2 images are partitioned into 16 blocks. Figure 3.4 shows an example of each block's image heat content which are all normalized to start at 1. This partition trick can not only create more combinations of local image heat content to make the feature extraction more precise but also allow to compare local image patches of images potentially. Further more, we can design arbitrary-sized overlapping image blocks at any position to accumulate the information at any level everywhere in only one random walk simulation process. We call this technique the multi-scale image heat content.

3.4 Image retrieval and classification based on the image heat content

In this section, five experiments were performed to examine the performance of our heat content feature. The first experiment is an example to show the performance

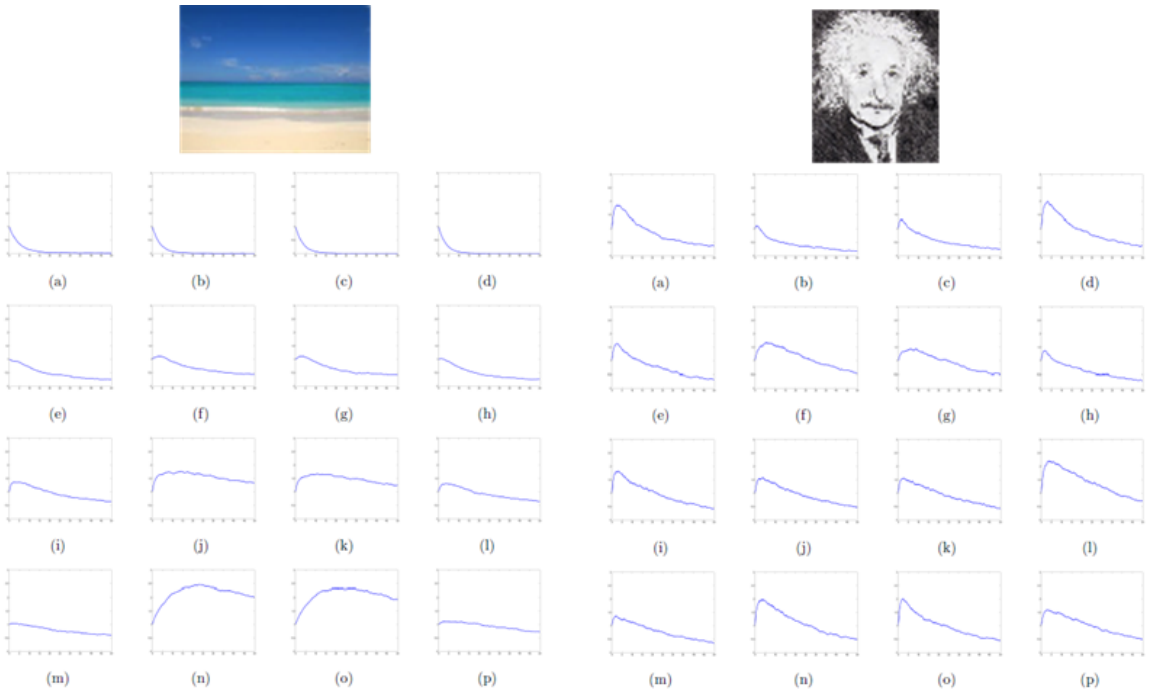


Figure 3.4. Local image heat content estimation

of the image heat content under image variations such as rotation, stretching and affine transforms. The second experiment if a facial recognition demo to show our algorithm's ability to handle pose-variations of human faces. The third experiment is an image retrieval experiment for natural images. In this experiment, we test our multi-scale image heat content by showing relative differences of all the image pairs in a dataset with natural images. The Fourth experiment is another retrieval experiment for spectrograms. The purpose of this experiment is to check the robustness of the heat content feature for similar-structure but largely distorted spectrogram images due to different reading speed and background environment. The experiment also shows the potential of applying our method on voice and speech recognition projects. In the last experiment, we apply the heat content feature in a typical classification task for hand written digits dataset MNIST and compare it with some widely used image features through standard machine learning models.

3.4.1 Affine-transformed images

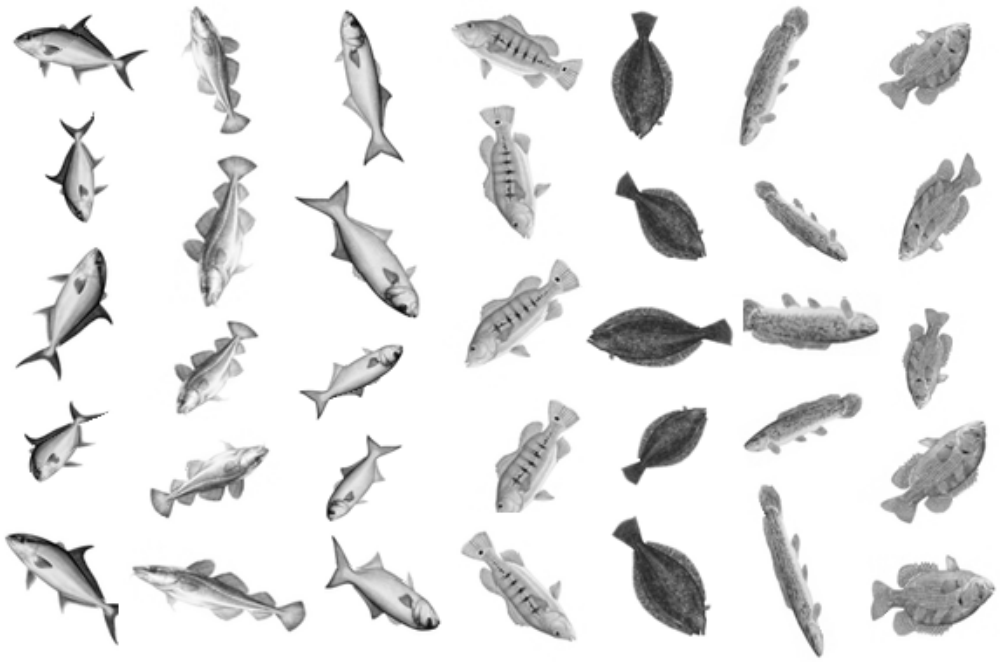
In the first experiment, we set up a dataset contains seven types of fish images. The original fish image is from [2]. To each type of fish image, five affine transforms under random but bounded (the variations are less than 30%) coefficients were applied to generate five similar fish images for each type. Figure 3.5(a) shows the result of fish images after affine transform.

The simulated heat contents of all of the fish images are plotted in Figure 3.5(b). The heat content curves of the same color corresponds to the same type of the fish images. We can see that even from the beginning part ($t < 10$), the heat content curves can successfully cluster the same types of fishes into one group. The result illustrates that the heat content feature is robust to rotation, stretching and affine transform of images, which is a useful property for image feature extraction.

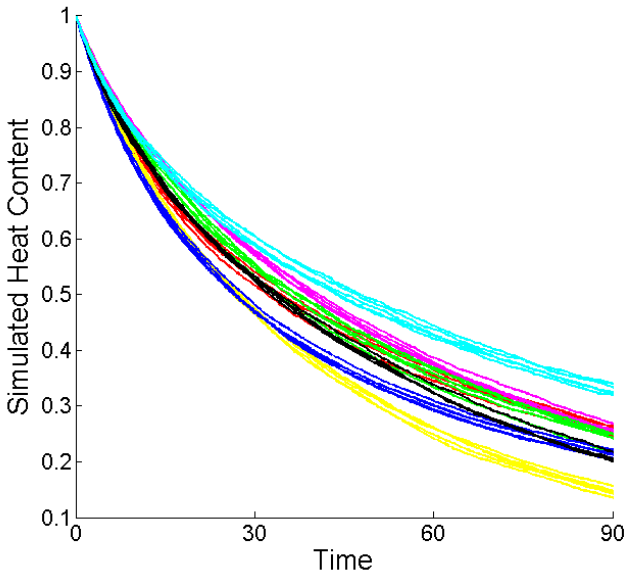
3.4.2 Facial images with pose variations

The second experiment is designed to test our algorithm's ability of handling pose-variations of human facial images. Face recognition is always considered to be a major challenge in computer vision society [37, 76, 78]. Specific algorithms are designed to only deal with the characteristics of the facial images [33, 57, 65].

Figure 3.6 shows a test set contains 36 facial images, in which there are 3 pose variations each for 12 different people. The test samples are chosen from part of the database in [69]. Instead of using natural boundary of the human face, a rectangular frame which are exactly combined by the out contour pixels of the image are treated as the boundary vertices in the graph. Suppose the image heat content for person i of pose j is $h_{ij}(t)$, and we define a distance measure of global image heat content to be $dis(h_{ij}(t), h_{mn}(t)) = \sqrt{\sum_t (h_{ij}(t) - h_{mn}(t))^2}$. We calculate the maximum distances among poses of one person and also the average distances between one person to the other. In this experiment we set the ridge factor to be 0.5. Tables 3.4.2 and 3.4.2



(a) Fish image examples



(b) Heat content curves

Figure 3.5. Affine transformed fish image examples and corresponding image heat content curves. In the graph generation process, $f(I_i, I_j) = |I_i - I_j|$, d is the square of distance, $\epsilon_1 = \epsilon_2 = 1$. In the Monte Carlo estimation, $\delta = 0.1$ and the number of initial random walkers per vertex is 1.

shows the result. We can see that the maximum distances among poses are smaller than the distances between different people, thus the algorithm can naturally cluster the same person's facial images into one community and the algorithm can be used to classify facial images.

Figure 3.6. Human facial images with pose variations

Person	1	2	3	4	5	6
Maximum Distance	0.396	0.571	0.372	0.417	0.475	0.574
Person	7	8	9	10	11	12
Maximum Distance	0.249	0.534	0.668	0.977	0.371	0.797

Table 3.1. Maximum distances of different poses for one person

Person	1	2	3	4	5	6
Average Distance	0	2.211	1.326	2.989	1.835	1.212
Person	7	8	9	10	11	12
Average Distance	1.619	2.181	3.099	1.508	1.715	1.445

Table 3.2. Average distances between person 1 to other people

3.4.3 Natural image retrieval

In this experiment, we selected some natural images online and downloaded to form a testing dateset. We then get rid of the color information of the image. Figure 3.7 shows all the images in the dataset. For this dataset, we compute the normalized multi-scale image heat content for each image and by comparing a distance matrix we illustrate the capability of only using image heat content to do image retrieval.

Figure 3.8 illustrate the distance matrix of the image heat content difference of the all pairs of images. We first compute a baseline heat content for all the images and then input all the images again to calculate the distances of each image to all the baselines, the result is shown in every raw. In the illustration, the dark red entries represent small distances and the dark blue ones represent large distances. In this experiment we set the ridge factor to be 2. Figure 3.9 shows that some examples of the "similar" pairs based on the distances. We can see that our algorithm captures the structural information of the image despite some "similar" pairs of image are actually quite different at pixel level.

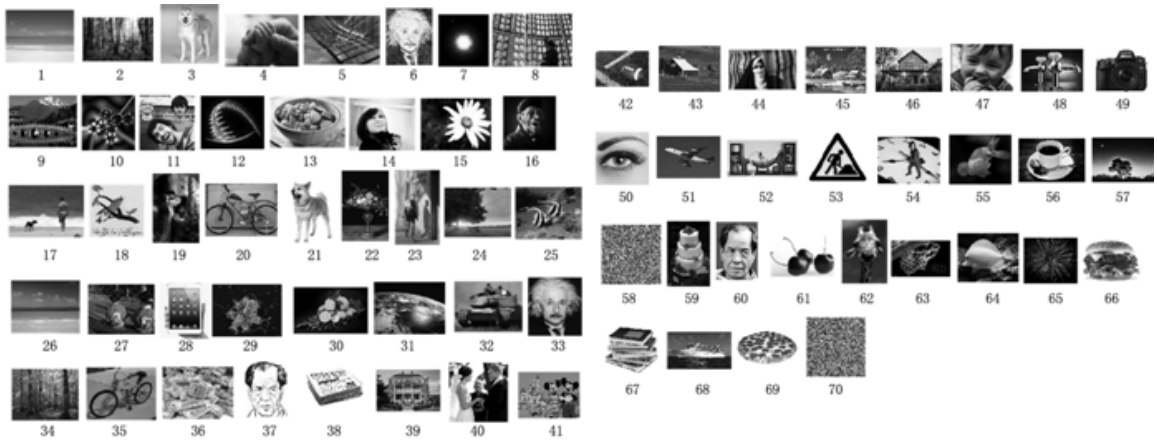


Figure 3.7. Image database

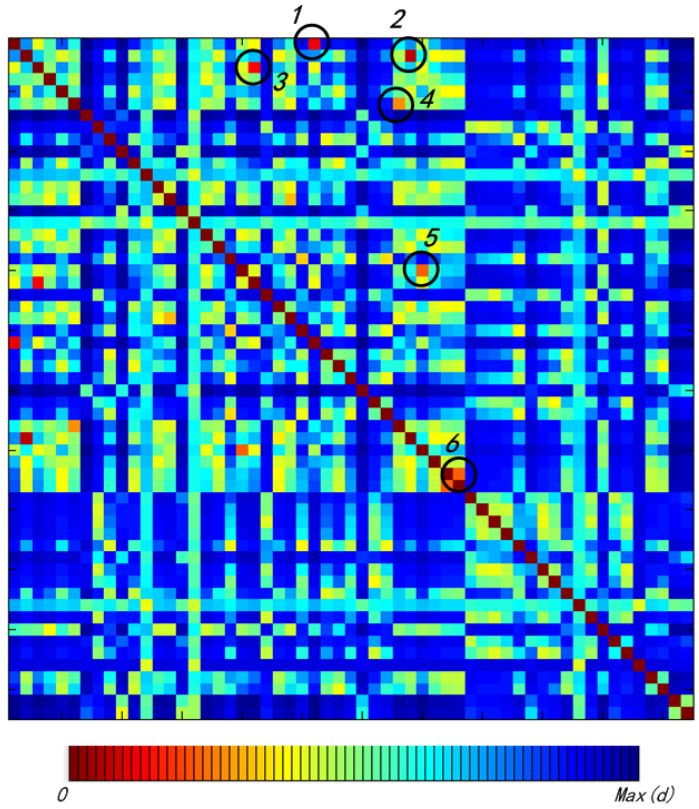


Figure 3.8. The distance matrix of all images

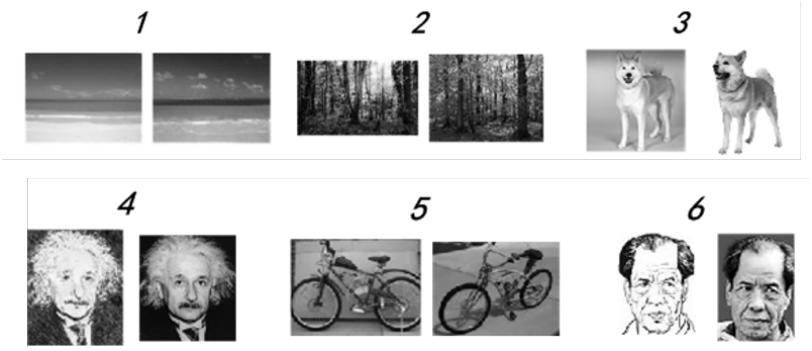


Figure 3.9. Similar pairs in the database

3.4.4 Speech and spectrogram retrieval

In this experiment, we test our feature by a spectrogram retrieval experiment. A spectrogram is an image, which is the result of a short time fourier transform (STFT) of an audio sample.

The experiment is set to check the similarity measurement of the same words in different reading speeds. We record 40 ".wav" files and all of them are 3-seconds speech samples. The 40 samples are divided into 2 groups. The first group is the reference group in which 20 words are read by one person in normal speed. These words are: "University, December, China, America, Massachusetts, Amherst, Christmas, Holiday, Wednesday, Computer, Engineering, Mathematics, Equation, Freedom, Five, October, January, Arsenal, Soccer, Ladygaga". The second group is the test set. In this set all the previous words are read by the same person again but in slower speed. The sampling frequency is $16,000$ Hz in this experiment.

First we do some pre-processing for all the samples. We first use an energy based algorithm to detect the duration time of the speech. Then we only keep the active speech part and remove the rest inactive periods. Afterwards, we calculate the spectrograms of all the 40 active speech samples. The STFT window sizes and the overlapping lengths are set to be the same as the typical widely used setups in speech recognition tasks. These spectrograms are used in the following similarity testing experiment. Figure 4.21 shows the spectrograms of several word examples in the reference group as well as the test group. We can see that in this experiment the test words are read in slower speed than the references. The goal is to check if our algorithm can find the similarities among the two speech samples of the same word despite the differences of reading speed. Figure 3.11 shows the exact ratio of durations between the 20 slower test samples and the 20 original reference samples. From the figure we can see that the reading speeds of the test samples are slower than the reference ones from 15% up to 80% and the average is around 45%.

For all the reference samples and the test samples, we uniformly convert all the spectrograms into the same-size images. Namely no matter how long the duration of the speech is, we resize the spectrogram to a fixed length. Then, for each resized spectrogram S, we use our graph generation algorithm to generate a big graph for the

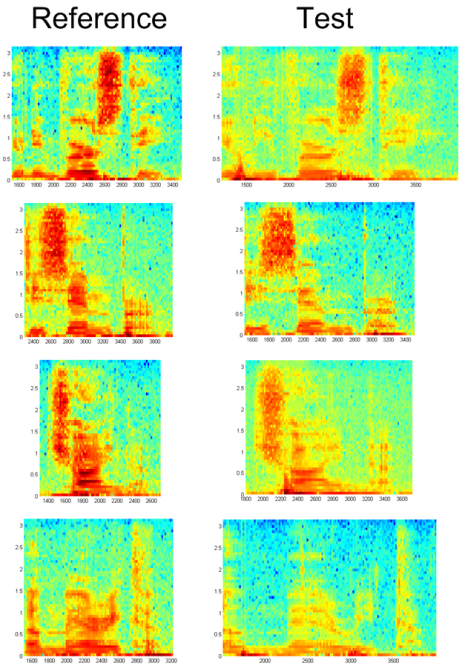


Figure 3.10. Spectrogram examples of the normal and slower samples

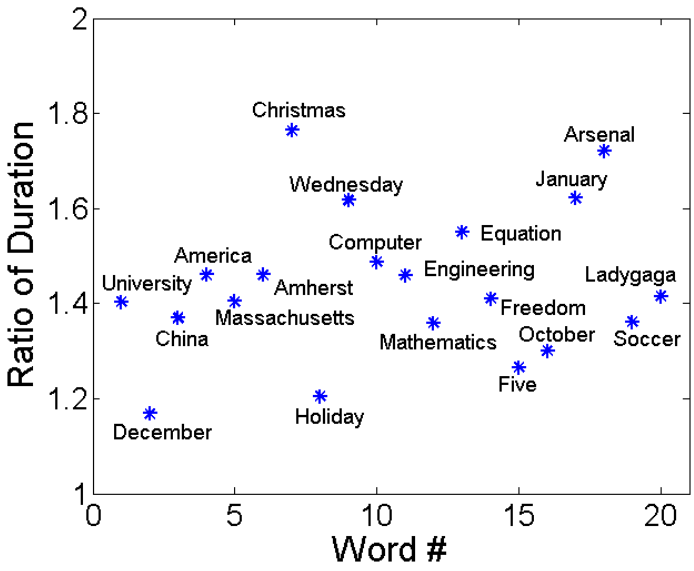


Figure 3.11. Ratio of durations between the test samples and reference samples

whole spectrogram and several small graphs for some image blocks in the spectrogram which can be denoted as G_1 to G_n. We want to capture the general structure as well as some details of the spectrogram. Then we compute the 10-steps heat contents for

all the graphs generated to be h_1 to th_n. All the heat contents are combined as a big vector h which is the feature vector of the spectrogram S. In the experiment, we use some 14 resolution blocks to capture details of the spectrogram.

