Hyper-Rectangle Based Segmentation and Clustering of Large Video Data Sets

Seok-Lyong Lee* and Chin-Wan Chung†

*Department of Information and Communication Engineering, †Department of Computer Science
Korea Advanced Institute of Science and Technology
373-1, Kusong-Dong, Yusong-Gu, Taejon 305-701, Korea
sllee@islab.kaist.ac.kr, chungcw@cs.kaist.ac.kr

Abstract

Video information processing has been one of great challenging areas in the database community since it needs huge amount of storage space and processing power. In this paper, we investigate the problem of clustering large video data sets that are collections of video clips as foundational work for the subsequent processing such as video retrieval. A video clip, a sequence of video frames, is represented by a multidimensional data sequence, which is partitioned into *video segments* considering temporal relationship among frames, and then similar segments of the clip are grouped into *video clusters*. We present the effective video segmentation and clustering algorithm which guarantees the clustering quality to such an extent that satisfies predefined conditions, and show its effectiveness via experiments on various video data sets.

Keywords: Clustering, Clustering algorithm, Video cluster, Video segment

1. Introduction

Recently, the video information has become widely used in many application areas such as news broadcasting, video on demand, and video conferencing, as digital storage technology and computing power have been significantly advanced in the last decade. These applications involve searching, consuming, or exchanging large volume of complex video data sets. To handle such voluminous data sources, it is essential that the video data should be effectively represented, stored, and retrieved.

A video database may contain a number of video clips that can be represented by multidimensional data sequences (MDS's). In our earlier work [11], we have formally defined an MDS S with K points in the n-dimensional space as a sequence of its component vectors, $S = \langle S[1], S[2], \ldots, S[K] \rangle$, where each vector $S[j]$ $(1 \le j \le K)$ is composed of n scalar entries, that is, $S[j] = (S^1[j], S^2[j], \ldots, S^n[j])$. A video clip consists of multiple frames in temporal order, each of which can be represented by a multidimensional vector in the feature space such as RGB or YCbCr color space. Thus, a video clip is modeled as a sequence of points in a multidimensional space such that each frame of the sequence constitutes a multidimensional point, whose components are feature values of a frame. By modeling a video clip to an MDS, the problem of clustering frames in a video clip is

transformed into that of clustering points of an MDS in a multidimensional space. Each sequence is partitioned into video segments (or video shots) and then similar segments are grouped into a video cluster. Figure 1 shows the hierarchical structure of video data.

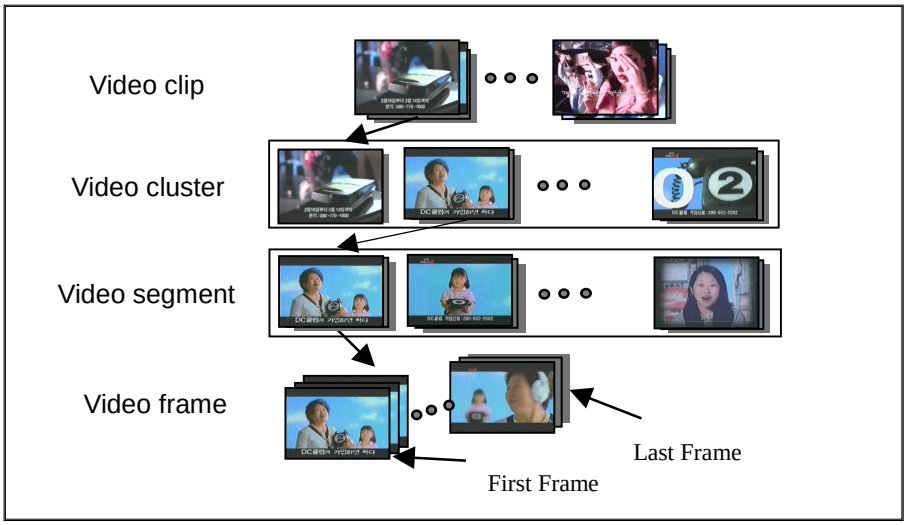


Figure 1. Hierarchical structure of video data

The clustering has attracted great interest in many database applications such as customer segmentation, sales analysis, pattern recognition, and similarity search. The task of clustering data points can be defined as follows: Given a set of points in a multidimensional space, partition the points into clusters such that points within each cluster have similar characteristics while points in different clusters are dissimilar. A point that is considerably dissimilar to or inconsistent with the remainder of the data is referred to as an *outlier* or a *noise*.

Various clustering methods have been studied in database communities, however the clustering of video data should be handled in a way different from the existing clustering methods in various aspects. First, in a video clustering, the temporal relationship among frames and among video segments should be considered importantly, since the temporal ordering of frames and video segments is an intrinsic feature of video data. Existing methods did not consider it. Second, a target object to be clustered in existing methods is mapped to a single point in a multidimensional space and thus belongs to a single cluster, while a video clip is represented by multiple points that can be partitioned into multiple separate clusters. Third, the shapes of clusters may also be considered differently. The existing methods attempt to look at quantitative properties of clusters, independent of how they will be used. They determine a certain number of clusters that optimize given criteria such as the mean square error. Thus, the shapes of clusters are determined arbitrarily depending on the distribution of points in the data space. However, we consider, in addition to the clustering itself, the subsequent retrieval process importantly, such as *'Find video clips that are similar to a given news video.'* Therefore, the shapes of clusters should be appropriate for this purpose.

