

**A GRAPH-BASED APPROACH FOR MODELING AND INDEXING
VIDEO DATA**

by

JEONGKYU LEE

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2006

Copyright © by JEONGKYU LEE 2006

All Rights Reserved

To my wife, Kyungwon, and my lovely children, Eunji and Eunsuk.

ACKNOWLEDGEMENTS

It is my pleasure to thank the many people who made this thesis possible.

I would like to gratefully acknowledge the enthusiastic supervision of Dr. JungHwan Oh to complete this thesis. My appreciation also goes to my other committee members: Dr. Ramez Elmasri, Dr. Leonidas Fegaras, Dr. Alp Aslandogan, and Dr. Jean Gao. Thanks to all of them not just for being on my committee but for all of their guidance and assistance.

I would also like to thank all my colleagues in the Multimedia Information Group (MIG) who have helped me a lot throughout my Ph.D life. Especially, I want to say special ‘Thank You’ to Mehemmed Emre Celebi and Kyungseo Park who have discussed my research topics whenever I encountered problems, and made persistent relationship with my family.

I am forever indebted to my parent for providing a loving environment for me, and to my wonderful wife, Kyungwon Kahng for her sacrifice, encouragement and patience. She is my best friend in my life.

June 21, 2006

ABSTRACT

A GRAPH-BASED APPROACH FOR MODELING AND INDEXING VIDEO DATA

Publication No. _____

JEONGKYU LEE, Ph.D.

The University of Texas at Arlington, 2006

Supervising Professor: JungHwan Oh

With the advances in electronic imaging, storage, networking and computing, the amount of digital video has grown tremendously. The proliferation of video data has led to significant amount of research on techniques and systems for efficient video database management. In particular, extensive research has been done on video data modeling to manage and organize the data that is semantically rich and complicated. However, the enormous amount of data size and its complexity have restricted the progress on video data modeling, indexing and retrieval. In order to get around the problems, we turn to a graph theoretical approach for video database. Since a graph is a powerful tool for pattern representation and classification in various applications, it can represent complicated patterns and relationships of video objects easily.

In this dissertation, in order to capture the spatio-temporal characteristics of video object, we first propose a new graph-based video data structure, called *Spatio-Temporal Region Graph* (STRG), which represents spatio-temporal features and the correlations among the video objects. A *Region Adjacency Graph* (RAG) is generated from each

frame, and an STRG is constructed by connecting RAGs. An STRG is segmented into a number of pieces based on its content for efficient processing. Then, each segmented STRG is decomposed into its subgraphs, called *Object Graph* (OG) and *Background Graph* (BG) in which redundant BGs are eliminated to reduce index size and search time.

Next, we propose a new indexing of OGs by clustering them using unsupervised learning algorithms for more accurate indexing. In order to perform the clustering, we need a proper distance measure between two OGs. For the distance measure, we propose a new measure, *Extended Graph Edit Distance* (EGED) because the existing measures are not very suitable for OGs. The EGED is defined in non-metric space for clustering OGs, and it is extended to metric space to compute the key values for indexing. Based on the clusters of OGs and the EGED, we propose a new indexing structure *STRG-Index* which provides efficient retrieval.

Based on the STRG data model and STRG-Index, we propose a graph-based query language named STRG-QL, which is extended from object-oriented language by adding several graph operations. To process the proposed STRG-QL queries, we introduce a rule-based query optimization that considers the hierarchical relationships among video segments. For more efficient query processing, we show how to use STRG-Index during the query processing.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iv
ABSTRACT	v
LIST OF FIGURES	xi
LIST OF TABLES	xiii
Chapter	
1. INTRODUCTION	1
1.1 Multimedia Data	2
1.1.1 Basic Data Types in Multimedia	2
1.1.2 Characteristics of Video Data	4
1.2 Video Database Management	6
1.3 Motivation of Dissertation Work	8
1.4 Contributions	9
1.4.1 Spatio-Temporal Region Graph for Video Modeling	12
1.4.2 Extended Graph Edit Distance	13
1.4.3 Graph-based Video Data Mining	13
1.4.4 Spatio-Temporal Region Graph Indexing	14
1.4.5 STRG Query Language	14
1.5 Organization	14
2. RELATED WORK	16
2.1 Video Data Modeling	17
2.2 Distance Measures	19
2.3 Video Data Mining	19

2.4	Video Indexing Techniques	21
2.5	Video Query Processing	21
3.	STRG PRODUCING	24
3.1	Region Adjacency Graph	24
3.1.1	Region Segmentation	25
3.1.2	Region Adjacency Graph	25
3.2	Spatio-Temporal Region Graph	28
3.2.1	Subgraph Isomorphism	30
3.2.2	Neighborhood Graph	31
3.2.3	Graph-based Tracking	32
3.3	Experimental Setup of This Dissertation	36
3.3.1	Real Video Data Set	37
3.3.2	Synthesized Data Set	38
3.4	Experimental Results of STRG Generation	39
3.5	Summary	42
4.	STRG PARSING	43
4.1	Graph-based Shot Boundary Detection	44
4.1.1	Graph Similarity Measure	44
4.1.2	Shot Boundary Detection	45
4.2	STRG Decomposition	47
4.2.1	Object Region Graph	48
4.2.2	Object Graph	49
4.2.3	Background Graph	52
4.3	Experimental Results	53
4.3.1	Efficiency of Graph Similarity Measure	54
4.3.2	Performance of Shot Boundary Detection	54

4.3.3	Performance of Object Graph Detection	56
4.4	Summary	56
5.	STRG CLUSTERING	58
5.1	Extended Graph Edit Distance	59
5.2	Clustering OGs Using EGED	63
5.2.1	EM Clustering with EGED	63
5.2.2	Optimal Number of Clusters	66
5.3	Conceptual Clustering	67
5.3.1	Model Formation	67
5.3.2	Model-based Formal Concept Analysis	68
5.3.3	Concept Graph Generation	73
5.4	Experimental Results	81
5.4.1	Results of Clustering OGs	81
5.4.2	Results of Conceptual Clustering	85
5.5	Summary	90
6.	STRG INDEXING	91
6.1	STRG-Index Tree Structure	91
6.2	STRG-Index Tree Construction	94
6.3	STRG-Index Tree Node Split	95
6.4	Size Analysis	96
6.5	Experimental Results	97
6.5.1	STRG-Index with EGED vs. Graph Edit Distance	97
6.5.2	STRG-Index vs. M-tree	98
6.5.3	Efficiency and Scalability of STRG-Index	100
6.6	Summary	102
7.	STRG QUERY PROCESSING	105

7.1	STRG Data Model	106
7.2	Basic Query Statement	107
7.3	Extended Functions	109
7.3.1	GDM()	110
7.3.2	EGED()	111
7.3.3	SUMMARY()	111
7.3.4	MAKECLIP()	112
7.4	Supported Query Types	112
7.4.1	Query By Feature	113
7.4.2	Query By Example	114
7.5	STRG Query Processing	115
7.5.1	Rule-based Query Optimization	116
7.6	Experimental Results	118
7.6.1	Performance of GDM() and EGED()	118
7.6.2	Performance of Optimization	119
7.6.3	Accuracy of Retrieval	119
7.7	Summary	121
8.	CONCLUSIONS AND FUTURE RESEARCH	123
8.1	Summary of Contributions	123
8.2	Future Research Directions	124
	REFERENCES	127
	BIOGRAPHICAL STATEMENT	136

LIST OF FIGURES

Figure	Page
1.1 A simple video clip and the descriptions of contents	6
1.2 An architecture of Graph-based Video Database Management System . . .	10
3.1 (a) Original two consecutive frames, and (b) Results of region segmentation for (a)	26
3.2 (a) Real frame #14, (b) Region segmentation for (a), (c) $Gr(f_{14})$ for (b), (d) Enlarging a part of (b), and (e) Enlarging a part of (c)	28
3.3 Visualization of STRG for frame #14 – #16: (a) Region segmentation results, (b) STRG, and (c) Enlarging a part of STRG in (b)	33
3.4 Two examples of neighborhood graphs for nodes v_{14-44} and v_{14-52}	33
3.5 Algorithm 1: Maximal Common Subgraph	35
3.6 Algorithm 2: Graph-based Tracking	36
3.7 Visualization of synthesized data set when noise level is 5%: (a) Vertical (12), (b) Horizontal (12), (c) Diagonal (8), and (d) U-turn (16)	40
3.8 Screenshot of STRG generation	41
4.1 Shot detection results for abrupt changes	46
4.2 Shot detection results for gradual changes	47
4.3 (a) Sample object segmented several parts, (b) Example of ORGs for (a), (c) Merged OG, and (d) Results of temporal edges	50
4.4 (a) A part of STRG in Figure 3.3 (c), and (b) BG from (a)	53
5.1 Example of EGED between two Object Graphs: (a) No gap, and (b) A gap for insertion in OG^s	61
5.2 A model-based concept lattice for the context K in Table 5.2	73
5.3 Concept similarities between two concepts in Figure 5.2	75
5.4 Algorithm 3: A model-based conceptual clustering	79

5.5	Result of model-based conceptual clustering: (a) A result of concept merging using <i>ConSim</i> with $T_{sim} = 0.5$, and (b) A concept graph of (a)	80
5.6	CER: (a) EM-EGED vs. EM-LCS, EM-DTW, (b) KM-EGED vs. KM-LCS, KM-DTW, and (c) KHM-EGED vs. KHM-LCS, KHM-DTW	83
5.7	EM-EGED performances against KM-EGED and KHM-EGED: (a) Clustering Error Rate, (b) Cluster building time, and (c) Distortion	84
5.8	Results of EM clustering with <i>EGED</i> for video (Surv 1)	86
5.9	The BIC values for finding the optimal number of clusters	87
5.10	Quality of generated concepts using Relaxation Error	89
5.11	Quality of generated concepts using F-measure	90
6.1	Example of STRG-Index tree structure	92
6.2	Algorithm 4: Building STRG-Index	95
6.3	Query performances of STRG-Index with $EGED_M$ vs. with GED: (a) Distance computation,(b) Total processing time, and (c) Accuracy	99
6.4	Query performances of STRG-Index vs. MT-RA and MT-SA: (a) Distance computation, (b) Total processing time, and (c) Accuracy	101
6.5	Efficiency and Scalability of STRG-Index: (a) Building index time, and (b) Scalability of STRG-Index	104
7.1	Four basic classes in STRG data model	107
7.2	ODL schema using STRG data model	108
7.3	STRG Query processing phases	116
7.4	Execution plan of Q7 query	118
7.5	Performance of GDM() and EGED()	119
7.6	Performance of Optimization	120
7.7	Accuracy of STRG-QL	121
7.8	Result of k-NN query using STRG-QL (k=2)	122

LIST OF TABLES

Table	Page
2.1 Summary of various query languages for video	23
3.1 Description of real data set	38
3.2 Performance of STRG generation	41
4.1 The average number of processed frames per second for the SBD using <i>GSM</i> and <i>GSM_{sim}</i> with different frame rates: 5, 10, and 15 FPS	55
4.2 Experimental results of SBD	56
4.3 Experimental results of Object Graph (OG) detection	57
5.1 Example of model-based formal context without threshold	71
5.2 Example of model-based formal context with threshold ($\epsilon = 0.01$)	71
5.3 Formal concepts for context in Table 5.2	72

CHAPTER 1

INTRODUCTION

With the recent advances in electronic imaging, video devices, computing power, and network technologies, the use of multimedia data in many applications has increased significantly. Some examples of these applications are distance learning, digital libraries, video surveillance systems, medical videos, and video-on-demand. As a consequence, there are increasing demands on modeling, indexing and retrieving these data. Particularly, there is a strong demand for organization and management of video data. Video is a medium of communication that delivers more information per second than any other elements of multimedia, such as text, images, and audio. However, since the complexities of video data and their sheer volume as well as the limitation of current video processing techniques, have restricted progress on video data modeling, indexing and retrieval. In order to address the challenging problems, including modeling and indexing unstructured video data, our research exploits a graph that is a powerful tool for pattern representation and classification. The primary advantage of graph-based representation is that it can represent patterns and relationships among data easily. To take this advantage into video databases, we consider the characteristics of video data including spatio-temporal features, huge size and contents of it.

In this chapter, we introduce background and some related work of video database management, motivations and contributions of this dissertation.

1.1 Multimedia Data

We define multimedia data as the use of one or more different media, such as text, audio, image, animation, and video, to convey information. As the name implies, the multimedia is a combination of ‘multiple’ and ‘media’. The word ‘medium’ (the singular of media) means a transmission channel. Therefore, the important thing in terms of databases is ‘information’ that is transmitted through the media. In this section we first describe the basic data types in multimedia. Then, we present the characteristics of video data that we are focusing on in this dissertation.

1.1.1 Basic Data Types in Multimedia

There are a number of data types that can be characterized as multimedia data types. These are typically the elements or building blocks of more generalized multimedia data. The basic multimedia data types can be described as follows:

1. **Text** Although text is the conventional data type and has simple data format, it still plays an important role in multimedia to represent semantics of data very clearly. For example, in addition to ASCII-based text files, text is used in word processor, databases, and annotations on multimedia objects. In multimedia, text is used for the following purposes: titles, menus, annotations, and contents. The basic format of text is ASCII code. However, it is getting complicated into binary format, closed caption in video, and various fonts in Graphical User Interface. Text is the least space intensive data type in terms of storage. For example, 8.5 by 11 inch page (i.e., letter size) of text requires only 2 KB of storage without compression.
2. **Audio** Audio is one of the increasingly popular data types in multimedia. The sound heard by the ear is analog that is a continuous waveform, such as sounds produced by acoustic instruments, and human voices. The analog sound wave is transferred into its digital representation consisting of discrete numbers, which is

called digital audio. For example, one of the most popular is Microsoft's wave file format, i.e., WAV file. Audio is quite space intensive data type in terms of storage. For example, one second of digitized sound can require several tens of kilobytes of storage.¹ In order to reduce the size, many compression techniques can be used after digitizing sound, such as MPEG, Adaptive Differential Pulse Code Modulation (ADPCM) and Voc File Compression. However, the better quality of audio during the compression you want, the more space is required.

3. **Digital Image** Digitized images are sequence of pixels that represent a region. Pixels are numbers interpreted to allow the display of a particular 'dot' with different values for luminance, color, and contrast. Pixels can be as simple as 0 or 1 including white or black for black and white still images. On the other hand, higher resolution color images can contain 8 bits, 16 bits, or 24 bits per pixel. They allow the representation of millions of colors in high resolution. Based on the resolution, size, complexity, and compression method, the space requirement of digital can be various. For example, the size of 8.5 by 11 inch image can vary from 10 KB for simple black and white image to megabytes for complex, high-resolution color image.
4. **Digital Video** Video is a medium of communication that delivers more information per second than any other elements of multimedia. The video objects are stored as a sequence of still images, which is called as frames. The process of digitizing analog video is called video capture. The digital video is one of the most space intensive multimedia data types. Depending on its resolution and size, a single frame can consume more than 1 MB of storage. In case that a video is digitized as 30 frames/second, the total size of 1 hour long video can consume over 100 GB of storage. To reduce the size, an original video is compressed to the small size when

¹Depending on the sample rate and sample size, its size can be various.

it is digitized. The popular examples of video data format are AVI from Microsoft, and Quicktime format from Apple.

5. **Animation** Animation is the art of creating an illusion of movement from a series of still drawings. The basic idea of animation is that the human eye can detect extremely small amounts of luminous energy, and have the persistence of vision. Thus, they prevents the appearance of any flicker when a motion-picture film is displayed on a screen at the rate of at least 16 screen illuminations per second. Typically at the speed of 15 frames per second, the eye sees smooth motion.

Among these basic data types in multimedia, a video is the most important medium, since it integrates or combines all other data types, such as text, images, audio, and animation. For example, a video has text information as the format of caption. Caption data usually helps people with hearing problems watch TV programs. Now it is broadly used by the main TV channels and many educational audio and visual materials. There are two types of captions, namely, open and closed captions. Open caption data is stored and displayed as a part of video frames. Closed caption data is stored separately from each video frame and displayed as an overlap on video frames. When video is generated, audio data including sound, voice and speech, are synchronized with visual information. Since video consists of a sequence of frames, its basic elements are a group of still images. When they are displayed, they can make smooth motion picture like animation. In this dissertation, we mainly focus on video data among other multimedia data types since it can handle other media processing.

1.1.2 Characteristics of Video Data

Generally video can be defined as storage formats for moving pictures. Since it combines all other basic data types in multimedia, there are some important character-

istics comparing to the others, such as text, audio, and images. These characteristics of video can be summarized as follows:

- **Spatial and temporal data** As described in the previous subsection, video consists of sequence of still images, called frames in terms of video. Each frame has a group of pixels that represent location of dot and its values for luminance, color, and contrast. Therefore, the pixel information can be used for the spatial features of blocks, regions, and objects. For example, when a frame is segmented into a number of homogeneous color regions, we can compute a centroid of each region. The computed centroid represents the spatial information of the region. Moreover, the spatial information in the frame can span across multiple frames. For example, a centroid of a region in the current frame may be changed in the next frame. The differences over multiple frames are the temporal feature in video. The temporal feature plays an important role in video processing, such as moving objects segmentation, tracking, and shot boundary detection.
- **Huge size of data** The digital video is one of the most space intensive multimedia data types. When it is digitized, it requires a lot of space at disk drive. Suppose that we have one surveillance camera in operation 24 hours a day for 1 year. The captured data is stored as different qualities; visual phone quality (64K bps), VCR quality (1M bps), and broadcasting quality (4M bps). The total size of each video using different qualities without compression is as follows:

- Visual phone quality: $365 \times 24 \times 60sec \times 64Kbps = 2.5TB$

- VCR quality: $365 \times 24 \times 60sec \times 1Mbps = 38TB$

- Broadcasting quality: $365 \times 24 \times 60sec \times 4Mbps = 153TB$

Such a huge amount of video prevents efficient processing and management. Although compression techniques can reduce the size significantly, it still requires a huge amount of space comparing to other media types.

- **Semantically rich data format** Multimedia data contains more semantics than conventional data type such as alpha-numeric data. Particularly, video has more information than any other media. For example, Figure 1.1 shows a simple video clip and semantics in it. We can get a lot of information from the video clip. From the contents, we know the name of player, his number, uniform color, and so on. Also, we can recognize more important information from the caption data on the top-left corner of frames, such as current score, teams, and name of stadium.

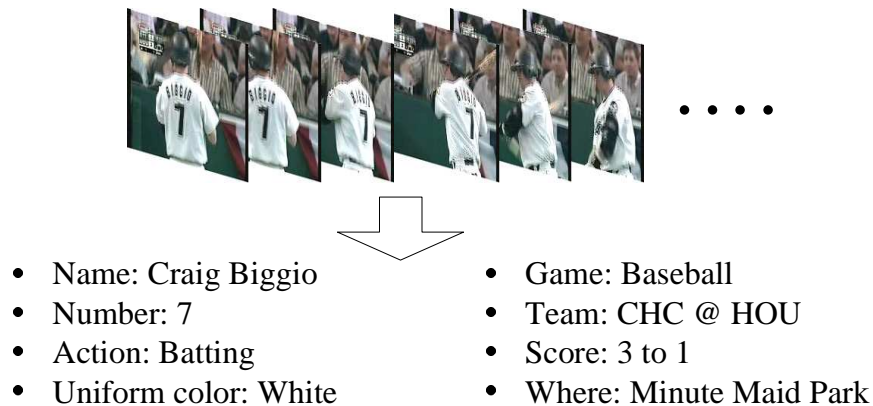


Figure 1.1. A simple video clip and the descriptions of contents.

Three characteristics of video mentioned above should be considered when it is processed and managed.

1.2 Video Database Management

Just like multimedia databases can be defined as various things, video database management systems (VDBMSs) mean different things to different people. The reason is that video can imply so many different things. For example, those who are using camcorder at home regard video databases as a storage for the video taken by the camera. Another example of video database is video-on-demand where users can select a video

from database and play it on their monitor. It employs high-speed parallel disk storage as well as high-speed networking.

In this dissertation, we define video database as the efficient management system for video that supports organization, data structure, and retrieving video objects. The following is a brief definition of the video database functions that should be considered in this dissertation.

- **Video data modeling** In a conventional database management system (DBMS), accessing data is based on distinct attributes of well-defined data developed for a specific application. For video data that has unstructured format, we can define similar attributes, i.e., low-level features such as color, shape, and texture. Due to the characteristics of video mentioned earlier, VDBMS should have different model to organize these attributes values. We first extract feature values from unstructured data. Next, the extracted data must be appropriately modeled by considering spatial and temporal characteristics of video.
- **Video data parsing** Due to the huge size of video data, it should be segmented into proper processing units, such as shots or scenes. In addition to video segmentation, a video needs to be decomposed into meaningful units for further processing. The examples of video parsing include moving objects detection and background modeling.
- **Video data indexing** In addition to data modeling and parsing, video data should be organized for easy access. The large information in a video data makes manual indexing or manual annotation labor intensive, time consuming and prone to errors. To address these, indexing structure in VDBMS should support an automatic building and fast accessing. Also, the indexing incorporates various levels of video objects, such as block, region, moving object, shot, and scene levels.

- **Video data mining** Whether dealing with video, video database applications tend to involve referencing relationships between video objects. In order to provide efficient access to the relationships of video objects, VDBMS should incorporate video data mining techniques. Among the existing data mining techniques, we can employ unsupervised learning algorithms, such as clustering, and conceptual clustering, to find the complex relationships and meaningful information in video objects.
- **Video data querying** Given the data and indexing structures mentioned above, queries video objects must be optimized. The query optimization in VDBMS need to consider the different levels of video objects and their relationships. Also, the query processing supports various types of user requests such as query by example and feature.

1.3 Motivation of Dissertation Work

Graph is a powerful tool for pattern representation and classification in various fields [1, 2, 3], such as image processing, video analysis, and biomedical applications. The primary advantage of graph-based representation is that it can represent patterns and relationships among data easily. To take this advantage into video analysis, several studies have proposed the graph-based techniques [4, 5, 6, 7]. In Region Adjacency Graph (RAG) [4, 5], segmented regions and spatial relationships among them are expressed as nodes and edges, respectively. However, RAG cannot represent the temporal characteristic of video which is its representative feature. Also, various graph matching algorithms such as bipartite matching [6] and error-correcting matching [7] have been used in video data. However, the existing graph matching algorithms still require high computational cost, and suffer from low accuracy since they consider only the spatial feature to match video data.

To address these, we propose a new graph-based data structure, called *Spatio-Temporal Region Graph* (STRG) representing spatial and temporal relationships among objects in a video sequence. The STRG is constructed by combining RAGs generated from each frame, and decomposed into its subgraphs, called *Object Region Graphs* (ORGs) representing the same corresponding regions. ORGs representing the same object over frames are merged into an *Object Graph* (OG) which represents each semantic object in a video sequence. For unsupervised learning, we cluster similar OGs into a group, in which we need to match two OGs. For this graph matching, we introduce a new distance measure, called *Extended Graph Edit Distance* (EGED), which can handle temporal characteristics of OGs. The EGED is defined in a non-metric space first for the clustering of OGs, and it is extended to a metric space to compute the key values for indexing. Based on the clusters of OGs and the EGED, we propose a new indexing method *STRG-Index*.

1.4 Contributions

This dissertation makes the contribution to modeling and indexing video data using a graph. To do this, we design and develop the algorithms of a graph-based video database management system (GVDBMS) focusing on representing spatial and temporal characteristics of video objects. Figure 1.2 illustrates the proposed GVDBMS architecture. GVDBMS consists of three primary components; *Video Processing*, *Repository* and *Query Processing* component.

Video Processing Component

The main purpose of this component is to generate STRG data from input videos. Unlike existing graph-based video processing techniques which consider only spatial information,

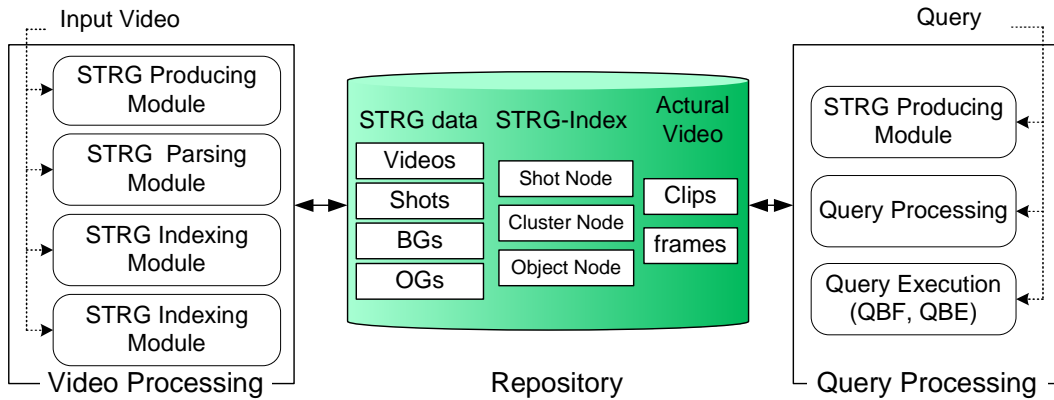


Figure 1.2. An architecture of Graph-based Video Database Management System.

an STRG represents temporal information among video objects additionally. The component includes several submodules to process input videos as follows:

1. *STRG producing module* generates Region Adjacency Graph (RAG) from each frame, and connects RAGs temporally to construct an STRG for each video. To do this, we first divide a frame into homogeneous color regions using region segmentation technique. The segmented regions and spatial information among them are expressed as nodes and edges in RAG, respectively. An STRG is constructed by connecting RAGs temporally. In order to connect RAGs, we use a graph-based tracking algorithm by graph matching.
2. *STRG parsing module* partitions an STRG into shots for more efficient processing. In order to do this, we compute a temporal connectivity between two consecutive RAGs (frames) by graph matching. If two RAGs are strongly connected to each other, the two frames corresponding to the two RAGs are considered to be in a same shot. Otherwise, the two frames are considered to be in two different shots. After detecting shot boundaries of an STRG, we compute a key RAG (key frame) of each shot to characterize it. Then, it extracts Object Graphs (OGs) and Background

Graph (BG) which are subgraphs of an STRG. An OG represents an object which appears over frames, and a BG represents a background of each shot.

3. *STRG clustering module* clusters similar OGs into a group to find the pattern of moving objects. The results of cluster will be used for indexing for efficient access. In order to find formal concepts of clusters, they are analyzed based on formal concept analysis techniques.
4. *STRG indexing module* creates STRG-Index for better performance of a query, which is a tree structure including shot nodes, cluster nodes and object nodes. Each node is pointing the actual units of videos, i.e., clips, shots and frames, to support various types of user queries.

Repository Component

The outputs of video processing component, such as STRG data, STRG-Index and actual units of videos are stored at the repository. Although we will not cover repository component in detail in this dissertation, this component however includes very important issues in practical systems.

Query Processing Component

The query processing is responsible for processing and responding user queries. A user can request various types of queries using query processing component. The proposed GVDBMS supports two types of queries: Query by Feature (QBF) and Query by Example (QBE). In QBF, a user retrieves salient objects and shots from database that satisfy the conditions given by feature values in a query statement. In QBE, a user retrieves those using a sample video clip in a query statement. To process QBE, a temporary STRG data model is generated from a query video by *STRG producing module*, which is the same module in the video processing component. Then, an STRG-Index and a rule-

based query optimization are used for an efficient execution of a query, which consider the hierarchical characteristics, such as video, scene, shot, and frame appropriately.

The detailed contributions of this dissertation work are discussed in this section.

1.4.1 Spatio-Temporal Region Graph for Video Modeling

First, we propose a new video data structure, STRG, based on graph representation. It can represent not only spatial features of video objects, but also temporal correlations among them. In addition, the STRG can handle various types of videos since it supports shot level as well as object level operations. Almost all produced videos, such as drama, news, and documentary videos, consist of a number of shots that are the basic processing units of video. On the other hand, non-produced videos, such as surveillance videos, are taken continuously without shot changes. Our proposed video data structure, STRG can support both types of videos: produced videos with shots, and non-produced videos without shots. To achieve this, we model a video from the level of region that is homogeneous color region forming an object in a video. The generated STRG is segmented into a number of shots to support produced videos.

In addition to shot boundary detection, the proposed STRG model is applied to perform other video processing techniques. For example, moving objects in a sequence of video frames are tracked by a graph matching. Since a graph-based tracking algorithm considers the relationships of moving objects represented as a graph, it provides more accurate and consistent results. Another example of video processing using STRG is video summarization. A long video can be summarized into shorter version of video clips or highlight. The last example of video processing is background modeling by overlapping nodes and edges in a graph over frames.

1.4.2 Extended Graph Edit Distance

In order to process video objects, we need to have proper similarity or dissimilarity measure of them. Thus, we propose a new distance function of OGs, EGED which is defined in both non-metric and metric spaces. Since non-metric EGED provides more accurate results, it is used for matching video moving objects represented as a graph (OG). On the other hand, metric EGED is used for indexing STRGs because of its efficiency.

We also introduce a Graph Similarity Measure (GSM) to measure the similarity while STRG is processed. For example, STRG is segmented into a number of smaller pieces, i.e., shots, by using GSM. In addition, GSM can be used for matching BGs to find similar one.

1.4.3 Graph-based Video Data Mining

For unsupervised learning, we cluster similar OGs into a group which represents a moving pattern. To do this, we exploit a model-based clustering algorithm (EM) with *EGED*.

Based on the results of a model-based clustering algorithm, we propose a model-based conceptual clustering (MCC) of spatio-temporal data to find its formal concepts. Unlike existing conceptual clustering algorithms that use a goodness measure for a clustering criterion, the proposed MCC exploits formal concept analysis to generate the concepts. Since formal concept analysis can find not only the formal concepts but also the relations among them, our conceptual clustering is more effective than traditional ones.

1.4.4 Spatio-Temporal Region Graph Indexing

Based on the clusters of OGs and the EGED, we propose a new indexing structure, STRG-Index. Since STRG-Index uses a metric version of EGED for the distance function, tree structure and data clustering, it provides fast and accurate indexing. The STRG-Index tree structure consists of three level of nodes; shot node, cluster node and object node. Shot node contains the BGs of segmented STRGs. Each record in the node has its identifier, an actual BG, and an associated pointer. Cluster nodes contain the centroid OGs representing cluster centroids. A record in the cluster node contains its identifier, a centroid OG of each cluster, and an associated pointer. Object nodes contain OGs belonging to a cluster. A record in the node has the index key, an actual OG, and an associated pointer.

1.4.5 STRG Query Language

Based on the STRG data model, we propose a new graph-based query language, named STRG-QL which is extended from object-oriented query language by adding several graph operation. To process the proposed STRG-QL queries, we propose a rule-based query optimization considering the logical structure of video and hierarchical relationships among video segments, which provides more efficient STRG-QL processing. In addition, we present two main query types that STRG-QL supports; Query by Feature (QBF) and Query by Example (QBE). While QBF is the basic query type which is compatible to ODMG OQL, QBE supports a query with a sample video clip.

1.5 Organization

The rest of the dissertation is organized as follows. Chapter 2 describes detailed background and related works of video database management systems including video data modeling, distance measure, video data mining, video indexing techniques, and

query processing. Chapter 3 discusses our proposed video data model: Region Adjacency Graph (RAG) and Spatio-Temporal Region Graph (STRG). Chapter 4 presents several video parsing techniques based on a graph theory, such as a graph-based shot boundary detection, STRG decomposition for Object Region Graph (ORG), Object Graph (OG), and Background Graph (BG). We also present a graph similarity measure that is used for the video parsing. In Chapter 5, we propose a distance measure for OGs by extending Edit Distance to a graph, which is called Extended Graph Edit Distance (EGED). Using the distance measure of OGs, we then proposed two data mining techniques for video to find concepts of video contents: EM clustering with EGED, and model-based conceptual clustering algorithms. Chapter 6 present a new indexing structure of video objects, named Spatio-Temporal Region Graph Indexing (STRG-Index). It begins with a description of indexing that is a tree structure consisting of shot, cluster, and object nodes. Then, we introduce how to construct and split the node. Chapter 7 describes our proposed video query processing techniques including STRG Query Language (STRG-QL) and rule-based query optimizations. Finally, Chapter 8 concludes this dissertation with future research directions.