For every reference speech samples S^1 to S^{20}, we now have the corresponding reference feature vectors h^1 to h^{20}. The next step is to compute the feature vectors for all the test samples and to check the differences between the test samples and all the reference samples. Figure 3.12 shows the Euclidean distance matrix of heat content vectors between all the reference and test samples. We can see that every test sample is correctly matched to its reference one.

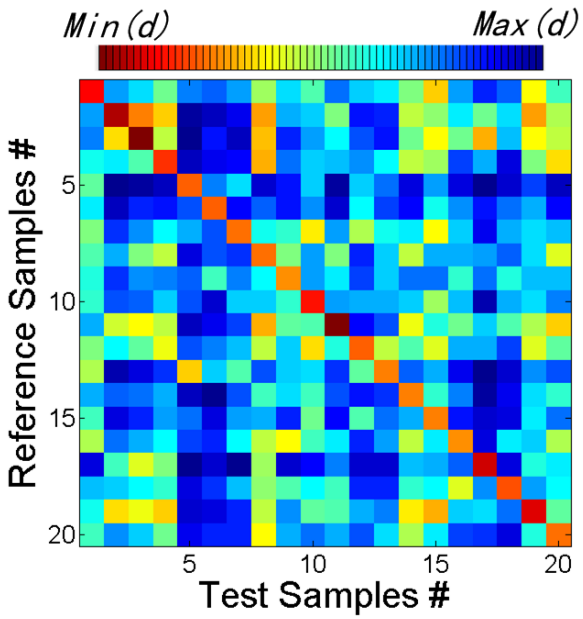


Figure 3.12. Similarity matrix of spectrograms

3.4.5 Hand written digits classification experiment

The MNIST database [42] is a benchmark widely used in image classification algorithm comparison. It contains a training group of $60,000$ images and a testing group of $10,000$ images. One property of this dataset is that all the images are hand

written digits with standard size and contrast, which is very "similar" already. Figure 3.13 shows some example images.

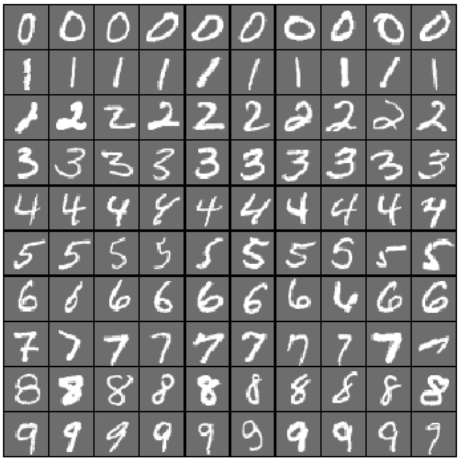


Figure 3.13. Hand-written digit image examples in MNIST database

We first compute the heat content feature for every image based on forty-nine 10×10 random position overlapping local image blocks. Five steps of heat content is used for each image block. Table 4.2 compares the classification error rates of the heat content feature and other blockwise similar-size low-level features including the intensity histogram, intensity moments, Gabor coefficients, gray-level co-occurrence matrix (GLCM) and edge directions histogram. All the features are normalized and the classification algorithm is a k-nearest-neighbors classifier with L2 norm as the distance measure. The result shows that although the heat content feature alone is not the best, it is still better than some single low-level features. More importantly, if we add the heat content feature to any other feature and form a combined feature, the error rate always drops. This result provides preliminary evidence that the heat content feature contains some useful image information which is not represented by the existing low-level image features.

Our next experiment is to simulate the real image retrieval task by combining three types of features (Intensity, texture and shape). We apply logistic and linear kernel

support vector machine (SVM) [12] classifiers to execute the classification. Table 3.4 shows that in all situations the performance of the combined feature with the heat content is better than the original combination in both classifiers. This result further illustrates that the heat content feature contains unique and useful image information as a new type of low-level feature which has the potential to improve the feature exaction step for current image retrieval systems.

Feature	Alone	with **HC**
Intensity histogram (Histogram)	10.76%	**6.89%**
Intensity moments (Moments)	8.94%	**7.04%**
Gabor coefficients (Gabor)	3.05%	**2.94%**
Gray-level co-occurrence (GLCM)	6.92%	**4.49%**
Edge directions (Edge)	3.68%	**3.15%**
Heat content (HC)	6.54%	N/A

Table 3.3. Classification error rate (k-NN classifier)

Feature	Logistic	SVM
Histogram + Gabor + Edge	2.12%	1.47%
Histogram + Gabor + Edge + **HC**	**2.01%**	**1.31%**
Histogram + GLCM + Edge	2.58%	1.62%
Histogram + GLCM + Edge + **HC**	**2.28%**	**1.54%**
Moments + Gabor + Edge	2.06%	1.30%
Moments + Gabor + Edge + **HC**	**1.90%**	**1.24%**
Moments + GLCM + Edge	2.36%	1.48%
Moments + GLCM + Edge + **HC**	**2.20%**	**1.41%**
Combined feature	1.82%	1.29%
Combined feature with **HC**	**1.78%**	**1.22%**

Table 3.4. Classification error rate (logistic/SVM classifier)

3.5 Summation

Feature extraction is the first and most fundamental step of a fast image retrieval system. In this section, we propose a graph-based image representation associated with a fast graph similarity comparison algorithm based on the asymptotic behavior of the heat content. The heat content feature is shown to be a robust, easily computed

image feature and has the potential to be an effective and efficient multi-scale feature for image retrieval. The heat content feature can also be combined with other existing low-level features to create a complex visual signature which may improve the following classification/retrieval tasks. Although we still need further experiments and analysis to thoroughly understand the advantages and drawbacks of the heat content feature, our preliminary results show that heat content computation could be a useful component of image retrieval.classification tasks.

CHAPTER 4

IMAGE UNDERSTANDING BASED ON SPECTRAL GRAPH INFORMATION

4.1 How to improve the heat content feature?

In the last chapter, our preliminary result shows that the heat content feature can improve the image retrieval/classification performance when combined with existing low level features. However, for a mid-size image with more than $10,000$ pixels, the image heat content feature can only be roughly estimated. Moreover, the functions that are summed to generate the feature decay exponentially with rates given by the graph Laplacian eigenvalues; hence, information carried by larger eigenvalues is mostly lost.

In this chapter, we first introduce several approaches to drastically reduce the size of the traditional pixel-based graph representation. By detecting corner/edge pixels or merging nodes, our model can generate a much smaller graph for the same image, which makes complicated feature extraction methods such as spectral analysis a possible option. We then formally propose an analysis of forming the asymmetric graph to represent the image, where the oscillatory heat content contains additional useful frequency information. We also propose a novel feature extraction model based on the spectral graph information. Experiments show that our re-designed features perform better than the original heat content feature and is a more effective supplement to further improve the performance of traditional feature-based image retrieval and classification.

4.2 Graph generation of large images

Traditional pixel-based graph generation precisely represent the pixel relations in the image. However, it is a challenging task to apply powerful tools such as spectral analysis when the graph is very large. Even when the graph is made to be sparse, the size is still too large for a mid-size natural image input. In this section, we propose three graph generation model which has a far smaller number of nodes compared to pixel-based graph generation models.

4.2.1 Graph generation by corner detection

We first introduce a typical widely used approach to construct a reduced-size graph which represents an arbitrary grey-level image based on Harris corner detection and Delauney triangulation. The first step of generating the network is to set up vertices of the graph on the given image. A typical method which is to use Harris Corner Detector to generate vertices for the graph. The Harris Corner Detector is based on the local auto-correlation function of a signal, where the local auto-correlation function measures the local changes of the signal with patches shifted by a small amount in different directions.

Given a shift $(\Delta x, \Delta y)$ and a point (x, y), the auto-correlation function is defined as,

$$c(x, y) = \sum_{W} [I(x_i, y_i) - I(x_i + \Delta x, y_y + \Delta y)]^2. \tag{4.1}$$

where $I(\cdot, \cdot)$ denotes the image function and (x_i, y_i) are the points in the window W (Gaussian) centered on $(x.y)$. The shifted image is approximated by a Taylor expansion truncated to the first order terms, Harris Corner Detector is the approximation of the function and the result is a corner response map R associated with the original image. We take the local maxima of R and they becomes the vertices we want.

A Delaunay triangulation for a set P of points in the plane is a triangulation DT(P) such that no point in P is inside the circumcircle of any triangle in DT(P).

And the next step is to set the outside nodes as boundary nodes, as Figure 4.1 shows, the blue points and lines are the interior vertices and edges of the network, the red points are the boundary vertices of the network.

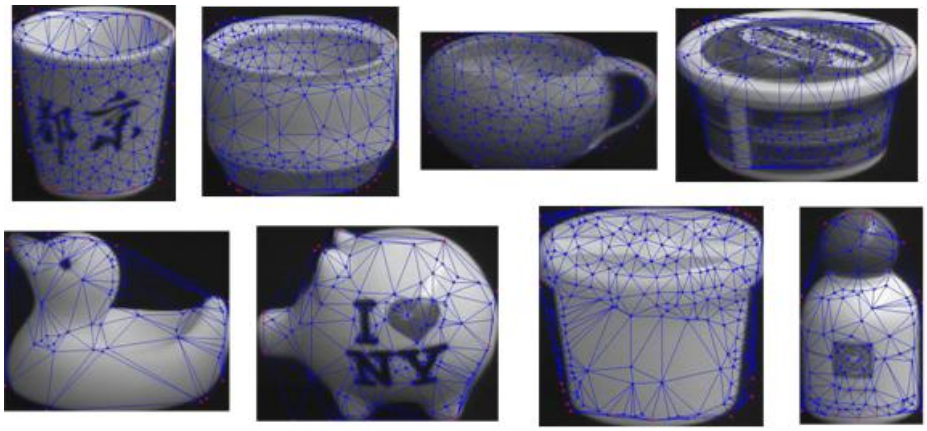


Figure 4.1. Delaunay network generation of the image

Then we build a weighted transition matrix in which the transition probability from vertex u to v is inversely proportional to the square of the distance of the link between vertices u and v. It means that a weight $W_E(u,v)$ of each edge (u,v) is defined by the square of the Euclidean distance between u and v which is $d^2_{(u,v)}$. So we have $W_E(u,v) = W_E(v,u) = 1/d^2_{(u,v)}$. Then the transition probability from vertex u to vertex v, $P(u,v)$ is

$$P(u,v) = \frac{W_E(u,v)}{w_V(u,u)} \tag{4.2}$$

We apply the heat content estimation in the experiment. In the test data set, we use the contours of fishes to be the original image. The basic fish contour database we used here contains 20 types of fish, as figure 4.2 shows.

To each type of fish, as we discussed before, a Harris-corner detector is used to locate and acquire the nodes of our graph and then we fix the boundary nodes according to its shape. Meanwhile we connect these nodes by a Delauney network. Then we do 20 affine transforms to the original fish shapes in each type. In the transform, the nodes (including the boundary nodes) are fixed but the Delauney

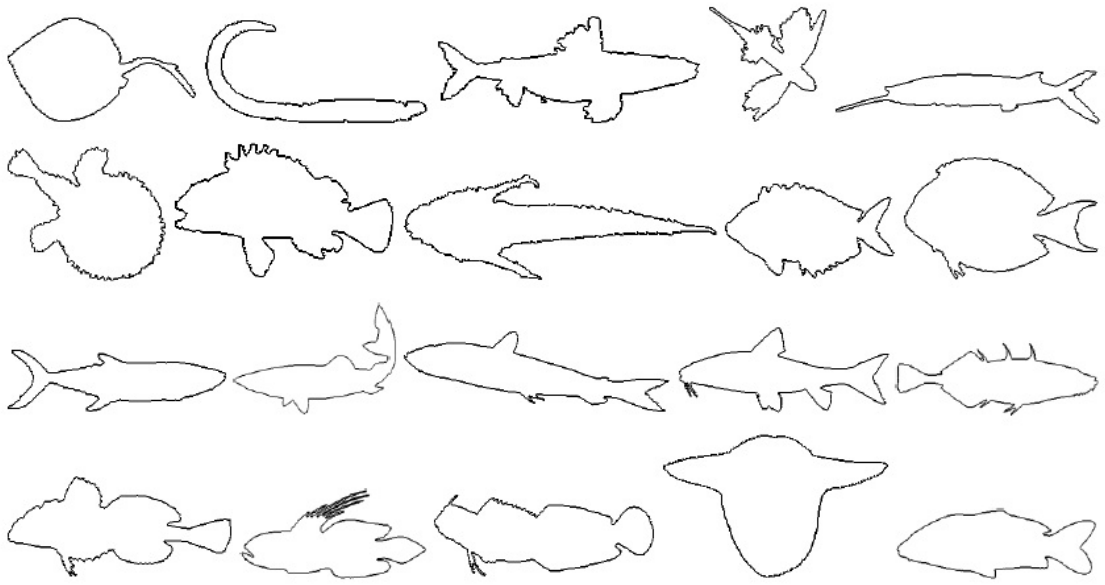


Figure 4.2. Fish Contour Database

network is reconstructed, namely the network has the same number of the total nodes and boundary nodes, but slightly different network structures which caused by the changes of shapes. Figure 4.3 shows some examples of such networks after the transform.

We randomly choose 10 fishes in each type to forms a training set. Then we compute the heat content estimation if all the fish shapes. Figure 4.4 shows the heat content curves of 7 types of fishes. We then use a distance defined as the following equation to do the recognition of fishes in the test set:

$$\| hc_1 - hc_2 \| = \sqrt{\int_0^t (hc_1(x) - hc_2(x))^2 dx}. \tag{4.3}$$

The successful rate is 94.5% in our experiment, which can even be improved if with more complicated recognition technique. How ever, this approach based on Delauney network and heat content simulation has difficulties in some other classification ap-

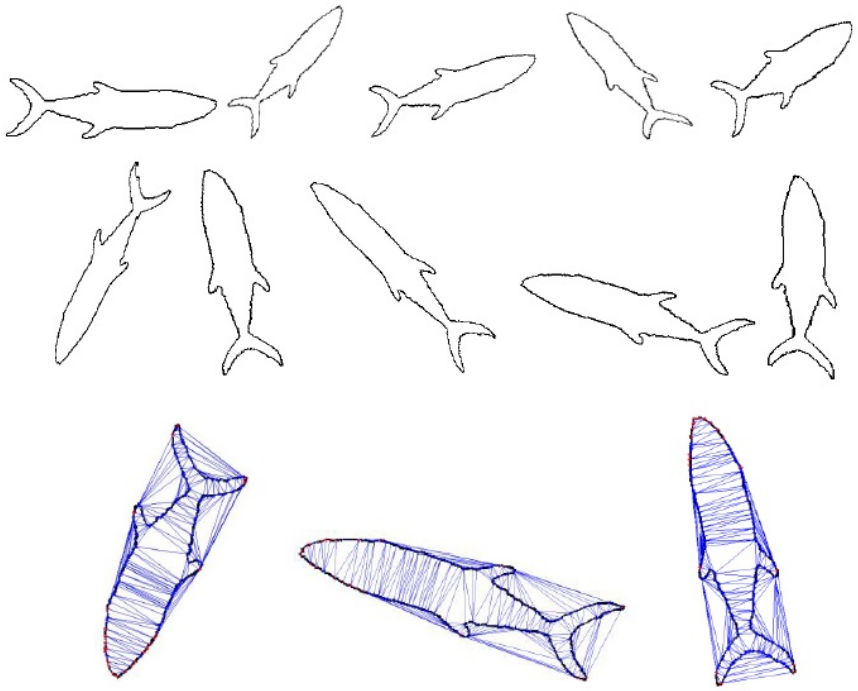


Figure 4.3. Sample fishes and networks after the transform

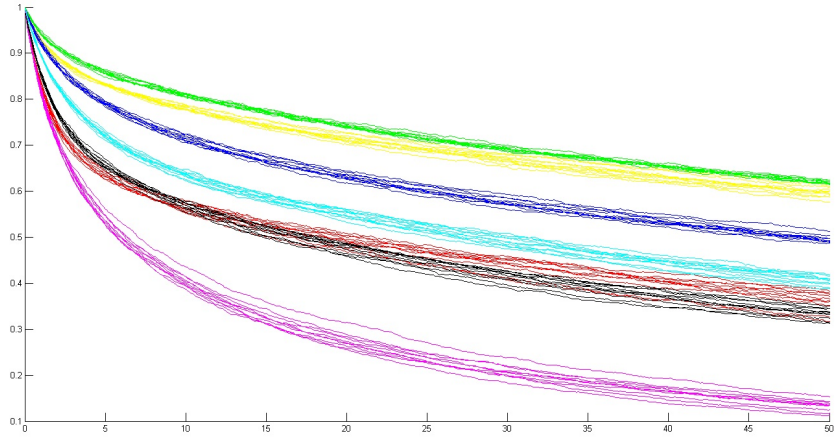


Figure 4.4. Heat content curves of 7 types of fish contours on Delauney graphs

plications. The major problem is that normally the network only contains hundreds of nodes and the boundary nodes plays a very important role in the upcoming heat content simulation process due to the small scale of these networks, and Harris corner

detector could not provide us stable output of vertex positions and total number of boundary nodes. That will cause serious problem in real classification problems. For example, if we use the algorithm on the previous facial image experiment, the result is not very good.

4.2.2 Graph generation by nodes merging

The first simple but effective small-size graph generation model is based on the super-pixel pre-segmentation of the image. We first divide the image into super-pixels using a very fast and effective SLIC-like algorithm [1]. In each super-pixel areas, the image area has very similar pixels with almost the same color and intensity contents. The original RGB colorful image is transformed to CIELAB format with the pixel representation of $[L, a, b, x, y]$. L is a general intensity measurement and a, b describe the visual color of the pixel. The SLIC-like super-pixel is generated by iteratively applying the following distance calculation of the similarity between a pixel and the local super-pixel center:

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

The color distance is defined as $d_c = \sqrt{(L_i - L_j)^2 + (a_i - a_j)^2 + (b_i - b_j)^2}$ and the location distance is defined as $d_s = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. S and m are controlling factors for the size of super-pixels. The center of the super-pixel is the local pixel with the lowest gradient position. Figure 4.5 shows a super-pixel segmentation of two images. The super-pixel pre-segmentation of the original image not only tremendously reduces the following computational complexity, but also captures most of the essential local edge information of the image. We construct the simple graph representation of the image based on the super pixel segmentation using the following equation

$$A_{sp}(i,j) = \begin{cases} k_1\left(\frac{d_c}{d_s^2}\right) & (i \neq j) \\ 0 & (i = j). \end{cases}$$

Figure 4.5. Super pixel segmentation of the image

Our second small-size graph generation model is derived from traditional pixel-based models. We can reduce the size of the graph by "nodes merging". Specifically, if some pixels form a homogenous area, they can be clustered into an image segment. In our graph generation model, we assign a single node for such an image segment. By doing so, the total number of nodes in the graph is greatly decreased.

