STRG-Index: Spatio-Temporal Region Graph Indexing for Large Video Databases

JeongKyu Lee
jelee@cse.uta.edu

JungHwan Oh
oh@cse.uta.edu

Sae Hwang
hwang@cse.uta.edu

Department of Computer Science & Engineering
University of Texas at Arlington
Arlington, TX 76019-0015 U.S.A.

ABSTRACT

In this paper, we propose new graph-based data structure and indexing to organize and retrieve video data. Several researches have shown that a graph can be a better candidate for modeling semantically rich and complicated multimedia 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 (i.e., images). In order to consider the temporal feature effectively and efficiently, we propose 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. The STRG is decomposed into its subgraphs in which redundant subgraphs are eliminated to reduce the index size and search time, because the computational complexity of graph matching (subgraph isomorphism) is NP-complete. In addition, a new distance measure, called *Extended Graph Edit Distance* (EGED), is introduced in both non-metric and metric spaces for matching and indexing respectively. Based on STRG and EGED, we propose a new indexing method STRG-Index, which is faster and more accurate since it uses tree structure and clustering algorithm. We compare the STRG-Index with the M-tree, which is a popular tree-based indexing method for multimedia data. The STRG-Index outperforms the M-tree for various query loads in terms of cost and speed.

1. INTRODUCTION

Recently, content-based video retrieval systems have developed for many applications, such as digital libraries, internet video search engines and surveillance systems. These systems can be divided into three categories as follows: (1) visual feature based retrieval systems [17] which use visual features of key frames to index frames, shots and scenes, (2) keyword based retrieval systems [21] which use manually extracted text information to represent the content of video segments, and (3) object based systems [4] which use sptio-temporal features among extracted objects. Three main issues discussed in the above systems are: (1) how to efficiently parse a long video into meaningful smaller units (i.e., shots or scenes), (2) how to compute the (dis)similarity between two units more accurately, and (3) how to index and retrieve these units more effectively.

For the first issue, the majority of existing techniques for video parsing use low level or high level features which are extracted from the frame level. Mostly used low level features are colors, shapes and textures. The frame level similarities are computed using these features to find segment boundaries [15, 22]. However, it is difficult to distinguish diverse semantic content in a video by using only low level features. On the other hand, we can represent the semantic content by high level features (i.e., various text information). However, extracting these high level features is a very tedious task since it can be done by manual annotations.

The second issue is to compute the (dis)similarity between two units according to the feature values extracted from each unit. Typically, the (dis)similarity is measured by the distance functions considering *time* which is the primary factor of a video. These distance functions include the traditional distance functions (i.e., Lp-norms), Dynamic Time Warping (DTW) [11], Longest Common Subsequence (LCS) [7], and Edit Distance (ED) [4]. Although traditional distance functions are easy to compute, they are not optimal to measure 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 distance functions which cannot be applied to the standard indexing algorithms.

The third issue is to index and retrieve these units effectively. There have been relatively little efforts on indexing and retrieving the units. A major difficulty is how to handle spatio-temporal relationships among the objects in these units. To address this, an indexing structure called 3DR-tree [26] was proposed. It indexes salient objects by treating the time (temporal feature) as another dimension in R-tree. However, simply treating the time as another dimension is not optimal since spatial and temporal features should be considered differently. A couple of other index structures, such as RT-tree [29] and M-tree [5], have been proposed to handle spatio-temporal features. However, they are ineffi-

SIGMOD 2005 June 14-16, 2005, Baltimore, Maryland, USA.

cient for various queries on moving objects since they cannot capture the characteristics of moving objects. Some studies [14, 20] have proposed a visual representation of video by constructing graphs. However, in these approaches only spatial features are considered. Temporal relationships among objects, which are more important characteristics of video data, are not considered thoroughly.

In order to address the above three issues, first 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) [20] is generated from each frame, and an STRG is constructed from RAGs. The STRG is decomposed into its subgraphs, called *Object Graphs* (OGs) and *Background Graphs* (BGs) in which redundant BGs are eliminated to reduce index size and search time. Then, we cluster OGs using Expectation Maximization (EM) algorithm [8] for more accurate indexing. To cluster OGs, we need a distance measure between two OGs. For the distance measure, we propose *Extended Graph Edit Distance* (EGED) because the existing measures are not suitable for the OG which is a special case of graph. 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*. Our contributions in this paper are as follows:

- We propose a new data structure, STRG for video data based on graphs. It can represent not only spatial features of video objects, but also temporal relationships among them.
- We propose a new distance function, EGED which is defined in both non-metric and metric spaces for matching and indexing respectively. It provides better accuracy.
- We propose a new indexing method, STRG-Index which provides faster and more accurate indexing since it uses tree structure and data clustering.

The remainder of this paper is organized as follows. In Section 2, we explain how to generate a RAG from each frame, how to construct an STRG from RAGs, and how to decompose STRG into OGs and BGs. In Section 3, we introduce EGED which is used for graph matching and indexing. The model-based clustering algorithm (EM) is employed to group similar OGs in Section 4. In Section 5, we propose STRG-Index for video data. The performance study is reported in Section 6. Finally, Section 7 presents some concluding remarks.

2. GRAPH-BASED APPROACH

In this section, we describe *Region Adjacency Graph* (RAG), and extend it to *Spatio-Temporal Region Graph* (STRG) which will be used for video indexing.

2.1 Region Adjacency Graph

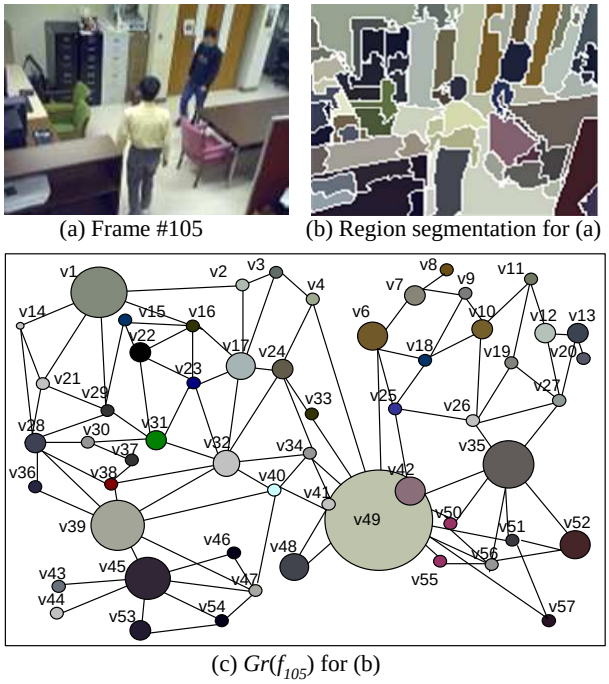
The first step in image and video processing is to decide the basic unit of processing such as pixel, block or region. Then, the features are computed from each unit for further processing. Recently, some studies focused on a graph-based approach to process image and video data [20, 23], since a graph 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.