CHAPTER 2

RELATED WORK

With the advances in electronic imaging, storage, networking and computing power, the amount of digital videos has grown tremendously. The proliferation of video data has led to significant amount of research on techniques and systems for efficient video database management. These research for video database management fall into five categories: video data modeling, similarity/dissimilarity function, video data mining, video data indexing, and video query processing. Unlike traditional data type, such as alpha-numeric data that can be modeled by relational model or object-oriented model, a video data cannot be organized easily since it is unstructured data format and has a huge amount of semantic contents. To address this issue, there have been a lot of research efforts of modeling video data [8, 9, 10, 11, 12, 13, 14, 15, 16, 17]. Video data requires a pre-processing step before it is managed: for example, shot boundary detection, and region segmentation. For these video processing, we need appropriate distance measures for video data. Since video data has temporal characteristics, the similarity or dissimilarity measure of video data should handle time characteristics [18, 19, 20, 21]. Another important issue in video database management is video data mining to find the knowledge in it. Specifically, finding concepts from video data can solve the problem of unstructured data format, such as conceptual clustering algorithm [22, 23, 24, 25, 26, 27]. The final goal of video database management is how to retrieve video data efficiently and effectively. To do this, there have been many research work for video indexing [28, 29, 30, 31, 32] and query processing [33, 34, 35, 36, 37, 38, 39, 40].

The rest of this chapter is organized as follows. In Section 2.1, we presented the existing video data models. Section 2.2 shows the distance measure for video data in detail. Video data mining techniques are discussed in Section 2.3. Section 2.4 and 2.5 introduce the existing video indexing and query processing techniques, respectively.

2.1 Video Data Modeling

Extensive research has been done on video data modeling to manage and organize the data which is semantically rich and complicated. The approaches to video data modeling can be divided into three categories as follows:

1. **Annotation based data models** [8, 9, 10] use manually extracted text descriptions to represent the content of video segments. The extracted content descriptions (annotations) attached to video streams are capable of conveying the semantic content. However, they have been criticized as computation demanding and inaccurate since manual annotations are needed.
2. **Segmentation based data models** [11, 12, 13] use visual features (i.e., color, shape, texture, etc.) to segment a video into a set of processing units, such as shots and/or scenes. Key frames are used to index the units. This model can support visual content-based video access, and its video processing can be fully automated. However, the visual features are limited to represent semantic content of video segment, consequently the visual feature-based retrieval cannot provide desirable results.
3. **Object based based data models** [14, 15, 16, 17] use spatio-temporal features extracted from video objects to address the limitation in representing their semantic content. The video objects are typically generated from video sequence by object tracking. Instead of indexing physical or logical units of video (shots and scenes), this approach attempts to use spatio-temporal correlations among video

objects to index the content. However, this method needs an efficient video object representation to extract spatio-temporal features, which is non-trivial.

Since the video data is semantically rich and complicated, an efficient video object representation for spatio-temporal features is essential for bridging the gap between the visual features and the semantic content.

In order to address this issue, several efforts have been made to model video data using graph representation [10, 13, 14]. Graph is a powerful tool for data representation and classification in various fields, such as image processing, video analysis, and biomedical applications. The primary advantage of graph-based representation is that it can represent patterns and correlations among data effectively. To take this advantage in video analysis, Kokkoras et. al. [10] propose Smart VideoText which is an annotation-based video data model using graph. Smart VideoText uses the conceptual graph to capture the semantic associations among the concepts in text annotations. It is capable of effective query, retrieval, and browsing based on the annotations. However, the conceptual graph cannot capture the visual content of video data. Moreover, it is difficult to implement an automated system since it requires manual annotation. The other example is a video summarization and scene detection technique using graph matching by Ngo et. al. [13]. A video is represented as a weighted undirected graph where the nodes and edges indicate the features extracted from shots and the similarity between two shots, respectively. Even though this method uses a graph representation, it is still limited to represent the semantic content of video since it is shot-based and does not consider its objects explicitly. Another example is the Video Semantic Directed Graph (VSDG) by Day et. al. [14], which represents a semantically unbiased abstraction for video objects using a directed-graph model. In order to construct user-independent view and semantic heterogeneity of the video data, a VSDG is created to model spatio-temporal

interactions among video objects. The spatio-temporal semantics are represented by conceptual spatial/temporal objects. However, it has difficulty in capturing the spatio-temporal characteristics (i.e., trajectories) of moving objects, and may contain redundant information as mentioned in [41]

2.2 Distance Measures

The next important topic in video database systems is how to compute the similarity or dissimilarity between two units according to the feature values extracted from each unit. Many similarity or dissimilarity measures have been proposed based on the feature values. The distance functions considering *time* which is a primary factor of videos, include the traditional distance functions (i.e., Lp-norms) [18], Dynamic Time Warping (DTW) [19], Longest Common Subsequence (LCS) [20], and Edit Distance (ED) [21]. Although the traditional distance functions are easy to compute, they cannot measure very well the difference between the units with multiple and complex features. The others are perceptually better than the traditional distance functions. However, they are non-metric, so they are not applicable to the standard indexing algorithms.

2.3 Video Data Mining

In the conceptual clustering, a *goodness measure* is usually defined for overall partitioning of objects, while the other clustering methods use *distance metrics* for the (dis)similarities of pairs of objects, such as Euclidean and Mahalanobis distances. COBWEB [22] is an incremental clustering algorithm based on probabilistic categorization trees. The search for a good clustering is guided by a goodness measure for partitions of data, i.e., *category utility* (*CU*). However, pure COBWEB only supports nominal attributes. In other words, it cannot be used for abstracting numeric data. In order to

incorporate numeric values, some extensions of the COBWEB system are proposed, such as COBWEB/3 [23], ECOBWEB [24], AUTOCLASS [25], Generality-based conceptual clustering (GCC) [26], and Error-based conceptual clustering (ECC) [27]. COBWEB/3 [23] combines the original COBWEB algorithm with the probabilities expressed in terms of the probability density function (*pdf*) to handle numeric attributes in the *CU* measure. In ECOBWEB [24], the probability distribution for numeric features is approximated by the probability distribution about the mean for the features. Both COBWEB/3 and ECOBWEB use modifications of the *CU* measure to handle numeric features. AUTOCLASS [25] uses a classical finite mixture distribution model on the data, and derives groupings of objects that locally maximize the posterior probability of individual clusters given by the feature distribution functions. GCC [26] method proposes a parameterized measure that allows a user to specify both the number of levels and the degree of generality of each level to build a symbolic hierarchical clustering model. ECC [27] method partitions the data set of one or more features into clusters that minimize a relaxation error as the goodness measure.

However, since all of the methods mentioned above use a goodness measure to find good clusters, they cannot represent the relations among the clusters. Recently, in order to address this, a few methods [42, 43] employ *formal concept analysis* (FCA) technique to conceptual clustering. FCA is a formal technique for knowledge representation and data analysis that was introduced by Wille [44, 45]. FCA uses formal context consisting of data objects, attributes, and their binary relations. For example, Hotho et al. [42] propose a conceptual clustering of text documents using FCA to describe the description of the resulting clusters. In their approach, the frequency of keywords in a document replaces a binary relation of formal context. Fuzzy conceptual clustering in FOGA (Fuzzy Ontology Generation frAmework) [43] combines fuzzy logics and FCA to generate concepts and relations of citation database. FOGA uses fuzzy membership values between data objects

and features in a formal context. The two major drawbacks of using FCA for conceptual clustering are as follows: (1) *scaling*: at the beginning of FCA, each attribute should be scaled to handle numeric data. However, there is no general solution to determine ideal scaling of spatio-temporal data, since they are not only high dimensional numeric features, but also a sequence of data in time, and (2) *large volume of structure*: the growing sets of objects and features bring exponential growth of the resultant concepts and relations, which may cause a decreasing of the performance on FCA. Moreover, such a high number of concepts and relations make their interpretations difficult.

2.4 Video Indexing Techniques

There has been relatively little effort on indexing and retrieving the segmented video units [28]. A major difficulty is how to handle spatio-temporal correlations among video objects. To address this, an indexing structure called 3DR-tree [29] has been proposed. It indexes salient objects by treating the time as another dimension in R-tree. Simply treating the time as another dimension is not optimal since spatial and temporal features should be considered differently. Several index structures, such as RT-tree [31], B+-tree [30], and M-tree [32], have been proposed to handle spatio-temporal features. However, they are inefficient for various queries on moving objects since they cannot capture the characteristics of moving objects properly.

2.5 Video Query Processing

In recent years, many video query languages have been proposed for video database management systems. In this subsection, we review and compare various query languages which can be classified into three categories: entirely new query language, SQL-based query language, and OQL-based query language.

First category is the entirely new query languages [33, 34] which are brand new systems without using any existing one. CVQL (Content-based logic Video Query Language) is proposed in [34], where users can specify query predicates by the spatial and temporal relationships of the content objects. Since CVQL is defined based on video frames, a user needs to have some prior knowledge about the video content. TVQL (Temporal Visual Query Language) [33] is developed for querying multidimensional range queries via visual interface. In these two languages, it may take some time for users to learn and use them since they are little flexible and domain specific.

Second category is the query languages which are extension of SQL [35, 36, 37, 38]. SQL-like query languages are the majority of existing video query languages since they are easy to implement and use. VideoSQL [35] is an SQL-like query language developed to retrieve video objects. It uses an inheritance mechanism which provides the sharing of common descriptive data among videos. However, the language does not include spatial or temporal information of videos. STQL (Spatio-Temporal Query Language) [36] tries to address this by integrating spatio-temporal data into existing data models as abstract data type. Recently, a rule-based spatio-temporal query language is proposed in BilVideo [38] which is a web-based VDBMS. BilVideo focuses on the task of spatio-temporal query processing and SQL-like query language. Since STQL and BilVideo focus on only moving objects in a video, there may be some difficulties to handle other logical structure of a video such as frame, shot and scene.

The last category is the query languages which are extensions of OQL [39, 40]. GOQL (Graph Object Query Language) [40] and MOQL (Multimedia Object Query Language) [39] are the extensions of OQL to handle multimedia data. GOQL queries are translated using an operator-based language, O-Algebra which is an object algebra designed for processing object-oriented database (OODB) queries. MOQL supports content-based spatial and temporal queries as well as query presentation.

Table 2.1 shows the summary of various query languages mentioned above. As seen in the table, most works are designed for the particular data models and applications. In addition, only limited number of query types are supported in each system.

Table 2.1. Summary of various query languages for video

Name	Year	Basis	DBMS	Algebra	Target Data	Data Structure	Query Type	Visualizaton
VideoSQL	1993	SQL	OVID	X	Video Frame	Video Object	Content query	X
TVQL	1996	new	MMVIS	X	Video Event	X	Visual query	O
MOQL	1997	OQL	ObjectStore	Object Algebra	Video Object	X	Content query	X
GOQL	1999	OQL	Vstore	O-Algebra	Multimedia	Graph	Sequence query	X
CVQL	1999	new	VDBMS	X	Video Frame	Content Object	Content query	X
STQL	2002	SQL	MDBMS	Extend Relational	Moving Object	Extend Relational	Temporal query	O
VQP	2003	SQL	VDBMS	X	Stream Video	Streaming Data	Temporal join	X
Rule-based	2004	SQL	BiVideo	Extend Relational	Video Segment	X	Knowledge query	X

CHAPTER 3

STRG PRODUCING

Graph is a powerful tool for pattern representation and classification in various fields, such as image processing, video analysis, and biomedical application. The primary advantage of graph-based representation is that it can represent patterns and relationships among data effectively. In order to take this advantage into video modeling, we propose a new graph-based data structure, called Spatio-Temporal Region Graph (STRG), which represents the spatio-temporal features and relationships among the objects extracted from video sequences. Region Adjacency Graph (RAG) [4] is generated from each frame, in which segmented regions and their spatial relationships are expressed as nodes and edges, respectively. By connecting the corresponding nodes along the RAGs, we construct an STRG for a video. The STRG is a new data structure for video data based on a graph. It can represent not only spatial features of video objects, but also temporal relationships among them.

The rest of this chapter is organized as follows. In Section 3.1, we explain how to segment each frame into a number of region, and how to generate a RAG for a frame from segmented regions. In Section 3.2, we construct an STRG from the constructed RAGs and corresponding nodes. Section 3.3 describes the experimental results of building STRG for real video sequences in detail. Section 3.4 provides the summary of the chapter.

3.1 Region Adjacency Graph

The first step in image and video processing is to decide the basic unit of processing such as pixel, block or region. Then, the features are computed from each unit for further

processing. Recently, some studies focused on a graph-based approach to process image and video data [4, 46], since a graph can represent not only these units but also their relationships. We first describe Region Adjacency Graph (RAG) which is the basic data structure for video indexing.

3.1.1 Region Segmentation

Assume that a video segment (S) has N frames. To divide a frame into homogeneous color regions, we use region segmentation algorithm called EDISON (Edge Detection and Image Segmentation System) [47]. There have been many research for region segmentation algorithm such as EDISON [47], JSEG [48], and ImageJ [49]. The reason we choose EDISON among other algorithms is that it is less sensitive to small changes over the frames, which occurs frequently in video sequence. Figure 3.1 shows the results of region segmentation using EDISON. In order to show the consistency, we run the EDISON on various video domains. The first row is the sample frames of colonoscopy video in a medical domain. The second row is the sample of traffic video, and the third is taken by surveillance camera. Figure 3.1 (a) is two consecutive frames for each video. As shown in Figure 3.1 (b), the results of region segmentation using EDISON are very consistent regardless of the types of videos.

3.1.2 Region Adjacency Graph

Using region segmentation algorithm, we have a number of homogeneous color regions from each frame. The segmented regions of a frame have several important characteristics as follows:

- Each region has its own spatial information represented by a position.
- Each region has its own characteristics, such as a representative color and a size of region.

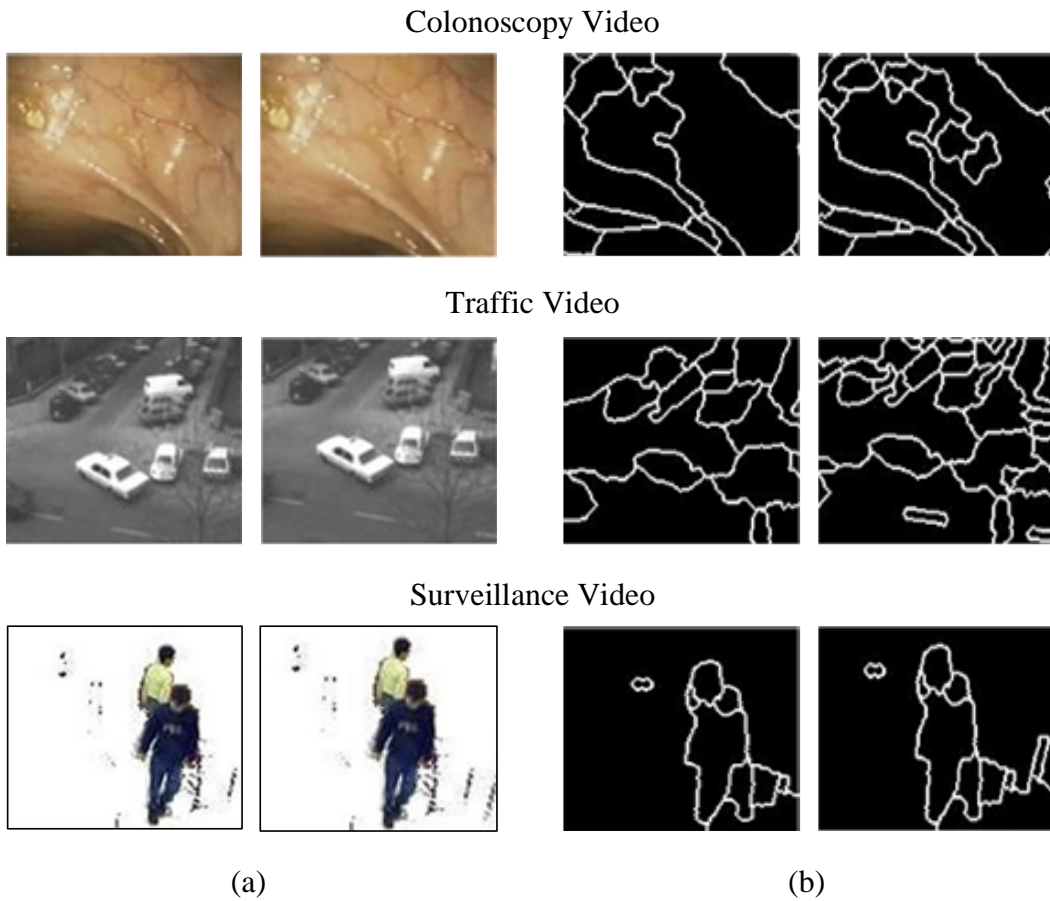


Figure 3.1. (a) Original two consecutive frames, and (b) Results of region segmentation for (a).

- Two adjacent regions may form either a same object or two different objects.

In order to represent the relationships and characteristics mentioned above, we employ a concept of Region Adjacent Graph (RAG) in image processing, which is defined as follows:

Definition 1. Given the n^{th} frame f_n in a video, a Region Adjacency Graph of f_n , $Gr(f_n)$, is a four-tuple $Gr(f_n) = \{V, E_S, \nu, \xi\}$, where

- V is a finite set of nodes for the segmented regions in f_n ,

- $E_S \subseteq V \times V$ is a finite set of spatial edges between two adjacent nodes in f_n ,
- $\nu : V \rightarrow A_V$ is a set of functions generating node attributes, and
- $\xi : E_S \rightarrow A_{E_S}$ is a set of functions generating spatial edge attributes.

A node ($v \in V$) corresponds to a region (r), and a spatial edge ($e_S \in E_S$) represents a spatial relationship between two adjacent nodes (regions). The node attributes (A_V) are location (x, y : centroid of a region), size (s : number of pixels in a region) and color (λ : mean of colors in a region) of the corresponding region. The spatial edge attributes (A_{E_S}) indicate relationships between two adjacent nodes such as spatial distance (d : spatial distance between centroids of two regions) and orientation (ϕ : angle between a line formed by two centroids and horizontal line). The attribute vectors of node (A_V) and spatial edge (A_{E_S}) can be defined as:

$$\begin{aligned} A_V &= (x, y, s, \lambda)^T \\ A_{E_S} &= (d, \phi)^T \end{aligned}$$

As illustrated in Figure 3.2, a RAG provides a spatial view of regions in a frame. Figure 3.2 (a) and (b) show a real frame and its segmented regions respectively. A RAG in Figure 3.2 (c), $Gr(f_{14})$ is constructed according to Definition 1. Each circle indicates a segmented region. And, the radius, the color, and the center of circle correspond to the node attributes; size (s), color (λ) and location (x, y), respectively. In addition, the length and angle of the edges represent the spatial edge attributes; spatial distance (d) and orientation (ϕ) between two adjacent nodes. In Figure 3.2 (d) and (e), we enlarge two parts in Figure 3.2 (b) and (c) respectively to show more details on how to construct a RAG from segmented regions. For example, a region r_{58} in Figure 3.2 (d) forms a node v_{58} in Figure 3.2 (e) preserving the attributes. Since a region r_{58} is adjacent to two regions (r_{39} and r_{45}), two spatial edges ($e_S(v_{58}, v_{39})$ and $e_S(v_{58}, v_{45})$) are created.

Unlike existing techniques using RAG, the RAG defined in Definition 1 is an attribute graph which represents not only spatial relationships among segmented regions, but also its properties using nodes and edges. A node attribute (A_V) and an edge attribute (A_E) provide more precise information.

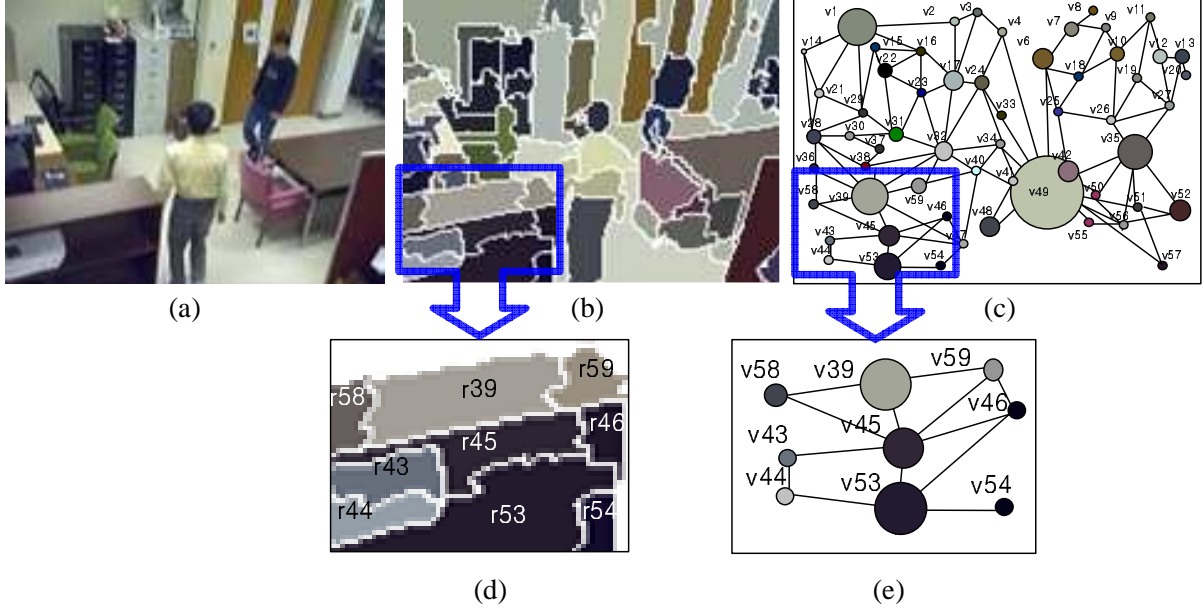


Figure 3.2. (a) Real frame #14, (b) Region segmentation for (a), (c) $Gr(f_{14})$ for (b), (d) Enlarging a part of (b), and (e) Enlarging a part of (c).

3.2 Spatio-Temporal Region Graph

While a RAG generated from each frame represents the spatial relationships among the nodes (regions), it cannot represent the temporal characteristics of video data, such as speed and direction of moving objects, and moving patterns. A node representing a region can span across multiple frames. Therefore, if the corresponding nodes in the consecutive frames are connected, its temporal characteristics can be represented. In order

to model this, we propose a new graph-based data structure of video, *Spatio-Temporal Region Graph* (STRG) that is temporally connected RAGs. The STRG can handle both temporal and spatial characteristics of video data. It is defined as follows:

Definition 2. *Given a video segment S , a Spatio-Temporal Region Graph, $Gst(S)$, is a six-tuple $Gst(S) = \{V, E_S, E_T, \nu, \xi, \tau\}$, where*

- V is a finite set of nodes for segmented regions from S ,
- $E_S \subseteq V \times V$ is a finite set of spatial edges between adjacent nodes in S ,
- $E_T \subseteq V \times V$ is a finite set of temporal edges between temporally consecutive nodes in S ,
- $\nu : V \rightarrow A_V$ is a set of functions generating node attributes,
- $\xi : E_S \rightarrow A_{E_S}$ is a set of functions generating spatial edge attributes, and
- $\tau : E_T \rightarrow A_{E_T}$ is a set of functions generating temporal edge attributes.

In an STRG, a temporal edge ($e_T \in E_T$) represents a relationship between two corresponding nodes in two consecutive frames such as velocity (ω : how much their centroids are changed) and moving direction (π : angle between a line formed by two centroids of the corresponding nodes and horizontal line). Figure 3.3 shows a part of STRG for frames #14 – #16 in a sample video generated by Definition 2. Figure 3.3 (a) is the region segmentation results by EDISON for frames #14 – #16, and Figure 3.3 (b) shows the constructed STRG by adding temporal edges between two consecutive RAGs (i.e., $Gr(f_{14}) - Gr(f_{15})$, and $Gr(f_{15}) - Gr(f_{16})$). In Figure 3.3 (c), we enlarge parts of Figure 3.3 (b) to show more details about the STRG construction. For example, two corresponding nodes v_{14-44} and v_{15-39} are connected by a temporal edge $e_{T_{14-17}}$ between RAG $Gr(f_{14})$ and $Gr(f_{15})$, and v_{15-39} and v_{16-42} are connected by $e_{T_{15-15}}$ between RAG $Gr(f_{15})$ and $Gr(f_{16})$.

An STRG becomes an extension of RAGs by adding a set of temporal edges (E_T) to them. E_T represents the temporal relationships between corresponding nodes in two consecutive RAGs. For example, E_T shows the tracked regions over time, and their properties, such as moving direction and speed. The main procedure of building an STRG is therefore, how to construct E_T , which is similar to *object tracking* in a video sequence. Although there have been numerous efforts [50, 51] on object tracking, it is still an open problem, since existing tracking algorithms usually use low-level features, but complicated patterns of moving objects cannot be interpreted easily by the low-level features. In order to overcome this problem, we propose a new graph-based tracking method, which considers not only low-level features but also relationships among the object regions. We use the *graph isomorphism* and the *maximal common subgraph* [3] to find the corresponding nodes in two consecutive RAGs.

3.2.1 Subgraph Isomorphism

To describe our graph-based tracking algorithm, we define subgraph isomorphism (See Definition 5). The subgraph isomorphism is based on the definitions of subgraph (See Definition 3) and graph isomorphism (See Definition 4). We first define subgraph between two graphs, Gr and Gr' as follows:

Definition 3. *Given a graph $Gr = \{V, E_S, \nu, \xi\}$, a subgraph of Gr is a graph $Gr' = \{V', E'_S, \nu', \xi'\}$ such that*

- $V' \subseteq V$ and $E'_S = E_S \cap (V' \times V')$,
- ν' and ξ' are the restrictions of ν and ξ to V and E_S , respectively, i.e.

$$\nu'(v) = \begin{cases} \nu(v) & \text{if } v \in V', \\ \text{undefined} & \text{otherwise.} \end{cases}$$

$$\xi'(e_S) = \begin{cases} \xi(e_S) & \text{if } e_S \in E'_S, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The notation $Gr' \subseteq Gr$ is used to indicate that Gr' is a subgraph of Gr . If two graphs are identical, we called them as isomorphic graphs, which is defined as follows:

Definition 4. *Two graphs $Gr = \{V, E_S, \nu, \xi\}$ and $Gr'' = \{V'', E''_S, \nu'', \xi''\}$ are isomorphic if there is a bijective function $f : V \rightarrow V''$ such that,*

- $\nu(v) = \nu''(f(v)) \forall v \in V$,
- *For any edge $e_S = (v_1, v_2) \in E_S$ there exists an edge $e''_S = (f(v_1), f(v_2)) \in E''_S$ such that $\xi(e''_S) = \xi(e_S)$, and for any edge $e''_S = (v''_1, v''_2) \in E''_S$ there exists an edge $e_S = (f^{-1}(v''_1), f^{-1}(v''_2)) \in E_S$ such that $\xi(e''_S) = \xi(e_S)$.*

The notation $Gr \cong Gr''$ is used to indicate that Gr and Gr'' are isomorphic. By Definitions 3 and 4, subgraph isomorphism between two graphs is defined as follows:

Definition 5. *A graph Gr is subgraph isomorphic to a graph Gr'' , if there exist an injective function $f : V \rightarrow V''$ and a subgraph $G' \subseteq G''$ such that Gr and Gr' are isomorphic (i.e., $Gr \cong Gr' \subseteq Gr''$).*

3.2.2 Neighborhood Graph

We are using *graph isomorphism* and the *maximal common subgraph* [3] to find the corresponding nodes in two consecutive RAGs. In other words, we track the corresponding regions based on graph processing. These algorithms are conceptually simple, but have a high computational complexity. In order to address this, we decompose a RAG

into its neighborhood graphs ($G_N(v)$) which are subgraphs of RAG as follows:

Definition 6. $G_N(v)$ is a neighborhood graph of a given node v in a RAG, if for any nodes $u \in G_N(v)$, u is an adjacent node of v , and has one edge such that $e_S = (v, u)$.

According to Definition 6, a neighborhood graph of a node consists of the node itself and its adjacent nodes. Figure 3.4 shows some examples of the neighborhood graphs of $Gr(f_{14})$ in Figure 3.3 (c). $G_N(v_{14-44})$ and $G_N(v_{14-52})$ are the neighborhood graphs for nodes v_{14-44} and v_{14-52} , respectively. As seen in the figure, each neighborhood graph consists of the node itself and its adjacent nodes. Although the neighborhood graph is a very simple graph where the depth is equal to 1, it can still represent the relationships among the nodes.

3.2.3 Graph-based Tracking

Let \mathbb{G}_N^m and \mathbb{G}_N^{m+1} be sets of the neighborhood graphs in m^{th} and $(m+1)^{th}$ frames respectively. For each node v in m^{th} frame, the goal is to find the corresponding node v' in $(m+1)^{th}$ frame. In order to decide whether two nodes are corresponding, we use the neighborhood graphs in Definition 6. Therefore, finding the corresponding two nodes, v and v' , is converted to find the corresponding two neighborhood graphs, $G_N(v)$ in \mathbb{G}_N^m , and $G_N(v')$ in \mathbb{G}_N^{m+1} , in which $G_N(v')$ is isomorphic or most similar to $G_N(v)$.

First, we find the neighborhood graph in \mathbb{G}_N^{m+1} , which is isomorphic to $G_N(v)$. Second, if we cannot find any isomorphic graph in \mathbb{G}_N^{m+1} , we find the most similar neighborhood graph to $G_N(v)$ using the following equation which computes the similarity between two neighborhood graphs (i.e., $G_N(v)$ and $G_N(v')$).

$$\delta(G_N(v), G_N(v')) = \frac{|G_C|}{\min(|G_N(v)|, |G_N(v')|)} \quad (3.1)$$

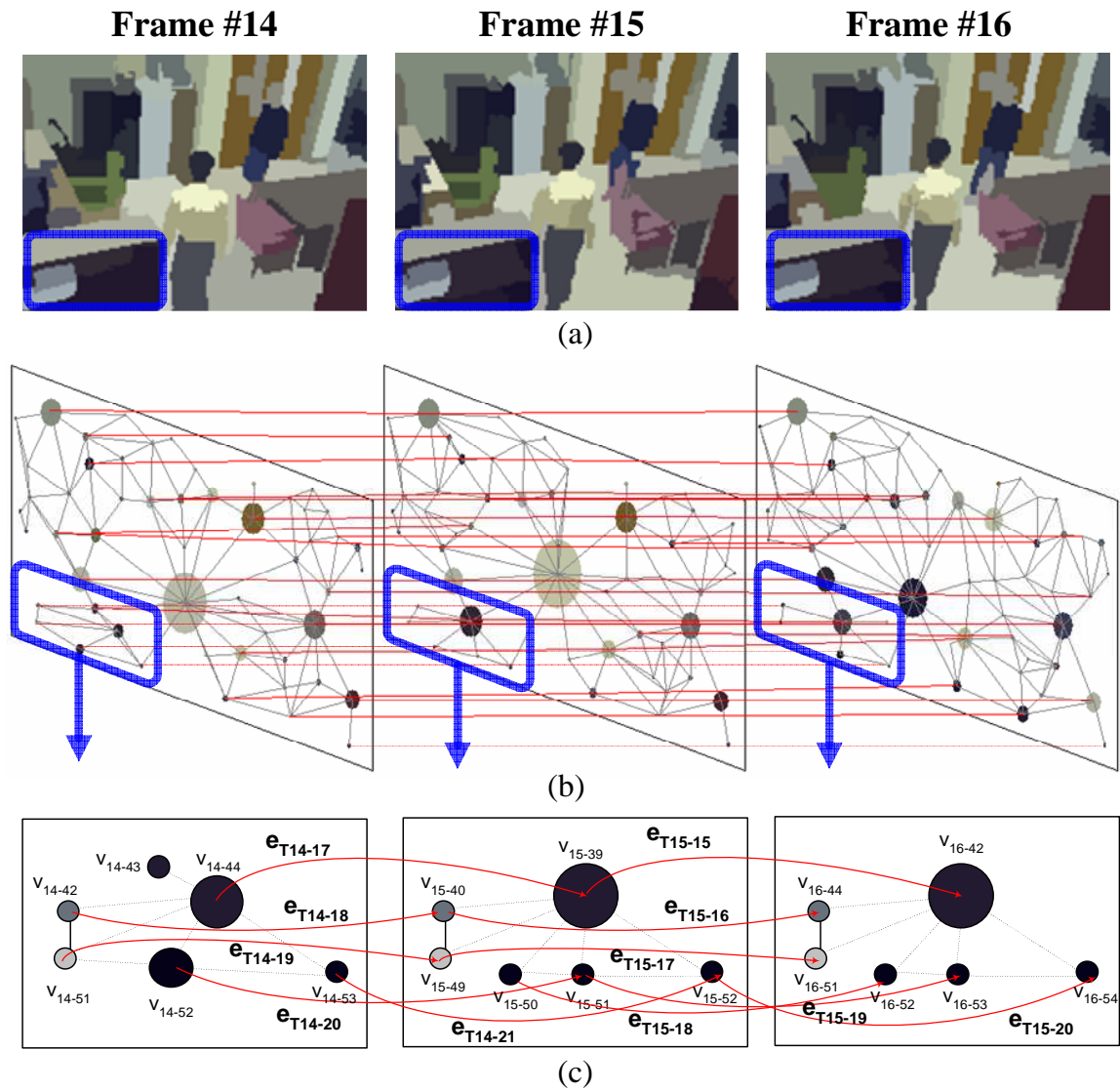


Figure 3.3. Visualization of STRG for frame #14 – #16: (a) Region segmentation results, (b) STRG, and (c) Enlarging a part of STRG in (b).