The first step of this model is similar to our original pixel-based graph model in the previous chapter. We generate a vertex for every pixel in the image and the weight of any edge is determined by a distance measurement between the corresponding pixels. For color images, like the super-pixel algorithm, the original RGB representation is transformed to the CIELAB format with every pixel represented by a five-dimensional vector $[L, a, b, x, y]$. L is an intensity measurement, a, b describes the visual color and x, y represents the location of the pixel on the image. We compute two distances to form the definition of the edge weight. For any two pixels p_i, p_j, the color distance is defined as $d_c = \sqrt{(L_i - L_j)^2 + (a_i - a_j)^2 + (b_i - b_j)^2}$ (for grayscale image we just use the intensity part) and the geometrical distance is defined as $d_s = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. The edge weight between any two pixels p_i and p_j

is defined as

$$w_{ij} = e^{-d_c^2/\sigma_I} \cdot e^{-ds^2/\sigma_X}, \tag{4.4}$$

where σ_I and σ_X are controlling parameters. On the other hand, we can also use our original graph generation model [38] in this step to produce pixel-based graph for the image to still focus on the high contrast part of the image.

The pixel-based graph describes similarities between pixels and captures important structural information on the image. Our goal is to merge similar nodes together while keeping the major structure of the graph intact. A MinMax K-Means clustering algorithm [70] is applied to acquire a fast segmentation for the image.

Suppose we generate k different-size segments S_1, S_2, ..., S_k after clustering. We merge all the nodes in each segment together and form k new nodes s_1 to s_k to be the vertices in our new graph. All the edges connected to any node in set S_i from outside in the original graph will then become links connected to the single node s_i in the new graph. At the same time, all the inside edges in each segment are summed up to be a self-loop edge for the corresponding new node s_i. Precisely, the weight between nodes s_m and s_n in the adjacency matrix of our "after-merging" graph is defined as

$$w_{mn} = \begin{cases} e^{-\frac{d_c^2}{\sigma_I}} \cdot e^{-\frac{ds^2}{\sigma_X}} & (d_s < th) \\ 0 & (d_s \geq th) \end{cases} \tag{4.5}$$

We can easily prove that the total degree of the graph does not change. The structure of the graph also remains intact. Considering a random walk on our new graph, the steady-state distribution of the random walkers can be easily calculated from the original distribution by adding entries from the same cluster together. Although there is some information loss inside every image segment, the general heat diffusion pattern is largely equivalent to the original one due to the fact that these image segments are mostly homogenous patches.

A natural image usually has large image segments. Depending on the parameters set for the K-Means clustering, for a standard configuration, the size of the graph normally reduces to less than one percent of the original size, yet most of the structural information of the image still remains. For example, the generated graph only contains hundreds of nodes for a typical 128×128 as shown input in figure 4.6. The model not only generate a much smaller graph which makes eigen-decomposition no longer a big challenge, but also naturally captures important image information through node merging of homogenous image segments.

Figure 4.6. Pre-segmentation of image

We can also combine the super-pixel graph generation model and the node merging graph generation model if the image is very large. Based on the super-pixel segmentation of the image, each super pixel is then represented by a similar 3-dimensional average property vector $[L, a, b]$ with the average position $[x, y]$. We can then calculate the similarity between neighboring super pixels too by equation 4.5 in which d_c and d_s are the distances of the color and location of two super pixels, repectively. The following procedure is the same and we can therefore generate a very concise graph representation even for a very large high-resolution natural image.

Interestingly, our new graph's degree distribution follows power law. Figure 4.7 shows two examples of power law degree distributions generated from natural images. Figure 4.8 shows a comparison between our new power law graph and the original pixel-based graph. The degree distributions and the weight spectrums are shown to demonstrate that the new power law graph is capable of capturing more high frequency behaviors as larger values in right part of the weight spectrums.

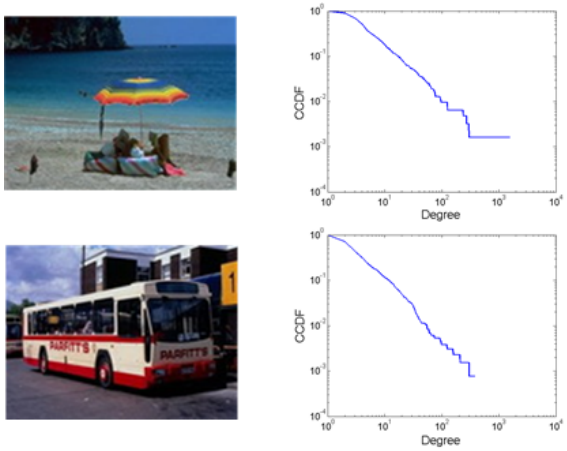


Figure 4.7. Images and their corresponding power law degree distributions

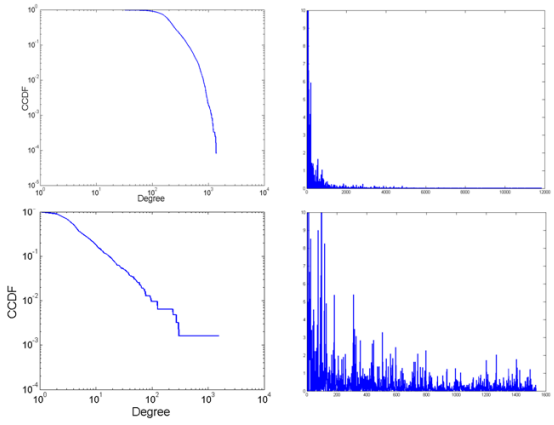


Figure 4.8. Degree distributions and weight spectrums (First row: original graph. Second row: power law graph. Left: CCDF. Right: weight spectrum.)

4.2.3 Graph generation based on edge detection

Another possible way to generate the reduced-size graph is based on the edge detection result of the image. We use the Canny edge detector to acquire the position of the edge pixels and we use the information in the intensity gradient of the corresponding pixels to define the graph representation of the image. Suppose the edge detection operator returns a value for the first derivative in the horizontal direction (G_x) and the vertical direction (G_y). From this the edge gradient and direction can be determined as

$$G = \sqrt{G_x^2 + G_y^2} \tag{4.6}$$

$$\Theta = \arctan G_y, G_x. \tag{4.7}$$

The following step in the graph generation process is to build a network using the information of Θ and the relative position of the pixel. Intuitively we want to make "corner point area" having larger weight in the graph. The weight of the edge between two pixels is defined by the following equation

$$w_{ij} = \begin{cases} \frac{|\Theta_i - \pi/2| - |\Theta_j - \pi/2|}{d_s^2} & (d_s < th) \\ 0 & (d_s \geq th) \end{cases}$$

in which d_s are the distances of the location of two edge pixels.

4.3 Improved image representation model based on spectral graph information

Spectral analysis is a powerful tool to extract useful structural information on the graph. For a large-size graph, fast eigen-decomposition becomes very difficult. The image heat content feature in the last chapter is designed to contain all the spectral

information which can also be estimated by a fast Monte-Carlo algorithm. However, by drastically reducing the size of the graph, we can directly acquire the precise result of the eigen-decomposition, which gives us more possibilities for designing features based on the spectral graph information.

Meanwhile, the heat content feature can be also improved in other aspects. We will first propose a modified oscillatory heat content feature based on an asymmetric graph in the following subsection. Second, instead of using a summation of exponentially decaying functions, which is largely determined by the component functions with small eigenvalues, we will propose a novel feature extraction model directly based on the spectral information of the graph.

4.3.1 Oscillatory image heat content

We first modify the symmetric graph into an asymmetric one by simply giving very small weights to edges from low average-intensity segments to high average-intensity segments. While this may seem arbitrary, the directional asymmetry in the resulting graph actually carries more information about the image structure.

The eigenvalues and eigenvectors of the normalized graph Laplacian $\mathcal{L}$ of the asymmetric graph can be complex valued, and exist as complex conjugates. The solution to the heat equation of this asymmetric graph Laplacian $\mathcal{L}$ is $H_t = \Phi e^{-\Lambda t}\Psi$. $\Lambda = diag[\lambda_i]$ is the diagonal eigenvalue matrix. Φ and Ψ are the right and left eigenvector matrices ($\Psi = \Phi^{-1}$). For any complex conjugates pairs of eigenvalues $\lambda = a + bi$ and $\overline{\lambda} = a - bi$ with corresponding eigenvectors ϕ, ψ, $\overline{\phi}$ and $\overline{\psi}$, suppose that $\sum_{uv} \phi(u)\psi(v) = \alpha + \beta i$, then we have

$$\alpha - \beta i = \overline{\sum_{uv} \phi(u)\psi(v)} \tag{4.8}$$

$$= \overline{(\sum \phi)(\sum \psi)} \tag{4.9}$$

$$= (\sum \overline{\phi})(\sum \overline{\psi}) \tag{4.10}$$

$$= \sum_{uv} \overline{\phi}(u)\overline{\psi}(v). \tag{4.11}$$

The summation is

$$e^{-\lambda t} \sum_{uv} \phi(u)\psi(v) + e^{-\overline{\lambda} t} \sum_{uv} \overline{\phi}(u)\overline{\psi}(v) \tag{4.12}$$

which is equal to

$$e^{-(a+bi)t}(\alpha + \beta i) + e^{-(a-bi)t}(\alpha - \beta i) = 2e^{-at}(\alpha \cos(bt)) + \beta \sin(bt)). \tag{4.13}$$

So the summation is still a real-valued function. Therefore, the total summation is still real valued and can be written as

$$Q(t) = \sum_{i=1}^{|iD|} e^{-a_i t}(\sqrt{\alpha_i^2 + \beta_i^2} \sin(b_i t + \arctan \frac{\beta_i}{\alpha_i})). \tag{4.14}$$

Compared to the original heat content, the new oscillatory heat content (OHC) is no longer a summation of purely exponentially decaying functions. It becomes an oscillatory function which contains components of different frequencies, amplitudes and phases. These variations can be helpful in classification tasks. We can also amplify the asymmetric part of $\mathcal{L}$ to generate more pairs of complex eigens by defining a new matrix $\mathcal{L}_k = (\mathcal{L} + \mathcal{L}')/2 + k(\mathcal{L} - \mathcal{L}')/2$ where $k > 1$.

We use the following example to show the components in the asymmetric graph heat content. The asymmetric graph shown here is a fully connected graph G=(V,E) with $|V| = 30$. The edges set $E = \{(u, v), u, v = 1, \ldots, |V|\}$ is separated into two

parts: $E = \{E_1, E_2\}$. Define the weight matrix $W = [w_{i,j}]$ with each element as below:

$$w_{i,j} = \begin{cases} \frac{100}{d_{i,j}} & (i,j) \in E_1 \\ \frac{1}{d_{i,j}} & (i,j) \in E_2 \end{cases}$$

where $d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ is the distance between vertex i and j. In Figure 4.9, the edges in E_1 are draw in the graph using arrows. Edges in E_2 are omitted in the figure.

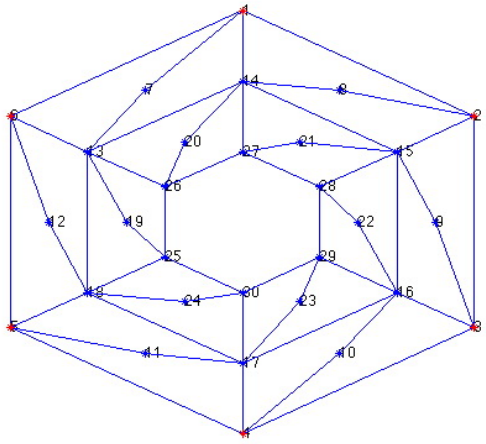


Figure 4.9. Asymmetric graph with edges in E_1

The eigenvalues and eigenvectors of the graph's Laplacian is shown in Table 4.3.1. In the table, two conjugated eigenvalues can be seen as one frequency component of the heat content function. For λ_2 and λ_3, the heat content component is

$$\begin{aligned} H_{2,3} =& (-2.1414 - 1.097i)e^{-(1.329+0.4435i)t} + (-2.1414 + 1.097i)e^{-(1.329-0.4435i)t} \\ =& e^{-1.329t}(-2.1414 - 1.097i)(\cos(-0.4435t) + i\sin(-0.4435t)) + \\ & e^{-1.329t}(-2.1414 + 1.097i)(\cos(0.4435t) + i\sin(0.4435t)) \\ =& e^{-1.329t}(-2 \times 2.1414\cos(0.4435t) - 2 \times 1.097\sin(0.4435t)) \\ =& e^{-1.329t}(-4.2828\cos(0.4435t) - 2.194\sin(0.4435t)). \end{aligned}$$

For real eigenvalues, the heat content component is simple, take λ_1 as an example,

$$H_1 = 40.6379e^{-0.0193t}.$$

k	1	2	3	4
λ_k	0.0193	1.329+0.4435i	1.329-0.4435i	1.4858
α_k	40.6379	-2.1414-1.097i	-2.1414+1.097i	1.1364
k	5	6	7	8
λ_k	0.7302+0.3524i	0.7302-0.3524i	1.36	1.0168+0.3138i
α_k	8.4138-13.046i	8.4138+13.046i	-15.2173	-12.365-10.6865i
k	9	10	11	12
λ_k	1.0168-0.3138i	0.7325+0.0686i	0.7325-0.0686i	1.1235+0.1867i
α_k	-12.365+10.6865i	-2.3982+2.9213i	-2.3982-2.9213i	8.0052+0.2153i
k	13	14	15	16
λ_k	0.899-0.0717i	0.9951+0.132i	0.9951-0.132i	0.9649+0.0865i
α_k	0.947-0.2215i	-0.0708-0.8629i	-0.0708+0.8629i	-0.8915-0.7349i
k	17	18	19	20
λ_k	1.1235-0.1867i	0.899+0.0717i	0.9649-0.0865i	1.077+0.0928i
α_k	8.0052-0.2153i	0.947+0.2215i	-0.8915+0.7349i	-0.1227+0.2333i
k	21	22	23	24
λ_k	1.077-0.0928i	1.1471+0.0127i	1.1471-0.0127i	1.1046
α_k	-0.1227-0.2333i	0.0108+0.3151i	0.0108-0.3151i	-1.3316

Table 4.1. Eigenvalues and eigenvectors of the graph's Laplacian

The asymmetric heat content totally have 14 frequency components. We draw the 14 components separately and compare them with the added up heat content, as shown in Figure 4.10. The dark black line is the total heat content curve. The red solid red line is curve for H_1, which is the low frequency component of the heat content. As shown in the figure, at the beginning part of the heat content curve, the high frequency components pull down the curve; and at last, the low frequency component dominates.

The asymmetric heat content can also be simulated by collecting traffic of random walk at each step with correction $(du/dv)^{1/2}$ for the transition starting at vertex u and ending at v. Take out the correction, the curve will be like the red curve in Figure 4.11. The green curve is the same as the heat content curve in previous part. Considering random walk with correction $(du/dv)^{2/3}$ and with correction du/dv, the

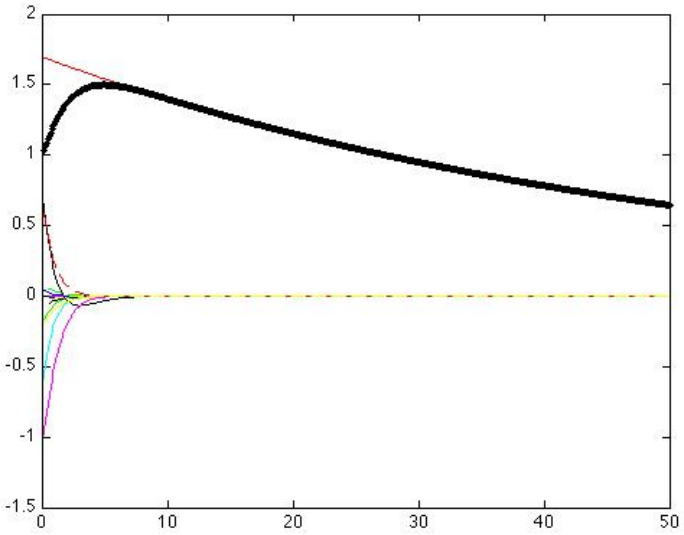


Figure 4.10. The 14 heat content components of the directed graph

curves are the blue and black curve in Figure 4.11 separately.

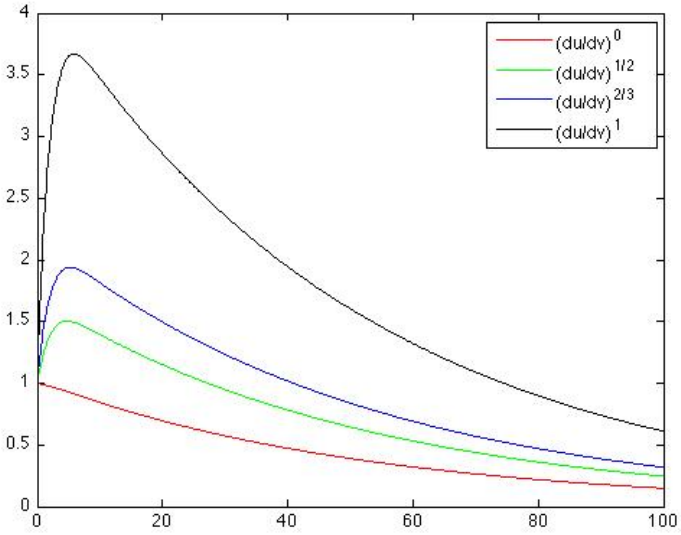


Figure 4.11. The curves under different du/dv corrections

4.3.2 Weighted eigenvalues

The second approach in our algorithm is the feature extraction of the graphs generated in the first section. The original heat content is first computed for the graph. Based on such result, our new feature extraction method is designed to be the combination of three component descriptors.

The first component descriptor in our improved feature is based on the information of the eigenvalue spectrum. Suppose we have the λ-spectrum to be $[\lambda_1, \lambda_2, \ldots, \lambda_n]$, the component descriptor $[f_1, f_2, \ldots, f_m]$ is defined as

$$f_j = \begin{cases} k_1 n & (j = 1) \\ k_2 \lambda_2 & (j = 2) \\ k_3(\lambda_n - 2) & (j = 3) \\ k_4 t_1 / n & (j = 4) \\ k_5 t_2 / n - 1 & (j = 5) \\ k_j |\lambda_{p_j n} - 1| & (j = 6, \ldots, m) \end{cases} .$$

t_1 and t_2 are locations on the spectrum that λ are very close to 1 in between, p_i and k_i are some coefficients. Figure 3 shows the original eigenvalues as well as our designed feature vectors. We can see that the original eigenvalues are very hard to distinguish, yet the proposed feature vectors are quite effective both in differentiating and clustering.