Assume that a video segment has N frames. To divide a frame into homogeneous color regions, we use region segmentation algorithm called EDISON (Edge Detection and Image Segmentation System) [6]. The reason we choose EDISON among other algorithms is that it is less sensitive to small changes over the frames, which occur frequently in video sequence. The relationships among segmented regions can be represented by a graph, 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 adjacent nodes in f_n,*
- *$\nu : V \rightarrow A_V$ is a set of functions generating node attributes,*
- *$\xi : E_S \rightarrow A_{E_S}$ is a set of functions generating spatial edge attributes.*

A node ($v \in V$) corresponds to a region, and a spatial edge ($e_S \in E_S$) represents a spatial relationship between two adjacent nodes (regions). The possible node attributes (A_V) are size (number of pixels), color and location (centroid) of corresponding region. The spatial edge attributes (A_{E_S}) indicate relationships between two adjacent nodes such as spatial distance and orientation between centroids of two regions. A RAG provides a spatial view of regions of frame as illustrated in Figure 1.



(a) Frame #105 (b) Region segmentation for (a)

(c) $Gr(f_{105})$ for (b)

Figure 1: Example of region segmentation and RAG

Figure 1 (a) and (b) show an actual frame and its segmented regions respectively. The RAG in Figure 1 (c), $Gr(f_{105})$, is constructed according to Definition 1. Each

circle indicates a segmented region. Here, the radius, the color and the center of circle correspond to the node attributes such as size, color and location, respectively. In addition, the lines in Figure 1 (c) represent the spatial edge attributes, i.e. spatial distance and orientation between two adjacent nodes.

2.2 Spatio-Temporal Region Graph

A RAG is generated from each frame. A node representing a region can span across multiple frames. In other words, the corresponding nodes in the consecutive frames need to be connected to represent its temporal characteristic. The temporally connected RAGs are called *Spatio-Temporal Region Graph* (STRG). 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,*
- *$\tau : E_T \rightarrow A_{E_T}$ is a set of functions generating temporal edge attributes.*

In an STRG, a temporal edge ($e_T \in E_T$) represents the relationships between two corresponding nodes (regions) in two consecutive frames, such as velocity (how much their centroids are changed) and moving direction. Figure 2 shows a part of STRG for frames #104 − #106 in a sample video. The horizontal lines between the frames indicate the temporal edges.

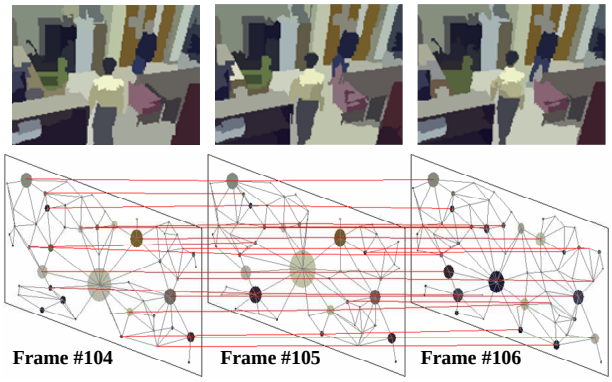


Figure 2: Visualization of STRG for frame #104 − #106

An STRG is an extension of RAGs by adding a set of temporal edges (E_T) to them. E_T represents temporal relationships between the corresponding nodes in two consecutive RAGs. Constructing E_T is similar to the problem of object tracking in video sequence.

Although there are numerous efforts [10, 13] for object tracking, it is still an open problem. The main reason is that most of the tracking algorithms use low-level features such as color, location and texture, but complicated moving patterns of 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 regions. To describe our graph-based tracking algorithm, we first define subgraph isomorphism as follows (see Definitions 3, 4 and 5).

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.*

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

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

The notation $Gr' \subseteq Gr$ is used to indicate that Gr' is a subgraph of Gr.

DEFINITION 4. *Two graphs $Gr = \{V, E_S, \nu, \xi\}$ and $Gr'' = \{V'', E''_S, \nu'', \xi''\}$ are isomorphic, denoted by $Gr \cong Gr''$, 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 $\xi(e''_S) = \xi(e_S)$.*

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., $Gr \cong Gr' \subseteq Gr''$).*

Now, the tracking problem can be converted to find the most common subgraph from two consecutive frames since each frame is represented by a graph. We define the most common subgraph between two given graphs in Definition 6.

DEFINITION 6. *G_C is the most common subgraph of two graphs Gr and Gr', where $G_C \subseteq Gr$ and $G_C \subseteq Gr'$, if and only if $\forall G'_C : G'_C \subseteq Gr \wedge G'_C \subseteq Gr' \Rightarrow |G'_C| \leq |G_C|$, where $|G|$ denotes the number of nodes of G.*

The graph-based tracking method starts with defining the neighborhood graph $G_N(v)$ in Definition 7.

DEFINITION 7. *$G_N(v)$ is the neighborhood graph of a given node v in a RAG, if for any nodes $u \in G_N(v)$, u is the adjacent node of v and has one edge such that $e_S = (v, u)$.*

G_N is a subgraph of a RAG. 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. To decide whether two nodes are corresponding, we use the neighborhood graphs in Definition 7. Therefore, to find 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 (1) which computes a similarity between two neighborhood graphs.

$$SimGraph(G_N(v), G_N(v')) = \frac{|G_C|}{min(|G_N(v)|, |G_N(v')|)} \quad (1)$$

where G_C is the most common subgraph between $G_N(v)$ and $G_N(v')$ (see Definition 6). The well-known algorithms computing G_C are based on the *maximal clique detection* [16] or the *backtracking* [18]. We exploit the idea of maximal clique detection to compute G_C since the neighborhood graph can be easily transformed into maximal clique. The higher the value of $SimGraph$, 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 $SimGraph$ 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}). In this way, we find all possible pairs of corresponding neighborhood graphs from $\mathbb{G}_N^m$ to $\mathbb{G}_N^{m+1}$. Algorithm 1 provides the pseudocode of graph-based tracking.