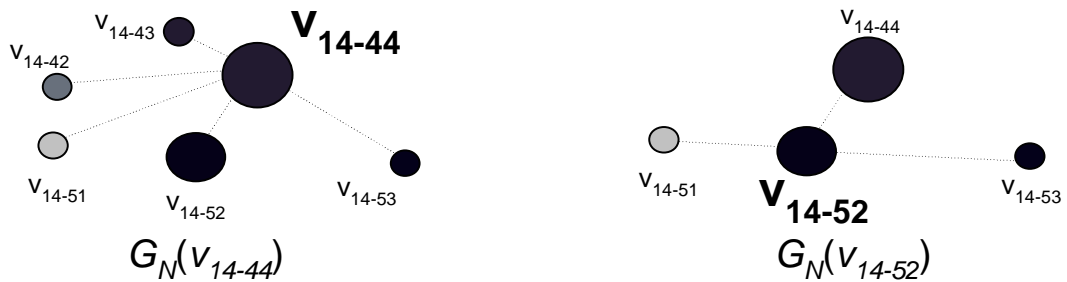


Figure 3.4. Two examples of neighborhood graphs for nodes v_{14-44} and v_{14-52} .

where $|\bullet|$ denotes the number of nodes of a graph, and G_C is the maximal common subgraph between $G_N(v)$ and $G_N(v')$. The well-known algorithms computing G_C are based on the *maximal clique detection* [52] or the *backtracking* [53]. We exploit the idea of maximal clique detection to compute $|G_C|$ since the neighborhood graph can be easily transformed into a maximal clique. For maximal clique and other graph matching operations, we need to compare two nodes or two edges using their attribute affinities. For these comparisons, we use *Mahalanobis* metric to consider the weights of attributes. The node affinity (W_V) between A_{V^i} and A_{V^j} , spatial edge affinity (W_{E_S}) between $A_{E_S^i}$ and $A_{E_S^j}$, and temporal edge affinity (W_{E_T}) between $A_{E_T^i}$ and $A_{E_T^j}$ are as follows:

$$\begin{aligned} W_V(i, j) &= (A_{V^i} - A_{V^j})^T \Sigma^{-1} (A_{V^i} - A_{V^j}) \\ W_{E_S}(i, j) &= (A_{E_S^i} - A_{E_S^j})^T \Sigma^{-1} (A_{E_S^i} - A_{E_S^j}) \\ W_{E_T}(i, j) &= (A_{E_T^i} - A_{E_T^j})^T \Sigma^{-1} (A_{E_T^i} - A_{E_T^j}) \end{aligned} \quad (3.2)$$

Σ^{-1} gives the weights of attributes whose values are predetermined depending on the content of video. Since we do not consider correlations among features, Σ^{-1} is a diagonal matrix.

In order to find G_C between $G_N(v)$ and $G_N(v')$, we first construct an *association graph* which is formed by creating nodes from each compatible pair of two nodes in $G_N(v)$ and $G_N(v')$, and inserting edges between nodes if they have equivalent relationships with the nodes v and v' . Then, G_C is obtained by finding the maximal clique in the association graph. Figure 3.5 illustrates the pseudo code of finding the maximal common subgraph from two given neighborhood graphs (Algorithm 1).

The higher the value of δ in Equation (3.1), the more similarity between $G_N(v)$ and $G_N(v')$. For $G_N(v)$ in \mathbb{G}_N^m , $G_N(v')$ is the corresponding neighborhood graph in \mathbb{G}_N^{m+1} , whose δ with $G_N(v)$ is the largest among the neighborhood graphs in \mathbb{G}_N^{m+1} , and greater than a certain threshold value (T_{sim}); i.e. $\arg_{G_N(v') \in \mathbb{G}_N^{m+1}} \max \delta(G_N(v), G_N(v'))$.

Algorithm 1: Maximal Common Subgraph

Input: two neighborhood graphs: $G_N(v)$ and $G_N(v')$
Output: the set of nodes V_C

- 1: V_A = a set of nodes of the association graph between $G_N(v)$ and $G_N(v')$;
- 2: $k = \min(|G_N(v)|, |G_N(v')|) + 1$;
- 3: **do**
- 4: $k = k - 1, V_C = \emptyset$;
- 5: **MCS** (V_C, V_A);
- 6: **until** ($|V_C| = k$);
- 7: **return** V_C ;

8: **Procedure** *MCS* (X, Y)

- 9: **begin**
- 10: $Y_1 = \{v \in V_A : v \text{ is connected to all nodes in } Y\}$;
- 11: $Y_2 = \{v \in Y - X : v \text{ is connected to all nodes in } X\}$;
- 12: **if** $Y_1 = \emptyset$ or $Y_2 = \emptyset$ **then**
- 13: **return** X ;
- 14: **else**
- 15: select a node v_y which is connected to all nodes in X ; $V_C = V_C \cup \{v_y\}$;
- 16: **MCS** ($X \cup \{v_y\}, Y$) \cup **MCS** ($X, Y - \{v_y\}$);
- 17: **end if**
- 18: **end Procedure**

Figure 3.5. Algorithm 1: Maximal Common Subgraph.

In this way, we find all possible pairs of corresponding neighborhood graphs (eventually corresponding nodes) from \mathbb{G}_N^m to \mathbb{G}_N^{m+1} . Figure 3.6 provides the pseudo code of graph-based tracking (Algorithm 2).

The complexity of the graph-based tracking algorithm can be analyzed as follows. First, to analyze Algorithm 1, let K_1 and K_2 be the number of nodes of two neighborhood graphs, $G_N(v)$ and $G_N(v')$, respectively. According to [54], the worst case (in number of states) to compute Algorithm 1 is:

$$S = (K_2 + 1)(K_2) \dots (K_2 - K_1 + 2) = \frac{(K_2 + 1)!}{(K_2 - K_1 + 1)!} \quad (3.3)$$

When $K_1 = K_2$, the complexity of Equation (3.3) is reduced to $O(K \cdot K!)$ where K is the number of nodes in a neighborhood graph. Therefore, when I and J are the

Algorithm 2: Graph-Based Tracking

Input: two Region Adjacent Graphs: $Gr(f_m)$ and $Gr(f_{m+1})$

Output: temporal edge set: E_T

```

1: let  $E_T = \emptyset$ ;
2: for each  $v \in Gr(f_m)$  do
3:   let  $g = G_N(v)$ ,  $maxSim = 0$ ,  $Sim = 0$ ,  $maxNode = null$ ;
4:   for each  $v' \in Gr(f_{m+1})$  do
5:     let  $g' = G_N(v')$ ;
6:     if  $g$  and  $g'$  are isomorphic then
7:        $E_T = E_T \cup \{e_T(v, v')\}$ ; break;
8:     else  $Sim = \delta(g, g')$  by Equation (3.1);
9:     end if
10:    if  $Sim > maxSim$  then
11:       $maxNode = v'$ ;  $maxSim = Sim$ ;
12:    done
13:    if no isomorphic graph of  $v$  and  $maxSim > T_{sim}$  then  $E_T = E_T \cup \{e_T(v, maxNode)\}$ ;
14:  done
15: return  $E_T$ ;

```

Figure 3.6. Algorithm 2: Graph-based Tracking.

number of nodes of two RAGs, $Gr(f_m)$ and $Gr(f_{m+1})$, the total complexity to compute Algorithm 2 is $O(I \cdot J \cdot K \cdot K!)$. However, since two consecutive frames are not much different in a same shot, the search area in $Gr(f_{m+1})$ for each node in $Gr(f_m)$ is very limited. In other words, for each node v in $Gr(f_m)$ we do not have to travel all nodes in $Gr(f_{m+1})$ to find its corresponding node. Consequently, the complexity can be reduced to $O(I \cdot K \cdot K!)$. In addition, in a real situation the neighborhood graph has a small number of nodes (i.e., K is 4 to 8), and is much simpler than general graphs. Therefore, the overall complexity of the graph-based tracking algorithm can be reduced significantly.

3.3 Experimental Setup of This Dissertation

In order to assess the proposed techniques in this dissertation, we performed a set of experiments. For the experiments, we prepared two different data sets; the real data

set and the synthesized data set. The real data set has various video clips having different contents. The data set will be used for the evaluations of the proposed techniques in this dissertation. On the other hand, the synthesized data set is generated to evaluate the performance of the proposed algorithms. Java 2 SDK 1.4.2 and JMF 2.1e are used for the experiments on an Intel Pentium IV 2.6 GHz CPU computer for the rest of experiments in this dissertation.

3.3.1 Real Video Data Set

To perform a set of experiments in this dissertation, we prepare a real video data set that is used for verifying the proposed video data model (see Section 3.4), shot boundary detection (see Section 4.3.2), video data mining (see Section 5.4.1), and query processing (see Section 7.6).

The real video data set consists of two groups. The first group has the produced videos with various contents, and the second has the surveillance videos captured by a video camera without any pre-defined moving patterns. Table 3.1 shows the description of the real data set used for the experiments in this dissertation. For the produced videos, we choose eight video clips from [55], which have various types of contents including drama, animation, talk show, movie, sport, and news. Each video clip has different types of shot changes, such as abrupt and gradual changes. The surveillance videos are taken from inside of building (Surv 1 and 2) and outside for traffic scenes (Traffic 1 and 2). As seen in Table 3.1, while the produced videos have various types of shot changes with simple object motions, the surveillance videos have complex patterns of object motions without any shot change. Our video data set lasts about 47 hours that is long enough to evaluate the proposed algorithms in this dissertation.

Table 3.1. Description of real data set

Group	Type	No.	Name	Duration (hh:mm:ss)	Abrupt Changes	Gradual Changes	Total Changes	Complexity of Object Motion
Produced Video	Drama	1	Silk Stalking	00:10:24	91	4	95	Very Low
	Cartoon	2	Scooby Dog Show	00:11:38	101	5	106	Low
	Sitcom	3	Friends	00:10:22	113	3	116	Low
	Talk Show	4	Divorce Court	00:11:11	154	6	160	Very Low
	Si-Fi	5	Star Trek	00:12:27	109	2	111	Low
	Soap Opera	6	All My Children	00:05:44	48	2	50	Very Low
	Sport	7	Tennis	00:03:53	114	5	119	Very High
	News	8	Local (ABC)	00:03:53	176	10	186	Low
Surveillance Video	Indoor	9	Surv 1	40:38:02	0	0	0	High
	Indoor	10	Surv 2	04:12:24	0	0	0	High
	Outdoor	11	Traffic 1	00:15:08	0	0	0	Very High
	Outdoor	12	Traffic 2	00:12:48	0	0	0	Very High
Total				46:54:55	906	37	943	

3.3.2 Synthesized Data Set

In order to demonstrate the performance of the proposed video data mining techniques (see Section 5.4.1), indexing (see Section 6.5), and query processing (see Section 7.6), the synthesized data are generated and used for the experiments. Since a moving object extracted from a video is a type of time-series data, we generate a set of new data by combining the Pelleg data set [56] which is widely used to test clustering algorithms, with the Vlachos data set [57] which is 2-D time-series data with noise. Our synthesized data is generated as follows.

1. We design 48 moving patterns: vertical (12), horizontal (12), diagonal (8) and U-turn (16). Each pattern has different sizes of objects and various time lengths.
2. We generate the time-series data using the approach described in [56], which is normally distributed with $\sigma = 5$.

3. We add noise to each data point based on the method presented in [57]. We add six different noise levels ranging from 5% to 30%.
4. The generated data is converted to a temporal subgraph format using the nodes, temporal edges, and their attributes that will be explained in Section 4.2.1 (see Definition 7).

Figure 3.7 shows the example of generated data set when the noise level is 5%. The data set has 48 moving patterns including 12 vertical, 12 horizontal, 8 diagonal, and 16 U-turn directions. Each moving pattern has different sizes of objects and various time lengths.

3.4 Experimental Results of STRG Generation

According to the definitions of STRG generation mentioned in this chapter, we developed the application for STRG generation using Java. The input can be either a real video stream from video camera or saved video files. Figure 3.8 shows the screenshot of developed STRG generation. The application can generate STRG for the input video. It has several display panels for real video frame, region segmentation, RAG, and STRG for three consecutive frames. The output is saved as the format of STRG for further processing.

In order to evaluate the performance of STRG generation, we generate STRGs for videos in Table 3.1. We investigate the size of STRG, and the average processed frames during STRG generation. Table 3.2 shows the performance of STRG generation for real video data set. As seen in table, the size of generated STRG is reduced significantly comparing to the size of video. Concerning to the efficiency of STRG generation, the last column of Table 3.2 clearly shows that the proposed STRG generation can process over 12 frames per second (FPS) regardless of the types of videos.

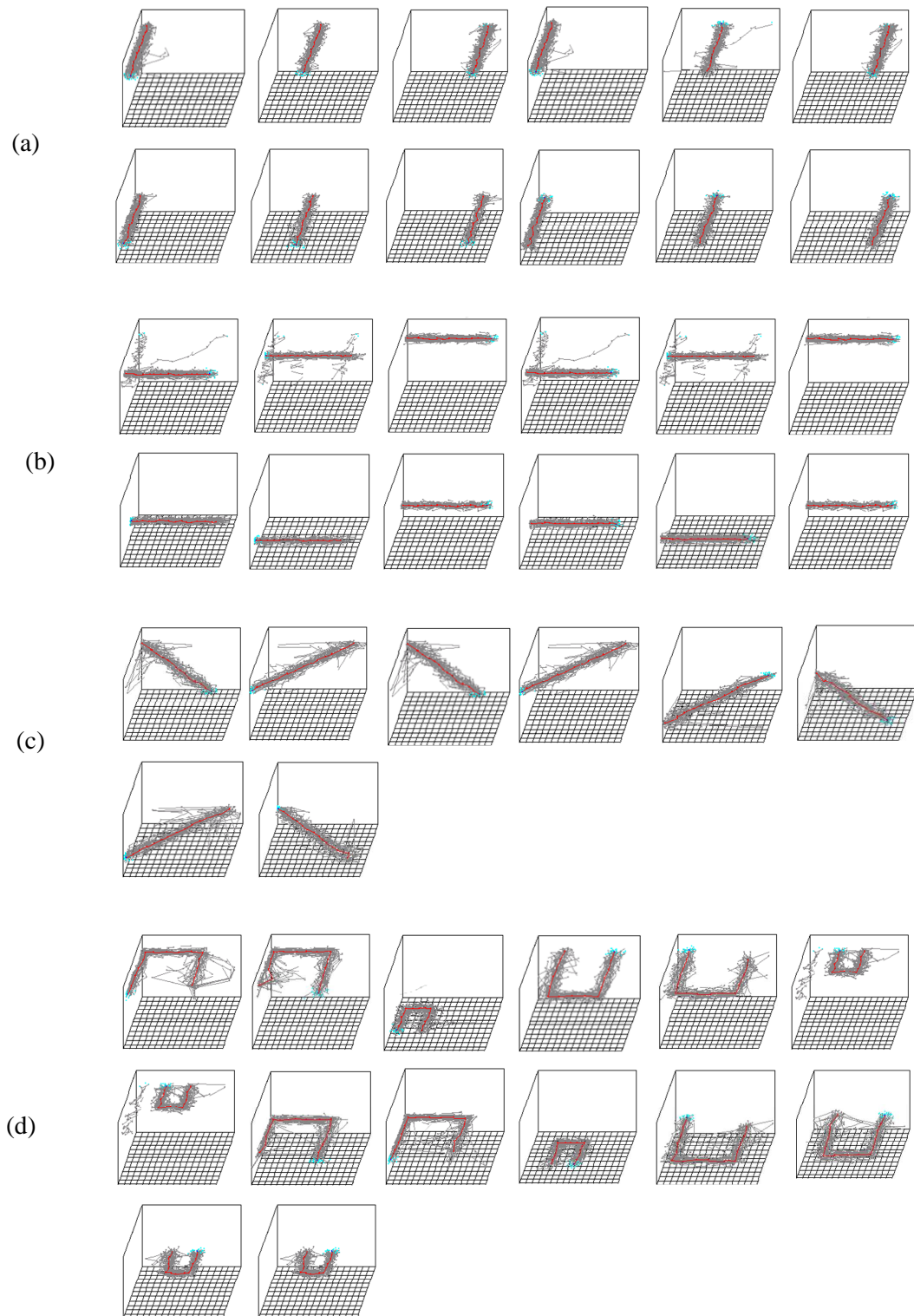


Figure 3.7. Visualization of synthesized data set when noise level is 5%: (a) Vertical (12), (b) Horizontal (12), (c) Diagonal (8), and (d) U-turn (16).

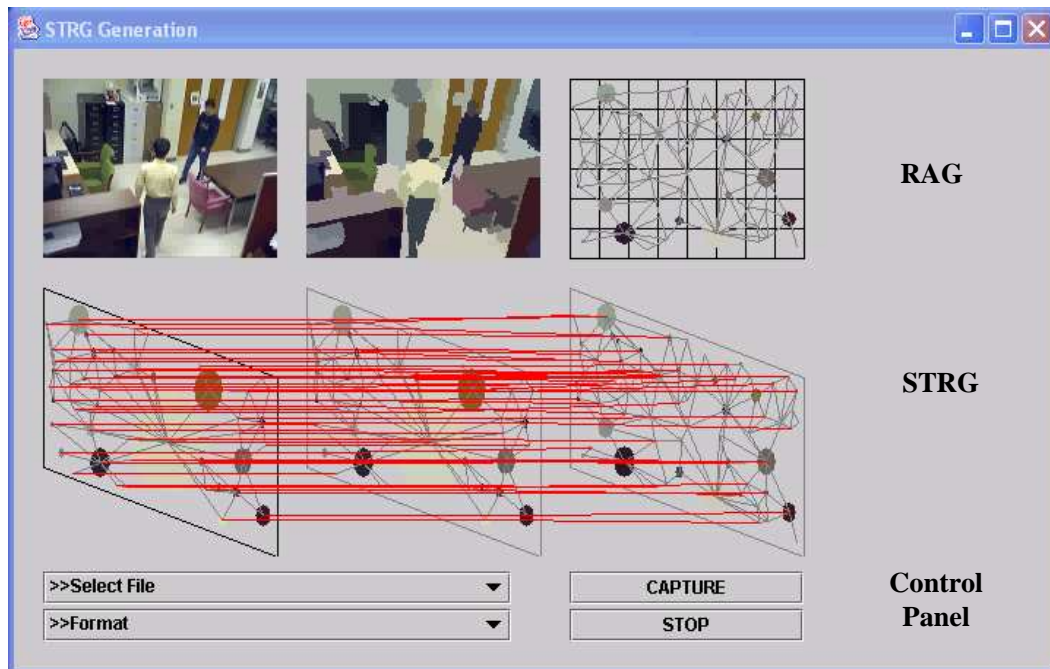


Figure 3.8. Screenshot of STRG generation.

Table 3.2. Performance of STRG generation

Video No.	Duration (hh:mm:ss)	STRG Size	Avg. Processed FPS
1	00:10:24	1.1 MB	13.4 FPS
2	00:11:38	1.3 MB	12.5 FPS
3	00:10:22	2.1 MB	12.3 FPS
4	00:11:11	1.0 MB	13.1 FPS
5	00:12:27	1.7 MB	12.3 FPS
6	00:05:44	0.9 MB	13.5 FPS
7	00:14:20	2.7 MB	12.0 FPS
8	00:30:27	4.3 MB	13.1 FPS
9	40:38:02	72.2 MB	14.1 FPS
10	04:12:24	6.4 MB	13.8 FPS
11	00:15:08	1.4 MB	13.1 FPS
12	00:12:48	1.2 MB	12.9 FPS

3.5 Summary

In this chapter, we proposed a new graph-based data structure to model video data. Several researches have shown that a graph can be a better candidate for modeling semantically rich and complicated video data. However, there are few methods that consider the temporal feature of video data, which is a distinguishable and representative characteristic when compared with other multimedia. In order to consider the temporal feature effectively and efficiently, we proposed a new graph-based data structure called *Spatio-Temporal Region Graph* (STRG). Unlike existing graph-based data structures which provide only spatial features, the proposed STRG further provides temporal features, which represent temporal relationships among spatial objects. First, we segment each frame into a number of homogeneous color regions using region segmentation algorithm. From the segmented region, we build a *Region Adjacency Graph* for each frame, where a set of nodes indicate the segmented region and a set of spatial edges represent the spatial relationships among the nodes, i.e., adjacency. After we have RAGs for the entire video, we construct an STRG of the video by adding a set of temporal edges that represent temporal characteristic. To add a set of temporal edges, we find the corresponding nodes between two consecutive RAGs. However, the graph-based approach requires a high computational complexity. In order to address this, we introduce a neighborhood graph of a node that is subgraph of RAG. Since a neighborhood graph is a simple graph, the computations for generating STRG are reduced significantly.

CHAPTER 4

STRG PARSING

In this chapter, we introduce several graph-based video parsing techniques such as shot boundary detection, object detection, and background modeling. These techniques of video parsing based on a graph provide more efficient and accurate video processing. An STRG is segmented into a number of smaller pieces corresponding to shots, which is the very first step in video processing. The segmented STRGs are used for the basic processing units such as shots or scenes. An STRG needs to be segmented into smaller units since it is a huge graph that requires heavy computation for further processing. Then, the small pieces of STRG are decomposed into *Object Region Graphs* (ORGs) and *Background Graphs* (BGs). In order to find ORGs from STRG, we extract all liner graphs whose nodes are connected by only temporal edges. Due to the limitation of region segmentation algorithm, a single moving object can be divided into several ORGs. To address this, the ORGs belonging to a single object are merged into *Object Graph* (OG). In addition, we model a background of STRG, called BG, since there are lots of redundant information in the background. The redundant BGs are eliminated to reduce the index size and search time.

The rest of this chapter is organized as follows. In Section 4.1, we describe the graph-based shot boundary detection technique. For the detection, we introduce a Graph Similarity Measure. Section 4.2 shows the way that STRG is decomposed into a number of OGs and BG. The performance study on real video data set is reported in Section 4.3. Section 4.4 presents the summary of the chapter.

4.1 Graph-based Shot Boundary Detection

An STRG needs to be segmented into smaller units since it is a huge graph that requires very high computation for further processing. A *shot*, defined as a collection of interrelated consecutive frames taken continuously by a single camera operation, is meaningful to serve as an elementary unit. Shot boundaries can be detected by employing a metric to measure the similarity or difference between two consecutive frames. This is based on the fact that the content remains nearly the same in a shot. If the similarity is more than a certain threshold, two frames are considered to be in a same shot. In our case, we exploit graph matching to measure the similarity between two consecutive frames (RAGs). Typically shot transitions can be classified into two groups: abrupt change (i.e., hard-cut) and gradual changes (i.e., dissolve, fade-in or fade-out). Our proposed shot boundary detection using graph matching can detect not only abrupt change, but also gradual changes in video.

4.1.1 Graph Similarity Measure

Shot boundaries in an STRG can be revealed as the temporal graph similarity between two consecutive RAGs by using the similarity between two neighborhood graph δ in Equation (3.1). In other words, if two RAGs are similar to each other, almost all of the nodes in one RAG are connected to the other RAG by E_T . The two frames corresponding to the two RAGs belong to the same shot. Otherwise, two frame corresponding the RAGs become a shot boundary.

Now, we measure the similarity between two RAGs using δ in Equation (3.1). Let $Gr(n-1)$ and $Gr(n)$ be RAGs of $(n-1)^{th}$ and n^{th} frame, respectively. *Graph Similarity Measure*, GSM between $Gr(n-1)$ and $Gr(n)$ can be defined as:

$$GSM(Gr(n-1), Gr(n)) = \frac{1}{\max(I, J)} \sum_{i=1}^I \sum_{j=1}^J \delta(G_N(v_i), G_N(v_j)) \quad (4.1)$$

where I and J are the number of nodes of $Gr(n - 1)$ and $Gr(n)$, respectively. However, GSM still requires expensive computation since it computes all pairs of neighborhood graphs between $Gr(n - 1)$ and $Gr(n)$. In order to reduce the computation time, we consider the neighborhood graphs within a certain range in $Gr(n)$. Since two consecutive RAGs in a shot are not much different, for each neighborhood graph in $Gr(n - 1)$ we do not have to compute the similarities (δ) with all neighborhood graphs in $Gr(n)$. Instead, we consider only neighborhood graphs within a certain range in $Gr(n)$, which can reduce the computation time of GSM . A simplified version of GSM in Equation (4.1), which is referred to GSM_{sim} , can be defined as:

$$GSM_{sim}(Gr(n - 1), Gr(n)) = \frac{1}{I} \sum_{i=1}^I \sum_{j'=1}^{J'} \delta(G_N(v_i), G_N(v_{j'})) \quad (4.2)$$

where $v_{j'}$ ($j' = 1, \dots, J'$) is the node in $Gr(n)$ such that the spatial distance between (x, y) of v_i and (x', y') of $v_{j'}$ is less than a given value. We will show the efficiency of GSM_{sim} in Section 4.3.

4.1.2 Shot Boundary Detection

We use GSM_{sim} in Equation (4.2) to detect shot boundaries, since each frame is represented as a graph (RAG) and it considers the spatial relationships of a RAG unlike existing similarity measures such as pair-wise pixel comparison [58] or color histogram [59]. Shot transitions can be classified into two groups: abrupt change (i.e. hard-cut) and gradual change (i.e. dissolve, fade-in and fade-out).

4.1.2.1 Abrupt Changes

Detecting abrupt change is relatively easier. If GSM_{sim} is more than a certain threshold value (T_{cut}), the two frames corresponding to the two RAGs are considered to be in a same shot. Otherwise, the two frames are considered to be in two different shots.

The main problem is how to select the threshold value (T_{cut}). We used several threshold values in our technique. However, these values are not based on a pre-selected training set or previously collected statistics. As seen in Figure 4.1 which has 14 actual cuts, the number of detected cuts is not sensitive to the threshold value. For example, 14 cuts are detected when $T_{cut} = (0.1, 0.2, 0.3)$, and 17 cuts are detected when $T_{cut} = 0.4$. Therefore, we can easily select an appropriate T_{cut} without doing any additional processing.

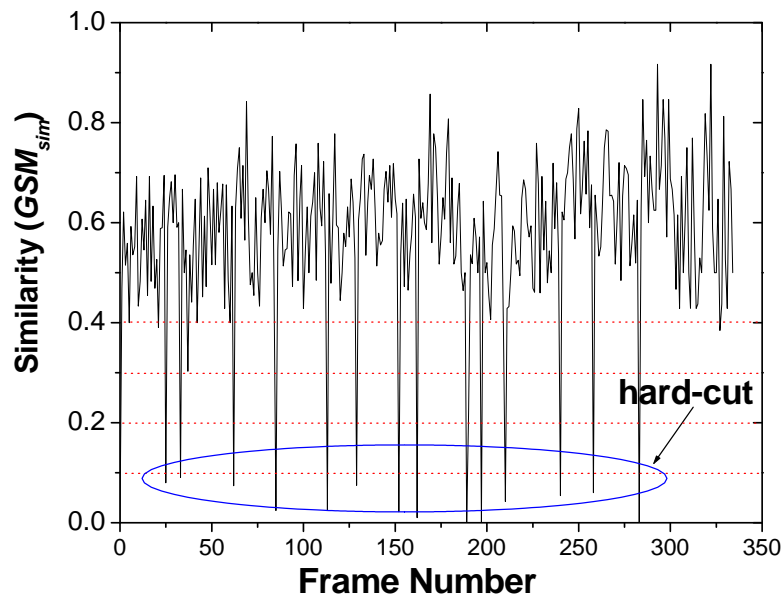


Figure 4.1. Shot detection results for abrupt changes.

4.1.2.2 Gradual Changes

As the name implies, a gradual change occurs throughout a number of frames. During a gradual change, GSM_{sim} generates lower similarity values regardless of the types of transitions since the similarity is based on matched nodes and edges which are rarely found in gradually changed frames. Therefore, the proposed method can detect starting and ending frames of gradual changes. Figure 4.2 shows some examples of shot

boundaries having gradual changes. An optimal threshold value (T_{cut}) for gradual change can be selected indubitably. As seen in the figure, 0.2, 0.3, 0.4 or 0.5 generates same boundaries.

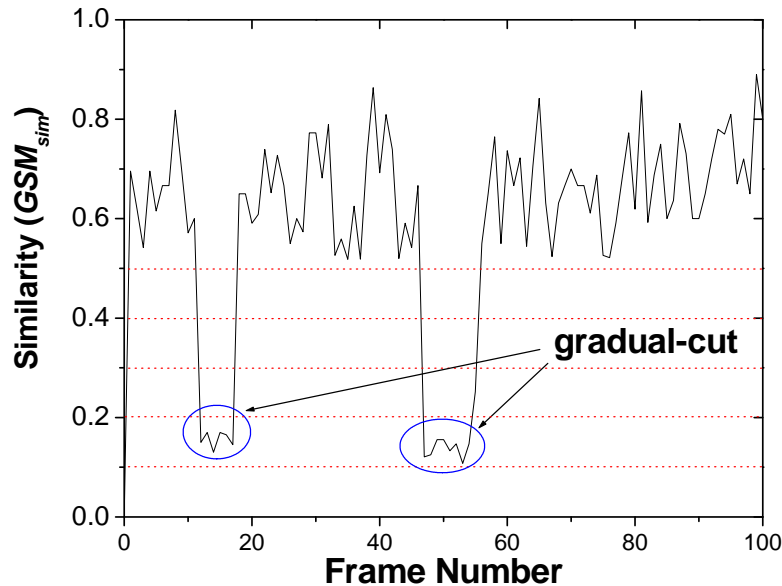


Figure 4.2. Shot detection results for gradual changes.

Unlike existing SBD techniques [55, 60] which use only feature values to discriminate the difference between two frames, the proposed SBD considers the relationships among segmented regions as well. Therefore, a gradual change is detected without any complicated additional processing, and the optimal threshold value (T_{cut}) can be selected reasonably.

4.2 STRG Decomposition

In this section, each segmented STRG (*strg*, in the following discussion), which corresponds to a shot, is decomposed into *Object Region Graphs* (ORGs) and *Background Graphs* (BGs). The ORGs belonging to a single object are merged into an *Object Graph*

(OG), and the redundant BGs are eliminated to reduce the searching area and the size of index.

4.2.1 Object Region Graph

One of the key characteristics of video data is that each spatial feature needs to be represented as a temporal feature, since it may change over time. In order to capture the temporal feature from an STRG, we define a temporal subgraph which is a set of sequential nodes connected to each other by a set of temporal edges (E_T) as follows:

Definition 7. Given a graph $Gst = \{V, E_S, E_T, \nu, \xi, \tau\}$, a temporal subgraph of Gst is a graph $Gst' = \{V', E'_S, E'_T, \nu', \xi', \tau'\}$ such that

- $V' \subseteq V$, $E'_S = E_S \cap (V' \times V')$ and $E'_T = E_T \cap (V' \times V')$
- ν' , ξ' and τ' are the restrictions of ν , ξ and τ to V , E_S and E_T , respectively, i.e.

$$\nu'(v) = \begin{cases} \nu(v) & \text{if } v \in V', \\ \text{undefined} & \text{otherwise,} \end{cases}$$

$$\xi'(e_S) = \begin{cases} \xi(e_S) & \text{if } e_S \in E'_S, \\ \text{undefined} & \text{otherwise,} \end{cases}$$

$$\tau'(e_T) = \begin{cases} \tau(e_T) & \text{if } e_T \in E'_T, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The notation $Gst' \subseteq_T Gst$ is used to indicate that Gst' is a temporal subgraph of Gst . In Definition 8, when the spatial edge set E_S is empty, the temporal subgraph Gst' can represent a trajectory of tracked regions. We refer to this trajectory as an *Object*

Region Graph (ORG). An ORG is a special case of temporal subgraph. An ORG has several characteristics as follows:

- It is a type of time-varying data since the temporal edges are constructed based on time. This is an important feature of an ORG that distinguishes it from other graphs.
- Unlike existing video indexing techniques [61] which consider only detected objects, an ORG considers spatial and temporal relationships between nodes.
- It is a linear graph in which each node has only temporal edges, E_T .