4.3.3 Weighted heat content spectrum

The second component in the improved model is motivated by the middle step of calculation the original theoretical heat content. Starting from the original heat content in (3.4), we define the heat content spectrum as $[\alpha_1, \ldots, \alpha_{|iD|}]'$, in which

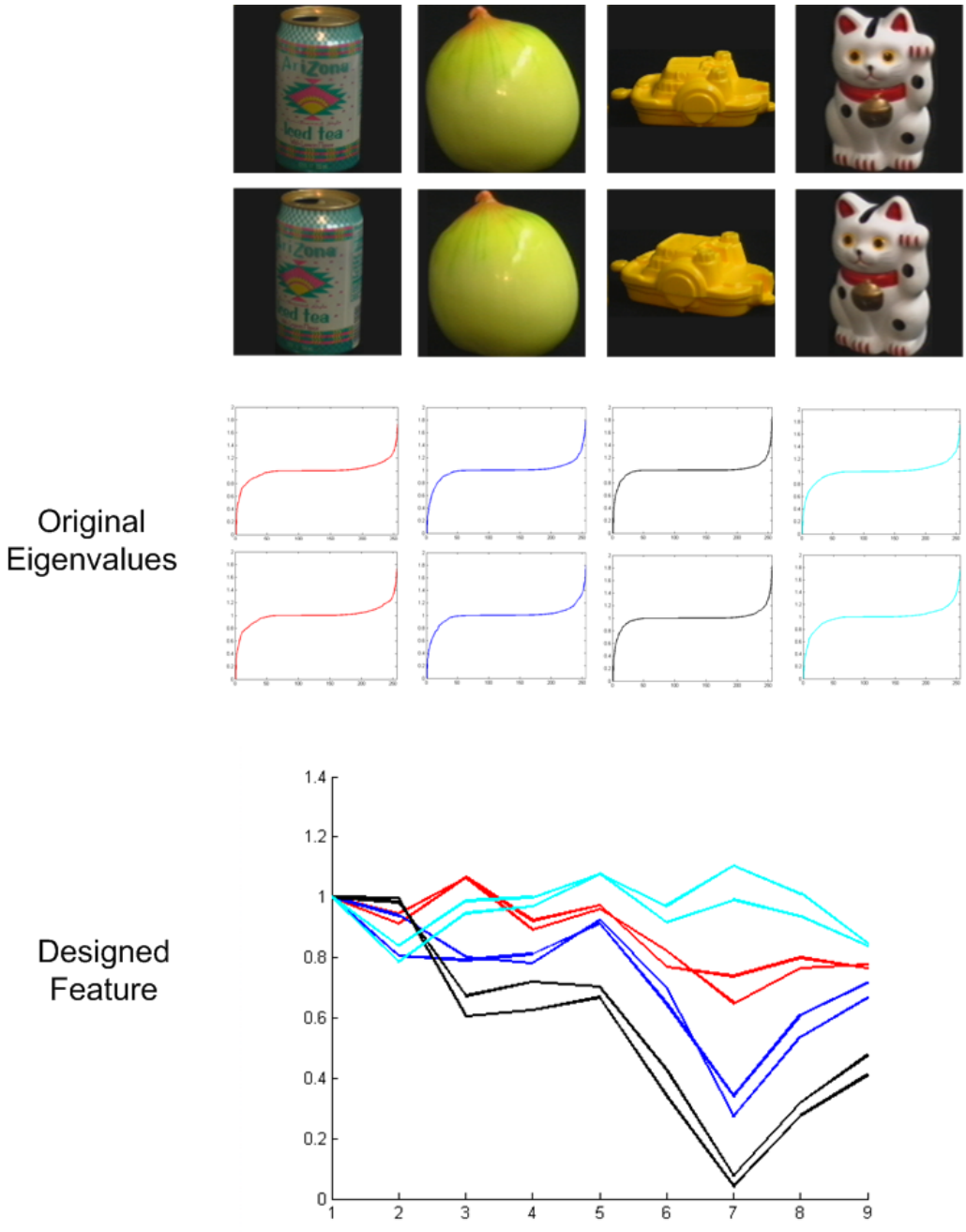


Figure 4.12. Eigenvalue distribution descriptor of the image

$$\alpha_i = \sum_{uv} \phi_i(u)\phi_i(v) = \left[\sum_{u} \phi_i(u)\right]^2 . \tag{4.15}$$

The original heat content can then be expressed as $Q(t) = \sum_{i=1}^{|iD|} \alpha_i e^{-\lambda_i t}$. The heat content spectrum has some interesting properties. First, the summation of all the αs equals the number of interior nodes. Because $\lambda_1 \approx 0$, we have the first eigenvector $\phi_1 \approx \left[\sqrt{\frac{d_i}{m|iD|}}, \ldots, \sqrt{\frac{d_n}{m|iD|}}\right]^T$ (m is the mean degree of the graph).

We use the following example to illustrate the property of the heat content spectrum. In this example illustration, we generated totally 32 graphs. Figure 4.13 shows the degree distributions of a powerlaw graph and a lognormal graph with the same mean degree. The graphs are:

(1) 4 Erdos-Renyi random graphs.

(2) 4 power law random graphs generated by Barabasi-Albert model.

(3) 12 power law random graphs generated by 3 different models (Molloy Reed model, Kalisky model, Model A [46]) based on the power law degree distributions in (2).

(4) 12 lognormal random graphs generated by 3 different models based on degree sequences sampled from lognormal distributions.

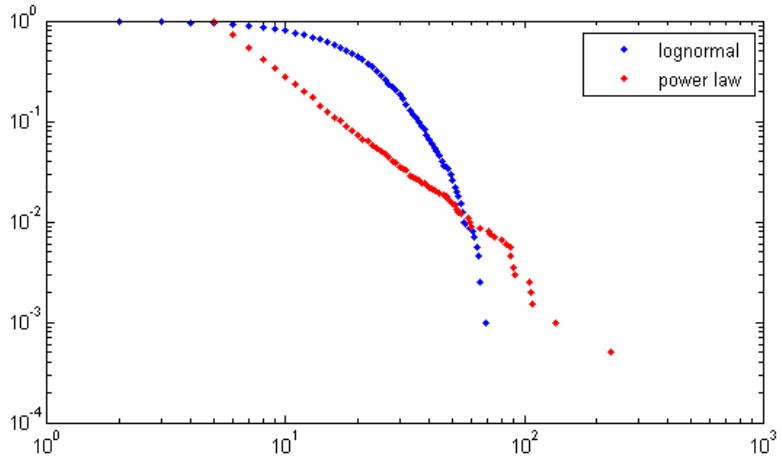


Figure 4.13. Degree distribution of graphs with power law and lognormal degree distributions

We calculate the heat content spectrums for every graph. Figure 4.14 shows the spectrums of all the graphs. We can see that graphs generated by the same model

share similar shape of the weight spectrum. We can also see that the magnitude of α_1 is related to the density of the network as well as the type of degree distribution of the network.

In general, α_1 increases when the mean degree of the network increases, which means that the rest αs will decrease accordingly because $\sum_{i=1}^{n} \alpha_i = n$. For networks which have the same average degree, the α_1 of the power law network, lognormal network, ER network follows a descent order. Figure 4.15 shows such an observation. The blue one is α_1s of ER random graphs, the green ones are α_1s of lognormal random graphs and the red ones are α_1s of power law graphs.

α_1 reaches its maximum $\max(\alpha_1) \approx |iD|$ only when all the vertices have the same degree in the graph. For the rest of the spectrum, the general shape of the weight spectrum captures some major structural information of the graph. Another observation is that the overall shape of the α spectrum is more related with the network models rather than the degree distributions. However, the differences between different distributions are still noticeable. On the other hand, α spectrum can be used to differentiate networks with the same degree distributions generated by different models.

We also demonstrate the ability of the heat content spectrum by the following graph mixture analysis experiment. We generated power law graphs which have the same degree sequence but with drastically different structure. Five types [32] of 2,000-nodes graphs as described in [46] were generated with the average degree set to be $m = 2$.

Figure 4.16 shows sorted (based on the ascending order of elements in the Fiedler vector [67]) adjacency matrices generated by Barabasi-Albert model (BA), Model A, Model B, Kalisky model and Molly-Reed (MR) model with corresponding weight spectrums. The shape of the weight spectrum captures different structural information of the graph very well. The demonstration is particularly clear for Model B as

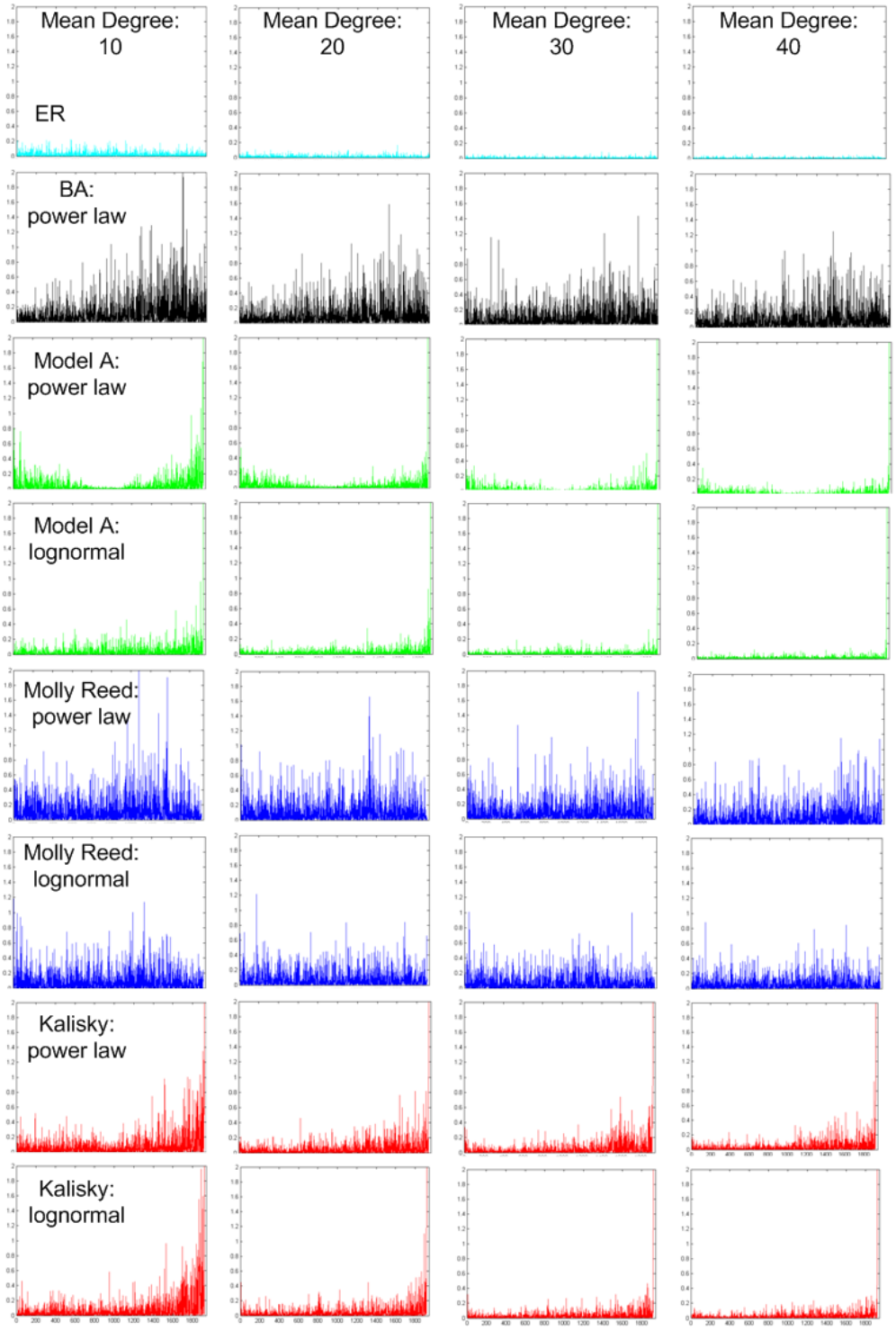


Figure 4.14. The heat content spectrums of power law graphs

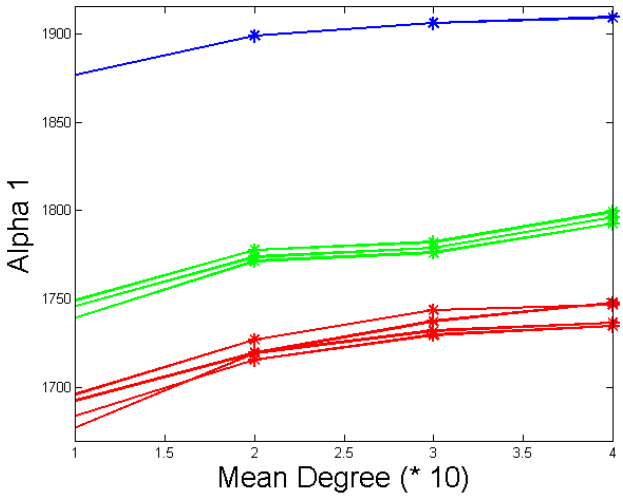
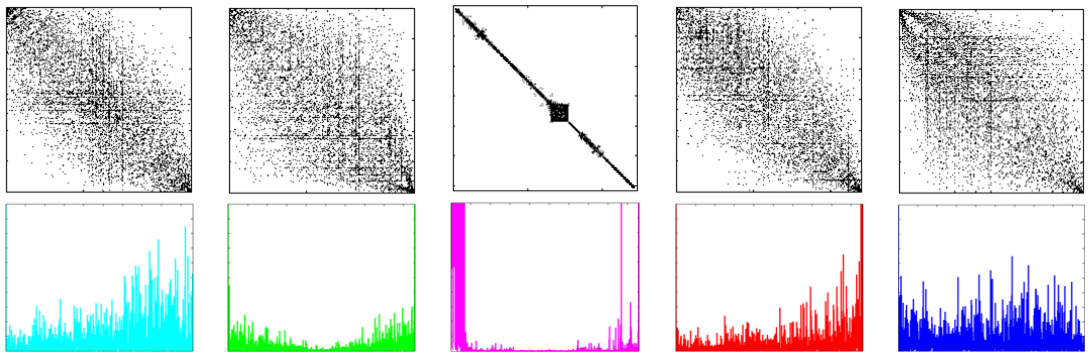


Figure 4.15. $\alpha 1$ and the network's mean degree



(a) BA (b) Model A (c) Model B (d) Kalisky (e) MR

Figure 4.16. Sorted adjacency matrices (first row) and weight spectrums (second row) of graphs with different structure.

the large αs on the left reflect the densely connected sub-graph. Figure 4.17 shows the weight spectrum of graphs generated by combining two models. The weight spectrum of the graph combination of same type models maintains the original shape, and the shape of the spectrum for the combined graph for two different models reflects properties from both models.

In the original heat content spectrum, we can notice that the entries are not evenly distributed and the length of the spectrum is determined by the size of the graph. In

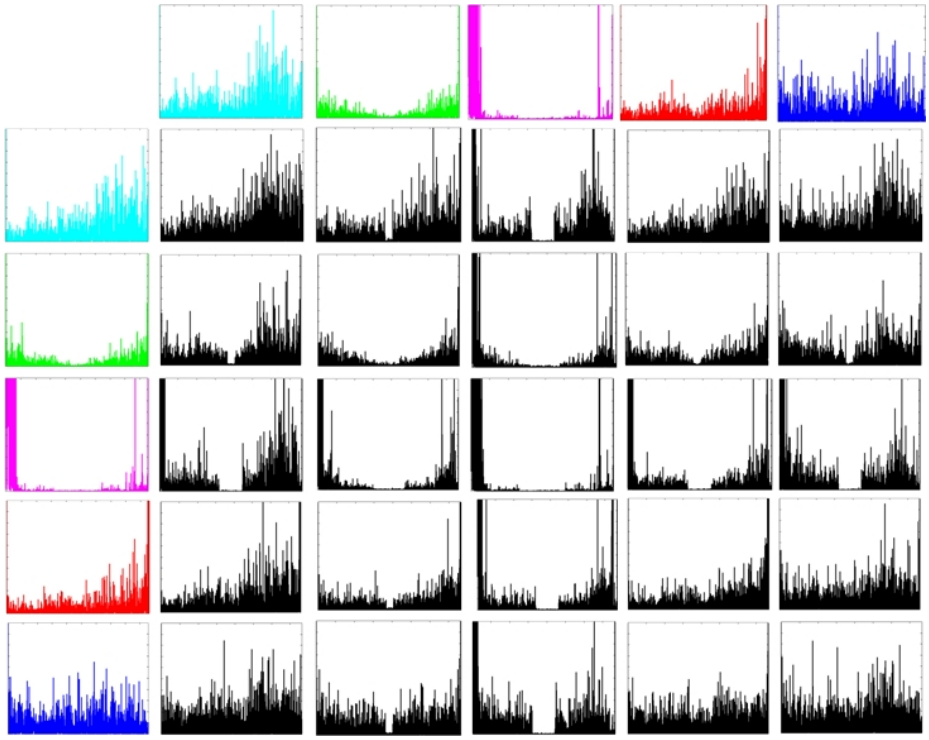


Figure 4.17. Heat content spectrums of graphs combined by two models. The color spectrums are for original graphs and the black spectrums are for combined graphs.

order to create a more general representation, we propose a model which utilizes the information of both the eigenvalue distribution and the heat content spectrum.

In this combined model, every eigenvalue λ_i of the normalized graph Laplacian satisfies $\lambda_i \in [0, 2]$. We first generate k overlapping intervals I_1 to I_k on $[0, 2]$. The middle point of each interval I_i is located on $(2i-1)/k$ and the length of the interval is $2/k + \varepsilon(1 - |(k - 2i + 1)/k|)$. The weighted heat content spectrum (WHCS) $f = [f_1, f_2, \ldots, f_k]$ is then defined as $f_x = \sum_{i:\lambda_i \in I_x} \alpha_i$. We call this the weighted heat content spectrum.

4.4 Experimental results

We first illustrate the proposed new features (OHC and WHCS) by two experiments that show the ability of the OHC feature to differentiate images in different

classes and the robustness of the WHCS feature to view point changes. We then show the improvement of our new features compared to the heat content feature combined with traditional low-level features in a handwritten digits recognition experiment.

4.4.1 Oscillatory heat content

We select thirty-five images from seven categories in the COREL dataset [73]. The images are shown in figure 4.18. We calculate both the heat content based on the original graph in [38] and the oscillatory heat content. Figures 4.19 and 4.20 shows the results. The same color curves represent images in the same category. We can see that although the original heat content can somewhat differentiate these images, the oscillatory heat content performs much better in either clustering or differentiating. The richer frequency behavior of the new feature better represent more diverse images and textures, which will eventually be helpful in a large scale image retrieval task.

Figure 4.18. Seven categories of similar images from COREL dataset

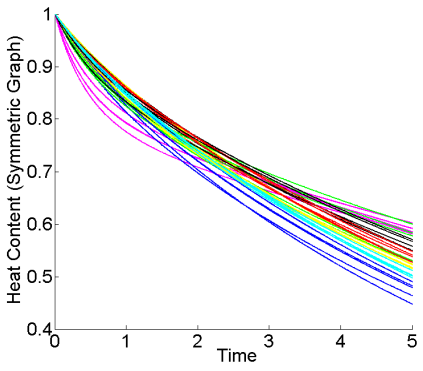


Figure 4.19. Heat Content

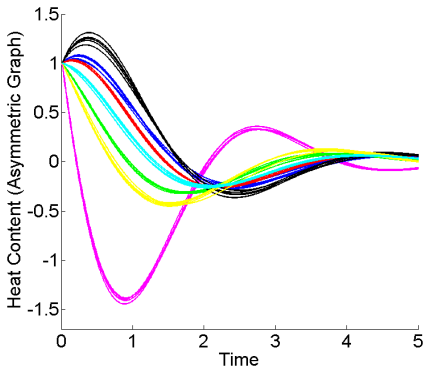


Figure 4.20. Oscillatory HC

4.4.2 Coil-100 color image retrieval

We test the weighted heat content spectrum with the COIL100 [52] dataset. K is set to be 6 in the K-Means clustering algorithm. $\sigma_I = 0.01$ and $\sigma_X = 4$ in the graph generation procedure. Figure 4.21 shows some examples of the images and their corresponding WHCS feature vectors. We can see that the WHCS is robust to small viewpoint changes yet still captures the differences between images. The feature is also effective at differentiating between different objects.

COIL100 contains 100 different objects in 72 different view angles. We select 8 different views for each object to form the reference set and the rest to be the testing set. Table 4.2 shows the resulting classification rate using a 1-nearest-neighbor classifier. The new WHCS is much better than the original HC because it is directly generated by the full-size image. When the feature is used with the multi-scale color histogram, the classification rate is only a little less than the state of the art hand-designed method.