Algorithm 1: Graph-Based Tracking

Input: two Region Adjacent Graphs: $Gr(f_m)$ and $Gr(f_{m+1})$
Output: temporal edge set: E_T

```
let E_T = ∅
for each v ∈ Gr(f_m) do
    let g = G_N(v), maxSim = 0, Sim = 0, maxNode = null;
        for each v´∈ Gr(f_{m+1}) do
            let g´= G_N(v´);
            if g and g´are isomorphic then
                E_T = E_T ∪ {e_T(v, v´) }; break;
            else
                Sim = SimGraph(g, g´) by Equation (1);
            end if
            if Sim > maxSim then
                maxNode = v´; maxSim = Sim;
        done
        if no isomorphic graph of v and maxSim > T_sim then
            E_T = E_T ∪ {e_T(v, maxNode) };
done
return E_T;
```

2.3 STRG decomposition

In this subsection, an STRG 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 search area and the size of index.

2.3.1 Object Region Graph (ORG)

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 8. *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.*

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

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

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

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: (1) 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, (2) unlike existing video indexing techniques [27] which consider only detected objects, an ORG considers spatial and temporal relationships between nodes, and (3) 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.

2.3.2 Object Graph (OG)

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 person may consists of several regions such as head, upper body and lower body. Figure 3 (a) shows an object (a person) which is segmented into four regions over two frames. Since there are four regions in each frame, we build four ORGs, i.e., (v_1, v_5), (v_2, v_6), (v_3, v_7) and (v_4, v_8) like Figure 3 (b). Since they belong to a single object, it is better to merge those ORGs into one. We refer to the merged ORGs as an *Object Graph* (OG).

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

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. See Appendix A for proof. □

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.

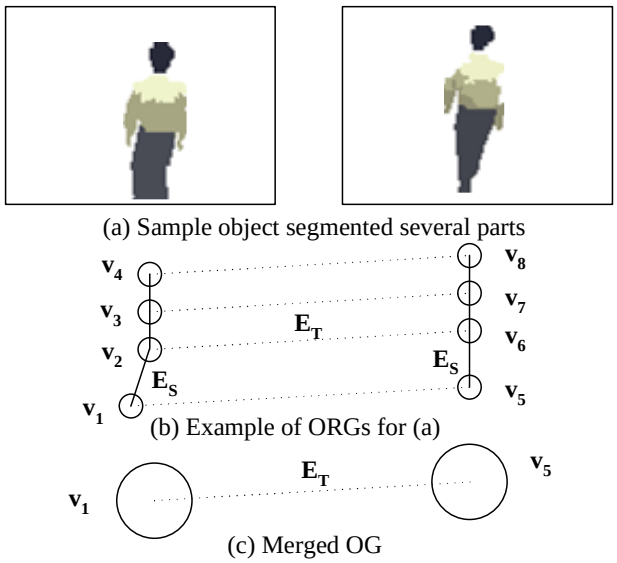


Figure 3: Example of subgraph merging

We need to merge ORGs which belong to a single object. To decide whether two ORGs are included in a single object, we consider the attributes of E_T; i.e., velocity and moving direction. If two ORGs have the same moving direction and the same velocity, these can be merged into a single OG. In Figure 3 (c), four ORGs are merged into a single OG; i.e., (v_1, v_5).

2.3.3 Background Graph (BG)

A video frame usually consists of two areas; foreground and background. A foreground is a main target on which a camera focuses, and a background is a supporting area that does not change significantly over time. Therefore, foreground/background distinction is a fundamental research area in video processing [22]. Moreover, from the perspective of graph-based video indexing, the distinction is more 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 video segment 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 (BGs). In order to eliminate redundant BGs, we overlap all remaining graphs using temporal edges (E_T).

3. EXTENDED GRAPH EDIT DISTANCE

As mentioned earlier, we index video data using clusters of OGs. To do that, we need a distance function to match OGs which are graphs. Among the existing graph matching algorithms, we select graph edit distance, and extend it to non-metric and metric spaces.

3.1 Extended Graph Edit Distance

There are many graph matching algorithms. The simplest one is to use inexact subgraph isomorphism [25]. In spite of its elegance and intuitiveness, this approach is not suitable for large databases due to its exponential time complexity. Messmer and Bunke [19] proposed a new algorithm for subgraph isomorphism which uses apriori knowledge about database models to reduce the time complexity. One limitation of their algorithm is that its inexact isomorphism detection uses an original edit distance which has been used for traditional string matching. It is not appropriate to handle image and video data. In order to address this, Shearer et al. [25] developed an algorithm to find the largest common subgraph (LCSG) by extending Messmer and Bunke's work with the decomposition network algorithm. Since LCSG is not suitable for graphs with temporal characteristics, we extend it by combining Chen's edit distance with real penalty (ERP) which allows to obey a metric space with local time shifting [3]. We call this new algorithm *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, deleting, and changing nodes, to transform one graph to the other. Since the main operations to edit graphs deal with nodes and their attributes rather than edges, we only consider nodes and their attributes of OGs. Let OG^s_m and OG^t_n be s^{th} and t^{th} OGs with m and n number of nodes, respectively.

$$OG^s_m = \{v^s_1, v^s_2, \ldots, v^s_m, \nu^s\}, \quad OG^t_n = \{v^t_1, v^t_2, \ldots, v^t_n, \nu^t\}$$

The distance function $EGED$ between OG^s_m and OG^t_n can be defined as follows.