It is usual in the video search that one or more key frames are selected for each video segment, and a query is processed on the selected frames [7]. But the search by the key frames does not guarantee the correctness since they cannot summarize all the frames of the segment. We proposed in [11] the similarity search scheme based on the hyper-rectangle that tightly bounds all points (or frames) in the segment, not on the key frames to prevent '*false dismissal*.' We believe that guaranteeing the correctness is one of important features in the similarity search. In addition, the shapes of clusters should be proper for the indexing mechanism. We use a hyper-rectangle as the shape of a cluster, since current dominant indexing mechanisms such as the R-tree [9] and its variants [3,4,14] are based on a minimum bounding rectangle (MBR) as their node shape.

1.1. Problem definition

The representation and the retrieval of video data place various special requirements on clustering techniques, motivating the need for designing a new clustering algorithm. Those requirements are categorized into two classes as follows: the geometric characteristics of clusters, and the temporal and semantic relationship among elements in a cluster

First, the cluster should be dense with respect to (wrt.) the volume and the edge for the efficient retrieval, by minimizing the volume and the edge of a cluster per point and maximizing the number of points per cluster. Next, the temporal and semantic relationship among elements in a cluster should be maintained. It means that the information on temporal ordering of elements in a cluster should be preserved, and elements in a cluster should be semantically similar. In addition to these requirements, it should be able to deal with outliers appropriately, and minimize the number of input parameters to the clustering algorithm. Considering these requirements, the clustering problem in this paper is formalized as follows:

Given: A data set of video clips and the minimum number of points *minPts* per video segment

Goal: To find the sets of video clusters and outliers that optimize the values of predefined measurement criteria.

An input parameter *minPts* is needed to determine outliers. In our method, each point in a sequence is initially regarded as a segment with a single point, and then closely related segments are repeatedly merged to form a cluster. If a certain segment has "*far fewer*" points than the average after the segmentation process, all points in it can then be considered as outliers. For instance, if a segment with 2 or 3 points is located away from other segments, it may be a set of outliers with high possibility. "*Far fewer*" is of course heuristically determined depending on applications. A too small value of *minPts* makes unimportant segments be indexed, degrading the memory utilization, while a too large value of *minPts* makes meaningful segments be missed. In this context, if a segment has points the number of which is less than a given *minPts* value after the segmentation process, all points in the segment are regarded as outliers. Those outliers are not indexed, but written out to the disk for later processing.

1.2. Brief sketch of our method

In the first step of our method, video clips are parsed to generate a data set of MDS's. Feature values are extracted from each frame of the video clip by averaging color values of pixels of a frame or segmented blocks of a frame. As an optional process, if the dimensionality of generated data is high, it is reduced to a low dimensionality to avoid '*dimensionality curse problem.*' It is usual that high dimensional data may not be used in reality since it needs huge amount of storage space and causes severe processing overhead.

In the next step, the generated MDS is partitioned into video segments such that predefined geometric and semantic criteria are satisfied. Outliers are also identified in this process. Finally, similar segments of a sequence are grouped into a video cluster in the clustering process to get the better clustering quality. In this way, a given video clip is represented by a small number of video clusters which will be indexed and stored into a database for later processing. In this paper, we focus on the segmentation and the clustering processes. The overall structure is shown in Figure 2.

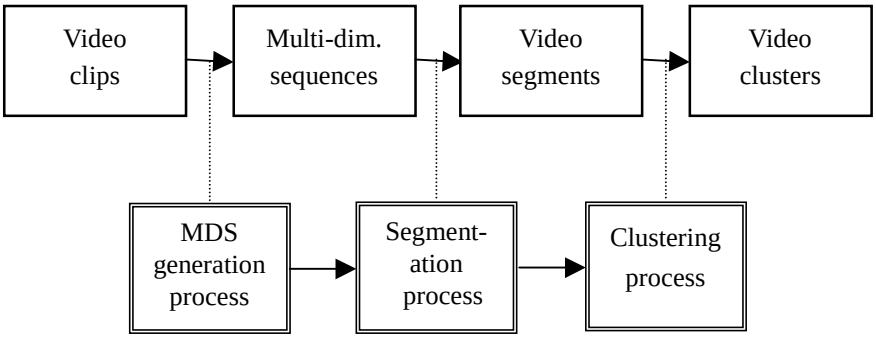


Figure 2. Overall structure of the proposed method

The segmentation and clustering method proposed in this paper is a foundational work for the creation of video databases, and can be used for various application domains such as video digital libraries, video on demand, news on demand, and tele-education systems. One of potential applications, which is emphasized in this paper, is the segmentation and clustering of video data sets, but we believe other application areas in which data can be represented in the form of MDS can also benefit. For examples, audio sequences, time series data, and various analog signals can be represented by MDS, and thus our method can be applied.

1.3. Paper Organization

The rest of the paper is organized as follows: Section 2 provides a survey of related works with a brief discussion on clustering data points and data sequences. Section 3 includes basic definitions, clustering characteristics, and various measurements of clustering quality. The segmentation process is described in Section 4 with an algorithm to produce video segments from an MDS. Section 5 provides the clustering process to generate video clusters by merging video segments.

Experimental results are presented in Section 6 and we give conclusions in Section 7.

2. Related works

Many excellent approaches on clustering data points in a multidimensional space have been proposed, such as CLARANS [13], BIRCH [15], DBSCAN [5], CLIQUE [2], and CURE [8].