Feature	Classification rate
HC + Color Hist	85.3%
WHCS + Color Hist	92.1%
Typical state of the art [74]	95%

Table 4.2. Classification rates of COIL100

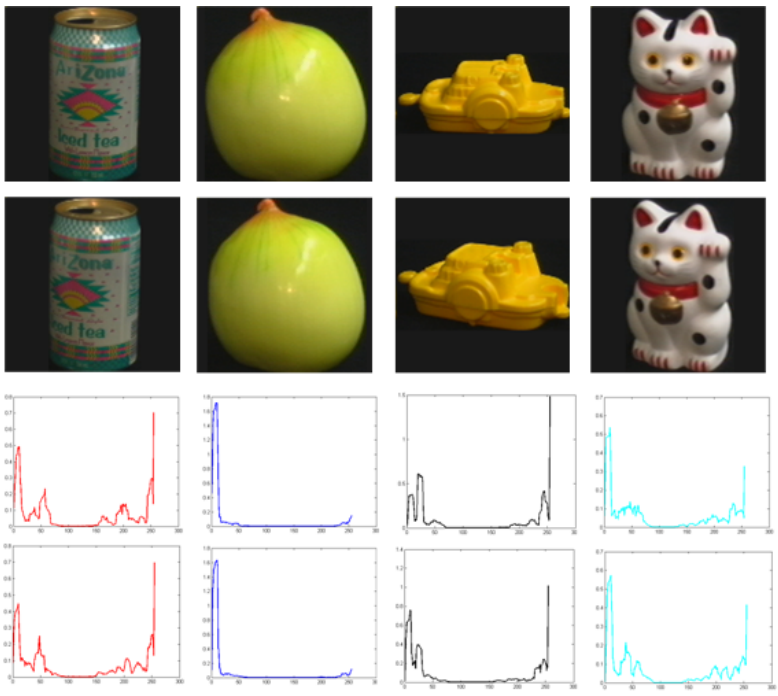


Figure 4.21. Weighted heat content spectrum

4.4.3 Outex-10 texture classification

4.4.4 Hand written digits classification

The MNIST database [42] is a widely used hand written digits benchmark containing a training set of $60,000$ images and a testing set of $10,000$ images. We simulate the image classification task using combinations of different basic features including intensity histogram (Hist), intensity moments (Mo), Gabor coefficients (Gabor), gray-level co-occurrence matrix (GLCM), edge directions histogram (Edge), heat content feature (HC) and the two proposed features oscillatory heat content (OHC) and weighted heat content spectrum (WHCS). The length of the HC, OHC and WHCS is set to be the 10. The combined feature contains all traditional features. All the features are computed for forty-nine 10×10 average-positioned blocks. Logistic and linear kernel supported vector machine (SVM) [12] classifiers are used in the experiment.

Logistic Classifier	Self	w/ HC	w/ OHC	w/ WHCS
Hist + Gabor + Edge	2.45%	2.34%	**2.18%**	2.20%
Hist + GLCM + Edge	2.77%	2.47%	2.41%	**2.39%**
Mo + Gabor + Edge	2.41%	2.37%	2.28%	**2.12%**
Mo + GLCM + Edge	2.54%	2.48%	**2.23%**	2.28%
Combined feature	2.29%	2.18%	2.08%	**1.98%**
SVM Classifier	Self	w/ HC	w/ OHC	w/ WHCS
Hist + Gabor + Edge	1.47%	1.45%	1.42%	**1.28%**
Hist + GLCM + Edge	1.69%	1.62%	**1.50%**	1.52%
Mo + Gabor + Edge	1.54%	1.48%	1.42%	**1.37%**
Mo + GLCM + Edge	1.60%	1.57%	1.45%	**1.42%**
Combined feature	1.39%	1.36%	1.31%	**1.23%**

Table 4.3. Classification error rate

Table 4.3 shows that the performance of the combined features with the OHC and the WHCS are better than the ones with the heat content feature in all situations. In most cases feature sets that include WHCS perform slightly better than those using OHC. This result further illustrates that the proposed two features based on graph spectral information contain unique and useful image information, and could be useful supplement to traditional feature extraction for image classification tasks.

4.5 Summary

In this section, we propose three approaches to reduce the size of pixel-based graph generation and to use graph spectral information for image feature extraction. The proposed image features improve on the original heat content feature and demonstrate an ability to serve as a useful supplement to traditional image feature extraction. Although there are still open questions on the theoretical side, in particular on details of the relationship between the heat content spectrum and graph structure, the experiment results support that the proposed method effectively captures useful image information.

CHAPTER 5

GRAPH-BASED IMAGE PROCESSING AND UNDERSTANDING SOFTWARE DEVELOPMENT

5.1 introduction

Fast image processing and understanding is not only a theoretical or algorithmic problem. A successful engineering application may not use the most state-of-the-art algorithms, unless it is available in a software system. In this chapter, we briefly describe two image related projects which focus on different aspects of the engineering problem. The first project is to apply our multi-scale image heat content to image retrieval software. The second project is a fast graph-matching based marker detection Android application with real-time efficiency and robustness.

5.2 Fast image retrieval/classification desktop application based on image heat content

The image retrieval software is designed to have two major parts. Part one is image retrieval module and part two is image comparison module. The image retrieval module is for standard task of image retrieval, namely by choosing an arbitrary image, the program should select several "visually similar" images to be the result of retrieval, and if the original image happens to be the target, it should become the first retrieved result. The image comparison module is designed for efficiency algorithm analysis when the algorithm need to be tuned. In this section, we provide a detailed description of the structure and the functionality of the platform. We first setup up a database.

The database is part of a standard image dataset contains 10 categories. Figure 5.1 shows one typical image in each categories.

Figure 5.1. Images in retrieve dataset

5.2.1 Image retrieval module

For image retrieval application, first we need a database contains varieties of images with its unique ID. For each image, a property vector is computed in advance, which includes a basic color-histogram vector, an edge-direction signature vector and multi-scale heat content feature. Namely for any I, the mechanism generates 19 vectors c, e, h and h_i ($i = 1$ to 16).

Once there is an incoming new image waiting for retrieval, the program automatically calculates all the vectors c, e and h, h_i for the image. We compare c, e and h separately first to reduce the search area and then compute weighted distances of the incoming image with image options. If the distance is below a given threshold, the program should output all the images satisfy the criteria. We use the following equation to estimate the distance between two images

$$d = \alpha\sqrt{\sum ||h_1^i - h_2^i||} + \beta||h_1 - h_2|| + \delta||c_1 - c_2|| + (1-\delta)||e_1 - e_2||. \qquad (5.1)$$

Figure 5.2 shows the interface of the module. The user can click the "Choose An Image" button to browse and upload arbitrary image, and it will be automatically shown in the button with fixed resized image. Below the button is an option panel

designed to let user choose the retrieval speed. There are three choices: fast, normal and precise retrieval. The main difference is due to the total number of the random walker which determines the variance of the heat content estimation. In the retrieval module, the platform provides two different approaches which are the proposed algorithm as well as a color-preferred google like one. Another useful function is that the user can add any image into the database by using "add to database" function. Once the button is triggered, the software will first check the input image to see if it's a duplicate version of exist images in database and if not, the software will compute all the features into the database and store everything including the original image into file. The right part of the interface shows the most similar retrieved images.

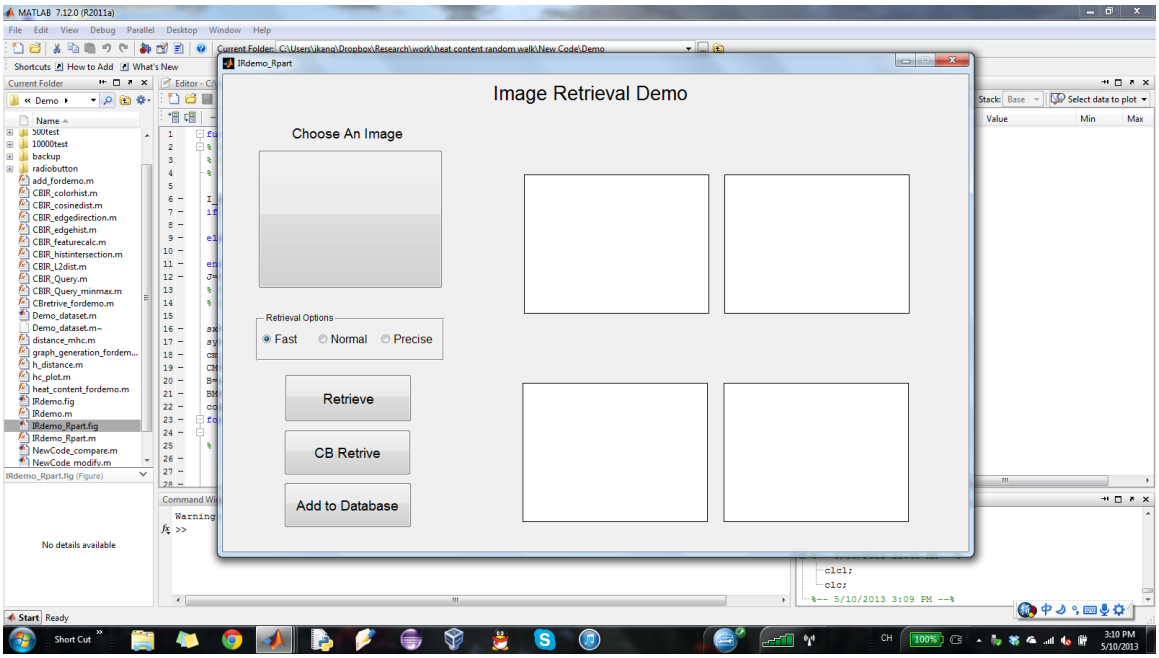


Figure 5.2. Images retrieve module

5.2.2 Image comparison module

Figure 5.3 shows the interface of the image comparison module. In this module, the user uploads 2 images for comparison. The option panel includes the total time of the image heat content and the similar precision options as in image retrieval module.

On the right the interface will automatically plot the total image heat content of the two inputs using red and blue curves. Each componentwise distances in the distance measure are also shown in the interface. By this function one can easily access the raw data and to adjust the weights on the measure.

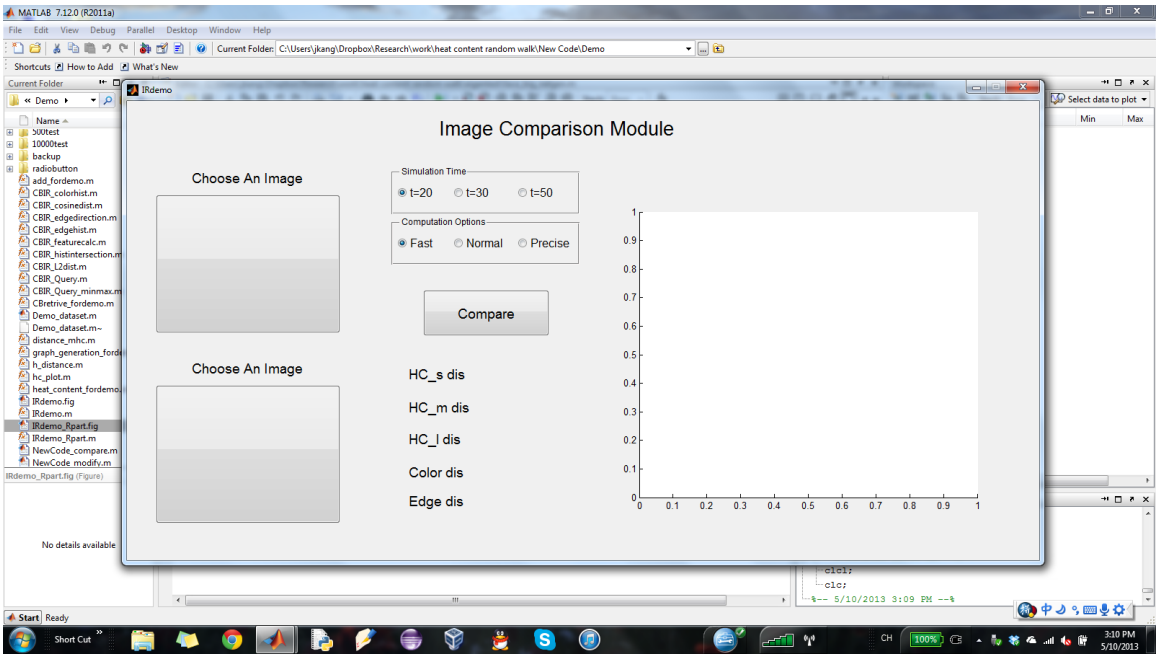


Figure 5.3. Images comparison module

5.2.3 Test Examples

In this section we show some examples of the working software. The examples include ordinary retrieval examples, adding new image example, comparison with google image search examples and image comparison examples. First we show two ordinary image retrieval examples. We observe that the retrieved original image is in the first place and the rest are some visually similar images. In normal cases, the image retrieval function works reasonably well.

We use some examples to show the the proposed image retrieval algorithm has some advantages on certain cases where it performs even better than Google image search engines. The first example is a photo taken of a Hawaiian mountain, the

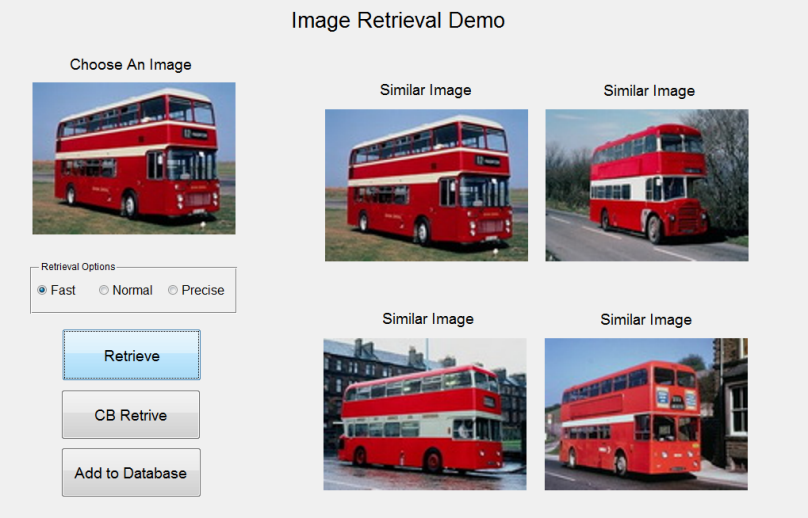


Figure 5.4. Retrieval example 1

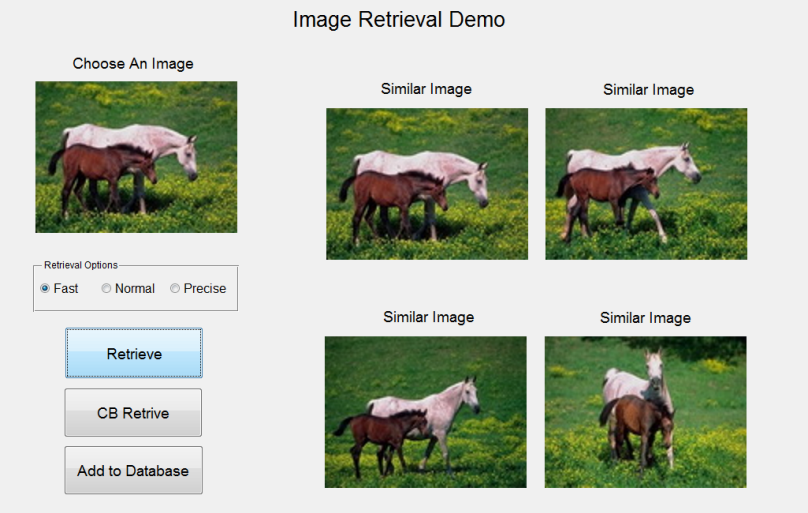


Figure 5.5. Retrieval example 2

original photo is in color but we remove the color and upload it to Google image search. Surprisingly, Google image search fails with this example because it gives color information too much weight. Figure 5.6 is the real google image search result and Figure 5.7 is our simulated color histogram based retrieval result. We can see that the color-based difference is not very large regarding to our original image and the first image that Google has retrieved, which might be the reason for this failure.

After we add some online images into our dataset (which are also in the Google image dataset for sure), figure 5.8 shows our retrieval result. This example may be trivial, but still, the combination of color and structure information seems to be very hard even for Google.

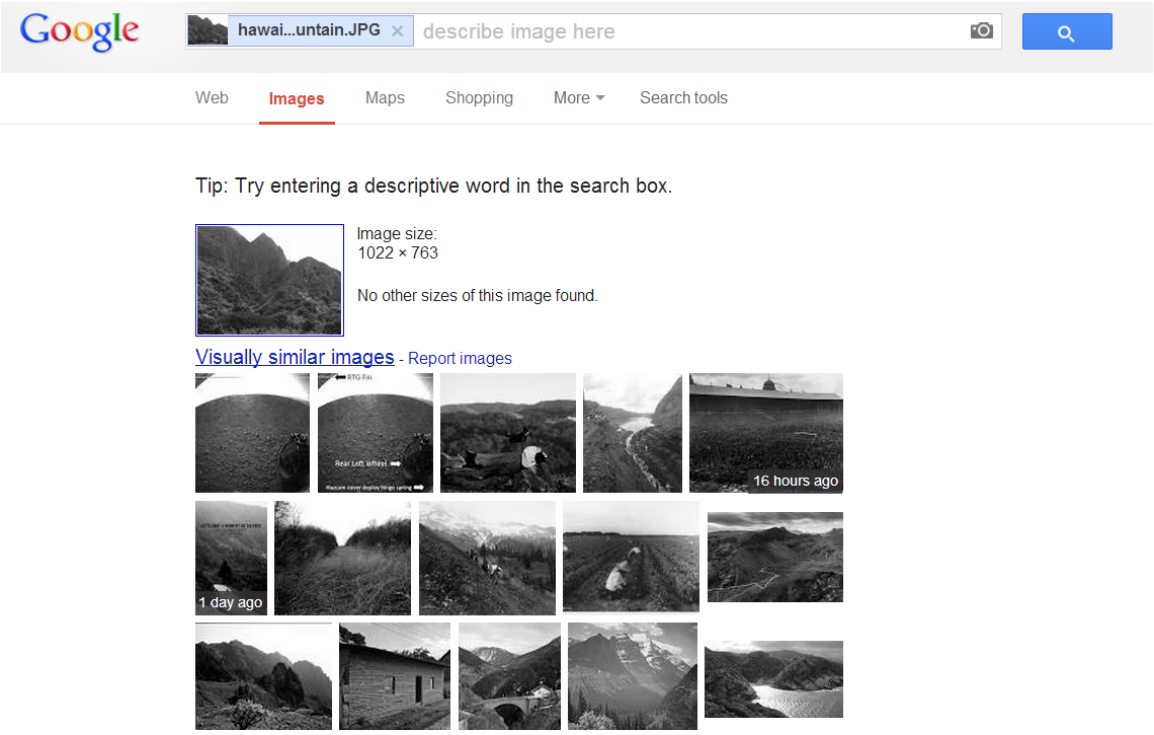


Figure 5.6. Real Google image search result

5.3 Real-time graph-matching based image marker detection Android application

5.3.1 Background introduction

The main goal for this project is to design an Android mobile application to help visually impaired to find their way in indoor environment. Since normally a RFID based system is used in this situation [31], our sub-system's goal is to locate these RFID tags in the building by searching for a visual marker placed near the RFID tag using smart phones equipped with a camera. The project delivered an application

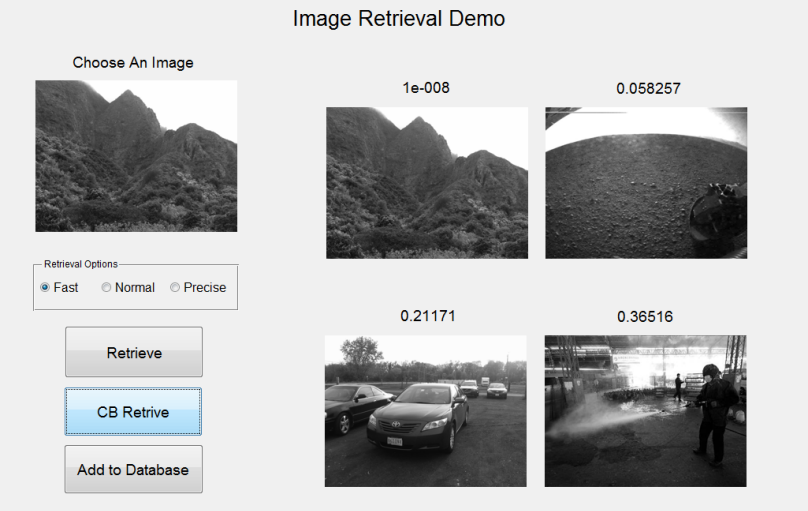


Figure 5.7. Simulated Google image search result

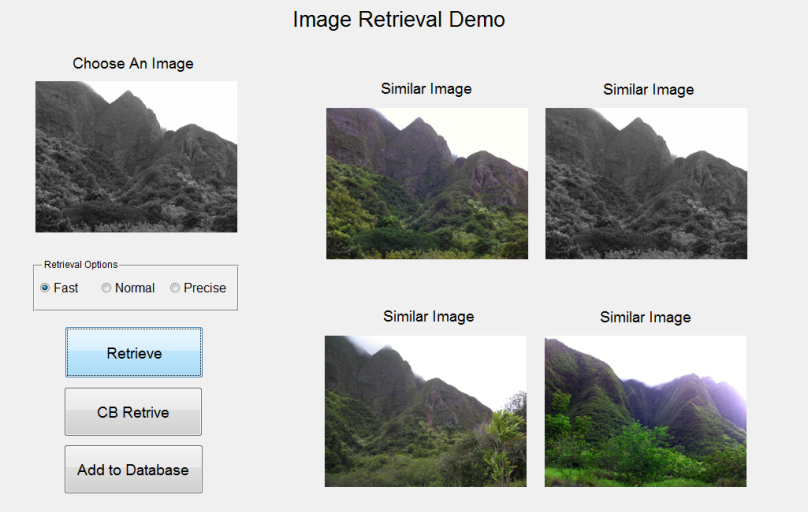


Figure 5.8. Retrieval based on our algorithm

for Android cell phones which implement the marker recognition algorithm with user interface to help visually impaired users locate RFID tags.