DEFINITION 9. *The Extended Graph Edit Distance (EGED) between two object graphs OG^s_m and OG^t_n is defined as:*

$$EGED(OG^s_m, OG^t_n) = \begin{cases} \sum_{i=1}^{m} |v^s_i - g_i| & \text{if } n = 1, \\ \sum_{i=1}^{n} |v^t_i - g_i| & \text{if } m = 1, \\ min[EGED(OG^s_{m-1}, OG^t_{n-1}) + dist_{ged}(v^s_m, v^t_n), \\ \quad EGED(OG^s_{m-1}, OG^t_n) + dist_{ged}(v^s_m, gap), \\ \quad EGED(OG^s_m, OG^t_{n-1}) + dist_{ged}(gap, v^t_n)] & \\ \quad \text{otherwise.} \end{cases}$$

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

$$dist_{ged}(v^s_i, v^t_j) = \begin{cases} |v^s_i - v^t_j| & \text{if } v^s_i, v^t_j \text{ are not a gap} \\ |v^s_i - g_j| & \text{if } v^t_j \text{ is a gap} \\ |v^t_j - g_i| & \text{if } v^s_i \text{ is a gap.} \end{cases}$$

Let v indicate a value $\nu(v)$ of node attribute for better readability. $dist_{ged}$ 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 DTW, which does not consider local time shifting. In our case, $g_i = \frac{v_{i-1}+v_i}{2}$ is used for $dist_{ged}$, which can handle local time shifting [3].

However, as long as the cost function $dist_{ged}$ replicates the previous nodes, EGED is no longer in 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 $7 > 2+4$. In order to satisfy the triangle inequality, EGED is specialized to be metric distance function (see Theorem 2) by comparing the current value with the fixed constant.

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

PROOF. See Appendix B for proof. □

$EGED_M$ is used to indicate the metric version of $EGED$. In $EGED_M$, we include the cases that $n = 0$ and $m = 0$ since each OG should be measured from the fixed points on metric space. Let us repeat the previous example 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.

4. 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 9 is applied for EM clustering algorithm. The results of clustering will be used for indexing in Section 5.

4.1 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 [1]. 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 Θ ($= \{\theta_1, \ldots, \theta_K\}$) 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

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

In Equation (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 [9]. 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 (2). Chris Fraley et al. [9] 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 (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 (2) can be rewritten as follows.

$$p(Y_j|\Theta) = \sum_{k=1}^{K} \frac{w_k}{2\pi^{1/2}|\sigma_k|} e^{-\frac{1}{2\sigma^2} EGED(Y_j,\mu_k)^2} \quad (3)$$

Equation (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 (3) as follows.

$$\mathcal{L}(\Theta|Y) = log \prod_{j=1}^{M} p(Y_j|\Theta) = \sum_{j=1}^{M} \log \sum_{k=1}^{K} w_k p_k(Y_j|\theta_k) \quad (4)$$

To find appropriate clusters we estimate the optimal values of the parameters (θ_k) and the weights (w_k) in Equation (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:

$$h_{jk} = P(k|Y_j, \theta_k) = \frac{w_k}{p_k(Y_j|\theta_k)} \quad (5)$$

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

$$\begin{aligned} w_k &= \frac{1}{M} \sum_{j=1}^{M} h_{jk} \quad (6) \\ \mu_k &= \frac{\sum_{j=1}^{M} h_{jk} Y_j}{\sum_{j=1}^{M} h_{jk}} \\ \sigma_k &= \frac{\sum_{j=1}^{M} h_{jk} EGED(Y_j, \mu_k)^2}{\sum_{j=1}^{M} h_{jk}} \end{aligned}$$

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 (4) are decided, each OG is assigned to a cluster, $\hat{k}$ in terms of the maximum posterior probability by the following equation.

$$\hat{k} = \arg \max_{1 \leq k \leq K} \frac{w_k p_k(Y_j|\theta_k)}{\sum_{k=1}^{K} w_k p_k(Y_j|\theta_k)} \quad (7)$$

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 (2) with M data sets of K clusters in d-dimension is $O(d^2KM)$ [1]. We are using Equation (3) instead of Equation (2). Therefore, the complexity of each iteration can be reduced to $O(KM)$ since $EGED$ reduces the complexity of covariance (d^2) to 1.

4.2 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 [13, 24] 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 $\mathcal{M} = \{\mathcal{M}_K : K = 1, \ldots, N\}$ 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

$$BIC(\mathcal{M}_K) = \hat{l}_K(Y) - \eta\mathcal{M}_K \log(M) \quad (8)$$

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 (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 (8).

5. STRG-INDEX STRUCTURE

In this section, we propose a graph-based video indexing method, called *Spatio-Temporal Region Graph Index* (STRG-Index) using $EGED_M$ distance metric and clustered OGs. We illustrate the STRG-Index structure, construction, management, and search.

5.1 STRG-Index Tree Structure

Now, we have compressed BGs and clustered OGs based on the techniques discussed in Section 2.3 and 4. The STRG-Index tree structure consists of three levels of nodes; root node, cluster node and leaf node as seen in Figure 4.

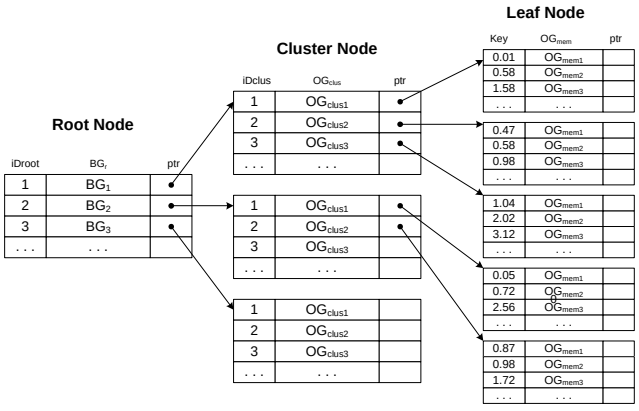
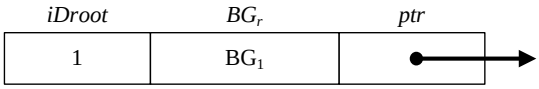
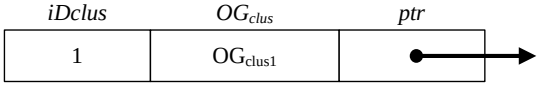


Figure 4: Example of STRG-Index Tree Structure

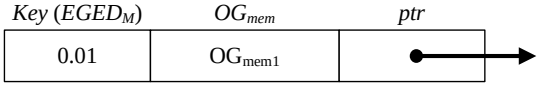
The top-level has the root node which contains the BGs. Each record in the root node has its identifier ($iDroot$), an actual BG (BG_r), and an associated pointer (ptr) which references the top of corresponding cluster node. The following figure shows the details of a record in the root node.