CLARANS is a clustering algorithm that is based on randomized search and gets its efficiency by reducing the search space using user-supplied input parameters. The algorithm BIRCH constructs a hierarchical data structure called the *CF-tree* for multiphase clustering by scanning a database and uses an arbitrary clustering algorithm to cluster leaf nodes of the *CF-tree*. It is the first approach to handle outliers effectively in the database area. DBSCAN tries to minimize requirements of domain knowledge to determine input parameters and provides arbitrary shapes of clusters based on the distribution of data points. Its basic idea is that for each point of a cluster, the neighborhood of the point within a given radius has to contain at least a given number of points. Thus, it needs only two input parameters, the radius and the number of points. CLIQUE identifies dense clusters automatically in subspaces of a high dimensional data space. The subspaces allow better clustering of data points than the original space. As input parameters, it needs the size of a grid that partitions the space and a global density threshold for clusters. The concept of clustering points in subspaces is extended to the projected clustering in [1] to pick particular dimensions on which data points are closely related and to find clusters in the corresponding subspace.

Another recent approach is CURE that identifies clusters having non-spherical shapes and wide variances in size. It achieves this by representing each cluster using multiple well-scattered points. The shape of a non-spherical cluster is better represented when more than one point are used. This algorithm finishes the clustering process when the number of clusters in the current level of the cluster hierarchy becomes k, where k is an input parameter. However, all approaches described above need multiple input parameters, and do not consider the temporal relationship among data points. Thus, they may not be applied to the clustering of data sequences, that have the temporal and semantic relationship among their elements, such as video clips.

The first clustering algorithm for sequences was proposed in [6], partitioning a sequence into subsequences, each of which is contained in an MBR (or a cluster). This algorithm uses the marginal cost (MCOST) which is defined as the average number of disk accesses divided by the number of points of the cluster. To determine MCOST, it considers the volume factor based on the volume increment of the current cluster when a point is included into the cluster. The MCOST method is initially designed to represent a time-series sequence by multiple rectangles. It was slightly modified in [10] to a two-pass algorithm running forward and backward to identify video shot boundaries, and also slightly modified in [11] to support the multidimensional rectangular query. The MCOST method can be used for the segmentation of video sequences in the sense that it can handle a sequence and a video can be represented by a multidimensional sequence. However, it

is not able to deal with outliers appropriately. Moreover, it considers the volume factor only during the clustering process, which is sometimes not sufficient. The edge of a cluster and the similarity between points in a cluster should also be considered as important factors in addition to the volume. We address this in Section 3.3 using some intuitive examples.

3. Preliminaries

In this section, we discuss various characteristics of a hyper-rectangle that is used to define a video segment and a video cluster, and clustering factors to be considered for effective segmentation and clustering. Table 1 summarizes symbols and definitions used in this paper.

Table 1. Summary of symbols and definitions

Symbol	Definition
S	Multidimensional data sequence (MDS)
$S[i]$	ith entry of S
N	Number of dimensions
N	Number of sequences in a database
HR	Hyper-rectangle
VC	Video cluster
VS	Video segment
P	Point, represented by $(P^1, P^2, \ldots, P^n)$ in the space $[0, 1]^n$
SP	Starting point of VC
K	Number of points in HR
$dist(*,*)$	Euclidean distance between two points
$Vol(HR)$	Volume of HR
$Edge(HR)$	Total edge length of HR
VPP	Volume per point
EPP	Edge per point
PPC	Number of points per cluster

3.1. Characteristics of a hyper-rectangle

A hyper-rectangle is a geometric polyhedron that tightly bounds all points in a video segment or a video cluster. First, we define it formally as follows:

Definition 1 (*Hyper-rectangle*). A hyper-rectangle HR with k points, P_j for $j = 1, 2, \ldots, k$ in the n-dimensional space, is represented by two endpoints, L(low point) and H(high point), of its major diagonal, and the number of points in the rectangle as follows: $HR = \langle L, H, k \rangle$, where $L = \{(L^1, L^2, \ldots, L^n) \mid L^i = min_{1 \le j \le k}(P_j^i)\}$, and $H = \{(H^1, H^2, \ldots, H^n) \mid H^i = max_{1 \le j \le k}(P_j^i)\}$ for $i = 1, 2, \ldots, n$. ■

We can represent a point P_j in the hyper-rectangular form by placing $L^i = H^i = P_j^i$ for all dimensions, that is, $\langle P_j, P_j, 1 \rangle$. This rectangle is denoted by $HR(P_j)$ which has zero volume and edge. It is sometimes convenient to describe operations of the segmentation and clustering if we regard a multidimensional point as a hyper-rectangle. The volume $Vol(HR)$ and the edge, i.e. total edge length, $Edge(HR)$ of HR are computed as:

$$Vol(HR) = \prod_{1 \le i \le n} (HR.H^i - HR.L^i) \quad (1)$$

$$Edge(HR) = 2^{n-1} \cdot \sum_{1 \le i \le n} (HR.H^i - HR.L^i) \quad (2)$$

Then, the volume and the edge per point of *HR*, *VPP*(*HR*) and *EPP*(*HR*) respectively, will be:

$$VPP(HR) = \frac{Vol(HR)}{HR.k} = \frac{\prod_{1 \le i \le n} (HR.H^i - HR.L^i)}{HR.k} \quad (3)$$

$$EPP(HR) = \frac{Edge(HR)}{HR.k} = \frac{2^{n-1} \cdot \sum_{1 \le i \le n} (HR.H^i - HR.L^i)}{HR.k} \quad (4)$$

Two hyper-rectangles can be merged during segmentation and clustering processes. We define a merging operator between two hyper-rectangles as follows:

Definition 2 (*Merging operator* ⊕). Let HR_1 and HR_2 be hyper-rectangles. Then, the merging operator ⊕ is defined as $HR_1 \oplus HR_2 = HR_3$ such that $HR_3.L = \{(HR_3.L^1, HR_3.L^2, \ldots, HR_3.L^n) \mid HR_3.L^i = min(HR_1.L^i, HR_2.L^i)\}$, $HR_3.H = \{(HR_3.H^1, HR_3.H^2, \ldots, HR_3.H^n) \mid HR_3.H^i = max(HR_1.H^i, HR_2.H^i)\}$ for $i = 1, 2, \ldots, n$, and $HR_3.k = HR_1.k + HR_2.k$. ■

By Definition 2, we can easily recognize that the operator ⊕ has a symmetric property, that is, $HR_1 \oplus HR_2 = HR_2 \oplus HR_1$. Consider a point *P* to be merged to a hyper-rectangle $HR = \langle L, H, k \rangle$. Merging *P* into *HR* produces a probably bigger hyper-rectangle, which causes changes in the volume, the edge, and the number of points. We are interested in the amount of change resulting from the merging process, since it is an important factor for clustering. The volume and edge increments, *ΔVol*(*HR*, *P*) and *ΔEdge*(*HR*, *P*) respectively, are formulated as follows:

$$\Delta Vol(HR, P) = Vol(HR \oplus HR(P)) - Vol(HR) \quad (5)$$

$$\Delta Edge(HR, P) = Edge(HR \oplus HR(P)) - Edge(HR) \quad (6)$$

3.2. Similarity between two points

The similarity of two points in a multidimensional space, each of which is represented by a multidimensional vector, is generally defined as a function of the Euclidean distance (hereafter, referred to as 'distance') between those two points. The similarity between video frames can be described as a function of the distance between the corresponding feature vectors. The value range of the similarity between two objects is usually [0,1] while the range of the distance is [0, ∞]. The distance is close to zero when two objects are similar, and becomes large if they are quite different. But the similarity is the opposite. It is close to 1 when two objects are similar, while it is close to zero when they are very dissimilar. The distance between two objects can be transformed into the similarity by an appropriate mapping function. In this paper, a data space is normalized in the $[0,1]^n$ hyper-cube, where the length of each dimension is 1, and thus the maximum allowable distance is $\sqrt{n}$, the length of a diagonal of the cube. This distance will be easily mapped to the similarity. We will use the distance for the similarity measure for simplicity. The distance between two adjacent

points in a *n*-dimensional sequence *S* is given as:

$$dist(S[j], S[j+1]) = \sqrt{\sum_{1 \le i \le n} (S^i[j+1] - S^i[j])^2} \quad (7)$$

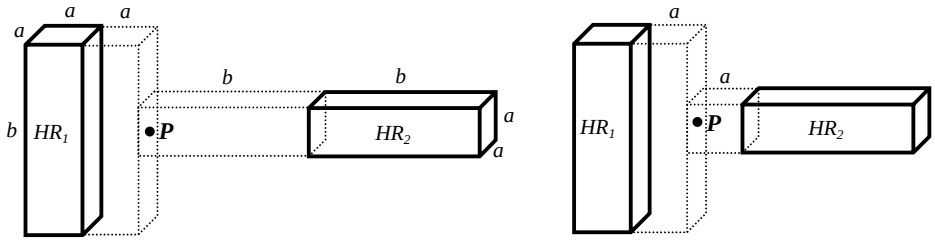
where $S^i[j]$ is a coordinate value of dimension *i* of the *j*-th point in sequence *S*.

3.3. Clustering Factors

In this section, we discuss two clustering factors, geometric and semantic factors, that should be considered for clustering MDS's. The former considers the geometric characteristics of hyper-rectangles, and is applied to both the segmentation and the clustering. On the other hand, the latter considers the semantic relationship among elements in hyper-rectangles, and is related to the segmentation only. We discuss those factors with intuitive examples in this section.

Geometric factor: Since geometric characteristics of a cluster have a great impact on the search efficiency, we need to consider this factor importantly for clustering. Apparently, a cluster with large volume in the search space has the higher possibility to be accessed by a query than that with small volume. However, the edge for the hyper-rectangular cluster should also be considered as an important factor in addition to the volume, as we have shown in [12]. Example 1 illustrates it.

Example 1. Let HR_1 and HR_2 be hexahedral clusters with sides *a*, *a*, and *b* (*a*<*b*), respectively in the 3-dimensional space as shown in Figure 3. We are going to determine the cluster into which a point *P* is being merged. In Figure 3.(a), we can see that $\Delta Vol(HR_1, P) = \Delta Vol(HR_2, P) = a^2 \cdot b$, $\Delta Edge(HR_1, P) = 4 \cdot a$ and $\Delta Edge(HR_2, P) = 4 \cdot b$. From the standpoint of the volume as a clustering factor, both HR_1 and HR_2 can be candidates. On the other hand, in Figure 3.(b), $\Delta Edge(HR_1, P) = \Delta Edge(HR_2, P) = 4 \cdot a$, $\Delta Vol(HR_1, P) = a^2 \cdot b$, and $\Delta Vol(HR_2, P) = a^3$. In this case, both HR_1 and HR_2 can be candidates if we consider the edge as a clustering factor. However, we observe intuitively that HR_1 is an appropriate candidate for the former case while HR_2 is good for the latter case, since $a < b$. It means that both volume and edge should be considered as factors for the clustering. ■



(a) Same volumes, different edges (b) Different volumes, same edges

Figure 3. Clustering factors: the volume and the edge

Semantic factor: Since consecutive points in a video segment are closely related, that is, semantically similar with each other, the distance between them needs to be considered as an

important clustering factor. If a point is spatially far from the previous point of a sequence, a new video segment should be started from the point. Example 2 shows this.