Generally, the whole project include the following tasks: first is to design a easily-to-make visual marker with proper size which can be put close to the RFID tag, and the second part is to design a corresponding marker detection algorithm to effectively output the result that if the marker exists on a particular image taken by the user, and

the third part of the project is to build an application which can apply the marker detection algorithm as well as help the user to locate the marker. The difficulties for the project include two major parts. One is to design a proper visual marker and the corresponding detection algorithm with very low false-detection rate, real-time performance and long detection range while the marker has to be as small as possible. Another difficulty is to design a visually impaired user friendly application that will not make the user frustrating while using.

5.3.2 Software design

The project is divided into three parts. In the first part, we introduce the design of a multi-color visual marker. In the second part, we briefly introduce the idea of the grid-graph based marker detection algorithm. Finally, we introduce our designs of the user-friendly application.

The first part of our project is to design an effective multi-color visual marker for the indoor environments in Knowles Engineering Building (KEB). Visual marker has a long history in computer vision society, and has a great number of applications [20,28,29]. One popular example is the bar-code which was invented decades ago [20]. However, A bar-code like marker is not suitable for our application. Normally there are dozens or hundreds (2-D code) of distinct areas which will make the detection miserable in long range. However, if we do not need to encode too much information in, this black-white type of markers are useful in the project. In [28], the authors proposed markers like ARToolkit tags or ARTags.

For detection of black-white markers, effective line detection or edge detection become very important. Basically, the first part is to scan the whole image and to detect edges. Based on the edge information, the marker can be detected with summarizing the edge information. Despite black and white markers, multi-color visual marker has an obvious advantage: there are three numbers for one pixel comparing to

a pure intensity-based approach for black and white marker. In this project, we use a pizza-like multi-color marker similar to the one in [34] but contains one additional color slice as in figure 5.9 shown.

We investigate different types of visual markers and analyze their variations of poses, illuminations, distances and etc. The key step is to collect samples under as many as possible different situations. We can visualize the color distribution of 3 different marker variations in Fig. 5.9.

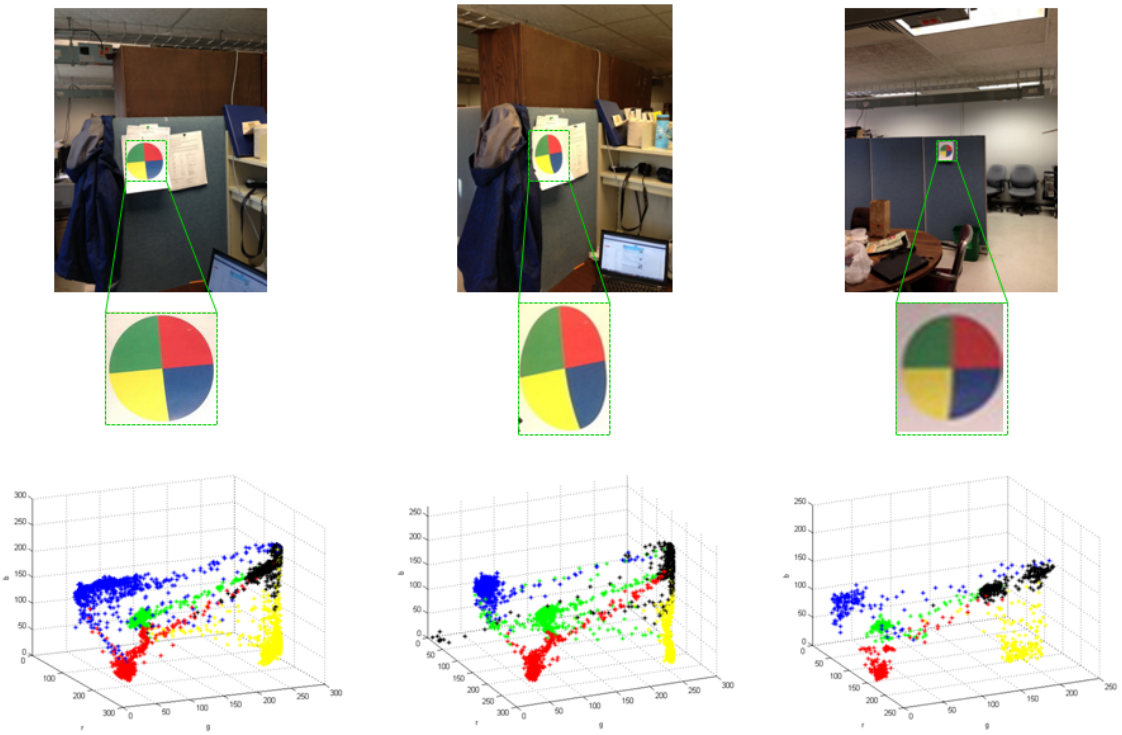


Figure 5.9. Color distribution of marker variations in real situations

These images which contains color tags are taken from real-world situations. From the figure we can see that not only geometrical position will cause the visual marker to transform and make the color distort, the illumination condition will also leads to different output of color and geometrical structure of the marker on the screen. The three dimensional plots at the bottom of the figure illustrate the color of all the pixels on the plate by their RGB point in a 3D Euclidean space. Despite some transition

pixels, most of the pixels in area with a certain color clustered pretty well. Although the exact RGB values for certain area such as "red" area in 3 different situations are not in the same range (for example, the right one's red pixels has a r value for around 150, but the middle one and the left one have r values for around 200). And meanwhile, the geometrical distances of pixels with different colors, and even the projection of the such distances on 3 dimensions follow some obvious patterns. We can see that the difference of RGB values between pixels in distinct color areas still lie in certain intervals. Thus we can utilize such patterns to rule out irrelevant blocks on the image and to detect our marker target. The idea which we will utilize in the following detection algorithm is that despite the exact RGB values of the pixels change a lot in different conditions, even vary a lot inside the same tag area which should be very close theoretically, the distances between the RGB values of pixels in different color areas should maintain certain underlying restrictions for our tag comparing to the background.

We briefly introduce our grid-graph based marker detection algorithm. The first step for the algorithm is the sampling, which is in order to compress the original input frame into a much smaller grid representation of the image in order to make the algorithm running faster. In Fig. 5.10, we illustrate the sampling process in our detection algorithm. We do not use any corner detection nor edge detection here, the original big input frame is sampled into a much smaller representation and form a 4-neighboring grid network. Let the distance of the original pixels between the two neighboring nodes is $2r$, the original frame is reduced by approximately $1/4r^2$ times in scale. Suppose the RGB values are recorded as this formation: (r_i, g_i, b_i) for i probe, we define the edge weight to be the maximum absolute difference of R, G, B componentwise as

$$w_{ij} = \max\{|r_i - r_j|, |g_i - g_j|, |b_i - b_j|\}. \tag{5.2}$$

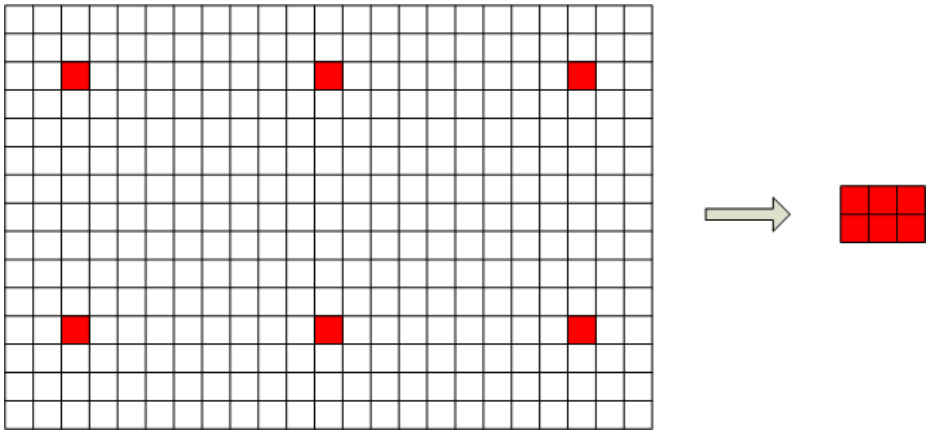


Figure 5.10. Sampling process in the detection algorithm

The next step is a sub-graph matching algorithm to detect the tag base on our compressed grid network. Cascading graph matching algorithm have one huge advantage is that they can be implemented with very fast running time because it can break up the iteration anytime if one comparison fails. In Fig. 5.11 we illustrate a cascade graph matching algorithm model. In the algorithm, we scanning the grid network in 2 by 2 sub-graphs. In each sub-graph, the algorithm compares the edge weight of vertex 1 and 2 first, and if the result pass our criteria, the algorithm then compares the edge weight of vertex 2 and 3, so on so forth. If all four comparisons are passed, there is a successful graph matching/marker detection of our visual tag.

In the final part, we discuss our considerations about user experience. Suppose we have detected a tag in the scanning process, what should the application do to guide the user to destination? Instead of informing the user about the precise location of the tag based on the calculation in one frame, we make our user toward the target all the time by suggesting the user to slide the cell phone toward a certain direction and with continuous vibrations to let the user know there is a target in the front. The trick is that when the tag is detected in central area of the screen, the guidance module of the application will notify the user to go straight and also inform the user how far the tag is approximately. And when the tag is not in central area, the application

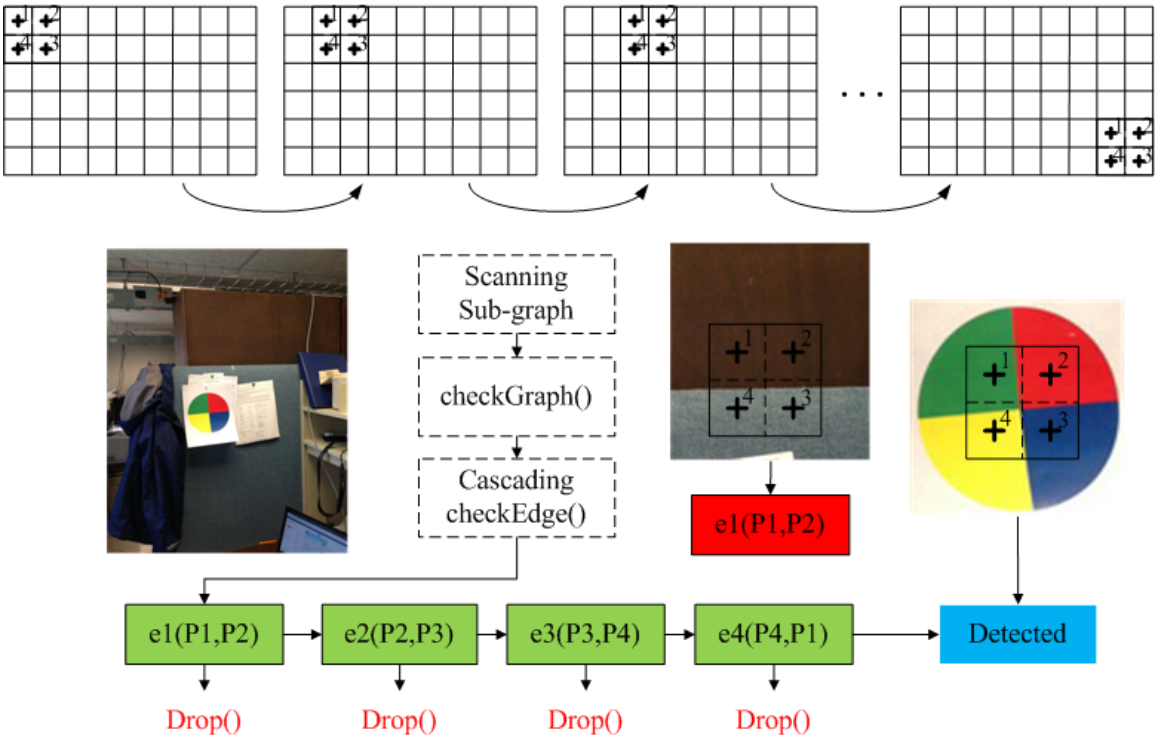


Figure 5.11. Sub-graph matching algorithm for marker detection

will guide the user to slide the cell phone toward the proper direction to make the tag into central area. The implementation details of the area partitions and sliding command suggestions are shown in next sub-section.

5.3.3 Implementation Details

Fig. 5.12 shows the overall software structure of our application. The application does not use any third party source code or development kit, but just based on pure Android SDK and JAVA. The test application are build into the first generation of Google's Samsung Galaxy cell phone. The preview resolution is set to be 1920×1080 which is the highest supported preview resolution.

The marker detection module is build in the onPreviewFrame() function. The the default radius of probe buffer r is 4, the default threshold is $t = 55$, and the default RGB reference values are listed in the source code. We have discovered that the

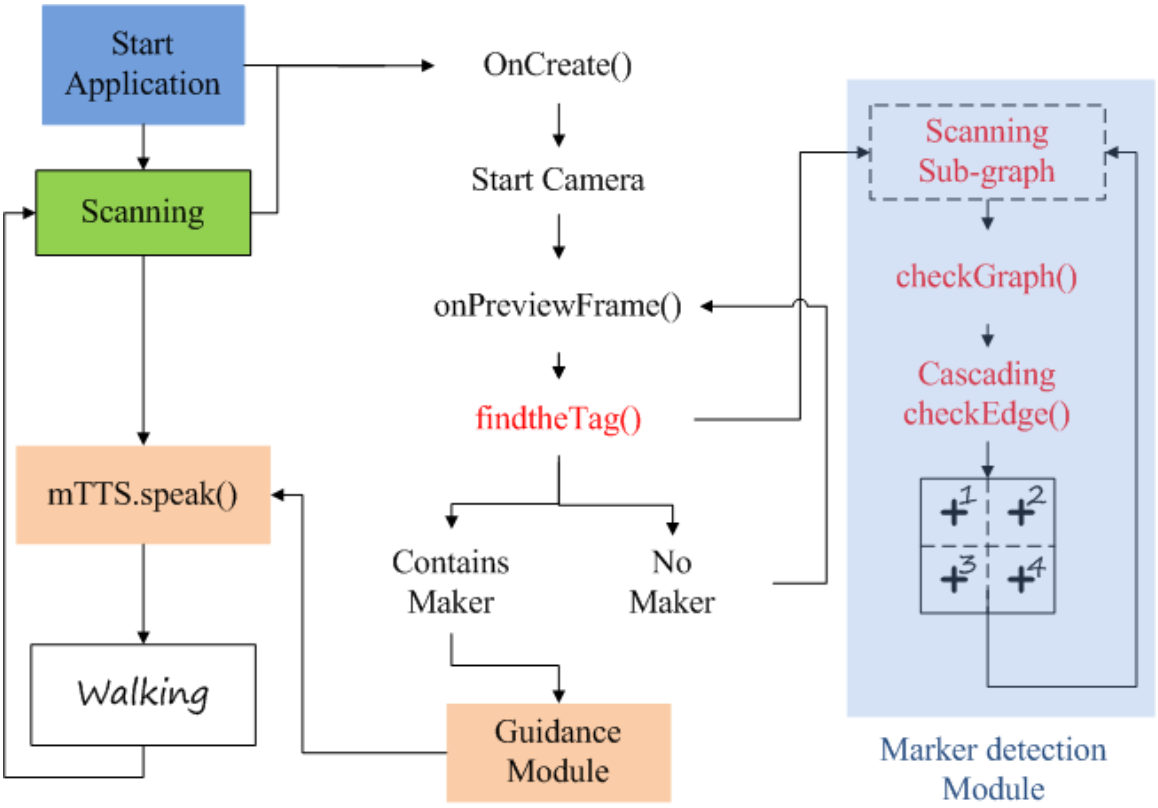


Figure 5.12. Overall structure of the application

chosen of r and t will affect multiple aspects of the application's performance but not in a trivial manner. In general, a larger r will lead to a shorted detection range but a more sensitive scanning feedback, but if we set r to be less than 4, we may encounter backfire. And a bigger t will generally increase the robustness of the application in more variations of situation such as lighting condition and pose variations but a big buffer may make the detection suffer a higher false alarm probability.

In the Marker detection module, once a tag has been detected, the application will draw a circle and cross according to the tag location in a canvas on a TagView which extends SurfaceView class. The guidance module includes two parts. If the tag is not in the central area (we define it to be the middle 1/2 height and width area), the application will notify the user to slide the cell phone. Fig. 5.13 shows the area partitions with their corresponding suggestions to user.