Thus, it is more efficient to process matching and indexing.

4.2.2 Object Graph

Due to the limitations of existing region segmentation algorithms, different colors of regions belonging to a single object cannot be detected as one region. Moreover, even a same color region may be segmented into two different regions because of small amount of illumination changes. This can be addressed by region merging. For instance, a body of a person may consist of several regions such as head, upper body and lower body. Figure 4.3 (a) shows an object (a person) that is segmented into four regions over three frames. For each corresponding pair of regions (nodes), we build four ORGs, i.e. (v_1, v_5, v_9) , (v_2, v_6, v_{10}) , (v_3, v_7, v_{11}) , and (v_4, v_8, v_{12}) seen in Figure 4.3 (b). Since these belong to a single object, it is better to merge them to one. For convenience, we refer to the merged ORGs as an *Object Graph* (OG).

In order to merge ORGs into OG, we first show how to merge two subgraphs in Theorem 1.

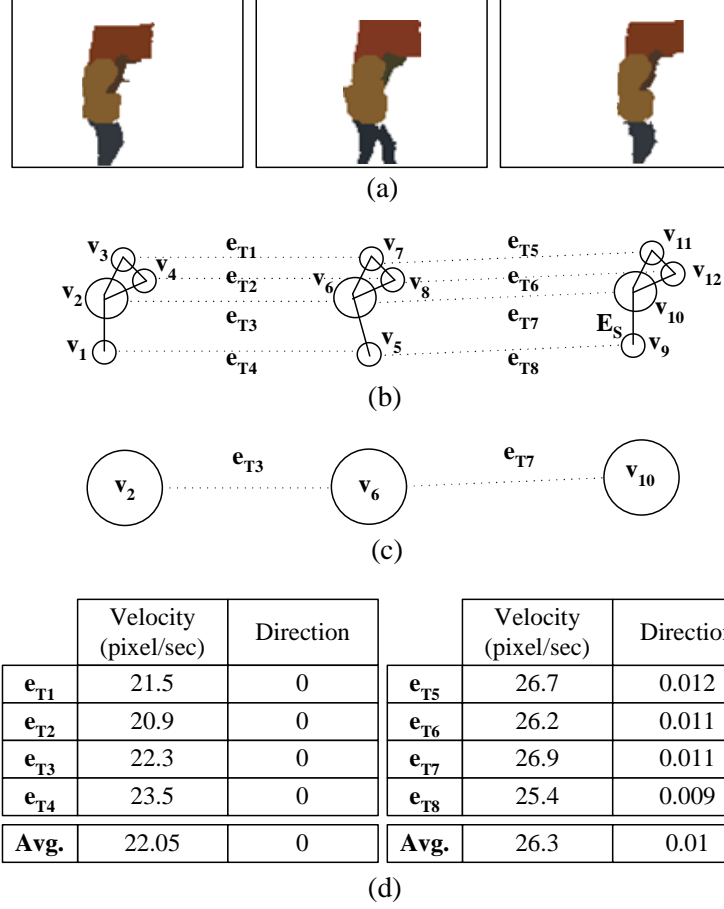


Figure 4.3. (a) Sample object segmented several parts, (b) Example of ORGs for (a), (c) Merged OG, and (d) Results of temporal edges.

Theorem 1. For given subgraphs G_1, G_2, G_1'' and G_2'' , if G_1 is subgraph isomorphic to G_1'' , and G_2 is subgraph isomorphic to G_2'' , then $G_1 \cup G_2$ is subgraph isomorphic to $G_1'' \cup G_2''$.

Proof. By the Definition 5,

\exists subgraph $G_1' \subseteq G_1'' \ni f_1$ is a graph isomorphism from G_1 to G_1' ,

\exists subgraph $G_2' \subseteq G_2'' \ni f_2$ is a graph isomorphism from G_2 to G_2' , and

$G_1' \cup G_2' \subseteq G_1'' \cup G_2''$.

$f_1 \circ f_2$ is a graph isomorphism from $G_1 \cup G_2$ to $G_1' \cup G_2'$, because

(1) $\nu_1(\nu_2(v)) = \nu'_1(\nu'_2(f_1(f_2(v)))) \forall v \in G_1 \cup G_2 \Rightarrow$

$$\nu_1 \circ \nu_2(v) = \nu'_1 \circ \nu'_2(f_1 \circ f_2(v)) \forall v \in G_1 \cup G_2.$$

(2) For any edge $e_S = (v_1, v_2)$ in $G_1 \cup G_2$

\exists an edge $e'_S = (f_1 \circ f_2(v_1), f_1 \circ f_2(v_2))$ in $G'_1 \cup G'_2$ such that $\xi(e'_S) = \xi(e_S)$, and

for any edge $e'_S = (v'_1, v'_2)$ in $G'_1 \cup G'_2$,

\exists an edge $e_S = (f_2^{-1} \circ f_1^{-1}(v'_1), f_2^{-1} \circ f_1^{-1}(v'_2))$ in $G_1 \cup G_2$ such that $\xi(e'_S) = \xi(e_S)$.

(1) and (2) satisfy the conditions of Definition 4 for a graph isomorphism between $G_1 \cup G_2$ to $G'_1 \cup G'_2$.

Since f_1 and f_2 are injective functions, $f_1 \circ f_2$ is an injective function too. Therefore, $G_1 \cup G_2$ is subgraph isomorphic to $G''_1 \cup G''_2$. \square

Theorem 1 states that two pairs of isomorphic subgraphs can be merged into one pair of isomorphic subgraphs. Suppose that ORG^s and ORG^t are two object region graphs which have isomorphic subgraphs with respect to the neighborhood graph of each node. Let v^s and v^t be nodes in ORG^s and ORG^t , respectively. Then, two neighborhood graphs $G_N(v^s)$ and $G_N(v^t)$ are obtained according to Definition 7. $G_N(v^s)$ and $G_N(v^t)$ have an isomorphic subgraph $G'_N(v^s)$ and $G'_N(v^t)$, respectively, because a temporal edge in an ORG is constructed by an isomorphic subgraph. By Theorem 1, $G'_N(v^s) \cup G'_N(v^t)$ is an isomorphic subgraph of $G_N(v^s) \cup G_N(v^t)$. After repeating this operation to all corresponding nodes in ORG^s and ORG^t , $ORG^s \cup ORG^t$ which is an OG is obtained eventually.

Since there are many types of video objects, which cannot be handled by a single algorithm, we assume in this paper that a target object has the following two conditions:

- An object consists of one or more ORGs that have the same moving direction and speed.

- Each ORG in a single object is adjacent to at least one other ORG (except an object having a single ORG).

For example, an object that moves and stops for a while can be detected as a moving object, while an object that does not move at all cannot be detected. The still object becomes a part of the background. We can also detect multiple objects that move together as long as they are not adjacent to each other. In order to merge two ORGs that belong to a single object, we consider the attributes (i.e. velocity and moving direction) of a temporal edge set (E_T). If two ORGs have the same values of velocity and moving direction, these can be merged into one. For example, in Figure 4.3 (d) the four temporal edges, i.e., e_{T_1} , e_{T_2} , e_{T_3} and e_{T_4} , have almost the same values for velocity (ω) and moving direction (π). The temporal edges and the corresponding nodes are merged into one temporal edge (e_{T_3}) and two nodes (v_2 and v_6). In Figure 4.3 (c), four more ORGs are merged into a single OG eventually, i.e. (v_2, v_6, v_{10}).

4.2.3 Background Graph

A video frame usually consists of two areas: foreground and background. The foreground is the main target on which a camera focuses, and the background is a supporting area that does not change significantly over time. Foreground/background separation is a fundamental research area in a content-based video processing [62, 63]. Moreover, from a prospect of graph-based video indexing, the separation is important because the size of index can be drastically reduced by eliminating the redundant backgrounds. Generally speaking, it is sufficient to maintain only one *Background Graph* (BG) for each segmented shot (*strg*) where there is little difference in the background over the frames.

We subtract all OGs from an *strg*, then the remaining graphs are considered as background graphs. In order to obtain a single BG for each shot, we overlap all remaining

graphs using temporal edges (E_T). The main idea of building a BG is inspired by adaptive background modeling and mosaic techniques [63].

The benefits of the *strg* decomposition can be summarized as follows:

1. It can reduce index size by eliminating BGs.
2. It can reduce searching time because the searching area is reduced to only OGs.

For convenience, we show Figure 3.3 (c) in Figure 4.4 (a) again. Figure 4.4 (b) shows the example of constructed BG from Figure 4.4 (a). The connected nodes by temporal edges become a single node, and their corresponding spatial edges are added to the BG.

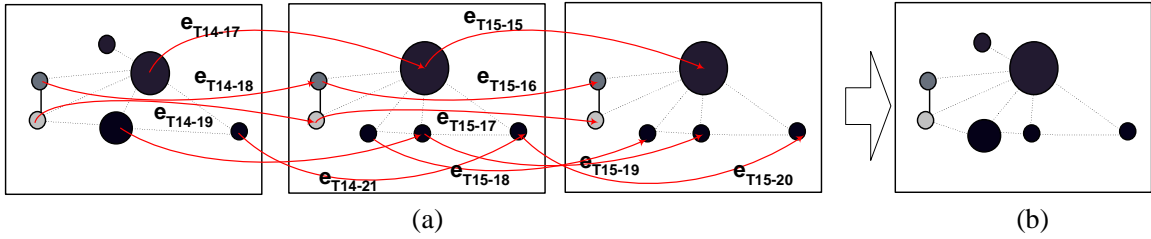


Figure 4.4. (a) A part of STRG in Figure 3.3 (c), and (b) BG from (a).

4.3 Experimental Results

To access the proposed methods, we have performed the experiments with real videos as shown in Table 3.1. All videos are encoded to AVI format with 15 frames per second and 160×120 pixel resolution. As seen in Table 3.1, the data set consists of various types of contents taken from TV programs. The sixth and seventh columns of Table 3.1 are the number of abrupt and gradual changes in videos, respectively.

We evaluate the performance of our proposed approaches by demonstrating that:

- The performance of GSM_{sim} in terms of processing speed is enough for a real time system.
- Our shot boundary detection technique using graph matching can detect and classify accurately both abrupt and gradual shot changes in video sequences.
- The proposed STRG decomposition technique can detect moving objects as a format of object graph (OG).

4.3.1 Efficiency of Graph Similarity Measure

The efficiencies of GSM and GSM_{sim} affect the performance of real time system since general graph matching algorithms are known as NP-complete [54]. In order to verify the efficiencies of GSM and GSM_{sim} , we check their processing capacity, i.e., the number of processed frames per second, when they run on the test videos with different frame rates (Frame Per Second, FPS). First, we digitize videos with 5 FPS, 10 FPS and 15 FPS. Then, the SBD mentioned in Section 2.3 is performed for each digitized sequence of frames using GSM and GSM_{sim} for similarity measures. Table 4.1 shows the results where each value is the average number of frames per second processed during the SBD. If the value is greater than corresponding FPS, it is able to process video data in real time. Otherwise, the processing will be delayed. As seen in Table 4.1, GSM_{sim} is applicable to video sequences with 5 FPS and 10 FPS, while GSM is only for 5 FPS. This is because GSM_{sim} is simplified by reducing searching area. Therefore, we use GSM_{sim} as similarity measure, and set up the system to process 10 frames for every second in our experiments.

4.3.2 Performance of Shot Boundary Detection

The performance of SBD is important for video abstractions since the proposed video abstraction algorithms are based on the shots. In order to evaluate the effective-

Table 4.1. The average number of processed frames per second for the SBD using *GSM* and *GSM_{sim}* with different frame rates: 5, 10, and 15 FPS

No.	<i>GSM</i>			<i>GSM_{sim}</i>		
	5 FPS	10 FPS	15 FPS	5 FPS	10 FPS	15 FPS
1	7.8	8.1	8.1	10.8	11.0	12.2
2	7.5	7.9	8.0	10.5	10.8	10.8
3	8.0	8.3	8.5	11.0	11.8	12.5
4	7.0	7.2	7.3	10.2	10.5	10.9
5	8.2	8.5	9.0	11.3	12.0	12.9
6	7.8	8.0	8.1	12.1	12.4	12.9
7	7.5	7.8	7.9	10.6	10.7	10.8
8	8.2	8.3	8.8	11.3	12.1	12.7
Avg	7.75	8.01	8.21	10.97	11.41	11.96

ness of the proposed SBD using graph matching (GM-SBD), we compare it with the SBD proposed by FXPAL Lab (FXPAL) [60], which has one of the best methods in TRECVID 2004. FXPAL combines pairwise similarities between images in the locality and supervised classification. All experiments of GM-SBD are performed with $T_{cut} = 0.2$, while FXPAL uses $T = 4$ for a threshold value and $L = 5$ for the number of neighborhood frames without random projection. We use ‘Recall’ and ‘Precision’ to verify the performance of those techniques. The recall and precision are defined as:

- Recall (H_r) is the ratio of the number of shots detected correctly over the actual number of shots in a given data set.
- Precision (H_p) is the ratio of the number of shots detected correctly over the total number of shots detected correctly or incorrectly.

The values of recall and precision are in the range of $[0, 1]$. The higher recalls indicate a higher capacity of detecting correct shots, while the higher precisions indicate a higher capacity of avoiding false matches.

The results are given in Table 4.2. As seen in Table 4.2, the results of GM-SBD are almost the same as those of FXPAL. We observe that the recalls and precisions

of the proposed SBD reach up to 94% and 90%, respectively, which indicate that it is competitive with existing techniques, and even consistent for different types of videos. In case of gradual changes, GM-SBD is more accurate than FXPAL.

Table 4.2. Experimental results of SBD

No.	GM-SBD				FXPAL			
	Abrupt change		Gradual change		Abrupt change		Gradual change	
	<i>Hr</i>	<i>Hp</i>	<i>Hr</i>	<i>Hp</i>	<i>Hr</i>	<i>Hp</i>	<i>Hr</i>	<i>Hp</i>
1	0.96	0.89	1.00	0.80	0.97	0.92	0.75	0.38
2	0.92	0.95	0.80	0.67	0.94	0.91	0.60	0.50
3	0.94	0.90	0.67	0.67	0.94	0.91	0.67	0.50
4	0.94	0.92	0.67	0.57	0.96	0.91	0.67	0.50
5	0.95	0.98	1.00	0.67	0.95	0.95	1.00	0.50
6	0.94	0.87	1.00	0.67	0.94	0.85	1.00	0.67
7	0.95	0.83	0.80	0.67	0.96	0.91	0.80	0.57
8	0.92	0.85	0.70	0.58	0.97	0.93	0.80	0.73
Avg	0.94	0.90	0.78	0.64	0.96	0.92	0.76	0.55

4.3.3 Performance of Object Graph Detection

To access the proposed method for STRG decomposition technique, we performed the experiments with the real video streams captured by surveillance camera as shown in Table 3.1. Table 4.3 shows the results of OG detection. The third and the fourth columns of Table 1 are the number of actual video objects and the number of correctly detected OGs, respectively. As seen in the last column, the accuracy of graph-based moving object detection, i.e., OG detection, reaches up to 94.7% on average.

4.4 Summary

In this chapter, we described several graph-based video parsing techniques such as shot boundary detection, object detection, and background modeling. The video

Table 4.3. Experimental results of Object Graph (OG) detection

No.	Duration (hh:mm:ss)	OG performance		
		Actual OGs	Found OGs	Accuracy
9	40:38:02	438	411	93.8%
10	04:12:24	159	147	92.5%
11	00:15:08	202	195	96.5%
12	00:12:48	210	203	96.7%
Total	45:18:22	1009	956	94.7%

parsing techniques are basis on a graph, which provide more efficient and accurate video processing. First, an STRG is segmented into a number of smaller pieces corresponding to shots. Then, each segmented STRG is decomposed into *Object Region Graphs* (ORGs) and *Background Graphs* (BGs). Due to the limitation of region segmentation algorithm, a single moving object can be divided into several ORGs. To address this, we merge ORGs belonging to a single object into *Object Graph* (OG). In addition, we model a background of STRG, called BG, since there are a lot of redundant information in the background. The redundant BGs are eliminated to reduce the index size and search time.

CHAPTER 5

STRG CLUSTERING

In the previous chapters, we proposed a new graph-based data structure, STRG that can represent the content of video sequence. After an STRG is constructed from a given video sequence, it is decomposed into its subgraphs called OGs, which represent the temporal characteristics of video objects. In this chapter, we introduce graph-based video data mining techniques, specifically unsupervised learning to find knowledge, i.e., moving patterns of objects. For the unsupervised learning, we cluster similar OGs into a group, in which we need to match two OGs. For this graph matching, we introduce a new distance measure, called *Extended Graph Edit Distance (EGED)*, which can handle the temporal characteristics of OGs. For actual clustering, we exploit *Expectation Maximization (EM)* with *EGED*. In addition, we propose a model-based conceptual clustering (MCC) of spatio-temporal data based on a formal concept analysis, which provides a user formal concepts of clusters. The proposed MCC can be applied to any types of spatio-temporal data, such as OGs, hurricane track data, and time-series data.

The remainder of this chapter is organized as follows. In Section 5.1, we introduce the *EGED* for graph matching. Section 5.2 shows EM clustering algorithm to group similar OGs. In Section 5.3, we propose a model-based conceptual clustering algorithm. The performance study is reported in Section 5.4. Finally, Section 5.5 presents the summary of the chapter.

5.1 Extended Graph Edit Distance

There are many graph matching algorithms. The simplest one is to use an inexact subgraph isomorphism [64]. In spite of its elegance and intuitiveness, this approach has an exponential complexity, therefore this is not suitable for our system that needs a significant amount of graph operations. Messmer and Bunke [65] proposed a new algorithm for subgraph isomorphism, which uses prior knowledge about the database model to reduce the computational complexity. One limitation of their algorithm is that it uses a traditional edit distance which has been used for string matching. It is not appropriate to handle complex graphs (i.e., STRG) which have various node and edge attributes. In order to address this, Shearer et al. [64] developed an algorithm to find the largest common subgraph (LCSG) by extending Messmer and Bunke’s work with the decomposition network algorithm. Since the LCSG is not suitable for graphs with temporal characteristics, we extend it by combining Chen’s edit distance with real penalty (ERP) [66], which allows to obey metric space with local time shifting. We call this new algorithm as *Extended Graph Edit Distance* (EGED) for convenience.

The purpose of the edit distance for graphs is to compute the minimum cost of graph edit operations such as adding, and changing nodes, which are necessary to transform one graph to another. However, a typical graph edit distance in [64] uses a simple edit cost function, in which all of the edit operations have equal cost. This simple edit cost function is not suitable for computing the distance between OGs because we need to consider time and various attributes of nodes and edges differently. In order to address this, we consider temporal characteristics and the node attributes of OG together to compute the distance (dissimilarity) between two OGs. Therefore, the main difference between the standard edit distance and EGED is that (1) EGED can handle a local time shifting using dynamic programming, which provides more accurate measure in time

sequence data, and (2) EGED has more realistic cost function for adding and changing nodes than that of edit distance.

Since the main operations to edit graphs deal with nodes and their attributes rather than edges, we consider only the nodes and their attributes. Let OG_m^s and OG_n^t be s^{th} and t^{th} OGs with m and n number of nodes; i.e., $OG_m^s = \{v_1^s, v_2^s, \dots, v_m^s, \nu^s\}$, and $OG_n^t = \{v_1^t, v_2^t, \dots, v_n^t, \nu^t\}$, respectively. The distance function $EGED$ between OG_m^s and OG_n^t can be defined as follows.

Definition 8. *The Extended Graph Edit Distance (EGED) between two object graphs OG_m^s and OG_n^t is defined as:*

$$EGED(OG_m^s, OG_n^t) = \begin{cases} \sum_{i=1}^m |v_i^s - g_i| & \text{if } n = 1, \\ \sum_{i=1}^n |v_i^t - g_i| & \text{if } m = 1, \\ \min[EGED(OG_{m-1}^s, OG_{n-1}^t) + dist_{ged}(v_m^s, v_n^t), \\ \quad EGED(OG_{m-1}^s, OG_n^t) + dist_{ged}(v_m^s, gap), \\ \quad EGED(OG_m^s, OG_{n-1}^t) + dist_{ged}(gap, v_n^t)] & \text{otherwise.} \end{cases}$$

where gap is an added or changed node, and g_i is a gap for i^{th} node. And,

$$dist_{ged}(v_i^s, v_j^t) = \begin{cases} |v_i^s - v_j^t| & \text{if } v_i^s, v_j^t \text{ are not a gap} \\ |v_i^s - g_j| & \text{if } v_j^t \text{ is a gap} \\ |v_j^t - g_i| & \text{if } v_i^s \text{ is a gap.} \end{cases}$$

Let v indicate all attribute values of a node for convenience. $dist_{ged}$ is the cost function for editing nodes. Depending on how to select a gap (g_i), various distance functions are possible. For example, when $g_i = v_{i-1}$, the cost function is the same as one in Dynamic Time Warping (DTW), which does not consider local time shifting.

In our case, $g_i = \frac{v_{i-1} + v_i}{2}$ is used for $dist_{ged}$. EGED can handle a local time shifting properly, since it is based on dynamic programming which is a suitable for time-varying

data [66], which can handle local time shifting properly. Figure 5.1 illustrates the computation of $EGED$ between two OGs, OG^s and OG^t . In Figure 5.1 (a), according to Definition 5, there is no gap between v_3^s and v_3^t since $EGED(OG_3^s, OG_3^t)$ has a minimum value when $dist_{ged}(v_3^s, v_3^t) = |v_3^s - v_3^t|$. Figure 5.1 (b) is the case that there is a gap in OG^s because $EGED(OG_5^s, OG_5^t)$ has a minimum value when $dist_{ged}(gap, v_5^t) = |v_5^t - g_5|$.

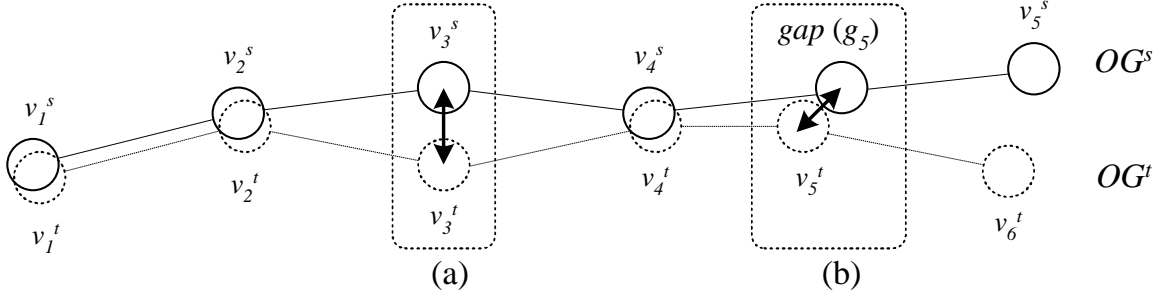


Figure 5.1. Example of EGED between two Object Graphs: (a) No gap, and (b) A gap for insertion in OG^s .

However, as long as the cost function ($dist_{ged}$) replicates the previous nodes, the EGED is no longer in a metric space since $dist_{ged}$ does not satisfy the triangle inequality. For example, given three simplified OGs; $OG^r = \{0\}$, $OG^s = \{1, 1\}$, and $OG^t = \{2, 2, 3\}$. Then, $EGED(OG^r, OG^t) > EGED(OG^r, OG^s) + EGED(OG^s, OG^t)$ because $7 > 2 + 4$, which does not satisfy the triangle inequality. In order to satisfy the triangle inequality, an EGED is restraint to be a metric distance function (see Theorem 2) by comparing a current value with a fixed constant.

Theorem 2. *If g_i is a fixed constant, then EGED is a metric.*

Proof. Suppose that R , S and T are OGs. It is obvious that EGED satisfies non-negative, symmetry and reflexivity. However, the non-trivial case is the triangle inequality, i.e.

$$EGED(R_l, T_n) \leq EGED(R_l, S_m) + EGED(S_m, T_n)$$

We prove the above triangle inequality by induction. By Definition 5, $EGED(R_l, T_n)$ can be written as

$$\begin{aligned} EGED(R_l, T_n) = \min[& EGED(R_{l-1}, T_{n-1}) + |v_l^r - v_n^t|, \\ & EGED(R_{l-1}, T_n) + |v_l^r - g|, \\ & EGED(R_l, T_{n-1}) + |v_n^t - g|] \end{aligned} \quad (5.1)$$

where g is a fixed constant. To compute the cost to transform v_l^r to v_n^t , consider the three terms in the right hand side of Equation (5.1), which correspond to the following cases:

1. Use $EGED(R_{l-1}, T_{n-1}) + |v_l^r - v_n^t|$ to edit v_{l-1}^r into v_{n-1}^t by replacing v_l^r with v_n^t .
2. Use $EGED(R_{l-1}, T_n) + |v_l^r - g|$ to edit v_{l-1}^r into v_n^t by deleting v_l^r .
3. Use $EGED(R_l, T_{n-1}) + |v_n^t - g|$ to edit v_l^r into v_{n-1}^t by adding v_n^t .

Since Equation (5.1) uses a fixed constant g , the three terms in the right hand side are optimal, which means that the term on the left is also optimal. For the last step of graph editing, one cannot do better than making a single change or not making any change at all. Therefore, EGED also satisfies the triangle inequality since $EGED(R_l, T_n)$ is optimal. \square

$EGED_M$ is used to indicate the metric version of EGED. The difference between $EGED_M$ and EGED is that the cases of $n = 0$ and/or $m = 0$ should be considered in $EGED_M$ (see Definition 5). It means that each data item (OG) should be measured from a fixed point in metric space. Let us repeat the previous example ($OG^r = \{0\}$, $OG^s = \{1, 1\}$, and $OG^t = \{2, 2, 3\}$) with $EGED_M$ and $g = 0$. $EGED_M(OG^r, OG^t) = 7$. Similarly, $EGED_M(OG^r, OG^s) = 2$ and $EGED_M(OG^s, OG^t) = 5$. Thus, $7 \leq 2 + 5$, which satisfies the triangle inequality.

5.2 Clustering OGs Using EGED

The proposed graph-based video indexing needs clusters of OGs for more effective indexing. For this clustering, we will employ a probabilistic clustering algorithm called *Expectation Maximization* (EM) to group similar OGs. For the distance measure used in clustering, *EGED* in Definition 8 is applied for EM clustering algorithm. The results of clustering will be used for a model-based conceptual clustering in Section 5.3, and indexing in Section 6.

5.2.1 EM Clustering with EGED

First, OGs are selected randomly from the M number of data items (OGs). Let Y_j be the j^{th} OG with a dimension d . Each OG is assigned to a cluster k with a probability of w_k such that $\sum_{k=1}^K w_k = 1$, which is the sum of the membership probabilities of all the measurements for Y_j to a cluster. A finite Gaussian mixture model is chosen to cluster OGs since it is widely used and easy to implement [67]. The density function ($p_k(Y_j|\theta_k)$) of Y_j , which is an observed data for individual j , is formulated as

$$p(Y_j|\Theta) = \sum_{k=1}^K w_k p_k(Y_j|\theta_k)$$

where $\Theta (= \{\theta_1, \dots, \theta_K\})$ is a set of parameters for the mixture model with K component densities.

Each θ_k is parameterized by its mean μ_k and covariance matrix Σ_k . The d -dimensional Gaussian mixture density is given by

$$p(Y_j|\Theta) = \sum_{k=1}^K \frac{w_k}{2\pi^{d/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(Y_j-\mu_k)^T \Sigma_k^{-1} (Y_j-\mu_k)} \quad (5.2)$$

In Equation (5.2), the covariance matrix Σ_k determines the geometric features of the clusters. Common cases use a restricted covariance, $\Sigma_k (= \lambda I)$, where λ is a scalar value, and I is an identity matrix in which the number of parameters per component grows as

a square of the dimension of the data. However, if the dimension of the data is highly relative to the number of data, the covariance estimates will often be singular, which causes the EM algorithm to break down [68]. Specifically, the data (i.e., OGs) in the time-dependant domain have different dimensions since their time lengths vary. Therefore, the covariance matrix Σ_k cannot have an inverse matrix, consequently we cannot compute Equation (5.2). Chris Fraley et al. [68] point out this problem, and suggest using different distance metrics between data points. Thus, we replace the Mahalonobis distance defined by the covariance and the mean of each component in Equation (5.2) with the *EGED* in Definition 9 with $g_i = \frac{v_{i-1}+v_i}{2}$. Since the covariance matrix is not needed in the *EGED*, the dimension of the Gaussian mixture density is reduced to one. Therefore, the Equation (5.2) can be rewritten as follows.

$$p(Y_j|\Theta) = \sum_{k=1}^K \frac{w_k}{2\pi^{1/2}|\sigma_k|} e^{-\frac{1}{2\sigma_k^2}EGED(Y_j,\mu_k)^2} \quad (5.3)$$

Equation (5.3) is a new *one*-dimensional Gaussian mixture density function with the *EGED* for OGs. This mixture model provides some benefits to handling OGs as follows. It can reduce the dimension, deal with various time lengths of OGs, and give an appropriate distance function for OGs in each cluster. Suppose that Y 's are mutually independent, the log-likelihood (\mathcal{L}) of the parameters (Θ) for a given data set Y can be defined from Equation (5.3) as follows.

$$\mathcal{L}(\Theta|Y) = \log \prod_{j=1}^M p(Y_j|\Theta) = \sum_{j=1}^M \log \sum_{k=1}^K w_k p_k(Y_j|\theta_k) \quad (5.4)$$

To find appropriate clusters we estimate the optimal values of the parameters (θ_k) and the weights (w_k) in Equation (5.4) using the EM algorithm, since it is a common procedure used to find the Maximum Likelihood Estimates (MLE) of the parameters iteratively.

The EM algorithm produces the MLE of the unknown parameters iteratively. Each iteration consists of two steps: E-step and M-step.

E-step: It evaluates the posterior probability of Y_j , belonging to each cluster k . Let h_{jk} be the probability of j^{th} OG for a cluster k , then it can be defined as follows:

$$h_{jk} = P(k|Y_j, \theta_k) = \frac{w_k}{p_k(Y_j|\theta_k)} \quad (5.5)$$

M-step: It computes the new parameter value that maximizes the probability using h_{jk} in E-step as follows:

$$\begin{aligned} w_k &= \frac{1}{M} \sum_{j=1}^M h_{jk} \\ \mu_k &= \frac{\sum_{j=1}^M h_{jk} Y_j}{\sum_{j=1}^M h_{jk}} \\ \sigma_k &= \frac{\sum_{j=1}^M h_{jk} EGED(Y_j, \mu_k)^2}{\sum_{j=1}^M h_{jk}} \end{aligned} \quad (5.6)$$

The iteration of E and M steps is stopped when w_k is converged for all k . After the maximum likelihood model parameters ($\hat{\Theta}$) in Equation (5.4) are decided, each OG is assigned to a cluster, \hat{k} in terms of the maximum posterior probability by the following equation.

$$\hat{k} = \arg \max_{1 \leq k \leq K} \frac{w_k p_k(Y_j|\theta_k)}{\sum_{k=1}^K w_k p_k(Y_j|\theta_k)} \quad (5.7)$$

The complexity of the proposed algorithm using the EM with the *EGED* can be analyzed as follows. The complexity of each iteration (one E-step and one M-step) in the EM using Equation (5.2) with M data sets of K clusters in d -dimension is $O(d^2 KM)$ [67]. We are using Equation (5.3) instead of Equation (5.2). Therefore, the complexity of each iteration can be reduced to $O(KM)$ since *EGED* reduces the complexity of covariance (d^2) to 1.