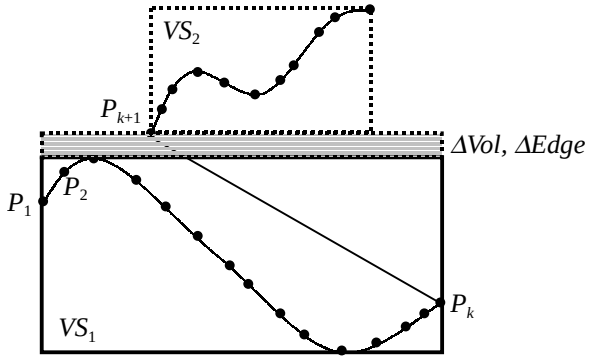


Figure 4. Semantic factor of clustering

Example 2. Let us consider an MDS which consists of a series of points P_j for j = 1, 2, …, k, k+1, …, as shown in Figure 4. We are going to determine whether a point P_{k+1} is to be merged into a video segment VS_1 or not. Let $\Delta Vol(VS_1, P_{k+1})$ and $\Delta Edge(VS_1, P_{k+1})$ be the volume and the edge increments respectively, resulting from the inclusion of P_{k+1} into VS_1, which are related to the shaded area in the figure. If we consider only the volume and the edge as clustering factors, P_{k+1} may be included in VS_1, since ΔVol and $\Delta Edge$ are relatively small. However, it will be better if a new video segment VS_2 is started from P_{k+1} because P_{k+1} is spatially far from P_k, that is, two points are dissimilar semantically. It shows that the distance between two consecutive points should also be considered as an important clustering factor. ■

3.4. Measurement of clustering quality

In this section, we introduce the criterion functions that can be used to measure the quality of clustering. As we mentioned the clustering requirements in Section 1.1, the clusters should be dense wrt. the volume and the edge for efficient retrieval. It is accomplished by minimizing the volume and the edge per point and by maximizing the number of points per cluster. As quantitative measures to evaluate the quality, we use three parameters: the volume per point (*VPP*), the edge per point (*EPP*), and the number of points per cluster (*PPC*). Suppose MDS S is represented by p hyper-rectangles, HR_1, …, HR_p. Then, *VPP*, *EPP*, and *PPC* of S are defined as follows:

$$VPP = \frac{\sum_{1 \le j \le p} Vol(HR_j)}{\sum_{1 \le j \le p} HR_j.k}, \quad EPP = \frac{\sum_{1 \le j \le p} Edge(HR_j)}{\sum_{1 \le j \le p} HR_j.k}, \quad PPC = \frac{\sum_{1 \le j \le p} HR_j.k}{p} \tag{8}$$

The MCOST method proposed in [6] considers the volume factor only when it generates clusters from sequences. However, as we claimed in Section 3.3 with some intuitive examples, the edge factor should also be considered importantly during the clustering process. In this context, the clustering quality should be evaluated wrt. both volume and edge factors.

4. Video Segmentation

Once multidimensional sequences have been generated from video clips, each sequence is partitioned into video segments. The segmentation is the repeating process of merging a point of the sequence into a hyper-rectangle if predefined criteria are satisfied. Consider a point P to be merged to a hyper-rectangle $HR = \langle L, H, k \rangle$ in the unit space $[0,1]^n$. Then, the segmentation is done in such a way that if the merging of P into HR satisfies certain given conditions then it is merged into the current segment, otherwise a new segment is started from the point. In this process, a *merging* object is a hyper-rectangle or a point (when a new segment is started), while a *merged* object is always a point. Let us start the discussion with the formal definition of a video segment as follows:

Definition 3 (*Video segment*). A video segment VS that contains k points in the temporal order, P_j for $j = 1, 2, \ldots, k$, is defined as follows: $VS = \langle sid, SP, HR \rangle$, where sid is the segment-id, SP is the starting point of VS, $HR = \langle L, H, k \rangle$ such that $L = \{(L^1, L^2, \ldots, L^n) \mid L^i = min_{1 \le j \le k}(P_j^i)\}$ and $H = \{(H^1, H^2, \ldots, H^n) \mid H^i = max_{1 \le j \le k}(P_j^i)\}$ for $i = 1, 2, \ldots, n$. ■

To merge a point into a segment during the segmentation process, our method uses predefined geometric and semantic criteria that should be satisfied. In the next subsections, we discuss those criteria and our proposed algorithm.

4.1. Geometric criterion

First, we introduce the geometric bounding condition with respect to the volume and the edge of a video segment. In [12], we introduced the concept of a *unit hyper-cube* which is defined as follows:

Definition 4 (*Unit hyper-cube*). Let HR_S be a hyper-rectangle that tightly bounds *all* K points in a video sequence S. Then, a unit hyper-cube *uCUBE* is defined as a cube in the space $[0,1]^n$, occupied by a single point assuming all points are uniformly distributed over the hyper-space of HR_S. If its side-length is e, its volume and edge will be:

$$Vol(uCUBE) = e^n = \frac{Vol(HR_S)}{K}, \quad Edge(uCUBE) = 2^{n-1} \cdot n \cdot e = 2^{n-1} \cdot n \cdot \sqrt[n]{\frac{Vol(HR_S)}{K}} \quad \blacksquare \qquad (9)$$