The mid-level has the cluster nodes which contain the centroid OGs representing cluster centroids. A record in the cluster node contains its identifier ($iDclus$), a centroid OG (OG_{clus}) of each cluster, and an associated pointer (ptr) which references the top of corresponding leaf node. The following figure shows the details of a record in a cluster node.



The low-level has the leaf nodes which contain OGs belonging to a cluster. A record in the leaf node has the index key (which is computed by $EGED_M(OG_{mem}, OG_{clus})$), an actual OG ($OG_{mem}$), and an associated pointer (ptr) which references the real video clip in a disk. The following figure shows the details of a record in a leaf node.



5.2 STRG-Index Tree Construction

Based on the STRG decomposition described in Section 2.3, an input video is separated into foreground (OG) and background (BG) as subgraphs of the STRG. The extracted BGs are stored at the root node without any parent. All the OGs sharing one BG are in a same cluster node. This can reduce the size of index significantly. For example, in surveillance videos a camera is stationary so that the background is usually fixed. Therefore, only one record (BG) in the root node is sufficient to index the background of the entire video.

We synthesize a centroid OG (OG_{clus}) for each cluster which is a representative OG for the cluster. This centroid OG is inserted into an appropriate cluster node as a record. This centroid OG is updated as the member OGs are changed such as inserting, deleting, etc. Also, each record in a cluster node has a pointer to a leaf node.

The leaf node has actual OGs in a cluster, which are indexed by $EGED_M$. To decide an indexing value for each OG, we compute $EGED_M$ between the representative OG (OG_{clus}) in the corresponding cluster and the OG (OG_{mem}) to be indexed. Since $EGED_M$ is a metric distance by Theorem 2, the value can be the key of OG to be indexed.

Algorithm 2 shows a psuedocode for building the STRG-Index tree for a given STRG.

5.3 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

Algorithm 2: Building STRG-Index

Input: Spation-Temporal Region Graph: *Gst*
Output: STRG-Index tree: *TR*

1: **let** *TR* = null;
2: create root node in *TR*;
3: *OG* = extracted *OG*s from *Gst* by Section 2.3.1 and 2.3.2;
4: *BG* = extracted *BG*s from *Gst* by Section 2.3.3;
5: *CLUS* = clusters of *OG*s by Section 4;
6: **for each** $BG_r \in BG$ **do**
7: create new cluster node;
8: **insert** tuple (*iDroot, BG$_r$, ptr(new cluster node)*) **into** root in *TR*;
9: **for each** $OG_{clus} \in CLUS$ **do**
10: create new leaf node;
11: **insert** tuple (*iDclus,OG$_{clus}$,ptr(new leaf node)*) **into** cluster in *TR*;
12: OG_{temp} = **sort**(OG_{mem}, $EGED_M(OG_{clus}, OG_{mem})$);
13: **for each** $OG_{mem} \in OG_{temp}$ **do**
14: **insert** tuple ($EGED_M(OG_{clus},OG_{mem})$, OG_{mem}, *prt*(real clip)) **into** leaf in *TR*;
15: **done**; **done**; **done**;
16: **return** *TR*;

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 described in Section 4. 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.

5.4 Size Analysis

In general, the performance of a database management system depends on the size of index structure and the memory utilization. As mentioned earlier, if the STRG-Index is stored in memory with their actual data items (OGs), it can provide better query processing performance. Let M be the number of OGs, and N be the total number of frames in a video segment which has a single background. The size of the STRG can be formulated as follows:

$$size(STRG) = \sum_{m=1}^{M} size(OG_m) + N \times size(BG) \quad (9)$$

On the other hand, the size of the STRG-Index tree is as follows:

$$size(STRG - Index) = \sum_{m=1}^{M} size(OG_m) + \sum_{k=1}^{K} size(OG_{clus_k}) + size(BG) \quad (10)$$

where K is the number of clusters. As seen in Equation (9) and (10), the difference between STRG and STRG-Index mainly depends on $N \times size(BG)$ and $\sum_{k=1}^{K} size(OG_{clus_k})$, since the size of a single BG is relatively small. Because N is usually much larger than K, the former term is much larger than the latter, i.e. $N \times size(BG) \gg \sum_{k=1}^{K} size(OG_{clus_k})$. Hence, the size of the STRG-Index is much smaller than the STRG.

5.5 Search Algorithm

Once the STRG-Index is constructed, we can search and retrieve OGs. From a query video segment q, we extract the background graph BG_q and object graphs OG_qs using the procedure described in Section 2. Search and retrieval are relatively straightforward tasks. We employ k-Nearest Neighbor (k-NN) search algorithm. At the query time, the BG_q is compared to each root node to determine which background OG_q belongs to, then each OG_q is compared to records in the cluster nodes. Finally, the search algorithm travels the leaf node to find similar OGs by comparing $EGED_M(OG_q, OG_{clus})$ with the index key values. Algorithm 3 presents a pseudocode of our search algorithm based on the k-NN search. When a query does not consider a background, Step 2 in Algorithm 3 is skipped. Instead, the search algorithm travels all cluster nodes in the STRG-Index to find the similar centroid OGs (OG_{clus}s).

Algorithm 3: k-NN Search Algorithm

Input: a query example ***q***, a STRG-Index *TR*, and *k*
Output: *k* most similar *OG*s

1: extract OG_q and BG_g from ***q***;
2: find the similar *BG* for BG_q in root node of *TR* using *SimGraph*;
3: find the similar OG_{clus} for OG_q in cluster node of *TR* using *EGED*;
4: $Key_q = EGED_M(OG_q, OG_{clus})$;
5: find k-nearest neighbor *OG*s to OG_q using *Key* in leaf node of *TR* and Key_q
6: **return** *OG*s;

6. EXPERIMENTAL RESULTS

We have performed the experiments with synthetic and real data to assess the proposed STRG-Index method on an Intel Pentium IV 2.6 GHz CPU computer.