5.2.2 Optimal Number of Clusters

The EM algorithm described above uses a pre-determined number of Gaussian densities (i.e., the number of clusters K). However, it is very difficult to decide the number reasonably at the beginning. Thus, estimating an optimal number of clusters is a key issue to improve the quality of EM clustering. Many criteria are proposed in the literatures [50, 56] to decide the optimal number of clusters with a known Gaussian model. The well-known criteria in the statistics literature are Bayesian Information Criterion (BIC), Akaike's Information Criterion (AIC) and Mallow's C_p . The basic idea of those criteria is penalizing the model in some way by offsetting the increase in log-likelihood with a corresponding increase in the number of parameters, and seeking to minimize the combination of log-likelihood and its parameters. We employ the BIC to select the number of clusters because it is convenient for model selection. Let $\mathcal{M} = \{\mathcal{M}_K : K = 1, \dots, N\}$ be the candidate models. \mathcal{M}_K is the finite Gaussian mixture model with K clusters. The BIC for \mathcal{M}_K is defined as

$$BIC(\mathcal{M}_K) = \hat{l}_K(Y) - \eta \mathcal{M}_K \log(M) \quad (5.8)$$

where $\hat{l}_K(Y)$ is the log-likelihood of the data Y by the K^{th} model, $\eta \mathcal{M}_K$ is the number of independent parameters for model \mathcal{M}_K , and M is the total number of data items. From Equation (5.4), the log-likelihood of the data Y is defined as follows:

$$\hat{l}_K(Y) = \log \prod_{j=1}^M p(Y_j | \Theta)$$

And, for a finite Gaussian mixture model of K component densities, the number of independent parameters is

$$\eta \mathcal{M}_K = (K - 1) + \frac{Kd(d + 3)}{2}$$

where d is a data dimension; i.e., $d = 1$ in our model because the dimension is reduced to 1 using the *EGED*. For a given data set Y , we can decide the number of clusters for the model \mathcal{M}_K whose BIC value is maximized in Equation (5.8).

5.3 Conceptual Clustering

Most of the previous clustering algorithm focus on finding a goodness measure of overall partitioning for data objects. However, these approaches are not very optimal for spatio-temporal data including moving objects in a video and time-series data that has high-dimensional attributes and very complicated relationships. As the spatio-temporal data is unlabelled and large volume of structure, clustering and concept analysis of the data remain challenging problems. In this section, we propose a model-based conceptual clustering (MCC) of spatio-temporal data based on a formal concept analysis. Our proposed MCC consists of three steps: ‘*model formation*’, ‘*model-based concept analysis*’, and ‘*concept graph generation*’. We then generate the concepts for spatio-temporal data using the concept graph obtained from the third step.

5.3.1 Model Formation

The use of spatio-temporal data in many applications such as financial data analysis, video surveillance systems, personal life storages, hurricane tracks, and medical videos, has increased enormously with the recent advances in sensor device, video capture device, storage, network technology and computing power. As a consequence, there are increasing demands on modeling, indexing, and retrieval of these data. A spatio-temporal data is typically defined as a sequence of observations of an object in time. Therefore, a d -dimensional spatio-temporal data ST_N , or simply ST , is a sequence:

$$ST_N = \langle (\vec{v}_1, t_1), \dots, (\vec{v}_N, t_N) \rangle \quad (5.9)$$

where \vec{v}_i is a feature vector of dimension d , and t_i is a time instant with $1 \leq i \leq N$ [69]. Each pair (\vec{v}_i, t_i) has the observed values of an object at time t_i . For example, if only one feature value is observed at each time (i.e., $d = 1$), ST is called a *time-series*. In case of a *trajectory*, \vec{v}_i can be two dimensional position, i.e., (x_i, y_i) . We can find many examples of high-dimensional ST s in real life data. One example of high-dimensional ST s can be found in hurricane track data [70, 71]. In addition to a spatial location of a hurricane track, i.e., longitude and latitude, there are more features to be considered for the hurricane data analysis, such as wind speed, pressure, and temperature. Another example is moving objects in video surveillance system [72, 73, 74] where a set of features includes not only a location of a moving object, but also size, color, and speed of an object. In this paper, we focus on high-dimensional spatio-temporal data where $d > 1$.

The process of ST model formation is the same as a model-based clustering of them mentioned in the previous section (Section 5.2), since the clustering conducts unsupervised learning to find models of ST s that are characterized by a set of parameters. Recall that the results of EM clustering algorithm are a number of clusters for ST . Each cluster indicate a model for the data. The benefits of model formation in the proposed conceptual clustering are as follows:

- It simplifies the computation of a conceptual clustering algorithm since the number of objects is reduced to the number of models that represent their specific patterns.
- Byproducts of the EM algorithm, i.e., the extracted model parameters (Θ) and the expectation (h_{jk}), play an important role in conceptual clustering since they provide the relationships between models and features.

5.3.2 Model-based Formal Concept Analysis

The second step of the proposed MCC is a model-based concept analysis to find formal concepts of modeled spatio-temporal data. Although the model formation using

EM clustering in Section 5.3.1 provides a set of models where similar ST s are grouped, it does not consider interpreting the obtained models. In other words, it is hard for a user to understand what the meaning of each model is, and how the models are formed. In order to address these, we employ the concept of formal concept analysis (FCA) that was introduced by Wille [44, 45]. FCA has been used for data analysis and knowledge representation. We propose a model-based concept analysis (MFCA), which incorporates a model-based statistical data analysis with FCA to represent the meanings of spatio-temporal data. The MFCA starts with a model-based formal context that is defined as:

Definition 9. *A model-based formal context is a triple $K = (G, F, I)$, where*

- G is a set of models characterized by Θ ,
- F is a set of features, and
- I is a set of relations between G and F (i.e. $I \subseteq G \times F$). Each relation $(g, f) \in I$ has a significance value $\lambda(g, f)$.

In the model-based formal context, G consists of the models determined in the previous subsection instead of all data objects. Each model is characterized by a set of parameters (Θ). The set of features (F) consists of d number of observed features in ST . The set of relations I indicates how much each feature is relevant to each model. The relevance between the k^{th} model g_k and the l^{th} feature f_l is represented as the *significance value* $\lambda(g_k, f_l)$, which is computed by exploiting a feature selection technique in data mining [75].

Let y_F and $y_{F-\{f_l\}}$ be the full feature vector, and the feature vector without the l^{th} feature, respectively. Consider two posterior probabilities $(P_{i,k,F}, P_{i,k,F-\{f_l\}})$ of the

k^{th} model based on the full feature vector (y_F), and the feature vector without f_l feature ($y_{F-\{f_l\}}$) as follows.

$$\begin{aligned} P_{i,k,F} &= P(y_{i,F}|\theta_{k,F}), \\ P_{i,k,F-\{f_l\}} &= P(y_{i,F-\{f_l\}}|\theta_{k,F-\{f_l\}}) \end{aligned}$$

where $\sum_{i=1}^m P_{i,k,F} = 1$ and $\sum_{i=1}^m P_{i,k,F-\{f_l\}} = 1$ with m number of objects in the k^{th} model. If f_l is a completely insignificant feature in the k^{th} model, then $P_{i,k,F}$ is equal to $P_{i,k,F-\{f_l\}}$. Otherwise, the difference between two probabilities provides a significance value of f_l in the model. In order to measure a difference between two probabilities, we use the Kullback-Leibler divergence (KLD) [76]. The significance value of the l^{th} feature (f_l) in the k^{th} cluster is defined as:

$$\lambda(g_k, f_l) = \sum_{i=1}^m \left| P_{i,k,F} \log \frac{P_{i,k,F}}{P_{i,k,F-\{f_l\}}} \right| \quad (5.10)$$

The higher the value of λ , the more significance of the feature in a model. Then, we determine all relations between a set of models G and a set of features F .

A model-based formal context K can be represented as a cross-table as shown in Table 5.1. The context in Table 5.1 has four models (i.e. g_1, g_2, g_3 and $g_4 \in G$), and three features (i.e. f_1, f_2 and $f_3 \in F$). The relation (I) between a model and a feature is measured by λ in Equation (5.10).

To remove relations that have very low significance values, we set a predetermined threshold ϵ . Table 5.2 shows the cross-table of the model-based formal context with $\epsilon = 0.01$.

According to a formal concept analysis, we can consider the features of a formal context as the description of the concept [44, 45]. Therefore, we can derive a model-based formal concept from a model-based formal context as follows (Definitions 10 and 11).

Table 5.1. Example of model-based formal context without threshold

	f_1	f_2	f_3
g_1	0.583	0.004	0.431
g_2	0.840	0.002	0.003
g_3	0.002	0.454	0.623
g_4	0.000	0.517	0.833

Table 5.2. Example of model-based formal context with threshold ($\epsilon = 0.01$)

	f_1	f_2	f_3
g_1	0.583	-	0.431
g_2	0.840	-	-
g_3	-	0.454	0.623
g_4	-	0.517	0.833

Definition 10. For a set of models $A \subseteq G$, we define $A' = \{f \in F \mid \lambda(g, f) > \epsilon \text{ for all } g \in A\}$, and a set of features $B \subseteq F$, we define $B' = \{g \in G \mid \lambda(g, f) > \epsilon \text{ for all } f \in B\}$

Definition 11. A model-based formal concept of the context $K = (G, F, I)$ with a threshold value ϵ is a pair (A, B) where $A \subseteq G$, $B \subseteq F$, $A' = B$, and $B' = A$. We call A the extent, and B the intent of the formal concept (A, B) .

The extent covers all models belonging to the formal concept, while the intent

comprises all features valid for all those models. Table 5.3 shows the complete list of concepts for the context in Table 5.2. For example, a concept C_4 has the extent $A = \{g_3, g_4\}$, and the intent $B = \{f_2, f_3\}$ (see highlighted cells in Table 5.2). The extent A and the intent B of the formal concept (A, B) are closely related by the relation I . In other words, the concept (A, B) means the models determined by features f_2 and f_3 .

Table 5.3. Formal concepts for context in Table 5.2

Concept	Extent (A)	Intent (B)
C_0	$\{g_1, g_2, g_3, g_4\}$	\emptyset
C_1	$\{g_1, g_2\}$	$\{f_1\}$
C_2	$\{g_1, g_3, g_4\}$	$\{f_3\}$
C_3	$\{g_1\}$	$\{f_1, f_3\}$
C_4	$\{g_3, g_4\}$	$\{f_2, f_3\}$
C_5	\emptyset	$\{f_1, f_2, f_3\}$

It is natural to have a hierarchical order between the model-based formal concepts of the context, i.e. *subconcept* or *superconcept* relation. In order to represent the relation, we exploit lattice theory [45] since it offers a natural way to formalize the ordering of objects. A model-based concept lattice is defined in Definition 12 and 13.

Definition 12. *If (A_1, B_1) and (A_2, B_2) are formal concepts of a model-based formal context, then (A_1, B_1) is the subconcept of (A_2, B_2) such that $A_1 \subseteq A_2$ (i.e., $B_2 \subseteq B_1$), denoted as $(A_1, B_1) \leq (A_2, B_2)$. In this case, (A_2, B_2) is superconcept of (A_1, B_1) .*

Definition 13. A model-based concept lattice of the model-based formal context $K = (G, F, I)$ is a set of all model-based formal concepts of K with the order \leq , denoted by $\underline{\mathfrak{B}}(K, \leq)$.

For the purpose of visualization of the lattice, we use a line diagram consisting of circles and lines for all models and features, respectively. Each circle represents a formal concept, and each line indicates a relation, i.e. subconcept (downward) or superconcept (upward) in Definition 5. Figure 5.2 shows the line diagram of the model-based concept lattice $\underline{\mathfrak{B}}(K, \leq)$ generated from the model-based formal context $K = (G, F, I)$ in Table 5.2.

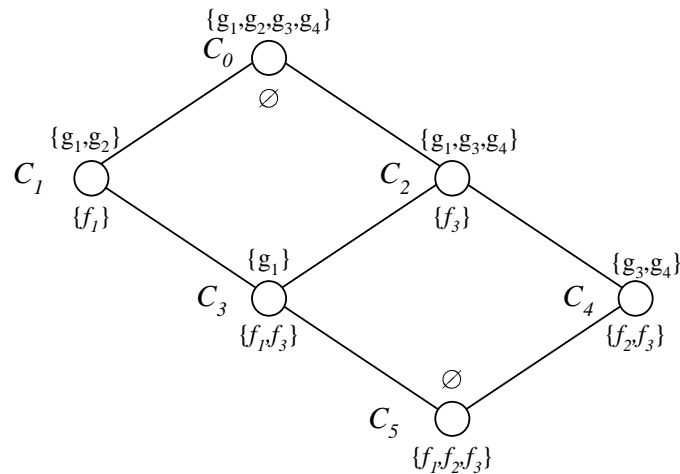


Figure 5.2. A model-based concept lattice for the context K in Table 5.2.

5.3.3 Concept Graph Generation

In this subsection, we introduce a concept graph generation that is the last step of the MCC. Although the generated formal concepts using MFCA provide formal descriptions of models, MFCA typically produces a high number of formal concepts. Among

them there exist some similar formal concepts that need to be merged into a single concept. In order to compact the formal concepts, we first propose a similarity measure *ConSim* between two formal concepts. Using the similarity measure, we compact the formal concepts to generate a concept graph. Then, we describe the entire process of the model-based conceptual clustering based on the concept graph.

5.3.3.1 Concept Similarity

To measure a similarity between two concepts generated by MFCA, we exploit the similarity of FOGA framework [43], then extend it to a model-based formal concept. In FOGA, the similarity between two formal concepts $C_1 = (\varphi(A_1), B_1)$ and $C_2 = (\varphi(A_2), B_2)$, where φ is a fuzzy membership function, is defined as $E(C_1, C_2) = \frac{|\varphi(A_1) \cap \varphi(A_2)|}{|\varphi(A_1) \cup \varphi(A_2)|}$. However, this similarity measure is not enough for our model-based formal concepts since it considers only a fuzzy membership for the similarity, i.e., $\varphi(A_1)$ and $\varphi(A_2)$, but the proposed concept consists of the extent and the intent. We extend it to handle both the extent and the intent of model-based formal concepts for the similarity. Let $C_1 = (A_1, B_1)$ and $C_2 = (A_2, B_2)$ be two model-based formal concepts in $\mathfrak{B}(K, \leq)$. The similarity measure *ConSim* between C_1 and C_2 is defined as follows.

Definition 14. *Given two model-based formal concepts C_1 and C_2 , a concept similarity (*ConSim*) between C_1 and C_2 is defined as:*

$$\text{ConSim}(C_1, C_2) = \gamma \frac{|A_1 \cap A_2|}{|A_1 \cup A_2|} + (1 - \gamma) \frac{|B_1 \cap B_2|}{|B_1 \cup B_2|}$$

where $|\text{set}|$ is the number of elements in set, and γ is a predefined weight of the extent concept such that $0 \leq \gamma \leq 1$.

The concept similarity *ConSim* in Definition 14 ranges 0 to 1. The higher the value

of *ConSim* is, the more similar two concepts are. The predefined weight (γ) plays a role to decide a priority between the extent and intent of the concept. For example, if a user has more confidence to the extent of concept (i.e., a set of models) than the intent (i.e. a set of features), γ is greater than 0.5. Otherwise, γ is set to less than 0.5. Throughout this paper, we set $\gamma = 0.5$, which means the extent and intent have the same priority for *ConSim*. Figure 5.3 shows the computed *ConSim* with $\gamma = 0.5$ of Figure 5.2.

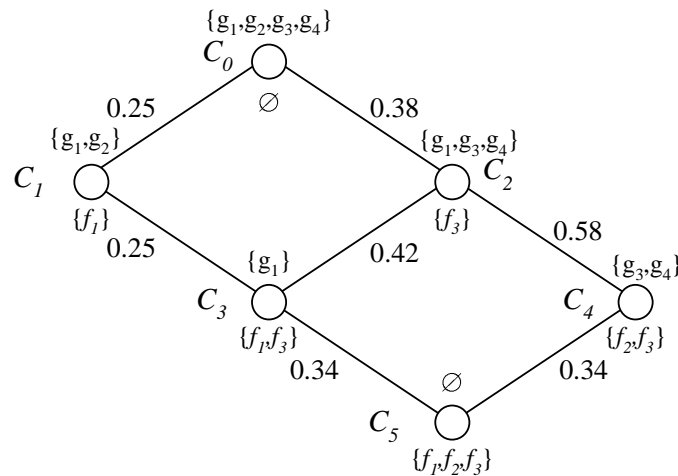


Figure 5.3. Concept similarities between two concepts in Figure 5.2.

5.3.3.2 Concept Graph Generation

We generate a set of models using EM, and perform the proposed MFCA on the models to find their formal concepts. Since the MFCA uses a statistical approach for the significance values between features and models, even a small difference of the significance values may cause similar concepts to be separated into different concepts. Moreover, the number of generated formal concepts is usually large since the MFCA allows a model to belong to more than one concept. Such a large number of formal concepts prevent a user from interpreting them easily. In order to address this, we introduce a technique of

merging two similar formal concepts into a single formal concept, and a concept graph where the merged concepts are represented.

Let $C_1 = (A_1, B_1)$ and $C_2 = (A_2, B_2)$ be two model-based formal concepts in $\underline{\mathfrak{B}}(K, \leq)$. The criteria of merging two concept (C_1 and C_2) are defined as follows:

Definition 15. *Given two model-based formal concepts C_1 and C_2 , $C_1 \cup_M C_2 = (A_1 \cup A_2, B_1 \cup B_2)$ is a merged concept if C_1 and C_2 satisfy*

- $(A_2, B_2) \leq (A_1, B_1)$ C_1 is superconcept of C_2 , and
- $ConSim(C_1, C_2) > T_{sim}$.

where T_{sim} is a predefined threshold value for the concept similarity. For an example of Figure 5.3, C_2 and C_4 can be merged into a single concept $C_2 \cup_M C_4$ with $T_{sim} = 0.5$. Since $C_4 \leq C_2$ and $ConSim(C_2, C_4) = 0.58 (> T_{sim})$, they satisfy the merging criteria in Definition 15.

However, if two formal concepts are merged into a single concept in $\underline{\mathfrak{B}}(K, \leq)$, the set of merged concepts is no longer model-based concept lattice, which is proven in the following proposition (Proposition 1).

Proposition 1. *For given model-based formal concepts (A_1, B_1) , (A_2, B_2) , and (A_3, B_3) in $\underline{\mathfrak{B}}(K, \leq)$ with $K = (G, F, I)$, if $(A_2, B_2) \leq (A_1, B_1)$, and $(A_3, B_3) \leq (A_1, B_1)$, then*

- (1) $(A_1 \cup A_2, B_1 \cup B_2)$ is not a model-based formal concept of K , and
- (2) $(A_3, B_3) \not\leq (A_1 \cup A_2, B_1 \cup B_2)$.

Proof. (1) Suppose that $(A_1 \cup A_2, B_1 \cup B_2)$ is a concept. By Definition 3 and 4, $(A_1 \cup A_2)' = (B_1 \cup B_2)$, i.e. for all $f' \in (A_1 \cup A_2)'$ satisfies $\lambda(g, f') > \epsilon$ for all $g \in (A_1 \cup A_2)$. However, if $f' \in A'_1$, but $f' \notin A'_2$, then $f' \notin (B_1 \cup B_2)$. Therefore, $(A_1 \cup A_2, B_1 \cup B_2)$ is not

model-based formal concept of K .

(2) Suppose that (A_3, B_3) is subconcept of $(A_1 \cup A_2, B_1 \cup B_2)$. By Definition 5, $(B_1 \cup B_2) \subseteq B_3$. However, if $f \in B_2$, but $f \notin B_1$, then $f \notin B_3$ because $B_1 \subseteq (B_2 \cap B_3)$. Therefore, $(A_3, B_3) \not\subseteq (A_1 \cup A_2, B_1 \cup B_2)$. \square

Proposition 1 (1) and (2) show that a merged concept cannot preserve the properties of a model-based concept lattice. Therefore, we cannot use a model-based concept lattice after merging concepts. In order to address this, we propose a concept graph to maintain the merged concepts and their relations, which is defined as follows:

Definition 16. A concept graph, \mathcal{G} , is a four-tuple $\mathcal{G} = \{V_c, E_c, \nu_c, \xi_c\}$, where

- V_c is a finite set of conceptual nodes,
- $E_c \subseteq V_c \times V_c$ is a finite set of relational edges between two concept nodes,
- $\nu_c : V_c \rightarrow A_{V_c}$ is a function generating the conceptual node attributes, and
- $\xi_c : E_c \rightarrow A_{E_c}$ is a function generating the relational edge attributes.

A concept graph \mathcal{G} is a directed graph whose edges are ordered pairs of node and attribute. A conceptual node ($v_c \in V_c$) corresponds to a (merged) concept where the node attributes (A_{V_c}) are sets of models and features. A relational edge ($e_c(v_c, v'_c) \in E_c$) represents a relation between two concepts, i.e., superconcept or subconcept for the edge attributes (A_{E_c}). For example, $e_c(v_c, v'_c)$ indicates that v_c is a superconcept of v'_c (i.e. v'_c is a subconcept of v_c). The relational edge is weighted by the similarity between two conceptual nodes using *ConSim* in Definition 14. The benefit of using the concept graph is that it is more flexible to maintain the merged concepts and their relations without loss of their semantics than a model-based concept lattice in Definition 13.

5.3.3.3 Model-based Conceptual Clustering

For a given model-based concept lattice $\underline{\mathfrak{B}}(K, \leq)$, we generate a concept graph in which similar formal concepts are merged into a single concept. Therefore, the main procedure of our model-based conceptual clustering is the concept merging. Figure 5.4 is the outline of the model-based conceptual clustering algorithm for ST s. The input is a set of spatio-temporal data ST_{set} , and a confidence threshold for a concept similarity T_{sim} , while the output is a concept graph \mathcal{G} .

First, we perform a model formation of ST_{set} to find a set of models \mathcal{M} (see line 1). For the models \mathcal{M} , we build a model-based formal context K , and a model-based concept lattice $\underline{\mathfrak{B}}(K, \leq)$ (see line 2 - 3). Then, we select the supremum concept of $\underline{\mathfrak{B}}(K, \leq)$ that is a concept having only subconcepts (see line 4). The supremum concept is a starting concept of the lattice traverse to find similar formal concepts. For each subconcept of the starting concept, we call a function MCC to determine if two formal concepts are similar using $ConSim$ in Definition 14. If a concept similarity is greater than T_{sim} , the corresponding subconcept is merged into its superconcept. Otherwise, leave it as it is (see line 12 - 23). We repeat the function for each subconcept recursively until the subconcept is the infimum concept of $\underline{\mathfrak{B}}(K, \leq)$ that is a concept having only superconcepts (see line 12). For the output, i.e. a concept graph \mathcal{G} , we maintain sets of models and features of each conceptual node, which describe the conceptual meanings of spatio-temporal data. On the other hand, each relational edge has the relationship between two corresponding nodes, i.e., a superconcept or a subconcept, and their similarity.

Figure 5.5 (a) is the result of concept merging of Figure 5.3 with $T_{sim} = 0.5$. We can observe that two formal concepts C_2 and C_4 can be merged into a single concept, since $ConSim(C_2, C_4) > 0.5$. Figure 5.5 (b) shows the concept graph of Figure 5.5 (a) in

Algorithm 3: Model-based Conceptual Clustering

Input: a set of spatio-temporal data ST_{set} , and a threshold T_{sim}

Output: a concept graph \mathcal{G}

/* \mathcal{M} is an empty set of models, K is an empty formal context, $\mathcal{B}(K, \leq)$ is an empty model-based concept lattice, and (V, E) is a pair of empty sets of node and edge */

- 1: $\mathcal{M} \leftarrow$ **perform** *model formation* of ST_{set} in Section 5.3.1;
- 2: $K \leftarrow$ **build** *model-based formal context* of \mathcal{M} in Definition 9;
- 3: $\mathcal{B}(K, \leq) \leftarrow$ **build** *model-based concept lattice* of K in Definition 13;
- 4: **let** C_{start} = the supremum concept of $\mathcal{B}(K, \leq)$;
- 5: $(V_c, E_c) = MCC(\mathcal{B}(K, \leq), C_{start})$;
- 6: $\mathcal{G} = (V_c, E_c, \mathcal{V}_c, \mathcal{E}_c)$;
- 7: **return** \mathcal{G} ;

8: **Function** $MCC(B, C)$

/* B is a model-based concept lattice, C is a starting concept, and (V_r, E_r) is a pair of node and edge sets for return */

- 9: **let** (V_r, E_r) be a pair of empty node and edge sets;
 - 10: **begin**
 - 11: **for each** subconcept C_{sub} of C in B **do**
 - 12: **if** C_{sub} is infimum concept of B **then**
 - 13: $V_t = V_r \cup \{ \mathcal{V}_c(C) \}$; $E_t = E_r \cup \{ \mathcal{E}_c((C, C_{sub})) \}$;
 - 14: **return** (V_r, E_r) ;
 - 15: **else**
 - 16: $(V_r, E_r) = MCC(B, C_{sub})$;
 - 17: **if** $ConSim(C, C_{sub}) > T_{sim}$ **then**
 - 18: $V_t = V_r \cup \{ \mathcal{V}_c(C \cup_M C_{sub}) \}$;
 - 19: $E_t = E_r \cup \{ \mathcal{E}_c(C \cup C_{sub}) \}$;
 - 20: **else**
 - 21: $V_t = V_r \cup \{ \mathcal{V}_c(C) \}$; $E_t = E_r \cup \{ \mathcal{E}_c((C, C_{sub})) \}$;
 - 22: **end if**
 - 23: **end if**
 - 24: **done**
 - 25: **return** (V_r, E_r) ;
 - 26: **end Function**
-

Figure 5.4. Algorithm 3: A model-based conceptual clustering.

which a set of nodes represents concepts of ST s, and a set of edges represents relations between two corresponding concepts.

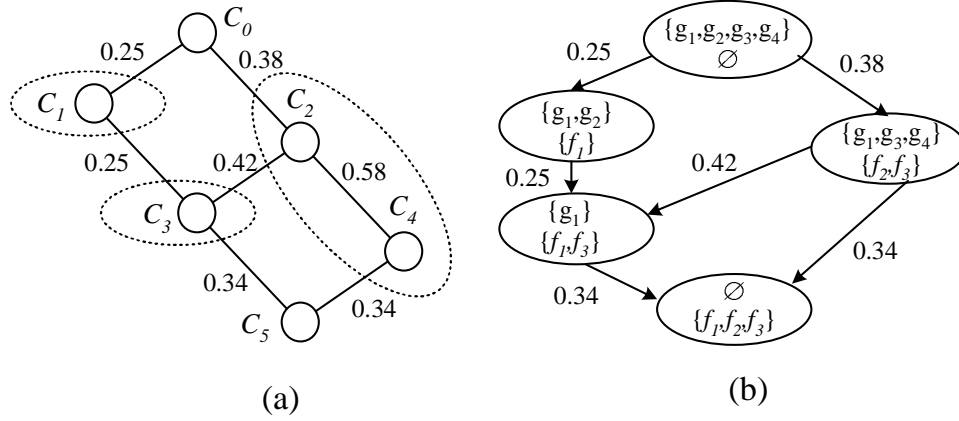


Figure 5.5. Result of model-based conceptual clustering: (a) A result of concept merging using *ConSim* with $T_{sim} = 0.5$, and (b) A concept graph of (a).

The characteristics of the proposed MCC can be summarized as follows: (1) Unlike existing conceptual clustering algorithms that use a goodness measure for a clustering criterion, the proposed MCC exploits FCA to generate formal concepts of ST s. This approach is suitable for the analysis of high-dimensional data such as ST since it is hard to find an appropriate goodness measure for such a high-dimensional data, (2) As in the definitions of MFCA, it uses a set of models instead of data objects themselves. The use of models in MFCA can reduce the computational cost, and provide more semantics than FCA using data objects since the number of models is much less than that of data objects, and each model has important information of the data objects such as patterns, and (3) Since FCA is developed only for labelled data, a scaling in which the data is divided into several intervals, is necessary for numeric data. However, this plain scaling is not suitable for ST since a series of data cannot be scaled easily. To address this, our MFCA uses a statistical model. Instead of using a binary relation of FCA, the MFCA

uses the *significance values* between models and features. Therefore, we do not have to scale the data since the significance values represent the relationships between features and models.

5.4 Experimental Results

5.4.1 Results of Clustering OGs

We have performed the experiments with synthetic and real data (see Table 3.1) to assess the performance of EM clustering OGs. First, we evaluate the performance of the EM clustering algorithm with the non-metric EGED on the synthesized data. Then, we apply the proposed EM clustering to real videos in Table 3.1.

5.4.1.1 Performance of EM clustering OGs

We evaluate the performance of the EM clustering algorithm with the non-metric EGED (EM-EGED) on the synthesized data. As stated earlier, the quality of clustering OGs is important in guaranteeing the performance of the STRG-Index. Our EM-EGED is compared with two other clustering algorithms; K-Means (KM) and K-Harmonic means (KHM). The detailed information about KM and KHM can be found in [77, 78]. Furthermore, to evaluate the performance of distance functions, we compare the EGED with Dynamic Time Warping (DTW) [19] and Longest Common Subsequence (LCS) [20]. We compare the performance of EM-EGED with EM-LCS and EM-DTW (Figure 5.6 (a)), KM-EGED with KM-LCS and KM-DTW (Figure 5.6 (b)), and KHM-EGED with KHM-LCS and KHM-DTW (Figure 5.6 (c)). For the evaluation, we use the clustering error rate defined as follows:

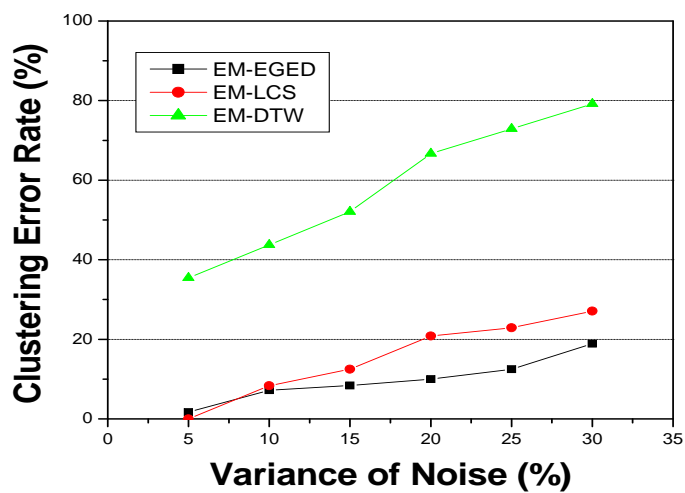
$$\begin{aligned} \text{Clustering Error Rate (CER, \%)} = & \hspace{15em} (5.11) \\ & \left(1 - \frac{\text{Number of Correctly Clustered OGs}}{\text{Total Number Of OGs}}\right) \times 100 \end{aligned}$$

The EGED based algorithms perform much better than those based on the LCS and the DTW as seen in Figure 5.6 (a), (b) and (c). Especially, EM-EGED outperforms EM-DTW since EM tends to break down when the distance function cannot compute the similarity properly (see Figure 5.6 (a)). The EGED measures the similarity between OGs more accurately than the others do. Figure 5.6 also shows that the EGED is much more robust to noise than either the LCS or the DTW is.

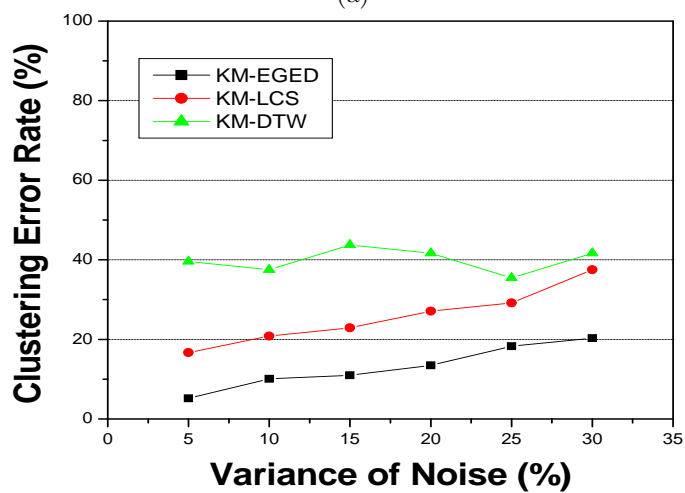
Figure 5.7 shows the performance of EM-EGED compared with those of KM-EGED and KHM-EGED. The clustering error rate of EM-EGED is a little better than that of KHM-EGED (see Figure 5.7 (a)). The reason why KHM-EGED has a similar clustering performance with EM-EGED is because its soft membership of data points is similar to h_{jk} of the EM in Equation (5.5), and its weight is similar to w_k in Equation (5.6). As far as the computation time is concerned, EM-EGED performs much better than KM-EGED or KHM-EGED. As shown in Figure 5.7 (b), EM-EGED runs much faster (around 1.5 to 2 times) than the others do. This can reduce the time to build the STRG-Index for a real-time system handling surveillance videos, for example. Figure 5.7 (c) shows the distortion values of each algorithm under different noise levels. The distortion is defined as the sum of the distances (i.e., in number of pixels) between the detected centroids and the true centroids. In terms of the distortion, EM-EGED is similar to KM-EGED, but much more accurate (about 2 times) than KHM-EGED. Overall, the quality of the EM-EGED proposed in this paper is superior to that of the other alternatives, such as KM-EGED, KHM-EGED, KM-LCS, KHM-LCS, KM-EDR, KHM-EDR as well as KM-DTW and KHM-DTW.