If all points of S are uniformly scattered into the space of HR_S, we can think one point is allocated to a unit hyper-cube. We can figure out intuitively that each point of S forms a hyper-rectangle whose shape is a unit hyper-cube. However, the uniform distribution is not likely to occur in reality. Points in a sequence usually show a clustered distribution in the real world. For instance, frames in a video segment are very similar, and thus the points of a segment are clustered together. The uniform distribution provides a bound in determining whether to merge a point into a video segment or not. The bounding thresholds wrt. volume and edge, τ_{vol} and τ_{edge} respectively for a

sequence *S*, are given as follows:

$$\tau_{vol} = Vol(uCUBE) = e^n, \quad \tau_{edge} = Edge(uCUBE) = 2^{n-1} \cdot n \cdot e \tag{10}$$

Definition 5 (*Geometric bounding condition*). Suppose a point *P* is to be merged into a video segment *VS* in the space $[0,1]^n$. Then, the *geometric bounding condition* is the condition that must be satisfied to merge *P* into *VS* and it is defined as follows:

$$\Delta Vol(VS, P) \le \tau_{vol} \wedge \Delta Edge(VS, P) \le \tau_{edge} \quad \blacksquare \tag{11}$$

Lemma 1. *The clustering that satisfies the geometric bounding condition guarantees better clustering quality than the case of the uniform distribution, wrt. VPP and EPP.*

Proof. See Appendix A.

4.2. Semantic criterion

Another important criterion discussed in Section 3.3 is a semantic factor. To determine whether to merge a point into a current video segment or not, the distance between the point and the previous point of it in the segment is examined. If the distance exceeds a predefined threshold, then a new segment is started from the point. Let us consider an MDS that has *K* points, P_j for $j = 1, 2, \ldots, K$. Then, the threshold τ_{dist} is the mean distance between all pairs of adjacent points in the sequence, and defined as follows:

$$\tau_{dist} = \frac{1}{K-1} \cdot \sum_{1 \le j \le K-1} dist(P_j, P_{j+1}) \tag{12}$$

Definition 6 (*Semantic bounding condition*). Consider a point P_{k+1} to be merged into a video segment *VS*, whose previous point is P_k, in the space $[0,1]^n$. Then, the *semantic bounding condition* is the condition that must be satisfied to merge P_{k+1} into *VS* and it is defined as follows:

$$dist(P_k, P_{k+1}) \le \tau_{dist} \quad \blacksquare \tag{13}$$

Satisfying this condition guarantees that the distance between any pair of two consecutive points in the video segment is equal to or less than the mean distance between all pairs of consecutive points in the sequence. It means that consecutive frames in a video segment have higher similarity than the mean similarity of those in the whole sequence.

4.3. Algorithm of video segmentation

Merging a point into a video segment is allowed if both conditions defined in Equation 11 and 13 are satisfied. For convenience, we can represent a point P_t in the video segment form by placing $sid \leftarrow NewSID()$, $SP \leftarrow P_t$, and $HR \leftarrow HR(P_t)$, that is, $\langle NewSID(), P_t, HR(P_t) \rangle$, where *NewSID*() is a function that generates the *sid* of a video segment. This video segment produced by a point P_t is denoted by $VS(P_t)$. To describe the process of merging a point, we introduce an algorithm *MERGE_POINT* that has two arguments with a positional order. The first argument is a *merging*

object that can be a video segment, while the second one is a *merged* object that is a point. This algorithm is described in Figure 5.

Algorithm *VIDEO_SEGMENTATION* in Figure 6 describes the segmentation process for a single MDS. It takes an MDS and *minPts* as input parameters, and returns the sets of video segments and outliers. In Step 0, it computes thresholds wrt. volume, edge, and distance for an MDS, to get bounding conditions. In Step 1, it evaluates geometric and semantic bounding conditions for each point of the MDS to determine whether to merge the point into the current segment or not. After this process, the number of points in each video segment is checked if it is less than *minPts*. All points in the segment with the value lower than *minPts* are regarded as outliers, and treated differently for the subsequent indexing and retrieval process.

Algorithm *MERGE_POINT*
***Input*:** video segment VS_{IN}, point P_t
***Output*:** video segment VS_{OUT}
Step 0: /* Merge a point into a video segment */
$VS_{OUT}.sid \leftarrow VS_{IN}.sid$
$VS_{OUT}.SP \leftarrow VS_{IN}.SP$
$VS_{OUT}.HR \leftarrow VS_{IN}.HR \oplus HR(P_t)$
Step 1: **return** VS_{OUT}

Figure 5. Algorithm *MERGE_POINT*

Algorithm *VIDEO_SEGMENTATION*
***Input*:** *MDS* S_i with K points, minimum number of points per video segment *minPts*
***Output*:** set of video segments $\mathbf{VS_i}$, set of outliers $\mathbf{O_i}$
Step 0: /* Initialization */
$\mathbf{VS_i} \leftarrow \phi$, $\mathbf{O_i} \leftarrow \phi$
compute τ_{vol}, τ_{edge}, and τ_{dist} for S_i
$VS_{current} \leftarrow VS$(First point P_1 of S_i)
Step 1: /* Video segment generation */
for each successive point P_j ($2 \le j \le K$) of S_i
if $\Delta Vol(VS_{current}.HR, HR(P_j)) \le \tau_{vol} \wedge$
$\Delta Edge(VS_{current}.HR, HR(P_j)) \le \tau_{edge} \wedge$
$dist(P_{j-1}, P_j) \le \tau_{dist}$ **then**
$VS_{current} \leftarrow$ ***MERGE_POINT***$(VS_{current}, P_j)$
else
if $VS_{current}.HR.k \le minPts$ **then**
$\mathbf{O_i} \leftarrow \mathbf{O_i} \cup$ {all points in $VS_{current}$}
else
$\mathbf{VS_i} \leftarrow \mathbf{VS_i} \cup \{VS_{current}\}$
$VS_{current} \leftarrow VS(P_j)$
end if
end if
end for
Step 2: **return** set $\mathbf{VS_i}$, set $\mathbf{O_i}$