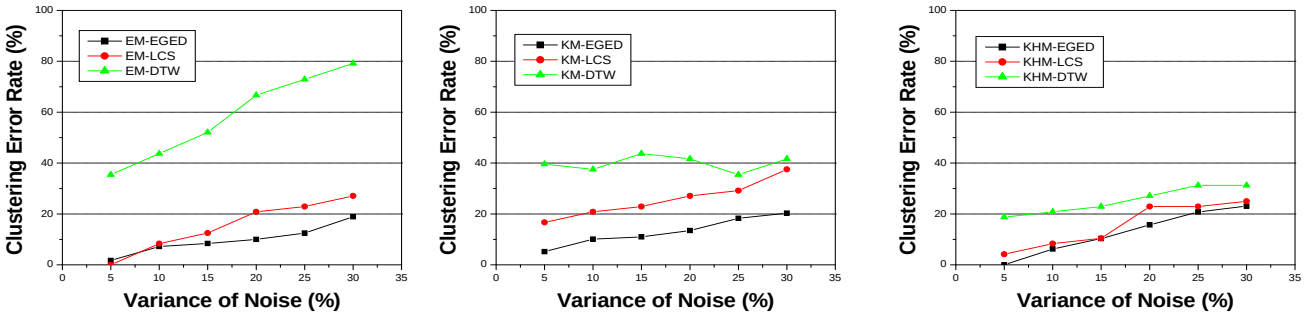
6.1 Experimental Setup

The real data are obtained from four video streams captured by a video camera. Table 1 shows the description of the real video data used in the experiments. The first two video streams (Lab1, Lab2) are taken from the inside of our laboratory, and the other two (Traffic1, Traffic2) from the outside, which have some traffic scenes. As seen in Table 1, our video data is about 45 hours, which is long enough to evaluate the proposed algorithms.

Table 1: Description of real data

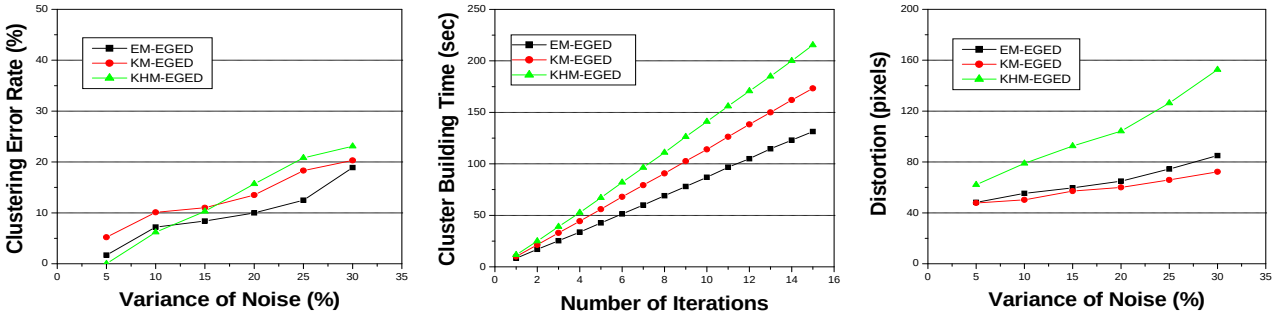
Video	# of OGs	Duration
Lab1	411	40 hour 38 min
Lab2	147	4 hour 12 min
Traffic1	195	15 min
Traffic2	203	12 min
Total	956	45 hour 7 min

To demonstrate the performance of the proposed STRG-Index more accurately, synthetic data is generated and used for the experiments. Since an OG is a type of time-series data, we generate new data by combining the Pelleg data set [24] which is widely used to test clustering algorithms, with the Vlachos data set [28] which is 2-D time-series data with noises. Our synthetic data is generated as follows. First, we design 48 moving patterns: vertical (12), horizontal (12), diagonal (8) and U-turn (16). Each pattern has two directions, different sizes of objects and various time lengths. Second, we generate time-series data following the approach



(a) EM-EGED vs. EM-LCS, EM-DTW (b) KM-EGED vs. KM-LCS, KM-DTW (c) KHM-EGED vs. KHM-LCS, KHM-DTW

Figure 5: Clustering error rates for noise variances 5%, 10%, 15%, 20%, 25% and 30%



(a) Clustering Error Rate (b) Cluster building time (c) Distortion

Figure 6: EM-EGED performances against KM-EGED and KHM-EGED

described in [24]. Here, the data set has 48 clusters and is distributed by Gaussian with $\sigma = 5$. Third, we add some noises to each data point based on the work in [28]. We generate 6 different types of data sets which have different amounts of noises ranging from 5% to 30%. Finally, the generated data are converted to OGs (temporal subgraph format) using the nodes, temporal edges and their attribute values given in Definition 8.

Using the synthetic data described above, the performance of the STRG-Index is compared with that of the M-tree index [5]. To be fair, we replace the data structure and the distance function used in the M-tree with OG and $EGED_M$, respectively. Our experiments demonstrate that:

1. *EGED* and EM clustering algorithm can prune the dissimilar data, which reduce the distance computations and provides better results for various similarity queries.

2. STRG-Index outperforms M-tree for various query loads in terms of cost and accuracy.

3. The size of STRG-Index is significantly smaller than that of STRG, which makes the index reside in memory for fast query processing.

6.2 Results of Clustering OGs

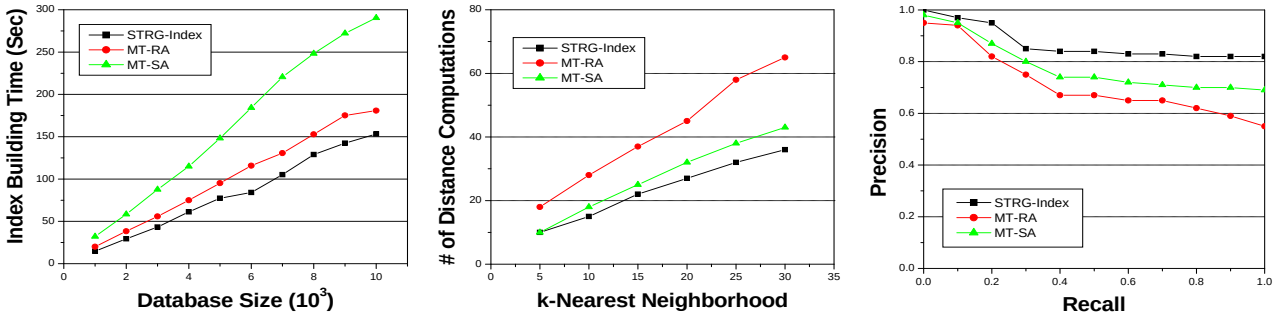
We first evaluate the performance of the EM clustering algorithm with the non-metric *EGED* (EM-EGED) on the synthetic data. As stated earlier, the quality of clustering OGs is important in guaranteeing the performance of the STRG-Index. Our EM-EGED is compared with two other clustering algorithms; K-Means (KM) and K-Harmonic means (KHM). The detailed information about KM and KHM can be found in [12]. Furthermore, in order to verify the performance of distance function, we compare the performance of EGED with Dynamic Time Warping (DTW) [11] and Longest Common Subsequence (LCS) [7].