5.4.1.2 Real Video Data Set

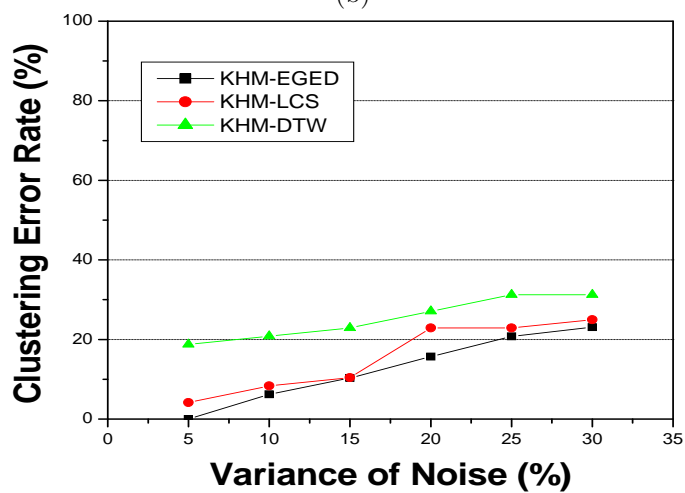
We apply the proposed EM clustering algorithm to surveillance videos in Table 3.1. Figure 5.8 shows the example of clustering result for the first surveillance video (Surv



(a)

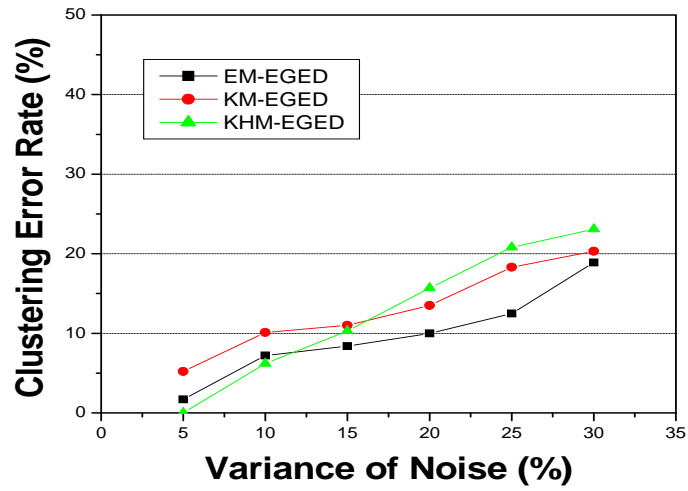


(b)

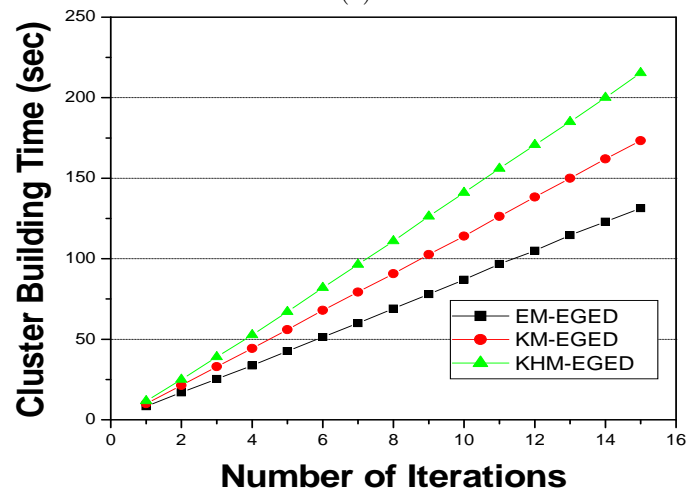


(c)

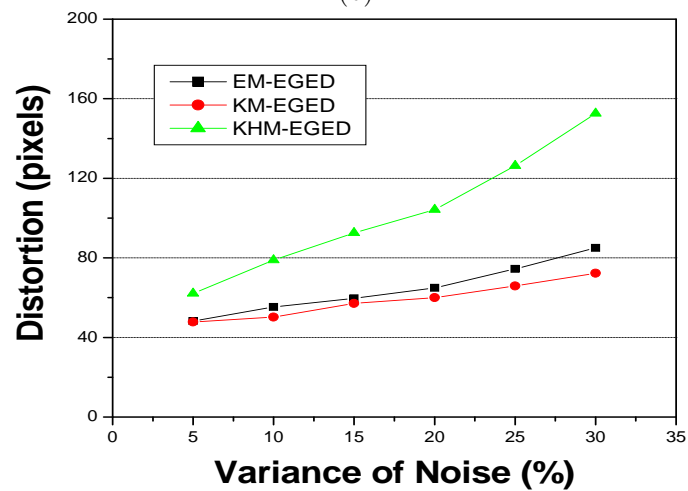
Figure 5.6. CER: (a) EM-EGED vs. EM-LCS, EM-DTW, (b) KM-EGED vs. KM-LCS, KM-DTW, and (c) KHM-EGED vs. KHM-LCS, KHM-DTW.



(a)



(b)



(c)

Figure 5.7. EM-EGED performances against KM-EGED and KHM-EGED: (a) Clustering Error Rate, (b) Cluster building time, and (c) Distortion.

1). As seen in this figure, OGs are grouped into 8 clusters. The first column indicates the number of clustered OGs, and the second column is the visualization of each cluster by plotting its members (OGs). Two sample OGs of each cluster are shown in the third column by some selected frames. The different clusters have different characteristics: for example, Cluster 2 has the objects moving bottom to top-right corner, and Cluster 3 has a similar pattern but with an opposite direction to Cluster 2. The interesting results are observed in Cluster 7 such that it has the noise data such as unexpected illumination changes at night. The algorithm clusters even those noise data into separated groups correctly.

Since the surveillance video are captured without any pre-defined moving patterns, it is hard to decide the optimal number of clusters, and the cluster membership of OGs to a certain cluster. We find the optimal number of clusters for each video stream using the BIC measure. The EM algorithm is performed for k (the number of clusters) ranging from 1 to 15. Then, the BIC values are computed using Equation (5.8). Figure 5.9 shows the BIC value corresponding to various number of clusters for each video. Here, the optimal number of clusters for a particular video is the peak value of the corresponding curve.

5.4.2 Results of Conceptual Clustering

In order to assess the proposed schemes, we have conducted several experiments with synthesized data sets in Section 3.3.2. Using the data sets, we evaluate the performances of our proposed approaches, and demonstrate that the quality of generated concepts based on MCC outperforms those of existing conceptual clustering algorithms.

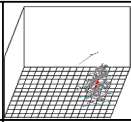

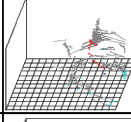

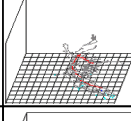

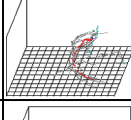

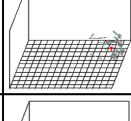

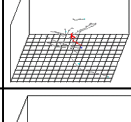

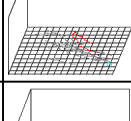

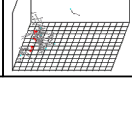
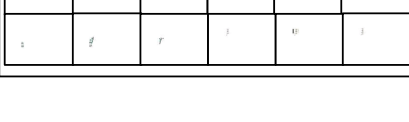
Cluster (# of OG)	Result	Examples of OG	Descriptions
0 (134)			Objects moving at bottom right corner.
1 (51)			Objects appear at right, then go out through door.
2 (45)			Objects moving bottom to top-right corner.
3 (33)			Objects moving top to bottom right corner.
4 (19)			Objects moving at top-right corner.
5 (15)			Objects moving bottom to top, then returning.
6 (15)			Objects moving right to left, then returning.
7 (98)			Noises caused by PC and illumination changes.

Figure 5.8. Results of EM clustering with *EGED* for video (Surv 1).

5.4.2.1 Evaluation Metrics

For our evaluation, we employ two evaluation metrics: the *relaxation error* [27], and *F-measure* [79]. The relaxation error (*RE*) measures the goodness of the generated concepts, which is a popular criterion of evaluating conceptual clustering algorithm. On the other hand, since the concept graph has a hierarchical tree structure, we need a specific metric that can analyze the entire hierarchical tree. We use F-measure (F_m) for this metric, which combines the precision and the recall from information retrieval.

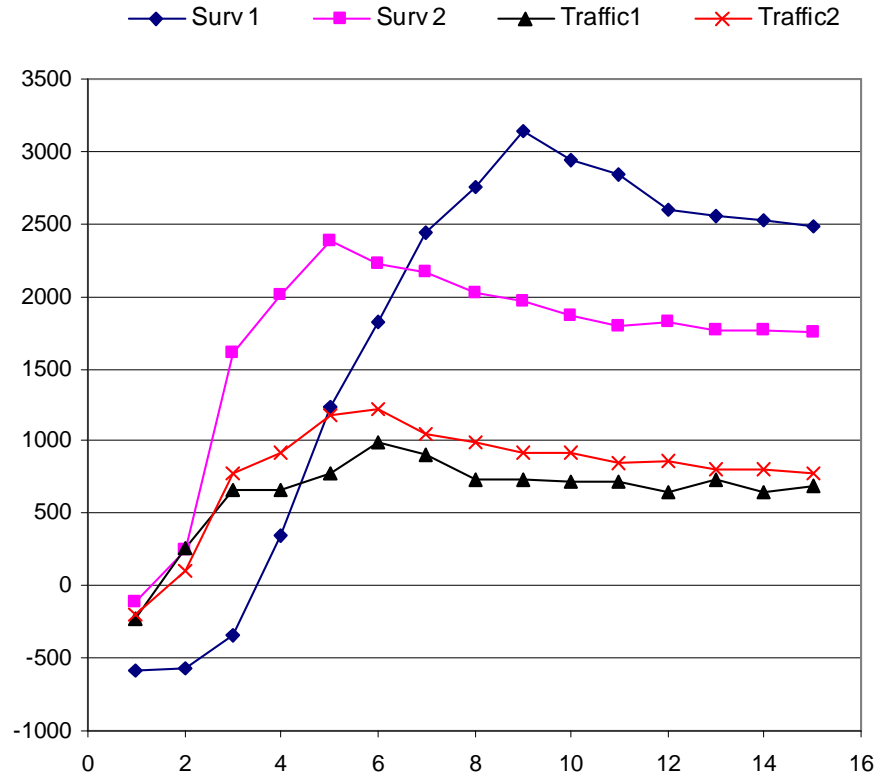


Figure 5.9. The BIC values for finding the optimal number of clusters.

Given a particular concept of d -dimensional feature ST s, i.e., $C_k = \{y_1, \dots, y_m\}$, the relaxation error (RE_{C_k}) for the concept C_k is defined as the average pair-wise distance using $STED$ in Definition 1 among ST s in C_k .

$$RE_{C_k} = \frac{1}{m^2} \sum_{i=1}^m \sum_{j=1}^m P(y_i)P(y_j)EGED(y_i, y_j)$$

where $P(y_i)$ and $P(y_j)$ are the probabilities of y_i and y_j occurring in C_k , respectively. RE_{C_k} implies the dissimilarities of data objects (ST s) in a concept C_k based on the distance function $EGED$. Therefore, the relaxation error ($RE(C)$) for entire concepts can be computed as follows:

$$RE(C) = \frac{1}{K} \sum_{k=1}^K RE_{C_k} \quad (5.12)$$

where K is the total number of concepts. A smaller value of $RE(C)$ corresponds to a higher quality of the concepts.

In addition to the relaxation error, we use F-measure for the evaluation metric. In F-measure, we consider each concept as if it were the result of a query, and each class as if it were the desired set of *STs* for the query. Given a particular concept C_k of size m_k , and a particular class R_j of size m_j , the precision (p) and the recall (r) for each C_k and R_j are

$$p_{kj} = \frac{m_{kj}}{m_k}, \quad r_{kj} = \frac{m_{kj}}{m_j}$$

where m_{kj} is the number of data items that are correctly clustered from R_j in C_k . The F-measure of concept C_k and class R_j is computed as follows:

$$F(C_k, R_j) = \frac{2 \cdot r_{kj} \cdot p_{kj}}{r_{kj} + p_{kj}} \quad (5.13)$$

Consequently, the F-measure for an entire hierarchy of any concepts ($F_m(C)$) is computed by taking the weighted average of all values for $F(C_k, R_j)$ in Equation (5.13) as follows:

$$F_m(C) = \sum_{j=1}^c \frac{m_j}{m} \max_k F(C_k, R_j) \quad (5.14)$$

where c and m is the total number of classes and *STs*, respectively. The values of $F_m(C)$ range from 0 to 1. The higher the F-measure is, the better the quality of concept is. We use both $RE(C)$ in Equation (5.12) and $F_m(C)$ in Equation (5.14) to compare the quality of generated concepts C with those of other conceptual clustering algorithms in the following subsection.

5.4.2.2 Results

First, we evaluate the quality of concepts using the relaxation error (RE). The REs computed by Equation (5.12) for the generated concepts are plotted in Figure 5.10. As shown in the figure, the proposed MCC outperforms ECC and GCC methods in terms of the goodness of clusters, i.e., our MCC is around 30% better than ECC. This

demonstrates the advantage of using the significance value λ in the model-based concept analysis in Section 3. Figure 5.10 also shows the robustness of MCC to noise.

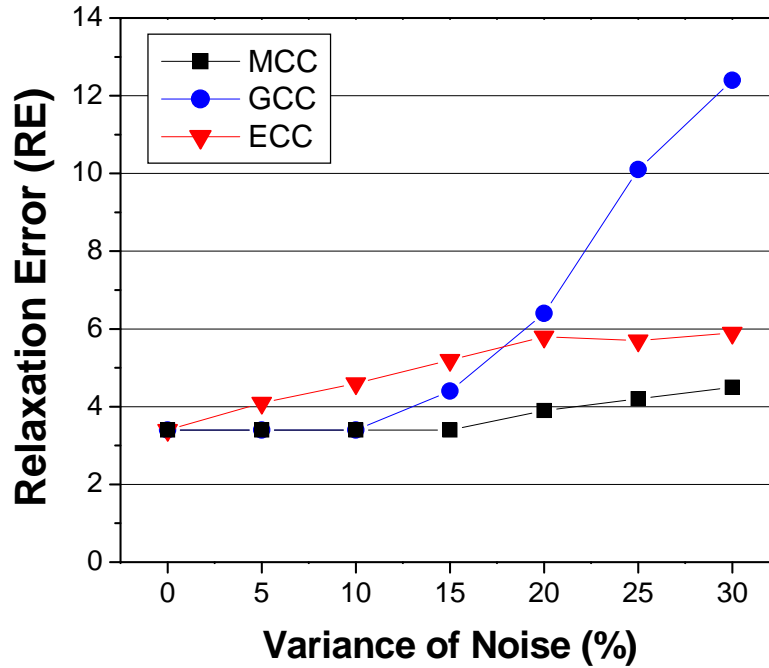


Figure 5.10. Quality of generated concepts using Relaxation Error.

However, RE metric is not enough to measure the hierarchical structures since it considers only the overall goodness measure of concepts. In order to address this, we evaluate the quality of concepts using F-measure that analyzes the entire hierarchical tree [79]. Figure 5.11 gives the quality of each method using F_m in Equation (5.14). From the figure, it is observed that the quality of ECC using F_m is decreasing significantly as the variance of noise gets larger unlike using RE . However, MCC is still the most robust conceptual clustering to the noise. Overall, the quality of the MCC proposed in this paper is superior to ECC and GCC in terms of the quality measured by a relaxation error and F-measure.

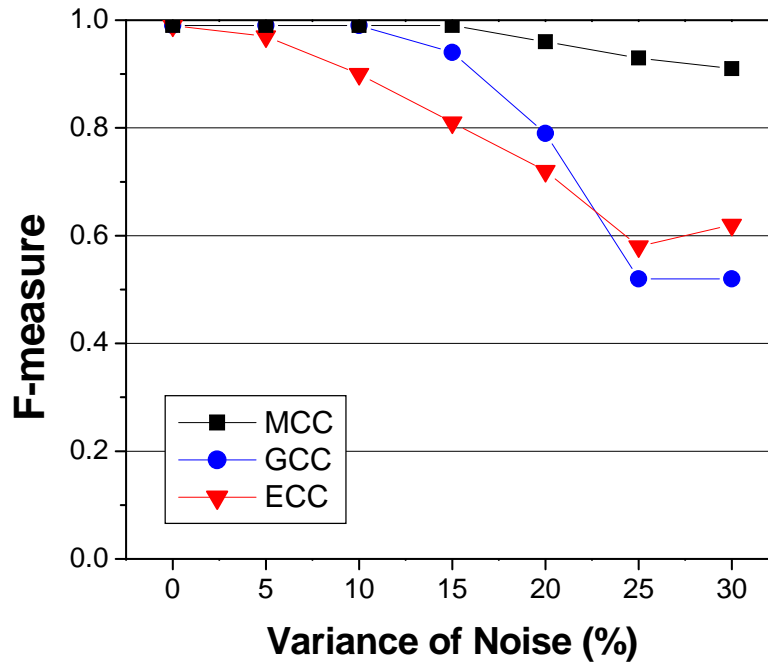


Figure 5.11. Quality of generated concepts using F-measure.

5.5 Summary

In this chapter, we introduced graph-based video data mining techniques, specifically unsupervised learning to find moving patterns of objects. For the unsupervised learning, we cluster similar OGs into a group using EM algorithm. For the graph matching, we introduce a new distance measure, called *Extended Graph Edit Distance (EGED)*, which can handle the temporal characteristics of OGs. For actual clustering, we exploit *Expectation Maximization (EM)* with *EGED*. In addition, we proposed a model-based conceptual clustering (MCC) of spatio-temporal data based on a formal concept analysis, which provides a user formal concepts of clusters. The proposed MCC can be applied to any types of spatio-temporal data, such as OGs, hurricane track data, and time-series data.

CHAPTER 6

STRG INDEXING

In this chapter, we propose a graph-based video indexing method, called *Spatio-Temporal Region Graph Index* (STRG-Index), which uses the $EGED_M$ as a distance measure in metric space, and clustered OGs. One of key issues in video database management systems is how to index video object for fast access. Recall that three characteristics of video data mentioned in Section 1.1.2 are (1) spatial and temporal data, (2) huge size of data, and (3) semantically rich data format. To address these in indexing structure, we uses clustered OGs and modeled BG, since they are decomposed from STRG by removing redundant information. In addition, the clustered OGs makes a query easy access in terms of similarity. We use $EGED_M$, metric version of EGED, for construction of STRG-Index because it makes a query fast.

The rest of this chapter is organized as follows. Section 6.1 presents STRG-Index tree structure consisting of Shot, Cluster, and Object Nodes. Section 6.2 shows how to construct STRG-Index. In Section 6.3, we discuss node split in STRG-Index that makes the size of nodes balance. The performance study is reported in Section 6.4. Finally, Section 6.5 presents the summary of the chapter.

6.1 STRG-Index Tree Structure

Now, we have a BG and clustered OGs for each *strg* based on the techniques discussed in Chapter 4 and 5. To build an index for video data, we adapt the procedure of tree construction proposed in M-tree [32] since it has a minimum number of distance computations and a good I/O performance. In M-tree, a number of representative data

items are selected for efficient indexing. There are several ways to select them such as Sampling or Random selection. In the STRG-Index, we employ the clustering results to determine the representative data items. The STRG-Index tree structure consists of three levels of nodes; shot node, cluster node, and object node as seen in Figure 6.1.

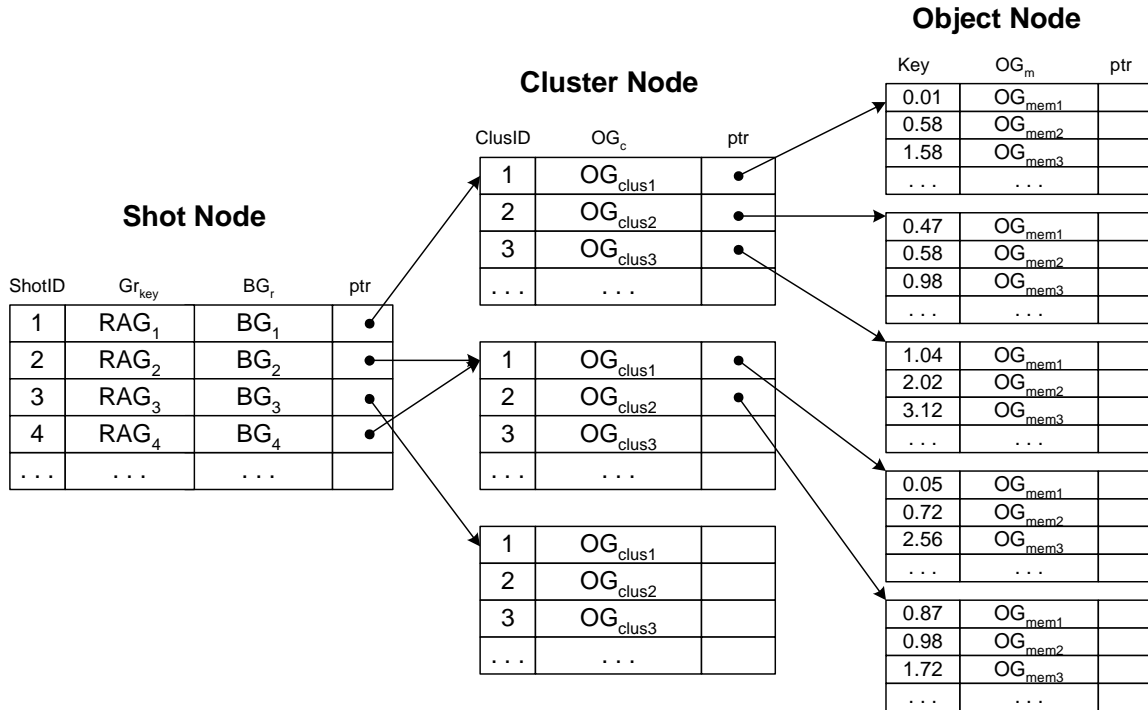


Figure 6.1. Example of STRG-Index tree structure.

The top-level has the shot node which contains the information of each shot in a video. Each record in the shot node represents a segmented shot whose frames share a background. The record has a shot identifier ($ShotID$), a key RAG (Gr_{key}), an actual BG (BG_r), and an associated pointer (ptr) which references the top of corresponding cluster node. The following figure shows an example of a record in the shot node.



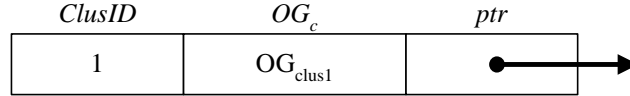
A key RAG is defined as follows.

Definition 17. For given N number of RAGs in i^{th} shot (\mathbf{strg}_i), a key RAG denoted by $Gr_{key}(i)$ is the most representative RAG in \mathbf{strg}_i , which has the largest summation of GSMs among all possible pairs of frames, such that

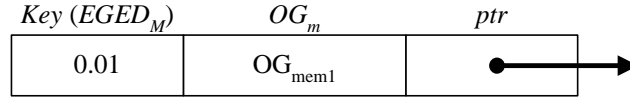
$$Gr_{key}(i) = \arg_{Gr \in \mathbf{strg}_i} \max \sum_{n=1}^N GSM(Gr, Gr(f_n))$$

The proposed key RAG selection is an extension of standard key frame selection techniques [80] by considering the temporal relations among RAGs in a *strg*.

The mid-level has the cluster nodes which contain the centroid OGs that represent cluster centroids. Each record indicates a representative OG among a group of similar OGs. A record contains its identifier (*ClusID*), a centroid OG (OG_c) of each cluster, and an associated pointer (*ptr*) which references the top of corresponding object node. The following figure shows an example of a record in a cluster node.



The low-level has the object nodes which contain OGs belonging to a same cluster. Each record in the object node represents an object in a video, and has the index key (which is computed by $EGED_M(OG_m, OG_c)$), an actual OG (OG_m), and an associated pointer (*ptr*) which references the actual video clip in the disk. The following figure shows an example of a record in the object node.



6.2 STRG-Index Tree Construction

Based on the STRG decomposition in Section IV, an input video is partitioned into shots. Each shot is divided into foreground and background. In other words, an *strg* is decomposed into OGs and a BG as its subgraphs. The extracted BGs and key RAGs are stored at a shot node. All the OGs sharing one BG are in a same cluster node. In addition, if two *strgs* have a same BG, they have a same cluster node as a child node. This can reduce the size of index significantly. For example, in a surveillance video, the camera is often stationary, therefore its background is fixed. In this case, only one record of a BG in a shot node is sufficient to index the background of the entire video.

The cluster nodes play an important role in the performance of search algorithm. Most of the indexing algorithms for time-varying data use non-metric distance functions, which are not applicable to traditional indexing structures such as M-tree [32], M+-tree [81] or B+-tree [30]. Therefore, they prune the data using a lower bound to reduce the search space. However, this still requires significant amount of computation, and results in high numbers of false positives. Instead of pruning data by a lower bound, the STRG-Index uses the EM clustering to group similar data. After the clustering is completed, we synthesize a centroid OG (OG_c) for each cluster which is a representative OG. This centroid OG is inserted into its corresponding cluster node as a record. Also, the centroid OG is updated when its member OGs are changed by insertion or deletion of OGs.

The object node has actual OGs in a cluster, which are indexed by the $EGED_M$. To decide an index value for each OG, we compute $EGED_M$ between the representative OG (OG_c) in the corresponding cluster and an OG (OG_m) to be indexed. Since the $EGED_M$ is a metric distance by Theorem 1, the values from $EGED_M$ can be the keys of

OGs for indexing. Figure 6.2 shows a pseudo code for building the STRG-Index tree for a given STRG, Gst (Algorithm 4).

Algorithm 4: Building STRG-Index

Input: Spation-Temporal Region Graph: Gst

Output: STRG-Index tree: TR

```

1: let  $TR = \text{null}$ ,  $\text{shotid} = 0$ ;
2:  $\text{strg} = \text{segment shots from } Gst \text{ by Section 4.1}$ ;
3: create shot node in  $TR$ ;
4: for each  $\text{strg}_r \in \text{strg}$  do
5:    $OG = \text{extracted } OGs \text{ from } \text{strg}_r \text{ by Section 4.2}$ ;
6:    $BG_r = \text{extracted } BG \text{ from } \text{strg}_r \text{ by Section 4.2}$ ;
7:    $Gr_{key} = \text{extracted key RAG from } \text{strg}_r \text{ by Section 6.1}$ ;
8:    $CLUS = \text{a cluster (of } OGs \text{ ) by EM clustering by Section 5.2}$ ;
9:   insert tuple ( $\text{shotid}$ ,  $Gr_{key}$ ,  $BG_r$ ,  $\text{ptr}(\text{new cluster node})$ ) into shot node in  $TR$ ;
10:   $\text{shotid}++$ ;  $\text{clusid} = 0$ ;
11:  if there exists  $BG$  in shot node  $\ni BG = BG_r$  then
12:    use the cluster node which  $BG$  belongs to;
13:  else create new cluster node; end if
14:  for each  $OG_c \in CLUS$  do
15:    create new object node;
16:    insert tuple ( $\text{clusid}$ ,  $OG_c$ ,  $\text{ptr}(\text{new object node})$ ) into cluster node in  $TR$ ;
17:     $\text{clusid}++$ ;
18:     $OG_{temp} = \text{sort}(OG_c, \text{EGED}_M(OG_c, OG_m))$ ;
19:    for each  $OG_m \in OG_{temp}$  do
20:      insert tuple ( $\text{EGED}_M(OG_c, OG_m)$ ,  $OG_m$ ,  $\text{prt}(\text{real clip})$ ) into object node in  $TR$ ;
21: done; done; done;; return  $TR$ ;

```

Figure 6.2. Algorithm 4: Building STRG-Index.

6.3 STRG-Index Tree Node Split

As new data are inserted into the database, the leaf nodes in low-level grow up arbitrary, which is inefficient for maintaining a balanced tree. In order to address this, the leaf node is split into two nodes if the node satisfies the following condition. If a leaf node has more OGs than a predefined value, we check whether splitting the node is

necessary by using the EM algorithm with $K = 2$ and the BIC value. In other words, if the BIC value when $K = 2$ is larger than the value when $K = 1$, the leaf node is split into two nodes. After splitting, the corresponding records in the cluster node are updated. Otherwise, the node remains unchanged. The split procedure enables the STRG-Index to keep the optimal number of leaf nodes, and provides more accurate results for similarity-based queries.

6.4 Size Analysis

In general, the performance of a database management system depends on the size of index structure and the memory utilization. If the STRG-Index is stored in memory with its actual data items (OGs), it can provide better performance in query processing. Let M be the number of OGs, and N be the total number of frames in a shot. The size of *strg* for a shot can be formulated as follows:

$$size(\mathbf{strg}) = \sum_{m=1}^M size(OG_m) + N \times size(BG) \quad (6.1)$$

On the other hand, the size of an STRG-Index tree for a shot is as follows:

$$size(STRG - Index) = \sum_{m=1}^M size(OG_m) + \sum_{k=1}^K size(CLUS_k) + size(BG) + size(Gr_{key}) \quad (6.2)$$

where K is the number of clusters, and $CLUS_k$ is k_{th} cluster. As seen in Equations (6.1) and (6.2), the difference between STRG and STRG-Index mainly depends on $N \times size(BG)$ and $\sum_{k=1}^K size(CLUS_k)$, since the size of a BG and a RAG (Gr_{key}) are relatively small. Because N is usually much larger than K , the former term is much larger than the latter, i.e. $N \times size(BG) \gg \sum_{k=1}^K size(CLUS_k)$. Hence, the size of STRG-Index is much smaller than that of STRG. However, when the database size increases, the number of shots, clusters, and objects will increase significantly. Consequently, it may not be

possible to load a whole STRG-Index into a memory. To address this, we enhance the STRG-Index to be *scalable* with respect to the number of clusters. An algorithm is said to be *scalable* if its complexity remains linear for a given fixed amount of main memory, when the number of data items increases arbitrarily large. Equation (6.2) is not scalable with respect to the number of clusters, since the number of OGs increases significantly as the number of clusters gets large. In order to make an STRG-Index scalable, we move the actual OGs in a object node to a disk, since they are the most memory consuming component in STRG-Index. For convenience, we refer to the STRG-Index where actual OGs are removed as an *scalable STRG-Index*. After removing OGs from the index, the complexity of Equation (6.2) becomes scalable with respect to the number of clusters, since the size of all key values in the object nodes is very small. Consequently, the number of disk I/O operations of the scalable STRG-Index is the same as the number of returned data. Therefore, the scalability of STRG-Index can maximize the memory utilization.

6.5 Experimental Results

In this section, we validate the performance of the STRG-Index in processing k-NN queries. Most experiments are based on the synthesized data sets for flexible comparisons.