Figure 6. Algorithm *VIDEO_SEGMENTATION*

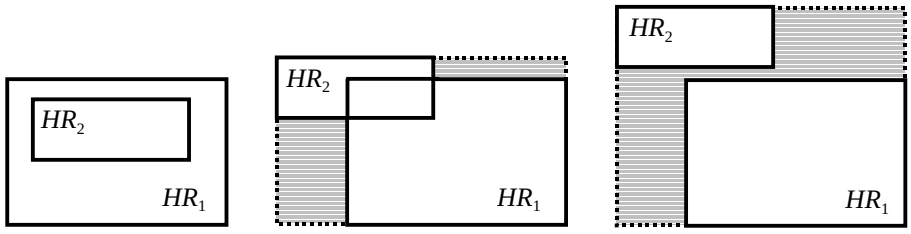
5. Video clustering

After video segments are generated from an MDS, those segments that are spatially close need to be merged together to promote the clustering quality defined in Equation 8. It is important to determine whether two hyper-rectangles of video segments or clusters are to be merged or not. Merging two hyper-rectangles is allowed as long as the predefined condition is satisfied. This process generates larger clusters gradually to optimize given measurement criteria. We formally define the video cluster as follows:

Definition 7 (*Video cluster*). A video cluster *VC* with *r* video segments in a temporal order, VS_j for $j = 1, 2, \ldots, r$, is defined as follows: $VC = \langle cid, slist, HR \rangle$, where *cid* is a cluster-id, *slist* is an ordered list of *sid*'s wrt. the temporal relationship among *VS*'s, $HR = \langle L, H, k \rangle$ such that $L = \{(L^1, L^2, \ldots, L^n) \mid L^i = min_{1 \le j \le r}(VS_j.HR.L^i)\}$ and $H = \{(H^1, H^2, \ldots, H^n) \mid H^i = max_{1 \le j \le r}(VS_j.HR.H^i)\}$ for $i = 1, 2, \ldots, n$, and $k = \Sigma_{1 \le j \le r}(VS_j.HR.k)$. ■

5.1. Placement of two hyper-rectangles

To determine whether two hyper-rectangles are to be merged or not, the spatial placement of them is important. There are three types of placements based on their relative positions: *inclusion*, *intersection*, and *disjunction*. In this section, we give an analysis on each placement with its possibility of merging. Figure 7 illustrates these placements.



(a) Inclusion (b) Intersection (c) Disjunction

Figure 7. Placement of two hyper-rectangles

Suppose that by merging two hyper-rectangles, HR_1 and HR_2, a merged hyper-rectangle HR_m is generated, that is, $HR_m = HR_1 \oplus HR_2$. Let VPP_m and EPP_m be the VPP and EPP of the merged hyper-rectangle, and VPP_n and EPP_n be those for the case of non-merging. Then, the following holds by Equation 8:

$$VPP_m = \frac{Vol(HR_m)}{HR_m.k} = \frac{Vol(HR_1 \oplus HR_2)}{HR_m.k}, \quad EPP_m = \frac{Edge(HR_m)}{HR_m.k} = \frac{Edge(HR_1 \oplus HR_2)}{HR_m.k} \tag{14}$$

$$VPP_n = \frac{Vol(HR_1) + Vol(HR_2)}{HR_1.k + HR_2.k}, \quad EPP_n = \frac{Edge(HR_1) + Edge(HR_2)}{HR_1.k + HR_2.k} \tag{15}$$

(a) *Inclusion* (Without loss of generality, we assume $HR_1 \supseteq HR_2$)

In this case, we derive the following using Equation 14 and 15 since $Vol(HR_m) = Vol(HR_1)$.

$$VPP_n = \frac{Vol(HR_1)+Vol(HR_2)}{HR_1.k + HR_2.k} = \frac{Vol(HR_m)+Vol(HR_2)}{HR_m.k} = VPP_m + \frac{Vol(HR_2)}{HR_m.k}.$$

Thus, $VPP_m = VPP_n - \frac{Vol(HR_2)}{HR_m.k}$.

By the assumption $HR_1 \supseteq HR_2$, $0 \le \frac{Vol(HR_2)}{HR_m.k} \le \frac{Vol(HR_1)}{HR_m.k} = \frac{Vol(HR_m)}{HR_m.k} = VPP_m$.

Therefore, $\frac{1}{2}VPP_n \le VPP_m \le VPP_n$. (16)

Similarly, since $Edge(HR_m) = Edge(HR_1)$, we derive:

$$EPP_n = \frac{Edge(HR_1)+Edge(HR_2)}{HR_1.k + HR_2.k} = \frac{Edge(HR_m)+Edge(HR_2)}{HR_m.k} = VPP_m + \frac{Edge(HR_2)}{HR_m.k}.$$

Thus, $EPP_m = EPP_n - \frac{Edge(HR_2)}{HR_m.k}$. Since $0 \le \frac{Edge(HR_2)}{HR_m.k} \le EPP_m$, the following hold :

$$\frac{1}{2}EPP_n \le EPP_m \le EPP_n \quad (17)$$

Since $VPP_m \le VPP_n$ and $EPP_m \le EPP_n$ by Equation 16 and 17, the clustering is always better than non-clustering. Therefore, when a hyper-rectangle is included in the other one, it is naturally allowed to merge two hyper-rectangles. The case of $VPP_m = VPP_n/2$ and $EPP_m = EPP_n/2$ occurs when two hyper-rectangles are identical.