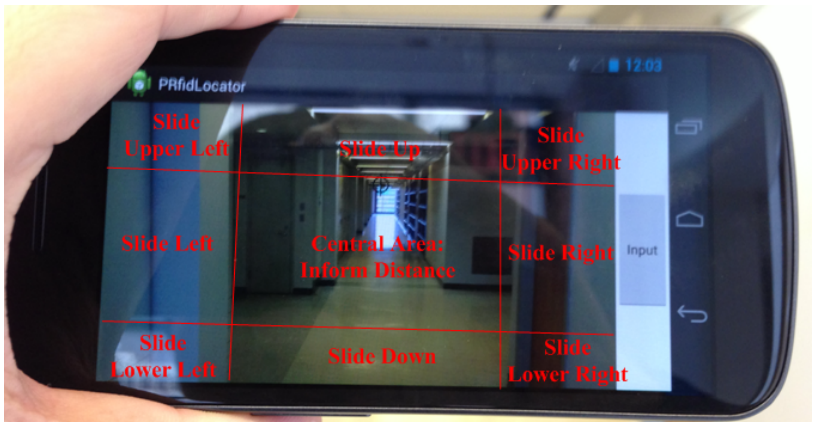


Figure 5.13. Notification of Sliding the cell phone

If the tag is in the central area, the guidance module will calculate the distance between the cell phone and the tag, then notify the user a rough number in integer meters. We first estimate the number of the tag's height h in one frame and acquire the proportion coefficient $h/1080$ (The height of one frame can be changed). Then we can use the coefficient $h/1080$ to estimate the distance by being divided by a constant C. C can be calculated theoretically and also can be acquired by try and error. Here we set C to be $340/1080$ and the result of the estimated distance is $340/h$ meters. This simple distance calculation algorithm requires some assumptions to be true. First, the center-located tag give us a great opportunity to estimate the distance directly by the number of pixels of the tag with enough preciseness. Second, because the tag should be put on hand-reachable level, we can make a reasonable assumption that the cell phone does not have a large vertical distance from the tag, and this assumption let us possible to use the height in estimating the distance which should be only related with distance.

In the application, text-to-speech (TTS) is implemented to generate speech suggestions for the user, along with vibration notification, provide a clear, helpful, but simple suggestions to the user. One additional functionality we want to add to the application is to be able to handle new multi-color tags with different patterns. (The

partition should be the same) Fig. 5.14 shows the user interface of the maker pattern update module. It is activated by press the "input" button in main interface. The only thing the user need to do is to put the tag inside the circle on screen and align the four color areas to the four sections inside the circle. By press the "take this" button on the upper right corner, the RGB values of pixels on the preview frame corresponding to the overlapping four black crosses have been consequently updated into the cell phone's sharedPreference file. The user can then use "go back" button to return to the main activity. The "reset" button provide a fast restoring functionality if the user want to reset the application to the default set up.

Figure 5.14. Pattern Update User Interface

5.3.4 Test Examples

The first set of test examples are shown in Fig.5.15 and 5.16. The purpose of this test is to estimate the maximum detection range. As we can see, the performance of the detection algorithm in our application is very good in normal conditions of KEB second floor hallway, especially when the tag is put in the middle of the hallway as

in Fig.5.15. The detection range can be up to 20 meters (actually the maximum is 23 but the performance is not stable if the distance is more than 20). In Fig.5.16, the tag is put at the end of the hallway, and the background contrast is larger in this situation, we have a shorter but still acceptable detection range to be approximately 15 meters.

Figure 5.15. Maximum Detection Range Example 1 ($\approx$ 20 meters)

Figure 5.16. Maximum Detection Range Example 2 ($\approx$ 15 meters)

Our second test example is to show the excellent robustness of tag detection under very large angles. As shown in Fig.5.17 and Fig.5.18, the tag can be detected even it is almost perpendicular to the cell phone in a certain range. The maximum detection

angle could be up to 80 degrees in this distance and the algorithm works well in all directions. The reason for this high robustness is that our proposed sampling and cascading algorithm only need dozens of pixels horizontally as well as vertically to detect the tag, thus the angle of the tag can be as large as possible and this is obviously to be very practical in environment such as KEB where three long hallways exist. The robustness under large-angle situation as well as long-distance detection ability is the major advantage of our application.

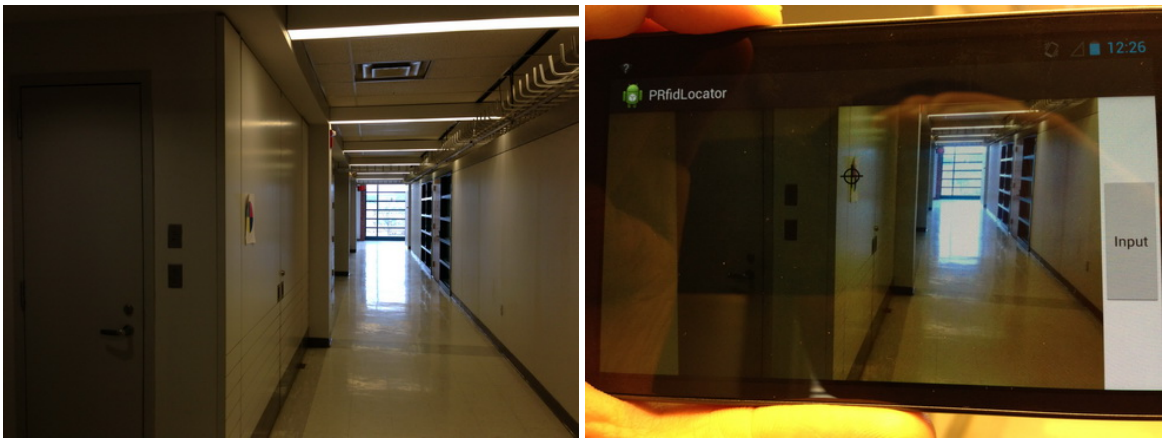


Figure 5.17. Maximum Detection Angle Example 1

Figure 5.18. Maximum Detection Angle Example 2

In this subsection, we present two tests on normal and dim lighting conditions. The application works well in both situations. Fig.5.19 is under normal lighting condition and Fig.5.20 is the same example in a dimmer situation. The dim lighting condition will not affect the performance of our application to detect the tag correctly because our the dim lighting will cause the RGB values to decrease while the differences of such values change less. However, if we turn off all the lights, nobody can detect any tag apparently.

“‘rt

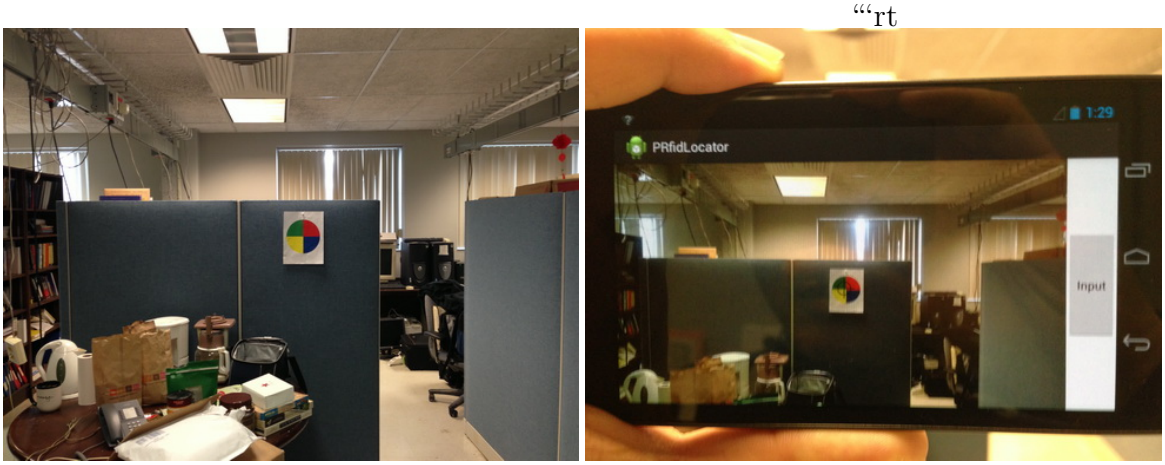


Figure 5.19. Normal lighting condition Example

Figure 5.20. Dim lighting condition Example

Another very important problem which we pay a lot of attention to is the false alarm. The user will suffer a lot if there is a false alarm. Although in previous real-world tests taken in KEB hallway, we do not encounter any false alarm situation, we still want to test our application under some sophisticated background. In Fig.5.21, the tag is put in front of a lot of books which contains varieties of colors and in Fig.5.22, we create another multi-color tag to be a distraction. In both cases the algorithm successfully detects the right tag and disregard any background area with no false alarm at all.

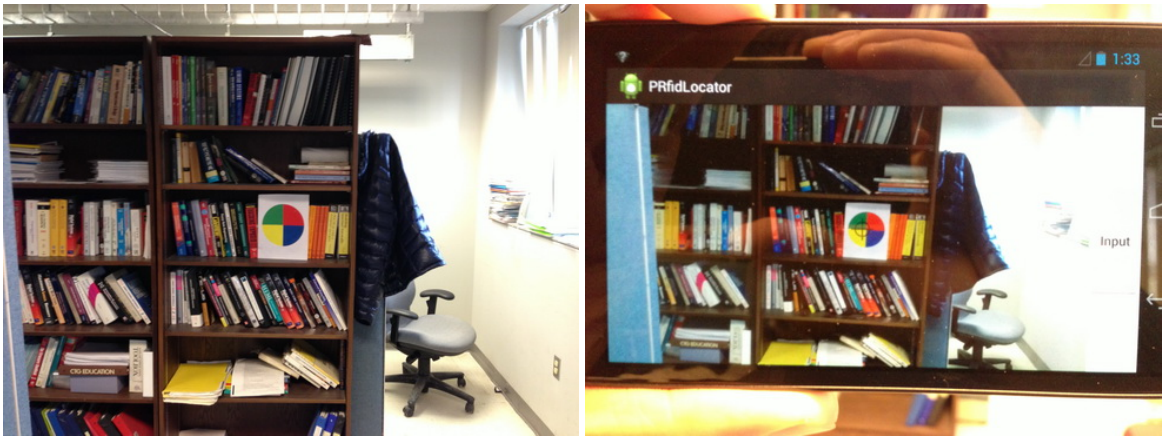


Figure 5.21. Tag Detection Under Sophisticated Background

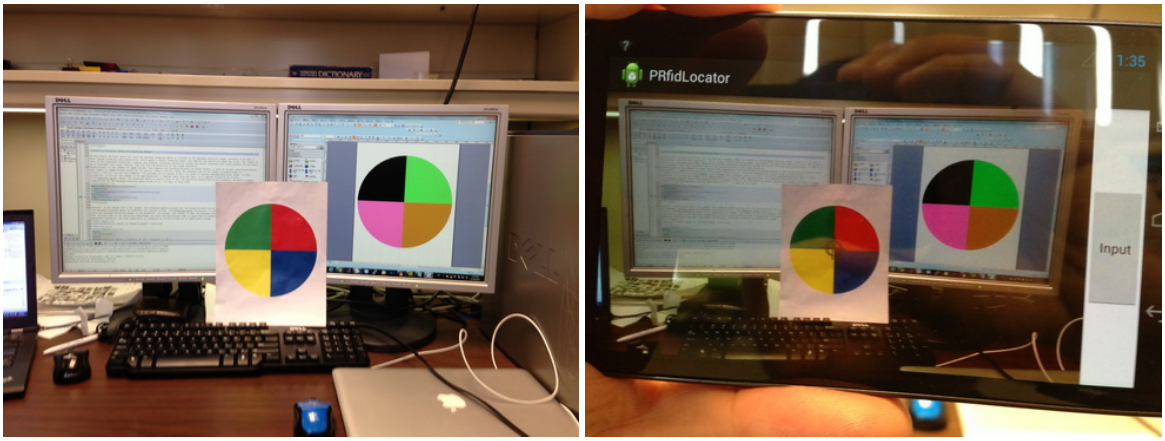


Figure 5.22. Distraction Background with Different Multi-Color Tag

Our application can handle new color patterns of our visual tag. For instance, in Fig. 5.23, we show the process of how to input the new color pattern. By putting the new tag inside the circle on screen, and let the four black crosses lay on the four color areas, the new color pattern is automatically being processed. We can then press the "take this" button to store the pattern into the cell phone. The right figure shows that now our application starts to detect the new tag instead of the old one.

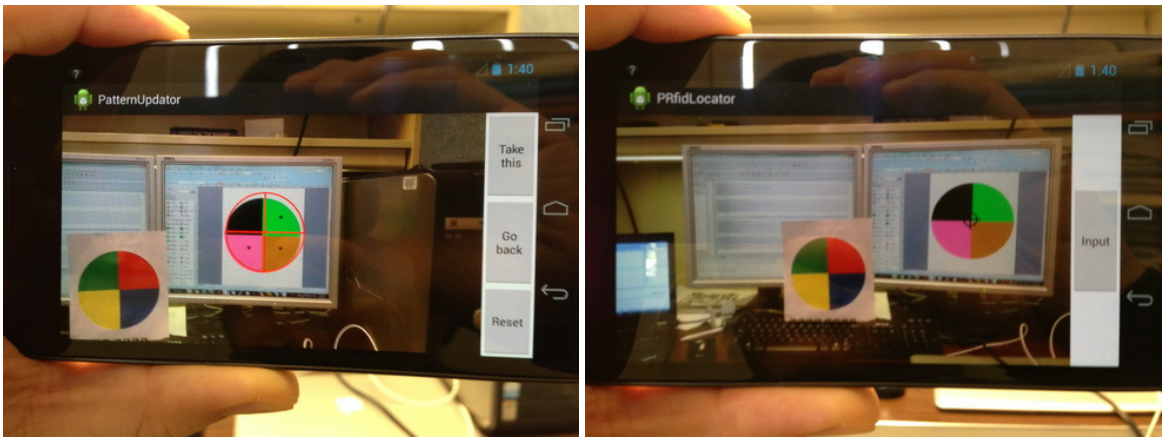


Figure 5.23. Input and Detect New Color Pattern

The maximum scanning speed is related to the maximum detection range. Actually if we want a more robust scanning user experience, which means a faster scanning speed, the smaller the detection range has to be. The reason is that in our proposed algorithm, suppose the average cascading operations inside one sub-network (contains four nodes) is k which can be treated as a constant, the total running time for the tag detection of one frame should be proportional to the number of image blocks. And suppose the distance between two probe pixels is $2r$ as we have introduced before, the number of image blocks should be $N \propto \frac{1}{4r^2}$. Thus the running time of one frame is proportional to $\frac{1}{4r^2}$. In our test samples, the minimum r is chosen to be 4 in order to achieve maximum detection range. In this situation, if we set the target to be approximately 1.5 meters away from the user, in a typical situation as in Fig.5.24 shows, the scanning speed is pretty slow to let the detection module functional nor-

mally. We give a roughly computation of the maximum scanning speed which is 10 - 20 cm/s in this case.

Figure 5.24. Example for Scanning Time

However, if we change r to be larger, the scanning speed increase but not proportionally as we assume. Table 5.3.4 shows the maximum scanning speed of different r values in Fig.5.24 situation (1.5 meters in typical Knoweles hallway), with the maximum detection range of the different r values. The reason is that the maximum scanning speed does not only depend on the running time of our detection algorithm, but also the processing time of acquiring the frames taken by the camera. But generally speaking, the user experience of scanning in $r = 16$ situation is comfortable enough.

	r=4	r=16
Maximum Detection Range	$\approx 20\ m$	$\approx 10\ m$
Maximum Scanning Speed	$10 - 20\ cm/s$	$30 - 40\ cm/s$

Table 5.1. Detection Range vs Scanning Speed

CHAPTER 6

DISCUSSION

In this chapter, we discuss the properties of our proposed graph-based image understanding model. First, we present an example to illustrate the eigenvectors in the Laplacian spectrum of the graph representation of image. The illustration shows that the eigenvectors do not only carry important information of the image but also follow an interesting pattern. Second, we investigate the relationship between our proposed features with traditional image features by using correlation analysis. The experiment shows that our model captures unique information not covered by traditional image features. We then discuss limitations and future directions of our model. We end this chapter by briefly summarizing the contributions of this dissertation.

6.1 The illustration of eigenvectors in the Laplacian spectrum

As we have discussed in previous chapters, the eigenvectors of the graph Laplacian contain important image structural information. We design the following example to intuitively show the properties of these eigenvectors. For convenience, we use the original pixel-based graph representation in this experiment. Every pixel p_i is represented by one vertex $v_i \in V$ in the graph $G = (V, E)$ with the adjacency matrix defined as

$$A(i,j) = \frac{\epsilon + I_i - I_j}{d(p_i, p_j)}, \tag{6.1}$$

in which I_i, I_j are intensities of pixel p_i and p_j. The normalized graph Laplacian is $L_n = D^{\frac{1}{2}}(D-A)D^{\frac{1}{2}}$. $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ are the eigenvalues of L_n and $\varphi_i, i = 1, \cdots, n$

are the corresponding eigenvectors. We then check every entry of the eigenvectors in detail. For eigenvector φ_i, we map every entry in the vector to their original corresponding position in the image and form a two-dimensional matrix. Every entry of this matrix is normalized into [0 255]. Each eigenvector then becomes a grayscale image. Figure 6.1 shows the eigenvector illustrations of 3 different images. The first row shows eigenvectors from φ_1 to φ_{10}; the second row shows eigenvectors from φ_{101} to φ_{800} (sampled consequently with same length 100); and the third row shows eigenvectors from φ_{891} to φ_{900} $(n = 900)$.

We notice that these eigenvectors form a decomposition of the original image. One interesting observation is that last several eigenvectors represent the shape again like the first few eigenvectors. But in the middle of the spectrum, the eigenvectors are periodical meaningless textures with different frequency behaviors (from 101 to 801). Another phenomenon is that the illustration of the high-frequency eigenvectors contain a lot of edge information of the image, which is more important to represent images than low-frequency information. The improvement from the original heat content to the heat content spectrum feature can be explained by such a phenomenon since the latter one focuses more on the high-frequency part of the Laplacian spectrum. Although the summary process in our model loses some information of the eigenvector, the spectrum still describes the general structure of the image. Meanwhile, the pattern showed by the illustration of the eigenvectors poses an interesting question alone: why do the low-frequency and high-frequency eigenvectors carry more information of the image structure while the middle ones contain less? We still don't know the answer yet.