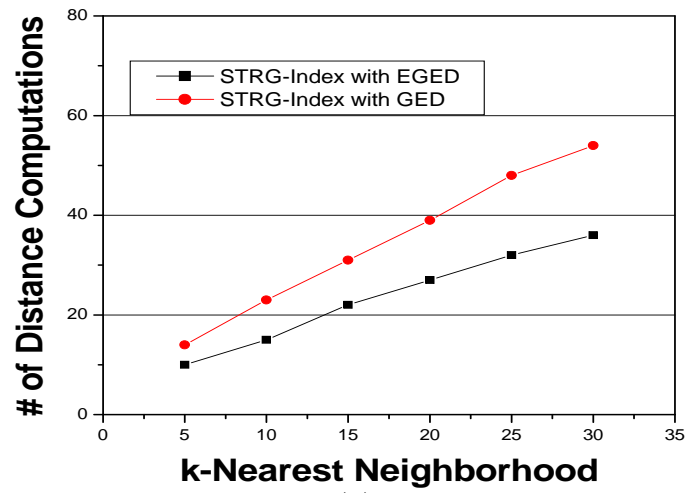
6.5.1 STRG-Index with EGED vs. Graph Edit Distance

In order to evaluate the effectiveness of STRG-Index and $EGED_M$, we first compare the performances of two versions of STRG-Index, which are using $EGED_M$ and the classical edit distance. For the classical edit distance, we use Bunke’s Graph Edit Distance (GED) with the simple cost function used in [21], where all the costs of editing nodes are set to one. Since the GED still obeys the triangular inequality [66], it is in a metric space. Therefore, we cluster OGs, and build an STRG-Index by replacing $EGED_M$ with GED. We compare the k-NN query performance of STRG-Index using $EGED_M$ with

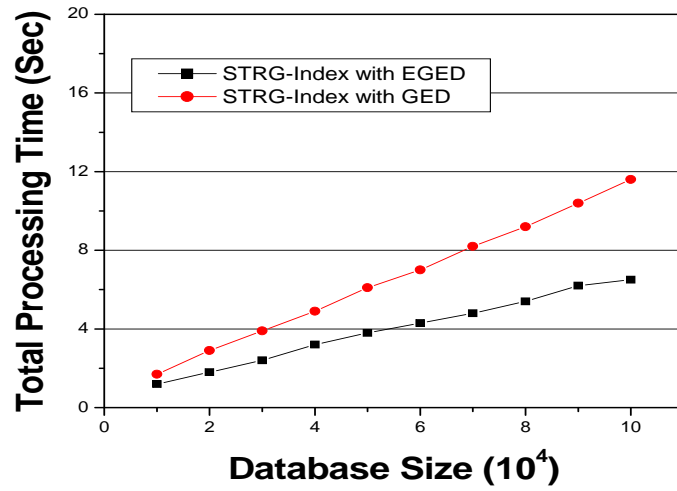
GED by considering the number of distance computations and the total processing time. Since the number of distance computations performed during a query processing is the dominant component [82], we consider it for evaluating the performance of k-NN query. k neighbors range from 5 to 30 on the synthesized data set which contains 5×10^4 objects in the 480 clusters. Figure 6.3 (a) shows that the number of distance computations for EGED_M is much smaller (average 30%) than that for GED. However, when the database size increases, the entire STRG-Index may not be fit to the memory. Therefore, we perform 10-NN queries using the STRG-Index on the data sets with various sizes ranging from 1×10^4 to 10×10^4 objects. Figure 6.3 (b) shows the total processing time which includes the distance computations and the disk I/Os for 10-NN queries. It shows that the total processing time for EGED_M is less than that for GED. Figure 6.3 (c) shows the accuracy of each indexing for the 10-NN query on a data set with 5×10^4 objects. In order to measure the accuracy, the precision and the recall of query results are computed and plotted. The query data is composed of OGs that are not in the data sets, and the query results are evaluated by the cluster memberships. From Figure 6.3 (c), it is obvious that the STRG-Index using EGED_M outperforms the STRG-Index using GED, since GED cannot handle the time characteristic of OGs appropriately. Overall, the STRG-Index is more effective when it uses EGED_M .

6.5.2 STRG-Index vs. M-tree

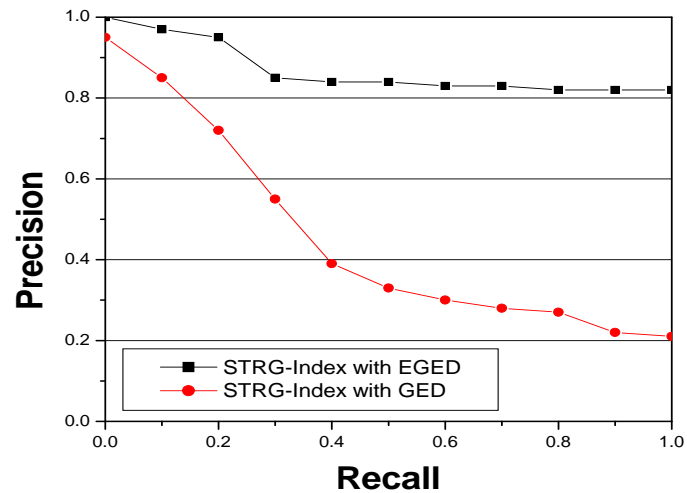
Next, we compare STRG-Index with M-tree (MT) index based on cost and accuracy. In the MT indexing, there are several possibilities depending on the criteria used to select the representative data items. RANDOM (MT-RA) and SAMPLING (MT-SA) methods are chosen for comparison purpose since MT-RA is the fastest, and MT-SA is the most accurate among the methods proposed in [32]. MT-RA selects the reference object(s) randomly. Although this is not optimal, it is the fastest one and used as a ref-



(a)



(b)



(c)

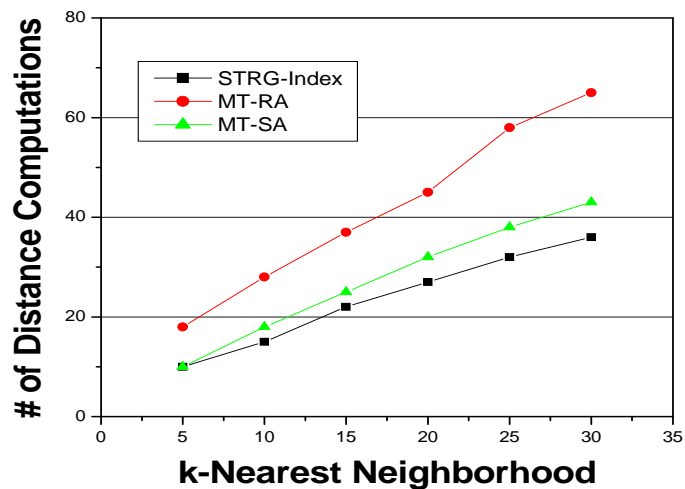
Figure 6.3. Query performances of STRG-Index with $EGED_M$ vs. with GED: (a) Distance computation, (b) Total processing time, and (c) Accuracy.

erence method. MT-SA also uses a random strategy, but the difference is that it iterates over some sample objects. Although its processing is slow, it can select the representative data objects properly. In STRG-Index, we use the EM clustering for selecting the representative nodes (cluster nodes). Actually, we cannot use the EGED for the MT since it needs a metric distance and does not use any explicit clustering. Therefore, we use the EGED_M for the MT construction where RANDOM or SAMPLE is used for selecting representative nodes. In STRG-Index, we use EGED_M for indexing, and EM clustering for selecting the cluster nodes.

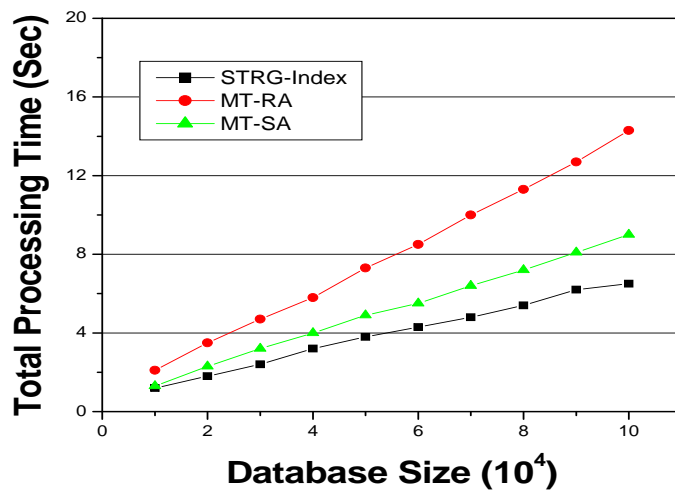
In order to validate the quality of the STRG-Index structure, we perform k-NN queries on the same synthesized data set used in the above experiments. As seen in Figure 6.4 (a), the number of distance computations to process k-NN queries using the STRG-Index is much smaller (average 22%) than that using either the MT-RA or the MT-SA. Figure 6.4 (b) shows the total processing time of 10-NN queries on the data sets ranging from 1×10^4 to 10×10^4 objects. The total processing time for 10-NN query using an STRG-Index is similar to that using MT-SA, and much less than that using MT-RA. This means that the performance of k-NN query using the STRG-Index is better than that using the MT index since both STRG-Index and MT use the same distance measure EGED_M. Figure 6.4 (c) shows the accuracy of each indexing structure for the k-NN query. As seen in the figure, the STRG-Index outperforms both MT-RA and MT-SA. These results demonstrate that the STRG-Index outperforms the M-tree index in terms of both cost and accuracy.

6.5.3 Efficiency and Scalability of STRG-Index

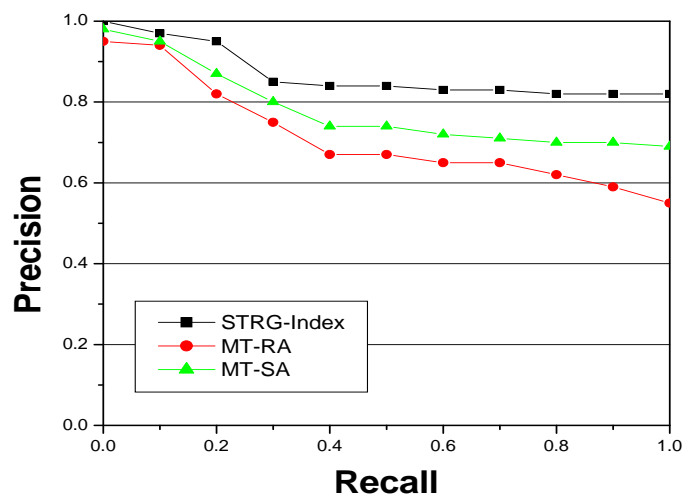
Figure 6.5 (a) shows the average time elapsed in building an index structure for databases of different sizes. From this figure, the time to build a STRG-Index is much less (15% to 50%) than that to build either MT-RA or MT-SA, even though both STRG-



(a)



(b)



(c)

Figure 6.4. Query performances of STRG-Index vs. MT-RA and MT-SA: (a) Distance computation, (b) Total processing time, and (c) Accuracy.

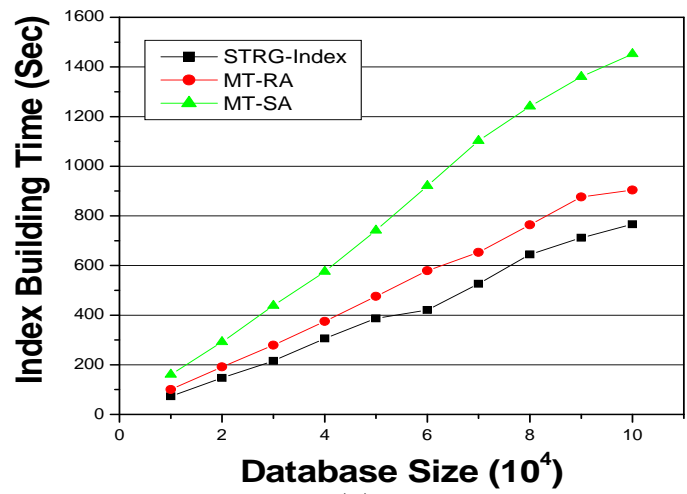
Index and MT have a similar tree structure. This is because the complexities of index construction for STRG-Index and MT are different; the STRG-Index is $O(KM)$ (Section V.B) and the MT is $O(RM \log_R M)$ by [82], where K is the number of cluster nodes, M is the size of the data set, and R is the overflow size. The complexity of building the STRG-Index is same as that of clustering because the index structure is built during the clustering process. However, the MT uses a split procedure during the index construction, which takes more time.

Concerning the scalability of STRG-Index with respect to the number of clusters, we compare the size of scalable STRG-Index, where actual OGs are removed, with that of the STRG-Index as the number of clusters increases from 48 to 480. Each cluster has 1,000 objects, which means the total number of objects increases from 4.8×10^4 to 4.8×10^5 . As seen in Figure 6.5 (b), the sizes of both scalable STRG-Index and general STRG-Index scale linearly with respect to the number of clusters, but the size of scalable STRG-Index is much smaller than that of general STRG-Index. This addresses the problem when the database size increases arbitrarily large.

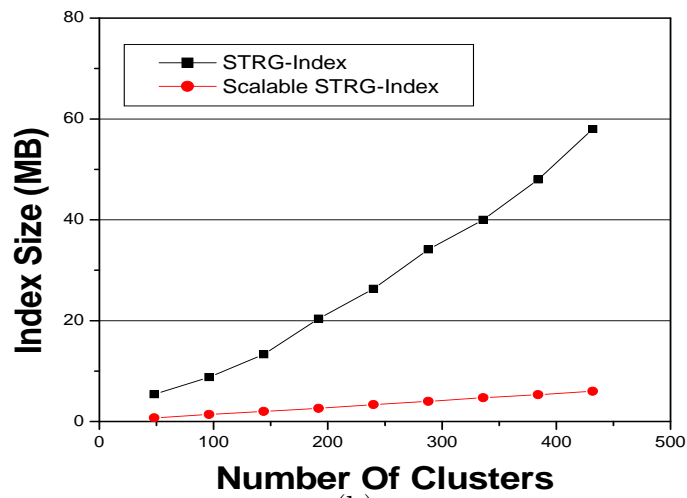
6.6 Summary

In this Chapter, we proposed a graph-based video indexing method, named as *Spatio-Temporal Region Graph Index* (STRG-Index). The STRG-Index tree structure consists of three level of nodes; shot node, cluster node and object node. Shot node contains the BGs of segmented STRGs. Each record in the node has its identifier, an actual BG, and an associated pointer. Cluster nodes contain the centroid OGs representing cluster centroids. A record in the cluster node contains its identifier, a centroid OG of each cluster, and an associated pointer. Object nodes contain OGs belonging to a cluster. A record in the node has the index key, an actual OG, and an associated pointer. The

experimental results demonstrate that the proposed STRG-Index performs remarkably well in terms of cost and speed.



(a)



(b)

Figure 6.5. Efficiency and Scalability of STRG-Index: (a) Building index time, and (b) Scalability of STRG-Index.

CHAPTER 7

STRG QUERY PROCESSING

In this chapter, we define *STRG data model* which is based on the object-oriented data model, and propose *STRG Query Language* (STRG-QL) from STRG data model. Existing Video Database Management Systems (VDBMSs) that are based on relational, object-relational, or object-oriented models, provide various types of query languages. However, a content-based video retrieval is still open problem and challenging area since the characteristics of video, i.e., spatio-temporal information of objects, are not considered properly. To address this, we first model video data using *Spatio-Temporal Region Graph* (STRG), which represents spatial and temporal information of video objects using graphs. An STRG data model is generated from STRG by exploiting object-oriented model where nodes and edges are used for the basic classes. Based on the STRG data model, we propose a new graph-based query language named STRG-QL, which is extended from object-oriented language by adding several graph operations. To process the proposed STRG-QL queries, we introduce a rule-based query optimization which considers the characteristics of video data, i.e., the hierarchical relationships among video segments. We illustrate the language features by examples.

The remainder of this chapter is organized as follows. Section 7.1, we present STRG data model to map a graph model into Object-Oriented Database. Section 7.2 shows the example of basic query statement in STRG-QL. In Section 7.3, we introduce four extended functions used in STRG-QL. Two supported query types, Query by Feature and Example, are presented in Section 7.4. Section 7.5 presents STRG query processing,

i.e., rule-based query optimization. Section 7.6 describes the experimental results of STRG-QL in detail. Section 7.7 provides the summary of this chapter.

7.1 STRG Data Model

The proposed STRG data model facilitates a high level manipulation of video content. We describe a mapping of STRG data model to Object-Oriented Database (OODB) objects using an ODMG ODL schema [83]. Figure 7.1 defines four basic classes in STRG data model; **Node**, **Edge**, **RAG**, and **STRG**. Each class represents both its class name and a type of objects belonging to the class.

- **Node** type is an ordinary object type which represents a segmented region in a video frame. It can have the attributes in which each value is a basic type, i.e., short. Objects of **Node** types are the nodes of **RAG** and **STRG**. A **Node** represents a single semantic object in a frame.
- **Edge** type is a sequence type, and its base type is a **Node** type. Objects of **Edge** type must be sequences of two **Nodes**. It can have the additional attributes in which each value is a basic type such as string and double. Objects of **Edge** types are the edges of **RAG** and **STRG**. An **Edge** represent either spatial relationship (adjacency) or temporal relationship (tracking) between two objects in a video.
- **RAG** type is a tuple type with two special attributes, **Node** and **Edge** which contain a set of node and edge objects, respectively. A **RAG** represents spatial information among objects in a frame.
- **STRG** type is a tuple type with three special attributes, **Node** and two types of **Edges**. One **Edge** is for spatial edges, and the other is for temporal edges. In an **STRG** object, its **Node** and **Edge** attributes contain a set of node and edge objects, respectively. A **STRG** represents spatial and temporal relationships among objects in a video segment.

```

class Node (extent Nodes)
{ attribute Short location;
  attribute Short size;
  attribute Short color;
};

class Edge (extent Edges)
{ attribute String e_type;
  attribute set<Node> nodes;
  attribute Double velocity;
  attribute Double direction;
};

class RAG (extent RAGs)
{ attribute set<Node> nodes;
  attribute set<Edge> Sedges;
};

class STRG (extent STRGs)
{ attribute set<Node> nodes;
  attribute set<Edge> sedges;
  attribute set<Edge> tedges;
};

```

Figure 7.1. Four basic classes in STRG data model.

We now define the ODL schema which describes a video database including videos, shots, moving objects, and backgrounds. Figure 7.2 shows four classes of the schema: **Video**, **Shot**, **OG** and **BG**. For every class we declare an extent, which refers to the current collection of all objects in that class. Each raw video is modeled as **Video** object. A **Video** provides some attributes describing the basic characteristics of a raw video data, i.e., video name and a set of shots that a video contains. A **Shot** models each segmented shot in a video. It provides some attributes describing a set of RAGs representing frames, a key RAG (frame), and moving objects and background belonging to it. An **OG** and a **BG** model individual object graph and background graph respectively, mentioned in Section 4.2.

7.2 Basic Query Statement

The query language STRG-QL is an OQL-like query enriched with the constructs to create, manipulate and query graph types of video objects. An STRG-QL is a superset of OQL. It recognizes the syntax of OQL but it also provides additional operators, functions and predicates to manipulate STRG data model.


```

class Video (extent Videos
              key v_id)
{ attribute Short v_id;
  attribute String name;
  relationship set<Shot> shots
    inverse Shot::video;
  other attributes;
};

class OG (extent OGs key
          o_id)
{ attribute Short o_id;
  attribute set<Node> nodes;
  attribute set<Edge> tedges;
  relationship Shot shot
    inverse Shot::ogs;
  other attributes;
};

class Shot (extent Shots
             key s_id)
{ attribute Short s_id;
  attribute set<RAG> frames;
  attribute RAG keyframe;
  relationship Video video
    inverse Video::shots;
  relationship set<OG> ogs
    inverse OG::shot;
  relationship BG bg
    inverse BG::shot;
  other attributes;
};

class BG (extent BGs key b_id)
{ attribute Short b_id;
  attribute set<Node> nodes;
  attribute set<Edge> sedges;
  attribute set<Edge> tedges;
  relationship Shot shot
    inverse Shot::bg;
  other attributes;
};

```

Figure 7.2. ODL schema using STRG data model.

As in OQL, STRG-QL uses the traditional “*select ... from ... where ...*” statements for querying. However, each clause has an extended meaning in terms of graphs. An STRG-QL query can be expressed by the following structure.

- SELECT: target of query
- FROM: range (or source) of query
- WHERE: condition (or predicate) of query

In SELECT clause, the target of a query is specified. The possible types are any classes defined in Figure 7.2, i.e., videos (**Video**), shots (**Shot**), moving objects (**OG**), backgrounds (**BG**), their constructors (i.e., key frames (Shot.keyframe), video identifier (Video.v_id)), etc. Also, video browsing functions can be used in SELECT clause, i.e., SUMMARY() or MAKECLIP(). The range (or source) of query is specified in FROM

clause, which defines the search space of a query. It can be a set of any classes in Figure 7.2. If the users have no idea about the possible sources where the target may come from, the symbol “*” can be used to represent all objects in database. In addition, a query video given by a user is specified in this clause, which makes STRG-QL support Query by Example. The qualification of a query is specified in WHERE clause. Objects in the FROM clause are evaluated by the specified predicates to get the results. Here is a typical example of STRG-QL.

**Q1: Find key frames of all shots which belong to
the video id = 1**

```
select s.keyframe
from s in Shots
where s.vid = 1;
```

This simple query retrieves the key frame of each shot. Full descriptions of STRG-QL syntax will be discussed in the following subsection.

7.3 Extended Functions

The proposed extended functions in STRG-QL are related to graph operations (i.e., graph matching and subgraph isomorphism), and video presentation (i.e., making a clip or video summary). Unlike existing video query languages which proposed spatial or temporal predicates, the proposed STRG-QL does not need to consider those predicates since STRG data model implicitly includes spatial and temporal information. In the following, new functions in STRG-QL are introduced.

7.3.1 GDM()

In order to compute the distance (dissimilarity) between two RAGs or BGs, we define a graph matching algorithm, called *Graph Dissimilarity Measure* (GDM), which uses the *maximal common subgraph* [3]. The graph dissimilarity measure *GDM* between two graphs G_1 and G_2 can be defined as follows.

Definition 18. *The Graph Dissimilarity Measure (GDM) between G_1 and G_2 is defined as:*

$$GDM(G_1, G_2) = 1 - \frac{|G_C|}{\max(|G_1|, |G_2|)}$$

where $|G|$ denotes the number of nodes of G , and G_C is the maximal common subgraph of G_1 and G_2 .

In Definition 18, G_C can be computed based on maximal common subgraph mentioned in Algorithm 1 (see Figure 3.5). In GDM(), the possible operands are any types of graphs (i.e., BG, RAG or STRG).

Q2: Find s_id of all shots of which background and key frame are same.

```
select b.shot.s_id
from b in BGs
where GDM(b.nodes, b.shot.keyframe.node) <  $\delta$ ;
```

In this query, all shots are returned, of which its background and key frame are same. Using GDM() to compare two RAGs or BGs, we allow a certain error margin to find similar graphs, since it is rarely happened to find identical graphs in STRG data model. In this case, we use a certain threshold value (δ) for the error margin.

7.3.2 EGED()

Since GDM() is designed for general purpose of graph matching such as RAGs and BGs, it is not suitable to compare two OGs which are time-varying data. In order to address this, we use a proposed distance function, EGED() between two OGs in Section 5.1.

Q3 gives a list of OGs whose moving pattern is same as those of the shot ($s_id = 1$) by EGED(). We allow a certain threshold value (ε) for the error margin like using δ of GDM().

Q3: Find o_id of all moving objects which have the same moving pattern to moving objects in the shot id = 1

```
select o.o_id
from o in OGs,
      q in ( select a.ogs from a in Shots
             where a.s_id = 1 )
where EGED(o.nodes, q.nodes) <  $\varepsilon$  ;
```

7.3.3 SUMMARY()

Since consecutive frames in a segmented shot are little different, it is efficient to summarize a long video without loss of main semantics when the frames are compared or browsed. In order to summarize videos or shots, we exploit a scene tree construction technique proposed in [55], and extend it to STRG data model. SUMMARY() requires three parameters as input, i.e., source object (a set of frames, shots or videos), summary level (L), and summary length (T_{len}). It returns a set of frames which is a summary without loss of original semantics. The detailed algorithm for the summarization can be seen in [55].

Q4: Summarize a video id = 1 with Level = 1 and $T_{len} = 0.9$

```
select SUMMARY(v.shots, 1, 0.9)
from v in Video
where v.v_id = 1;
```

Q4 returns the summary of a video. Moreover, it can be used in FROM and WHERE clauses to reduce the processing time.

7.3.4 MAKECLIP()

MAKECLIP() is a browsing function to generate a video clip from input objects such as RAGs, Shots, and Videos. Since STRG data model is expressed as types of graphs, it is inconvenient to verify the retrieved data with nodes and edges. MAKECLIP() is used in SELECT clause to help the visualization of query results. It has two parameters as operands, i.e., source objects and a destination of generated clip. Q5 shows an example of MAKECLIP() to generate a video clip from retrieved key frames.

Q5: Find key frames of all shots which belong to the video id = 1, then generate clip to C:\out.avi

```
select MAKECLIP( s.keyframe, C:\out.avi )
from s in Shots
where s.vid = 1;
```

7.4 Supported Query Types

In this section, we present two main query types that STRG-QL supports; Query by Feature (QBF) and Query by Example (QBE). While QBF is the basic query type

which is compatible to ODMG OQL, QBE supports a query with a sample video clip. We provide some examples of each query type.

7.4.1 Query By Feature

This type of query is used to retrieve salient objects from database that satisfy the conditions given by feature values. Since the query uses feature values of objects in database, it is fully compatible to a standard OQL. A typical example of QBF is given below:

Q6: Find moving objects of all OGs of which node has the following trajectory: ([10,60], [30,60], [50,60], [70,60], [90,60], [110,60])

```

select o.o_id
from o in OGs
where EGED(o.nodes,
            set(struct Node{ [10,60], nil, nil },
              struct Node{ [30,60], nil, nil },
              .....
              struct Node{ [110,60], nil, nil } )) <  $\epsilon$  ;

```

In this query, a set of Nodes is constructed from given feature values by following the locations of moving object. The constructed nodes are compared to each OGs.nodes to find salient objects. However, as seen in Q6 it is inconvenient to make a query statement using feature values. Sometimes it is not possible to make an appropriate query statement from given conditions. In order to address this, we propose more convenient query type using examples.

7.4.2 Query By Example

Query By Example (QBE) was developed originally by IBM in the 1970s to help users in their retrieval of information from the database using query templets [84]. We employ the idea of QBE to address the limitation of QBF. QBE in STRG-QL makes users retrieve data using a sample video clip. In other words, a sample video clip instead of feature values is given to a query. In order to support a query video, the FROM expression in OQL grammar is extended as follows.

```

<from_clause> := <variable_name> IN <expression>
               | <variable_name> IN <expression>, <from_clause>
               | <variable_name> IN <expression> OF <videolist>
<videolist> := [<videolist> ',' ] <name>
<name> := '[A-Za-z][A-Za-z0-9_#]*'

```

The query videos are assigned to OF clause. When a query has multiple videos, a comma is used for the separator. If OF clause is recognized in a compile time, a query processor creates temporary objects for a query video (i.e., **tmpVideo**, **tmpShot**, **tmpOG** and **tmpBG**) using STRG data model in memory. During the query execution phase, each data object is extracted from the query videos before retrieval. The temporary objects are removed from the memory after the query is completed. The temporary objects, **tmpVideo**, **tmpShot**, **tmpOG** and **tmpBG**, are inherited from **Video**, **Shot**, **OG** and **BG**, respectively.

In Q7, all moving objects in a query video (i.e., 'query.avi') are first extracted by STRG producing module, and will be stored at **tmpOG** as OG type. Then, the remaining query is processed like a normal query. Other STRG objects, such as **tmpVideo**, **tmpShot**, and **tmpBG**, can be created in memory if needed.

Q7: Find moving objects of OGs in a video with $v_id = 1$ whose moving patterns and background are the same as moving objects and background in a given video clip (query.avi)

```
select o.o_id
from v in Videos, o in OGs,
      q in tmpOGs of 'query.avi'
where v_id = 1 and
      o.shot in v.shots and
      EGED(o.nodes, q.nodes) <  $\epsilon$  and
      GDM(o.shot.bg, q.shot.bg) <  $\delta$  ;
```

7.5 STRG Query Processing

An STRG-QL query is processed in the query processor to retrieve data from a database, which is called a *STRG query processing*. Figure 7.3 shows four main phases of STRG query processing: query recognition, query decomposition, query optimization, and query execution. Query recognition performs syntactic and semantic checks of input query, and generates a parse tree. Query decomposition constructs a query tree from a parse tree using algebra expressions. The query tree is optimized for an efficient execution in query optimization phase. Finally, the query execution retrieves the data in database according to the execution plan. A set of results is returned to a user who requests the query. Among query processing phases, the most important one is query optimization since it determines the overall performance of STRG query processing. In this section, we focus on the query optimization strategy. First, we introduce a new index structure for STRG data model, which is called *STRG-Index*. Then, a rule-based optimization strategy of STRG-QL is discussed.

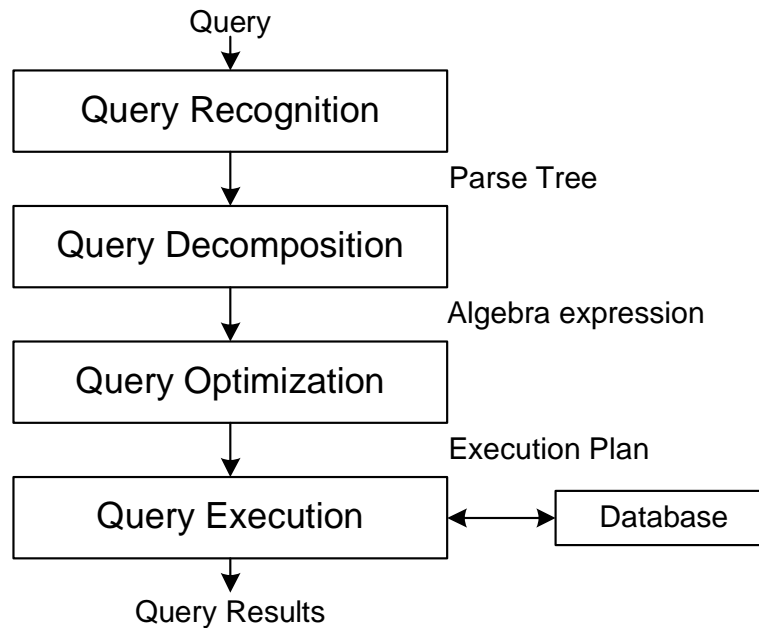


Figure 7.3. STRG Query processing phases.

7.5.1 Rule-based Query Optimization

A query optimization is the activity of choosing an efficient execution strategy for processing a given query. Since STRG data model deals with video data which have the hierarchical characteristics, such as video, scene, shot, and frame, we need a different strategy to process STRG-QL. For example, suppose that a query includes two join operations such as video to video or frame to frame levels. A conventional query processing uses a statistical information to reduce the cost of these operations. However, video level operation can be processed in advance without investigating additional information because video always has lower cardinality than frame does. Therefore, using logical and hierarchical structure of a video not only saves a cost of a query optimization, but also improves performance of a query execution. The rules of query optimization in STRG-QL are as follows:

Rule 1: Perform selection operation as early as possible. Like other query optimization, selection can reduce the cardinality of object. Therefore, it makes sense to move the selection operations as far down the tree as possible.

Rule 2: Perform the operations related to OGs after all operations related to Video and Shot operation. Since OG is an object with high cardinality, OG should be cascaded before it is processed. In order to do this, Rule 2 moves the operations related to OG later than others.

Rule 3: Perform extended operations of STRG-QL as late as possible. In general, the extended functions of STRG-QL (i.e., GDM(), EGED(), SUMMARY() and MAKECLIP()) mentioned in Section 5.3 have a significant amount of processing compared with conventional operations of OQL. Therefore, it is better to perform them as late as possible to reduce the cost of extended functions.

Rule 4: Use STRG-Index for the operations related to Shot, BG and OG. In order to avoid a sequential scanning in selection, and cartesian product in join operation, we use an STRG-Index mentioned in previous subsection.

Using above rules, we build an execution plan for the example query in Q7. First, Q7 query is decomposed into the following subqueries:

- Subquery 1: $v_id = 1$ and $o.shot$ in $v.shots$
- Subquery 2: $EGED(o.nodes, q.nodes) < \varepsilon$
- Subquery 3: $GDM(o.shots.bg, q.shots.bg) < \delta$

By Rule 1 and 2, Subquery 1 is executed at first. Then, Subquery 3 performs a join operation with GSM() between the output of Subquery 1 and \mathbf{q} (tmpOGs) according to Rule 3. Subquery 2 is carried out after Subquery 3 because of Rule 2. Figure 7.4 shows the execution plan constructed by our rule-based query optimization. For the selection

operations (σ) and the join operations (\bowtie), two STRG-Index for \mathbf{v} and \mathbf{q} can be used for more efficient query processing.