We compare the performance of EM-EGED with EM-LCS and EM-DTW (Figure 5 (a)), KM-EGED with KM-LCS and KM-DTW (Figure 5 (b)), and KHM-EGED with KHM-LCS and KHM-DTW (Figure 5 (c)). In order to evaluate the clustering algorithms, we use the clustering error rate defined as;

$$Clustering\ Error\ Rate\ (\%) = \quad (11)$$
$$(1 - \frac{Number\ of\ Correctly\ Clustered\ OGs}{Number\ Of\ Total\ OGs}) \times 100$$

The EGED based algorithms perform much better than those based on the LCS and the DTW as seen in Figure 5 (a), (b) and (c). Especially, EM-EGED outperforms EM-DTW since EM tends to break down when the distance function cannot compute the similarities properly (see Figure 5 (a)). The EGED measures the similarity between OGs better than others do. Figure 5 also shows that the EGED is much more robust to noise than either the LCS or the DTW is.

Figure 6 shows the performance of EM-EGED compared with that of KM-EGED and KHM-EGED. The clustering error rate of EM-EGED is a little better than that of KHM-EGED (see Figure 6 (a)). The reason KHM-EGED has a



(a) Building index time (b) k-NN query performance (c) Accuracy by precision and recall

Figure 7: Performance of STRG-Index comparing with MT-RA and MT-SA

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), and its weight is similar to w_k in Equation (6). As far as the computation time is concerned, EM-EGED performs much better than KM-EGED and KHM-EGED. As shown in Figure 6 (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 real-time systems such as video surveillance. Figure 6 (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 (average about 2 times) than KHM-EGED. Overall, the quality of the EM-EGED proposed in this paper is superior to the other alternatives, such as KM-EGED, KHM-EGED, KM-LCS, KHM-LCS as well as KM-DTW and KHM-DTW.

6.3 Indexing Power

In this section, we validate the performance of the STRG-Index on the synthetic data. The proposed STRG-Index is compared with the M-tree (MT) index based on total index building time and accuracy. In the MT indexing, there are several methods based on the criteria selecting 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 [5].

Figure 7 (a) shows the average elapsed times to build an index structure for different sizes of databases. It shows that the time for building the STRG-Index is much less (average 15% to 50%) than that for MT-RA or MT-SA, even though the STRG-Index and MT consist of tree structure. This is because their complexities for the index construction are different: the STRG-Index is $O(KM)$ (Section 4.1) and the MT is $O(RM \log_R M)$ by [2], 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 the 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 makes it slower.

The performance of the k-NN query is shown in Figure 7 (b). In [2], the total time (T) to evaluate a query can be formulated as:

$$T = \#\ of\ distance\ evaluation\ \times\ complexity\ of\ d() + extra\ CPU\ time\ +\ I/O\ time$$

where $d()$ is a distance function. T should be minimized for better performance. However, evaluating $d()$ is so costly that the other components (extra CPU time and I/O time) can be neglected. In other words, the number of distance evaluations performed during query processing is the dominant component for the performance of search. Thus, we count the number of distance computations to evaluate the performance of k-NN query in which k ranges from 5 to 30. Figure 7 (b) shows that the number of distance computations to build the STRG-Index is smaller (average 22%) than that of MT-RA which is the fastest split policy in M-tree. Since both STRG-Index and MT use the same distance measure $EGED_M$, the performance of k-NN query using STRG-Index is better than using the MT index.

Figure 7 (c) shows the accuracy of each indexing structure for k-NN query. In order to measure the accuracy, the precision and the recall of query results are computed and plotted. Recall is the ratio of relevant items retrieved to the total number in the database, and Precision is the ratio of relevant items retrieved to the total number of items retrieved. The query data is composed of OGs that are not presented in the data sets, and the query results are verified by their cluster memberships. From Figure 7 (c), it is obvious that the STRG-Index outperforms both the MT-RA and the MT-SA. These results show that the STRG-Index outperforms the M-tree index in terms of both accuracy and speed.

6.4 Real Video Data Set

We apply the proposed STRG-Index to real videos in Table 1. Because each video data is 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 (8). Figure 8 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. For example, the optimal clusters of Lab1 video is 9. From the third and fourth

columns of Table 2, we can see that there is little difference between the actual number of clusters and the number of clusters found using the BIC measure. Also, the second column of the table shows the clustering error rate of the EM-EGED algorithm. Note that the clustering error rates for the traffic videos (Traffic1 and Traffic2) are lower than those of the indoor videos (Lab1 and Lab2). This is because the traffic videos have more uniform content such as bidirectional movement of vehicles.

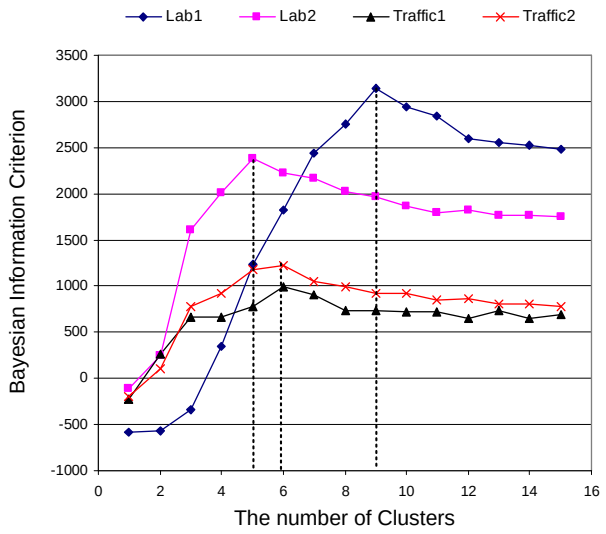


Figure 8: The BIC values for finding the optimal number of clusters

Concerning the index size, the last two columns of Table 2 clearly show that the STRG-Index is 10 to 15 times smaller than that of the STRG. The small size of the STRG-Index allows it to be stored entirely in memory for fast processing.