6.2 Correlation analysis of image features

In order to intuitively show the relationship between our model and other traditional image features, we propose the following correlation analysis experiment. We

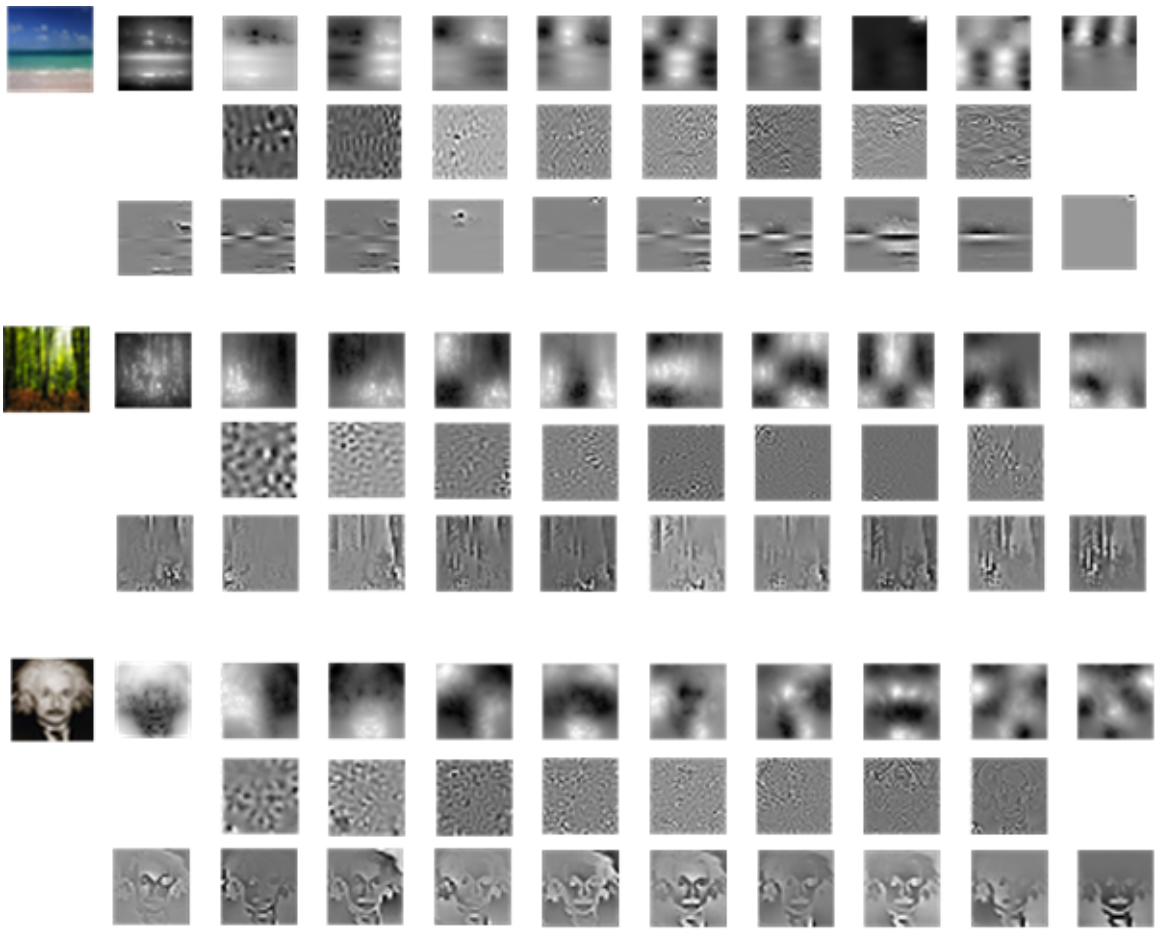


Figure 6.1. Image illustrations of the eigenvectors

use the data generated by our MNIST handwritten digits classification experiment. The raw data contains all the feature vectors of every image patch in the training set. For each image patch, there are total eight image feature samples. We then use these samples to compute the correlation matrix of these eight different image features.

Figure 6.2 shows the correlation matrix between all the eight image features tested in the MNIST experiment. We see that the heat content feature is less correlated with every other feature. While small correlation is evidence for the argument that the heat content feature contains new information, it is also a sign that the heat content feature may lose some important information represented by traditional successful features. The relatively low accuracy rate for the heat content feature alone in the k-nearest-

neighbor classification experiment may also be explained by such low correlation. In contrast, our improved features are moderately correlated to other features but still maintain their own uniqueness, which could probably explain the better performances in the digits classification experiment.

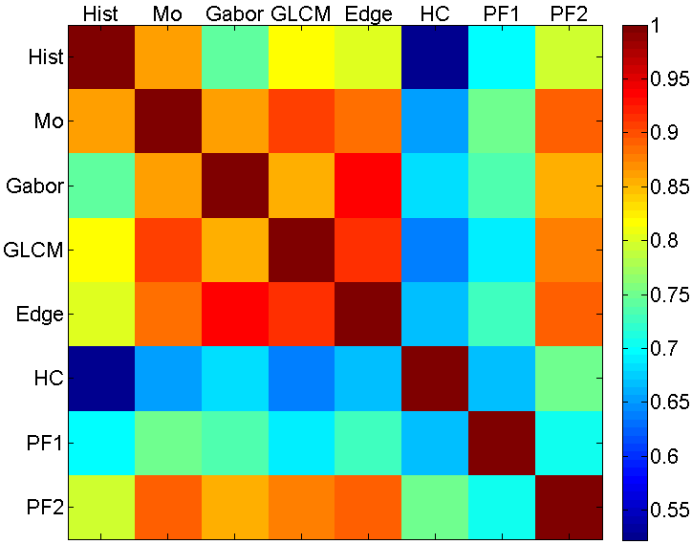


Figure 6.2. Correlation between different image features. (PF1: Oscillatory heat content, PF2: Weighted heat content spectrum)

Correlation analysis gives us an intuitive idea of what information our new model captures. However, we still have difficulties to fully understand the mechanism and the relationship between our model and visual similarity. On the other hand, we believe that our model can still be used as a new weapon and contributes to the image understanding arsenal.

6.3 Limitations of our model

A recent development of image understanding is the deep-learning based model. Very large scale datasets like ImageNet are used to test the performance of image classification and understanding in recent researches. And nearly all the models with the best performances are based on deep neural networks.

The major advantage of deep-learning based approaches is the depth of the neural network. These models usually have millions of trainable coefficients. Like our model, deep neural networks can also be seen as a special type of graph-based method. With the new development in efficient training algorithms and engineering breakthroughs such as GPU-based computing, the flexibility and the representing power of these models is remarkable because of the large number of coefficients. On the contrary, the only trainable part in our model is still based on the traditional approach, which inevitably limits our model's potential in complicated image understanding tasks.

One possible improvement for our model is to add a training process into the algorithm. The training process might be added to the graph generation component in our model, since the rest procedures are mostly fixed. However, we do not know how to form the training algorithm yet and one-layer shadow model might still not be enough for high-level classification and understanding tasks. On the other hand, it is always possible to attach more layers on top of the generated features, but normally it is of little use because the complexity has already vanished in the feature extraction process. In general, the major weakness of our model is the lack of flexibility in applying a meaningful training process.

6.4 Contribution of this dissertation

We end the discussion part by briefly summarizing the contributions of this dissertation.

First, we proposed a fast and effective graph similarity testing algorithm based on the concept of heat content. We also design estimation algorithms for large dense graphs. Compared to previous work, our algorithm has the following advantages in graph similarity testing. First, the heat content method maps the graph to one-dimensional data to compare. Second, the difference of two graphs can be presented at the very beginning part of the heat content curves. The first two advantages help

reduce the comparison time significantly. Meanwhile, our algorithm does not need a given nodes correspondence, which means it can be applied into more practical problems in real world. Our method is also robust to minor changes in the graph.

Based on the graph similarity testing algorithm, we designed a graph-based image processing and understanding framework. We proposed a graph-based image representation model and invent several useful tools for image analysis tasks, such as the image heat content feature, the weighted heat content spectrum and the oscillatory heat content. We also analyzed and discussed the properties of our model by designing several experiments focusing on different aspects of image variations. Our proposed image features are shown to be robust and easily computed. The model has the potential to become an effective and efficient multi-scale feature for image retrieval. The model can also be combined with other traditional image features to create a complex visual signature. Although we still need further experiments and analysis to thoroughly understand the advantages and drawbacks of our model, the proposed image features demonstrate an ability to serve as a useful supplement to traditional image retrieval and classification. There are still open questions on the theoretical side, in particular on details of the relationship between the heat content spectrum and graph structure. However, the experiment results show that the proposed methods effectively capture some useful and unique image information.

Finally, we presented the design of two real-life desktop and mobile applications which are related to specific image processing and understanding tasks. We demonstrate that graph-based image understanding models are efficient enough for real-time needs.

BIBLIOGRAPHY

[1] Achanta, Radhakrishna, Shaji, Appu, and Smith, Kevin. The heat kernel as the pagerank of a graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence(Volume: 34, Issue: 11, Nov. 2012) 33(11* (2012), 2274 – 2282.

[2] Association, International Game Fish. Igfa fish database. Available at `http://www.igfa.org/fish/fish-database.aspx`.

[3] Backes, A.R., Casanova, D., and Bruno, O.M. A complex network-based approach for boundary shape analysis. *Pattern Recognition 42* (2009), 54–67.

[4] Baeza-Yates, R., and Valiente, G. An image similarity measure based on graph matching. *Proceedings of the Seventh International Symposium on String Processing Information Retrieval (SPIRE'00)* (2000), 27–29.

[5] Bayati, M., Gleich, David F., Saberi, Amin, and YingWang. Message passing algorithms for sparse network alignment.

[6] Berg, M. Van Den, and Gilkey, Peter B. Heat content asymptotics of a riemannian manifold with boundary. *Journal of Function Analysis 120* (1994), 48–71.

[7] Bi, G., and Poo, M. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of Neuroscience 18(24)* (1998), 10464–10472.

[8] Brang, D., and Ramachandran, V.S. Survival of the synesthesia gene: Why do people hear colors and taste words. *PLoS Biology 9(11)* (2011), e1001205.

[9] Bullmore, Ed, and Sporns, Olaf. Complex brain network: graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience 10* (2009), 186–198.

[10] Butler, S. Interlacing for weighted graphs using the normalized laplacian. *Electronic Journal of Linear Algebra 16* (2007), 90–98.

[11] Buzsaki, Gyorgy, and Draguhn, Andreas. Neuronal oscillations in cortical networks. *Science 304(5679)* (2004), 1926–1929.

[12] Chang, Chih-Chung, and Lin, Chih-Jen. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology 2* (2011), 27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[13] Chung, F. The heat kernel as the pagerank of a graph. *Proceedings of the National Academy of Sciences 104(50* (2007), 19735–19740.

[14] Conte, D., Foggia, P., Sansone, C., and Vento, M. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI) 18(3)* (2004), 265–298.

[15] Datta, R., Joshi, D., Li, J., and Wang, J. Z. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys 40(2)* (2008), 5:1–5:60.

[16] Datta, Ritendra, Li, Jia, and Wang, James Z. Content-based image retrieval: Approaches and trends of the new age. In *Proceedings of the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval* (2005), pp. 253–262.

[17] Deselaers, T., Keysers, D., and Ney, H. Features for image retrieval: An experimental comparison. *Information Retrieval 11* (2008), 77–107.

[18] Felzenszwalb, PedroF., and Huttenlocher, DanielP. Efficient graph-based image segmentation. *International Journal of Computer Vision 59*, 2 (2004), 167–181.

[19] Felzenszwalb, P.F. Efficient graph-based image segmentation. *International Journal of Computer Vision 59(2)* (2004), 167–181.

[20] Fiala, M. Artag, a fiducial marker system using digital techniques. *CVPR* (2005), 590–596. IEEE Computer Society.

[21] Gao, X., Xiao, B., Tao, D., and Li, X. A survey of graph edit distance. *Pattern Anal Applic 13* (2010), 113–129.

[22] Gong, W. Can one hear the shape of a concept? In *Proceedings of the 31st Chinese Control Conference (Plenary Lecture)* (Hefei, China, July 2012), pp. 22–26.

[23] Gong, W. Transient response functions for graph structure addressable memory. In *Proceedings of the 52th IEEE Conference on Decision and Control* (Florence, Italy, December 2013).

[24] Grady, L. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28(11)* (2006), 1768–1783.

[25] Harchaoui, Z., and Bach, F. Image classification with segmentation graph kernels. In *IEEE Conference on Computer Vision and Pattern Recognition* (Minneapolis, USA, June 2007), pp. 1–8.

[26] Hart, P., Nilsson, N., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions of Systems, Science, and Cybernetics 4* (1968), 100–107.

[27] Hinton, G. E., Osindero, S., and Teh, Y. A fast learning algorithm for deep belief nets. *Neural Computation 18* (2006), 1527–1554.

[28] Hirzer, Martin. Marker detection for augmented reality applications.

[29] Homayoun Bagherinia, Roberto Manduchi. Robust real-time detection of multi-color markers on a cell phone. *Journal Real-time Image Processing 6* (2011).

[30] Howarth, P., and Ruger, S. Evaluation of texture features for content-based image retrieval. *Image and Video Retrieval, Lecture Notes in Computer Science* (2004), 326–334.

[31] J. Coughlan, R. Manduchi, H. Shen. Cell phone-based wayfinding for the visually impaired. *1st International Workshop on Mobile Vision* (2006).

[32] J. H. H, Grisi-Filho, Ossada, R., Ferreira, F., and Amaku, M. Scale-free networks with the same degree distribution: Different structural properties. *Physics Research International* (2013), 1–9.

[33] Jafri, R., and Arabnia, H. A survey of face recognition techniques. *Journal of Information Processing Systems 5(2)* (2009), 41–68.

[34] James Coughlan, Roberto Manduchi. Color targets: Fiducials to help visually impaired people find their way by camera phone. *J. Image Video Process. 2* (2007).

[35] Jeh, Glen, and Widom, Jennifer. Simrank: a measure of structural-context similarity. *In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining 2* (2002), 538–543.

[36] Jiang, Y.G., Dai, Q., J. Wang, C.W. Ngo, Xue, X.Y., and Chang, S.F. Fast semantic diffusion for large-scale context-based image and video annotation. *IEEE Transactions on Image Processing 21(6)* (2012), 3080–3091.

[37] Kachare, N.B., and V.S., Inamdar. Survey of face recognition techniques. *International Journal of Computer Applications 1* (2010), 29–33.

[38] Kang, Jieqi, Lu, Shan, Gong, Weibo, and Kelly, Patrick A. A complex network based feature extraction for image retrieval. In *IEEE International conference on image processing* (2014).

[39] Kellogg, R. Using graph distance in object recognition. *In Proc. 18th ACM Annual Computer Science Conf* (1990), 43–48.

[40] Koutra, D., Parikh, A., Ramdas, A., and Xiang, J. Algorithms for graph similarity and subgraph matching. Available at `https://www.cs.cmu.edu/~jingx/docs/DBreport.pdf`.

[41] Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[42] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86(11)* (1998), 2278–2324.

[43] LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. Deep learning. *Nature 521* (2015), 436–444.

[44] LeCun, Yann, Kavukcuoglu, Koray, and Farabet, Clment. *Convolutional networks and applications in vision*. 2010, pp. 253–256.

[45] Liu, Ying, Zhang, Dengsheng, Lu, Guojun, and Ma, Wei-Ying. A survey of content-based image retrieval with high-level semantics. *IEEE Transactions on Pattern Analysis and Machine Intelligence 40(1)* (2007), 262–282.

[46] Lu, S., Kang, J., Gong, W., and Towsely, D. Complex network comparison using random walks. In *The Sixth Annual Workshop on Simplifying Complex Networks for Practitioners* (May 2014).

[47] Markram, H., Lbke, J., Frotscher, M., and Sakmann, B. Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps. *Science 275(5297)* (1997), 213–215.

[48] Mcdonald, P., and Meyers, R. Diffusion on graphs, poisson problems and spectral geometry. *Transaction of the American Mathematical Society 354(12)* (2002), 5111–5136.

[49] Mcdonald, P., and Meyers, R. Isospectral polygons, planar graphs and heat content. *Proceedings of the American Mathematical Society 131(11)* (2003), 3589–3599.

[50] Melnik, Sergey, Garcia-Molina, Hector, and Rahm., Erhard. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. *In 18th International Conference on Data Engineering (ICDE 2002)* (2002).

[51] Mller, Henning, Michoux, Nicolas, Bandon, David, and Geissbuhler, Antoine. A review of content-based image retrieval systems in medical applicationsclinical benefits and future directions. *International Journal of Medical Informatics 73(1)* (2004), 1–23.

[52] Nene, S. A., Nayar, S. K., and Murase, H. Columbia object image library (coil-100). *Technical Report CUCS-006-96* (1996).

[53] Papadimitriou, P., Dasdan, A., and Carcia-Molina, H. Web graph similarity for anomaly detection. *Journal of International Service and Applications 1(1)* (2010), 19–30.

[54] Park, In Kyu, Yun, II Dong, and Lee, Sang Uk. Color image retrieval using hybrid graph representation. *Image and Vision Computing 17* (May 1999), 465–474.

[55] Park, J., and Kim, K. The heat energy content of a riemannian manifold. *Trends in Mathematics, Information Center for Mathematical Sciences 5(2)* (2002), 125–129.

[56] Park, Soo Beom, Lee, Jae Won, and Kim, Sang Kyoon. Content-based image classification using a neural network. *Pattern Recognition Letters 25*, 3 (2004), 287 – 300.

[57] Patil, A. M., Kolhe, S.R., and Patil, P. M.. 2d face recognition techniques: A survey. *International Journal of Machine Intelligence 2(1)* (2010), 74–83.

[58] Rajam, I. Felci, and Valli, S. A survey on content based image retrieval. *Life Science Journal 10(2)* (2013), 2475–2487.

[59] Riesen, K., and Bunke, H. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing 27(7)* (2009).

[60] Robles-Kelly, A., and Hancock, E. R. Edit distance from graph spectra. *Ninth IEEE International Conference on Computer Vision (ICCV'03) 1* (2003), 234.

[61] Rubinov, Mikail, and Sporns, Olaf. Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage 52* (2010), 1059–1069.

[62] Sanfeliu, A., and Fu, K. A distance measure between attributed relational graphs for pattern recognition. *IEEE Trans. on Systems, Man and Cybernetics 13(3)* (1983).

[63] Schmidhuber, Jurgen. Deep learning in neural networks: An overview. *Neural Networks 61* (2015), 85 – 117.

[64] Shi, Jianbo, and Malik, Jitendra. Normalized cuts and image segmentation. *IEEE Transaction on Pattern Anylysis and Machine Intelligence 22* (2000), 888–905.

[65] Sirovich, L., and Meytlis, M. Symmetry, probability and recognition in face space. *PNAS- Proceedings of the National Academy of Sciences 106(17)* (2009), 6895–6899.

[66] Smeulders, Arnold W.M., Worring, Marcel, Santini, Simone, Gupta, Amarnath, and Jain, Ramesh. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22(12)* (2002), 1349–1380.

[67] Spielman, Daniel A., and Teng, Shang-Hua. Spectral partitioning works: Planar graphs and finite element meshes. In *37th Annual Symposium on Foundations of Computer Science* (Burlington, VT, 1996), pp. 96–105.

[68] Stoianov, I., and Zorzi, M. Isospectral emergence of a 'visual number sense' in hierarchical generative models. *Nature Neuroscience 15* (2012), 194–196.

[69] Tarrs, F., and Rama, A. Gtav face database. Available at `http://gps-tsc.upc.es/GTAV/ResearchAreas/UPCFaceDatabase/GTAVFaceDatabase.htm`.

[70] Tzortzis, Grigorios, and Likas, Aristidis. The minmax k-means clustering algorithm. *Pattern Recognition 47* (2014), 2505–2516.

[71] Vailaya, A., Figueiredo, M. A.T., Jain, A. K., and Zhang, Hong-Jiang. Image classification for content-based indexing. *Trans. Img. Proc. 10*, 1 (2001), 117–130.

[72] Vicente, Sara, Kolmogorov, Vladimir, and Rother, Carsten. Graph cut based image segmentation with connectivity priors. In *IEEE Conference on CVPR* (2008), pp. 1–8.

[73] Wang, James Z., Li, Jia, and Wiederhold, Gio. Simplicity: Semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23(9)* (2001), 947–963.

[74] Yang, Allen Y., Maji, Subhransu, Christoudias, C. Mario, Darrell, Trevor, Malik, Jitendra, and Sastry, S. Shankar. Multiple-view object recognition in band-limited distributed camera networks. In *Third ACM/IEEE International Conference on Distributed Smart Cameras* (Como, Italy, 2009), pp. 1–8.

[75] Zager, L., and Verghese, G. Graph similarity scoring and matching. *Applied Mathematics Letters 21(1)* (2008), 86–90.

[76] Zhang, C., and Zhang, Z. A survey of recent advances in face detection.

[77] Zhang, L., Wang, L., and Lin, W. Generalized biased discriminant analysis for content-based image retrieval. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics 42(1)* (2012), 282–290.

[78] Zhao, W., R., Chellappa, A., Rosenfeld, and Phillips, P.J. Face recognition: A literature survey. *Computing Surveys* (2003), 399–458.