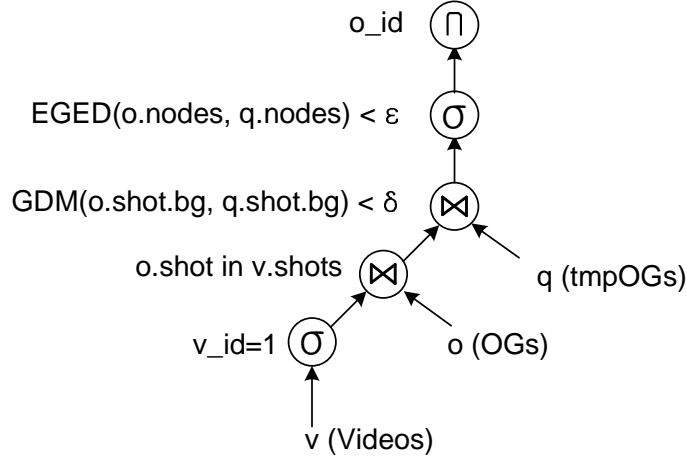


Figure 7.4. Execution plan of Q7 query.

7.6 Experimental Results

7.6.1 Performance of GDM() and EGED()

We first evaluate the performance of the proposed graph operational functions (GDM() and EGED()) on the synthetic data set, which are the main features to decide the performance of STRG-QL. Example query Q2 and Q3 are used for the evaluation of GDM() and EGED(), respectively, since Q2 and Q3 have those functions. Figure 7.5 shows the precision-recall values for two queries. The precision of Q2 (GDM()) is over 60% over any values of recall. And, the precision of Q3 (EGED()) is over 80% over any values of recall. These indicate that the proposed functions (GDM() and EGED()) are nearly optimal to find the similar objects. The performance of GDM() is a little less than that of EGED() since GDM() is designed for general purpose.

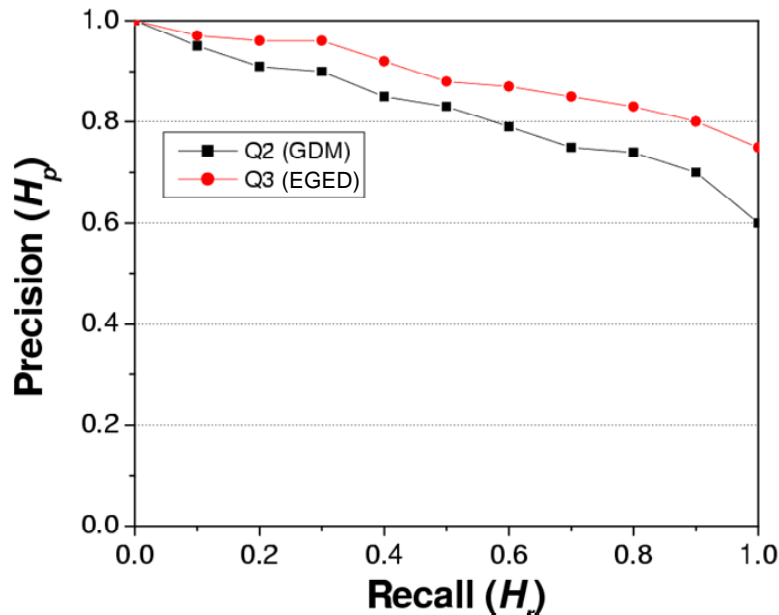


Figure 7.5. Performance of GDM() and EGED().

7.6.2 Performance of Optimization

Figure 7.6 shows the number of disk accesses to validate the efficiency of a rule-based query optimization. We use the same k-NN query in the previous subsection. We count the number of disk accesses when a query is executed with and without query optimization. As seen in the figure, a rule-based query optimization can reduce the significant number of disk accesses, which makes the performance of STRG-QL better.

7.6.3 Accuracy of Retrieval

To demonstrate the overall accuracy of STRG-QL and its query processing, we use the real videos in Table 3.1. First, all the videos are processed by the proposed STRG processing techniques mentioned in Chapter 3 and 4. The outputs are stored at the repository as the forms of STRG data model, i.e., Video, Shot, OG and BG. Then, STRG-Index is constructed for all the videos (see Section 6.1). We use a k-NN

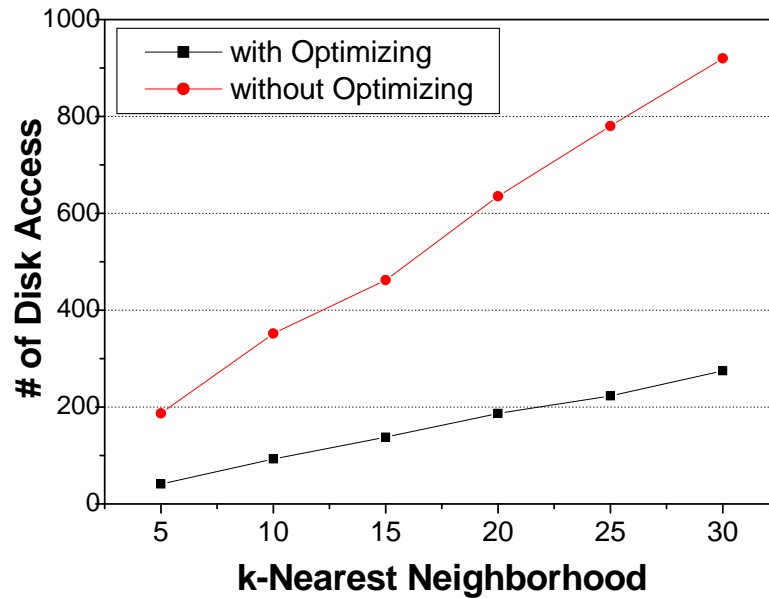


Figure 7.6. Performance of Optimization.

query which is an extension of example query Q7. The k-NN query can be expressed by STRG-QL as follows.

```

select o.o_id
from o in OGs,
      q in tmpOGs of 'query.avi'
where EGED(o.nodes, q.nodes) <  $\varepsilon$  and
      GDM(o.shot.bg, q.shot.bg) <  $\delta$  and
      rownum < k
order by EGED(o.nodes, q.nodes) asce

```

where k is a number of returned data, $\varepsilon = 0.1$ and $\delta = 0.1$. Two query videos are selected from the database; the first query video is from surveillance group, and the second is from produced group in Table 3.1. The results are verified with the ground truth. As seen in Figure 7.7, we observe that the precision of the first query is over 80%,

and that of the second query is over 70% at any values of recall. In other words, the proposed STRG-QL provides consistent accuracy for various video types.

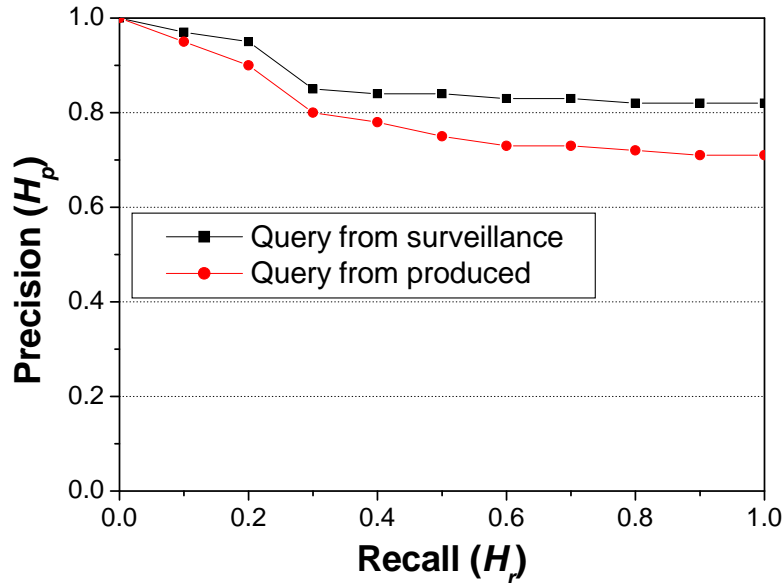


Figure 7.7. Accuracy of STRG-QL.

Figure 7.8 shows a result of the first query when $k = 2$. The first row shows some selected frames of the query video. Two matched videos are in the second and third rows. As seen in Figure 7.8, STRG-QL is able to find the objects with similar moving pattern.

7.7 Summary

In this chapter, we introduce a graph-based query language STRG-QL and its query processing for content-based video retrieval system. In order to develop a general-purpose video query language, first we model video data using Spatio-Temporal Region Graph (STRG). An STRG data model is generated from STRG using object-oriented model where nodes and edges are used for the basic classes. Based on the STRG data model, we propose a new graph-based query language named STRG-QL, which is an extension

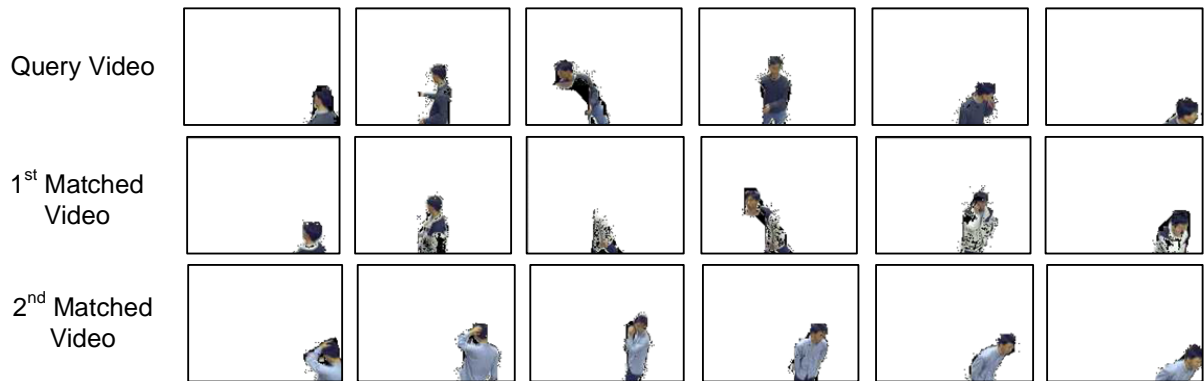


Figure 7.8. Result of k-NN query using STRG-QL ($k=2$).

of OQL by adding several graph operations. To process the proposed STRG-QL queries, we introduce a rule-based query optimization which considers the characteristics of video data. Some query examples are given to illustrate STRG-QL and its query processing. Experimental results on both synthetic data and real video streams show the effectiveness and accuracy of the proposed approaches.

CHAPTER 8

CONCLUSIONS AND FUTURE RESEARCH

The use of multimedia data in many applications has increased enormously with the recent advances in video capture device, storage, network technology and computing power. As a consequence, there are increasing demands on modeling, indexing, and retrieving these data. However, since the huge amount of data size and its complexity have restricted the progress on video data management systems. This dissertation address the problems of video data management systems by using a graph, such as video data modeling, indexing, and query processing. In this chapter, I summarize the contributions of this dissertation, and provide future research directions based on my work.

8.1 Summary of Contributions

In this dissertation, we proposed a new graph-based video data structure, named *Spatio-Temporal Region Graph* (STRG). STRG can represent spatio-temporal features and the correlations among the video objects. A *Region Adjacency Graph* (RAG) is generated from each frame, and an STRG is constructed by connecting RAGs. An STRG is segmented into a number of pieces based on its content for efficient processing. Then, each segmented STRG is decomposed into its subgraphs, called *Object Graph* (OG) and *Background Graph* (BG) in which redundant BGs are eliminated to reduce index size and search time.

In addition, we proposed a new indexing of OGs by clustering them using unsupervised learning algorithms for more accurate indexing. In order to perform the clustering, we need a distance measure between two OGs. For the distance measure, we propose a

new measure, *Extended Graph Edit Distance* (EGED) because the existing measures are not very suitable for OGs. The EGED is defined in non-metric space for clustering OGs, and it is extended to metric space to compute the key values for indexing. Based on the clusters of OGs and the EGED, we propose a new indexing structure *STRG-Index* which provides efficient retrieval.

Finally, we proposed a graph-based query language named STRG-QL by extending object-oriented language. To process the STRG-QL queries, we introduced a rule-based query optimization that considers the hierarchical relationships among video segments. For more efficient query processing, we present STRG-Index structure and show how to use it during query processing.

8.2 Future Research Directions

In the future, I would like to move from modeling and indexing multimedia data towards developing multimedia database system and its applications of various data types, which support a high-level user request. I am also interested in studying knowledge discovery to find hidden concepts within complex types of data, such as high-dimensional spatio-temporal data. To achieve this, I would rather use sound mathematical approach that emphasizes understanding and digs to the bottom, while still avoiding pure scholasticism, keeping in mind practical applications. I anticipate that the applications of this research will be numerous and diverse, for the simple reason that accurate modeling and indexing of multimedia data have immediate relevance to any systems that involves spatial, temporal, or spatio-temporal data. The remainder of this section briefly describes some specific potential research projects.

- **Developing GVDBMS: Graph-based Video Database Management System.** Based on the algorithms and techniques developed in my research work, I plan to develop a Graph-based Video Database Management System (GVDBMS)

that integrates graph-based modeling, video data mining, ontology, STRG-Index, and query processing. Developing GVDBMS, I am going to address the following issues: (1) The system can handle not only low-level feature values of the data, but also high-level human perceptions, i.e., concept of data. To support the high-level human perceptions, we extend conceptual query to STRG-QL that integrates with STRG-Index, (2) The system should be scalable to the size of data, since the huge amount of multimedia data may restrict to improve performance of the system, which is not acceptable for the practical systems, and (3) The system can be easily extended to other applications whose data types can be represented as a graph (STRG): e.g. hurricane tracks data, and mobile agent in sensor network.

- **Investigating a broad set of applications of GVDBMS.** The primary benefit of using a graph is that it can represent spatial and temporal relationships among semantically rich and complicated data. Thus, I extend GVDBMS to other application domains where the data is represented by a graph-based modeling. The first application domain is medical videos, specifically colonoscopy video and capsule endoscopy video. Unlike produced videos that consist of a group of shots, the medical videos are usually generated by a single camera operation without shot, which makes it difficult to manage and analyze them. However, a graph-based approach may model and index these types of videos since it is based on both low-level features and relationships among the data. The second application domain is high-dimensional spatio-temporal data, specifically hurricane track data and mobile agent in sensor network. The application systems generate their own ontologies to represent concepts and knowledge of the data. Using the generated ontology, the system can support high-level user requests, i.e., conceptual queries.
- **Investigating data mining techniques that discover Knowledge of data.** The formal concept used in the ontology and conceptual clustering is mathematical

representations of the concepts. In other words, it is still far from human perception that is based on linguistic terms. To reduce the gap, I plan to continue research on data mining techniques to discover and learn the knowledge of data, which should be closer to human understating. Among many data mining algorithms, I am interested in unsupervised learning algorithm, i.e., clustering, since almost all multimedia data are unlabeled, and semantically rich. Moreover, I am also interested in mathematical approaches for knowledge representation, i.e., formal concept analysis, and lattice theory. Both mathematical approach and data mining technique will be integrated into a knowledge discovery system.

REFERENCES

- [1] D. Conte, P. Foggia, C. Sansone, and M. Vento, “Graph matching applications in pattern recognition and image processing,” *Proc. of ICIP '03*, pp. 14–17, 2003.
- [2] S. Lu, M. Lyu, and I. King, “Video Summarization by Spatial-Temporal Graph Optimization,” in *Proceedings of the 2004 International Symposium on Circuits and Systems*, vol. 2, Vancouver, Canada, May 2004, pp. 197–200.
- [3] H. Bunke and K. Shearer, “A Graph Distance Metric based on the Maximal Common Subgraph,” *Pattern Recognition Letters* 19, pp. 255–259, 1998.
- [4] O. Miller, E. Navon, and A. Averbuch, “Tracking of moving objects based on graph edges similarity,” *Proceedings of the ICME '03*, pp. 73–76, 2003.
- [5] C. Yuan, Y.-F. Ma, and H.-J. Zhang, “A Graph-Theoretic Approach to Video Object Segmentation in 2D+t Space,” MSR, Tech. Rep., March 2003.
- [6] H. T. Chen, H. Lin, and T. L. Liu, “Multi-object tracking using dynamical graph matching,” *Proc. of the 2001 IEEE Conf. on CVPR*, pp. 210–217, 2001.
- [7] C. Gomila and F. Meyer, “Tracking Objects by Graph Matching of Image Partition Sequences,” *Proc. of 3rd IAPR-TC15 Workshop on GRPR*, pp. 1–11, 2001.
- [8] R. Hjelsvold and R. Midtstraum, “Modelling and Querying Video Data,” in *Proceedings of the Twentieth International Conference on Very Large Databases*, Santiago, Chile, 1994, pp. 686–694.
- [9] D. Montesi and H. Elmagarmid, “Videotext database systems,” in *Proceedings of the 1997 International Conference on Multimedia Computing and Systems (ICMCS '97)*. IEEE Computer Society, 1997, p. 344.

- [10] F. Kokkoras, H. Jiang, I. Vlahavas, A. K. Elmagarmid, E. N. Houstis, and W. G. Aref, "Smart videotext: a video data model based on conceptual graphs," *Multimedia Systems*, vol. 8, no. 4, pp. 328–338, 2002.
- [11] A. Hanjalic, R. L. Lagendijk, and J. Biemond, "Automated high-level movie segmentation for advanced video-retrieval systems," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 9, no. 4, pp. 580–588, 1999.
- [12] W. Mahdi, M. Ardebilian, and L. M. Chen, "Automatic video scene segmentation based on spatial-temporal clues and rhythm," in *International Journal of Networking and Information Systems*, January 2001.
- [13] C.-W. Ngo, Y.-F. Ma, and H.-J. Zhang, "Video summarization and scene detection by graph modeling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 2, pp. 296–305, February 2005.
- [14] Y. F. Day, S. Dagtas, and M. Iino, "Object-oriented conceptual modeling of video data," in *Proc. of 11th Int'l Conference on Data Engineering*, Taipei, Taiwan, March 1995, pp. 401–408.
- [15] S. Chang, W. Chen, H. J. Meng, H. Sundaram, and D. Zhong, "Content-based Video Search Engine Supporting Spatio-Temporal Queries," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 5, pp. 602–615, Sep. 1998.
- [16] B. Gedik, K.-L. Wu, P. Yu, and L. Liu, "Motion Adaptive Indexing for Moving Continual Queries over Moving Objects," in *CIKM*, 2004, pp. 427–436.
- [17] X. Zhu, X. Wu, A. K. Elmagarmid, Z. Feng, and L. Wu, "Video Data Mining: Semantic Indexing and Event Detection from the Association Perspective," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 5, 2005, pp. 665–677.
- [18] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," in *Proceedings 1994 ACM SIGMOD*

- Conference, Mineapolis, MN, 1994, pp. 419–429. [Online]. Available: cite-seer.ist.psu.edu/faloutsos94fast.html*
- [19] H. Gish and K. Ng, “Parametric Trajectory Models for Speech Recognition,” in *Proc. of The Fourth International Conference on Spoken Language Processing*, Philadelphia, PA, October 1996.
- [20] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithm*. Massachusetts: The MIT Press, 2001.
- [21] H. Bunke, “On a relation between graph edit distance and maximum common subgraph,” *Pattern Recogn. Lett.*, vol. 18, no. 9, pp. 689–694, 1997.
- [22] D. H. Fisher, “Knowledge Acquisition Via Incremental Conceptual Clustering,” *Machine Learning*, vol. 2, no. 2, pp. 139–172, 1987.
- [23] K. McKusick and K. Thompson, “COBWEB/3: A Portable Implementation,” *Technical Report No. FIA-90-6-18-2*, pp. 139–172, 1990.
- [24] Y. Reich and S. J. Fenves, “The Formation and Use of Abstract Concepts in Design,” *Concept formation knowledge and experience in unsupervised learning*, pp. 323–353, 1991.
- [25] P. Cheeseman and J. Stutz, “Bayesian classification (AutoClass): theory and results,” *Advances in knowledge discovery and data mining*, pp. 153–180, 1996.
- [26] L. Talavera and J. J. Béjar, “Generality-Based Conceptual Clustering with Probabilistic Concepts,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 196–206, February 2001.
- [27] W. W. Chu, K. Chiang, C.-C. Hsu, and H. Yau, “An Error-based Conceptual Clustering Method for Providing Approximate Query Answers,” *Communications of the ACM*, vol. 39, no. 12es, p. 216, 1996.

- [28] C. Böhm, S. Berchtold, and D. A. Keim, “Searching in High-Dimensional Spaces: Index Structures for Improving The Performance of Multimedia Databases,” *ACM Comput. Surv.*, vol. 33, no. 3, pp. 322–373, 2001.
- [29] Y. Theodoridis, M. Vazirgiannis, and T. K. Sellis, “Spatio-temporal indexing for large multimedia applications,” in *International Conference on Multimedia Computing and Systems*, 1996, pp. 441–448. [Online]. Available: citeseer.ist.psu.edu/article/theodoridis96spatiotemporal.html
- [30] C. S. Jensen, D. Lin, and B. C. Ooi, “Query and Update Efficient B+-Tree based Indexing of Moving Objects,” in *VLDB*, 2004, pp. 768–779.
- [31] X. Xu, J. Han, and W. Lu, “Rt-tree: An improved r-tree index structure for spatiotemporal databases,” in *4th International Symposium on Spatial Data Handling(SDH)*, January 1990, pp. 1040–1049.
- [32] P. Ciaccia, M. Patella, and P. Zezula, “M-tree: An efficient access method for similarity search in metric spaces,” in *the VLDB Journal*, 1997, pp. 426–435. [Online]. Available: citeseer.ist.psu.edu/article/ciaccia97mtree.html
- [33] S. Hibino and E. A. Rundensteiner, “MMVIS: Design and Implementation of a Multimedia Visual Information Seeking Environment,” in *ACM Multimedia*, 1996, pp. 75–86.
- [34] T. C. T. Kuo and A. L. P. Chen, “Content-Based Query Processing for Video Databases,” *IEEE Transactions on Multimedia*, vol. 2, no. 1, pp. 1–13, 2000.
- [35] E. Oomoto and K. Tanaka, “Ovid: Design and implementation of video-object database system,” in *IEEE Transactions on Knowledge and Data Engineering*, Aug. 1993, pp. 629–643.
- [36] M. Erwig and M. Schneider, “Spatio-Temporal Predicates,” *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 4, pp. 881–901, 2002.

- [37] W. G. Aref, M. A. Hammad, A. C. Catlin, I. F. Ilyas, T. M. Ghanem, A. K. Elmagarmid, and M. S. Marzouk, "Video query processing in the VDBMS testbed for video database research," in *MMDB*, 2003, pp. 25–32.
- [38] Ö. Ulusoy, U. Güdükbay, M. E. Dönderler, E. Saykol, and C. Alper, "BilVideo Video Database Management System," in *VLDB*, 2004, pp. 1373–1376.
- [39] J. Li, M. T. Ozsü, and D. Szafron, "Modeling of moving objects in a video database," in *International Conference on Multimedia Computing and Systems*, 1997, pp. 336–343. [Online]. Available: citeseer.ist.psu.edu/li97modeling.html
- [40] L. Sheng, Z. M. Özsoyoglu, and G. Özsoyoglu, "A Graph Query Language and Its Query Processing," in *ICDE*, 1999, pp. 572–581.
- [41] L. Chen, M. T. Özsu, and V. Oria, "Modeling of Moving Objects in a Video Database," in *Proceedings of the IEEE ICME 2002*, Lausanne, Switzerland, 2002, pp. 217–221.
- [42] A. Hotho and G. Stumme, "Conceptual Clustering of Text Clusters," in *Proceedings of FGML Workshop*, 2002.
- [43] T. T. Quan, S. C. Hui, and T. H. Cao, "FOGA: A Fuzzy Ontology Generation Framework for Scholarly Semantic Web," in *Proceedings of the 2004 Knowledge Discovery and Ontologies Workshop*, September 2004, pp. 37–48.
- [44] R. Wille, "Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts," *In: Ival Rival (ed.), Ordered Sets*, pp. 445–470, 1982.
- [45] B. Ganter and R. Wille, *Formal Concept Analysis - Mathematical Foundations*. Berlin - Heidelberg: Springer-Verlag, 1999.
- [46] J. Y. Pan, H. J. Yang, C. Faloutsos, and P. Duygulu, "Automatic multimedia cross-modal correlation discovery," in *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM Press, 2004, pp. 653–658.

- [47] D. Comanicu and P. Meer, “Mean shift: A robust approach toward feature space analysis,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, May 2002.
- [48] Y. Deng, , and B. S.Manjunath, “Unsupervised segmentation of color-texture regions in images and video,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI '01)*, vol. 23, no. 8, pp. 800–810, Aug 2001.
- [49] M. Abramoff, P. Magelhaes, and S. Ram, “Image processing with imagej,” *Biophotonics International*, vol. 11, no. 7, pp. 36–42, 2004.
- [50] R. Hammoud and R. Mohr, “Probabilistic hierarchical framework for clustering of tracked objects in video streams,” in *Irish Machine Vision and Image Processing Conference*, The Queen’s University of Belfast, Northern Ireland, August 2000, pp. 133–140. [Online]. Available: <http://www.inrialpes.fr/movi/publi/Publications/2000/HM00c>
- [51] L. M. Fuentes and S. A. Velastin, “People tracking in surveillance applications,” in *Proc. of Second IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, Kauai, Hawaii, December 2001.
- [52] G. Levi, “A Note on the Derivation of Maximal Common Subgraphs of Two Directed or Undirected Graphs,” *Calcols 9*, pp. 341–354, 1972.
- [53] J. J. McGregor, “Backtrack Search Algorithms and the Maximal Common Subgraph Problem,” *Software Practice and Experience*, vol. 12, pp. 23–34, 1982.
- [54] H. Bunke, P. Foggia, C. Guidobaldi, C. Sansone, and M. Vento, “A Comparison of Algorithms for Maximum Common Subgraph on Randomly Connected Graphs,” in *SSPR/SPR*, 2002, pp. 123–132.
- [55] J. Oh and K. A. Hua, “An efficient and cost-effective technique for browsing and indexing large video databases,” in *Proc. of 2000 ACM SIGMOD Intl. Conf. on Management of Data*, Dallas, TX, May 2000, pp. 415–426.

- [56] D. Pelleg and A. Moore, “X-means: Extending k-means with efficient estimation of the number of clusters,” in *Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco: Morgan Kaufmann, 2000, pp. 727–734.
- [57] M. Vlachos, D. Gunopulos, and G. Kollios, “Robust similarity measures for mobile object trajectories,” in *Proc. of 5th International Workshop Mobility In Databases and Distributed Systems*, Aix-en-Provence, France, September 2002, pp. 721–728.
- [58] E. Ardizzone and M. Cascia, “Automatic video database indexing and retrieval,” *Multimedia Tools and Applications*, vol. 4, pp. 29–56, 1997.
- [59] H. Yu and W. Wolf, “A visual search system for video and image databases,” in *Proc. IEEE Int’l Conf. on Multimedia Computing and Systems*, Ottawa, Canada, June 1997, pp. 517–524.
- [60] J. Adcock, A. Girgensohn, M. Cooper, T. Liu, E. Rieffel, and L. Wilcox, “FXPAL Experiments for TRECVID 2004,” in *Proceedings of TRECVID 2004 Workshop*, March 2004.
- [61] R. Tusch, H. Kosch, and L. Boszormenyi, “Videx: An integrated generic video indexing approach,” in *Proc. of ACM Multimedia 2000*, LA, CA, Oct. 2000, pp. 448–451.
- [62] E. Durucan and Tebrahimi, “Change detection and background extraction by linear algebra,” *Proceedings of The IEEE (Special Issue on Video Communications, Processing and Understanding for Third Generation Surveillance Systems)*, vol. 89, no. 10, pp. 1355–1367, Oct. 2001.
- [63] R. C. Jones, D. Dementhon, and D. S. Doermann, “Building mosaics from video using mpeg motion vectors,” *ACM Multimedia*, pp. 29–32, March 1999.
- [64] K. R. Shearer, S. Venkatesh, and D. Kieronska, “Spatial indexing for video databases,” *Journal of Visual Communication and Image Representation*, vol. 7, no. 4, pp. 325–335, December 1997.

- [65] B. T. Messmer and H. Bunke, “Subgraph isomorphism detection in polynomial time on preprocessed model graphs,” in *Recent Developments in Computer Vision*. Berlin,: Springer, 1995, pp. 373–382.
- [66] L. Chen and R. Ng, “On the marriage of lp-norms and edit distance,” in *The 30th VLDB Conference*, Toronto, Canada, 2004, pp. 1040–1049.
- [67] R. Chandramouli and V. K. Srikantam, “On mixture density and maximum likelihood power estimation via expectation-maximization,” in *Proceedings of the 2000 conference on Asia South Pacific design automation*. ACM Press, 2000, pp. 423–428.
- [68] C. Fraley and A. E. Raftery, “Model-based clustering, discriminant analysis, and density estimation,” *Journal of the American Statistical Association*, vol. 97, no. 458, pp. 611–631, June 2002.
- [69] Y. Cai and R. T. Ng, “Indexing Spatio-Temporal Trajectories with Chebyshev Polynomials,” in *Proceedings of the 2004 ACM SIGMOD*, June 2004, pp. 599–610.
- [70] T. H. Jagger, X. Niu, and J. B. Elsner, “A Space-time Model for Seasonal Hurricane Prediction,” *International Journal of Climatology*, vol. 22, pp. 451–465, March 2002.
- [71] J. B. Elsner, “Tracking Hurricanes,” *Bulletin of the American Meteorological Society*, vol. 84, pp. 353–356, 2003.
- [72] L. Chen, M. T. Özsu, and V. Oria, “Robust and Fast Similarity Search for Moving Object Trajectories,” in *Proceedings of the 2005 ACM SIGMOD*, Baltimore, MD, June 2005, pp. 491–502.
- [73] J. Lee, J. Oh, and S. Hwang, “STRG-Index: Spatio-Temporal Region Graph Indexing for Large Video Databases,” in *Proceedings of the 2005 ACM SIGMOD*, Baltimore, MD, June 2005, pp. 718–729.

- [74] M. Schneider, “Evaluation of Spatio-Temporal Predicates on Moving Objects,” in *Proceedings of the 21st International Conference on Data Engineering*, April 2005, pp. 516–517.
- [75] M. H. C. Law, A. K. Jain, and M. A. T. Figueiredo, “Feature Selection in Mixture-Based Clustering,” in *Proceedings of Advances in Neural Information Processing Systems*, December 2002, pp. 625–632.
- [76] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York, NY: Wiley-Interscience, 1991.
- [77] C. Elkan, “Using the Triangle Inequality to Accelerate K-Means,” in *ICML*, 2003, pp. 147–153.
- [78] G. Hamerly and C. Elkan, “Alternatives to the k-means algorithm that find better clusterings,” in *Proceedings of the eleventh international conference on Information and knowledge management*. ACM Press, 2002, pp. 600–607.
- [79] B. Larsen and C. Aone, “Fast and Effective Text Mining using Linear-Time Document Clustering,” in *Proceedings of the fifth ACM SIGKDD*, 1999, pp. 16–22.
- [80] B. Günsel and A. M. Tekalp, “Content-based Video Abstraction,” in *IEEE International Conference on Image Processing*, Chicago, Illinois, 1998, pp. 128–132.
- [81] X. Zhou, G. Wang, J. X. Yu, and G. Yu, “M+-tree : A New Dynamical Multidimensional Index for Metric Spaces,” in *ADC*, 2003, pp. 161–168.
- [82] E. Chavez, G. Navarro, R. Baeza-Yates, and J. L. Marroquin, “Searching in metric spaces,” *ACM Comput. Surv.*, vol. 33, no. 3, pp. 273–321, 2001.
- [83] R. G. G. Cattell and D. K. Barry, *The Object Data Standard: ODMG 3.0*. Morgan Kaufmann, 2000.
- [84] M. M. Zloof, “Query-By-Example,” *IBM Systems Journal*, vol. 16, no. 34, pp. 324–343, 1977.

BIOGRAPHICAL STATEMENT

Jeongkyu Lee was born on April 8, 1969 in Seoul, the capital city of Republic of Korea. In 1996, he received the B.S. in Mathematics Education from Sungkyunkwan University, Seoul, Korea. As soon as he finished his B.S., he worked as a junior Database Administrator in Hana bank, Korea for 5 years, and worked as a senior Database Administrator in IBM e-Business Hosting Team, Korea for 2 years. While he was working, he completed the M.S. in Computer Science at Sogang University, Seoul, Korea in 2001.

In fall 2002, he entered the Doctoral program in Computer Science and Engineering at the University of Texas at Arlington. Since then, he has been a member of Multimedia Information Group (MIG) and working with Dr. JungHwan Oh. In MIG at UTA, he has worked as a research assistant for two years.

His primary research interest is in the multimedia database management system based on graph theory including graph-based multimedia data modeling, graph-based indexing structure, query processing, and summary. His work also includes techniques for multimedia data mining, video processing, multimedia ontology, and medical imaging.

Mr. Lee is a student member of the IEEE and the ACM.