Table 2: Clustering Error Rate and the number of clusters for video streams

Video	EM-EGED	Optimal Cluster	Found Cluster	STRG size	STRG-Idx size
Lab1	16.8%	9	9	72.2MB	0.4MB
Lab2	14.4%	6	5	6.4MB	0.1MB
Traffic1	8.8%	6	6	1.4MB	0.2MB
Traffic2	9.5%	6	6	1.2MB	0.2MB

7. CONCLUDING REMARKS

In this paper, we propose an efficient method for indexing and retrieving video data, called STRG-Index. It expresses the video content as spatio-temporal region graph (STRG), and uses a tree structure and EM clustering algorithm for fast and accurate video indexing. Unlike existing graph-based data structures which provide only spatial view of individual frames, the STRG provides temporal relationships between consecutive frames. In addition, Extended Graph Edit Distance (EGED) is introduced for graph matching. The EGED is defined on both non-metric and metric spaces. The non-metric EGED is used for clustering which provides more accurate and fast indexing, and the metric EGED is used to index the STRG. Experimental results on both synthetic data and real video data show the effectiveness and accuracy of the proposed approach.

8. ACKNOWLEDGMENTS

This research is partially supported by the National Science Foundation grant EIA-0216500.

9. REFERENCES

[1] 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*, pages 423–428. ACM Press, 2000.

[2] E. Chavez, G. Navarro, R. Baeza-Yates, and J. L. Marroquin. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

[3] L. Chen and R. Ng. On the marriage of Lp-norms and edit distance. In *The 30th VLDB Conference*, pages 1040–1049, Toronto, Canada, 2004.

[4] L. Chen, M. T. Ozsu, and V. Oria. Symbolic representation and retrieval of moving object trajectories. In *Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, pages 227–234. ACM Press, 2004.

[5] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *the VLDB Journal*, pages 426–435, 1997.

[6] 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.

[7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithm*. The MIT Press, Massachusetts, 2001.

[8] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood estimation from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

[9] C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458):611–631, June 2002.

[10] L. M. Fuentes and S. A. Velastin. People tracking in surveillance applications. In *Proceedings of 2nd IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, Kauai, Hawaii, December 2001.

[11] H. Gish and K. Ng. Parametric trajectory models for speech recognition. In *Proceedings of The Fourth International Conference on Spoken Language Processing*, Philadelphia, PA, October 1996.

[12] 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*, pages 600–607. ACM Press, 2002.

[13] R. Hammoud and R. Mohr. Probabilistic hierarchical framework for clustering of tracked objects in video streams. In *Irish Machine Vision and Image Processing Conference*, pages 133–140, The Queen's University of Belfast, Northern Ireland, August 2000.

[14] A. Hanjalic, R. L. Lagendijk, and J. Biemond. Automated high-level movie segmentation for advanced video-retrieval systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(4):580–588, 1999.

[15] A. Jain and A. Vailaya. Image retrieval using color and shape. *Pattern Recognition*, 29(8):1233–1244, 1997.

[16] G. Levi. A note on the derivation of maximal common subgraphs of two directed or undirected graphs. *Calcols 9*, pages 341–354, 1972.

[17] 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.

[18] J. J. McGregor. Backtrack search algorithms and the maximal common subgraph problem. *Software Practice and Experience*, pages 23–34, 1982.

[19] B. T. Messmer and H. Bunke. Subgraph isomorphism detection in polynomial time on preprocessed model graphs. In *Recent Developments in Computer Vision*, pages 373–382. Springer, Berlin,, 1995.

[20] O. Miller, E. Navon, and A. Averbuch. Tracking of moving objects based on graph edges similarity. *Proceedings of the ICME '03*, pages 73–76, 2003.

[21] M. Naphade, S. Basu, J. Smith, C. Y. Lin, and B. Tseng. Modeling semantic concepts to support query by keywords in video. In *Proceedings of International Conference on Image Processing*, volume 1, pages 145–148, 2002.

[22] J. Oh, K. A. Hua, and N. Liang. A content-based scene change detection and classification technique using background tracking. In *SPIE Conf. on Multimedia Computing and Networking 2000*, pages 254–265, San Jose, CA, Jan. 2000.

[23] J. Y. Pan, H. J. Yang, C. Faloutsos, and P. Duygulu. Automatic multimedia cross-modal correlation discovery. In *Proceedings of the 2004 ACM SIGKDD*, pages 653–658. ACM Press, 2004.

[24] D. Pelleg and A. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the 7th International Conference on Machine Learning*, pages 727–734, San Francisco, 2000.

[25] K. R. Shearer, S. Venkatesh, and D. Kieronska. Spatial indexing for video databases. *Journal of Visual Communication adn Image Representation*, 7(4):325–335, December 1997.

[26] Y. Theodoridis, M. Vazirgiannis, and T. K. Sellis. Spatio-temporal indexing for large multimedia applications. In *Proceedings of IEEE ICMCS*, pages 441–448, 1996.

[27] R. Tusch, H. Kosch, and L. Boszormenyi. Videx: An integrated generic video indexing approach. In *Proceedings of ACMMM 2000*, pages 448–451, LA, CA, Oct. 2000.

[28] M. Vlachos, D. Gunopulos, and G. Kollios. Robust similarity measures for mobile object trajectories. In *Proceedings of 5th International Workshop Mobility In Databases and Distributed Systems*, pages 721–728, Aix-en-Provence, France, September 2002.

[29] 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)*, pages 1040–1049, January 1990.

APPENDIX

A. PROOF OF THEOREM 1

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 $G_1'' \cup G_2''$. □

B. PROOF OF THEOREM 2

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 it by induction. $EGED(R_l, T_n)$ can be written as follows by Definition 9.

$$\begin{aligned} EGED(R_l, T_n) = min[&EGED(R_{l-1}, T_{n-1}) + |v_l^r - v_n^t|, \\ &EGED(R_{l-1}, T_n) + |v_l^r - g|, \\ &EGED(R_l, T_{n-1}) + |g - v_n^t|] \end{aligned} \quad (12)$$

where g is a fixed constant for g_i. To compute the cost to transform v_l^r to v_m^t, consider the three terms in the right hand side of Equation (12), 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 $EGED(R_l, T_{n-1}) + |g - v_n^t|$ to edit v_l^r into v_{n-1}^t by adding v_n^t.

Since Equation (12) 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. □