(b) *Intersection* ($HR_1 \cap HR_2 \neq \phi$)

To get better clustering quality wrt. VPP and EPP than the case of non-clustering, $VPP_m \le VPP_n$ and $EPP_m \le EPP_n$ should hold. By Equation 14 and 15, we derive:

$$VPP_m = \frac{Vol(HR_m)}{HR_m.k} = \frac{Vol(HR_1 \oplus HR_2)}{HR_m.k} \le \frac{Vol(HR_1)+Vol(HR_2)}{HR_1.k + HR_2.k} = VPP_n \quad (18)$$

$$EPP_m = \frac{Edge(HR_m)}{HR_m.k} = \frac{Edge(HR_1 \oplus HR_2)}{HR_m.k} \le \frac{Edge(HR_1)+Edge(HR_2)}{HR_1.k + HR_2.k} = EPP_n \quad (19)$$

By Equation 18 and 19, the conditions to get the better quality is: $Vol(HR_1 \oplus HR_2) \le Vol(HR_1) + Vol(HR_2)$ and $Edge(HR_1 \oplus HR_2) \le Edge(HR_1) + Edge(HR_2)$. Let us consider the condition wrt. the edge. When two hyper-rectangles intersect, then the edges of those two rectangles intersect in every dimension. By geometric characteristics of a hyper-rectangle, the following lemma holds:

Lemma 2. *When two hyper-rectangles, HR_1 and HR_2, intersect, then the following always holds*:

$$Edge(HR_1 \oplus HR_2) \le Edge(HR_1) + Edge(HR_2) \quad (20)$$

Proof. See Appendix A.

Since $Edge(HR_1 \oplus HR_2) \le Edge(HR_1) + Edge(HR_2)$ always holds by Lemma 2, the condition to get the better quality will be: $Vol(HR_1 \oplus HR_2) \le Vol(HR_1) + Vol(HR_2)$

(c) *Disjunction* ($HR_1 \cap HR_2 = \phi$)

When two hyper-rectangles are disjoint, it is clear from Figure 7.(c) that $Vol(HR_1 \oplus HR_2)$ is always greater than $Vol(HR_1) + Vol(HR_2)$, while the relationship between $Edge(HR_1 \oplus HR_2)$ and $Edge(HR_1) + Edge(HR_2)$ varies. Using Equation 14 and 15, we derive:

$$VPP_m = \frac{Vol(HR_m)}{HR_m.k} = \frac{Vol(HR_1 \oplus HR_2)}{HR_m.k} > \frac{Vol(HR_1) + Vol(HR_2)}{HR_1.k + HR_2.k} = VPP_n \quad (21)$$

Because we consider both *VPP* and *EPP* as the clustering quality, we conclude that when two hyper-rectangles are disjoint, then the clustering quality becomes worse if we merge two rectangles, regardless of *EPP*. Thus, the merging of hyper-rectangles is not allowed in this case.

By considering all three cases that are discussed above, we finally conclude that the following lemma holds.

Lemma 3. *Merging two hyper-rectangles, HR_1 and HR_2, of video segments or video clusters guarantees better clustering quality wrt. VPP, EPP, and PPC than non-merging if the following condition holds:*

$$Vol(HR_1 \oplus HR_2) \le Vol(HR_1) + Vol(HR_2) \quad (22)$$

Proof. See Appendix A.

Lemma 3 states that when two hyper-rectangles are merged, the expanded volume *ExpVol* by merging, depicted as the shaded space in Figure 7.(b), must be equal to or less than the volume of the intersected space of two rectangles. That is:

$$Vol(HR_1 \oplus HR_2) = Vol(HR_1 \cup HR_2) + ExpVol = Vol(HR_1) + Vol(HR_2) - Vol(HR_1 \cap HR_2) + ExpVol$$

$$Vol(HR_1 \oplus HR_2) - (Vol(HR_1) + Vol(HR_2)) = - Vol(HR_1 \cap HR_2) + ExpVol$$

By Equation 22, $- Vol(HR_1 \cap HR_2) + ExpVol \le 0$. Therefore, $ExpVol \le Vol(HR_1 \cap HR_2)$ holds.

5.2 Algorithm of video clustering

Merging two hyper-rectangles is allowed only when the condition specified in Equation 22 is satisfied. We can represent a video segment *VS* in the video cluster form by placing $cid \leftarrow NewCID()$, $slist \leftarrow AddItem(VS.sid)$, and $HR \leftarrow VS.HR$, that is, $\langle NewCID(), AddItem(VS.sid), VS.HR \rangle$. Here, *NewCID*() is a function that generates the *cid* of a video cluster, and the function *AddItem*(*VS.sid*) adds the *sid* of *VS* to the *slist*. This video cluster generated by the video segment *VS* is denoted by *VC*(*VS*). Algorithm *MERGE_CLUSTERS* merges two clusters. It takes two input video clusters and generates a merged video cluster, as described in Figure 8.

Algorithm *VIDEO_CLUSTERING* shown in Figure 9 performs the clustering of video segments that are generated by algorithm *VIDEO_SEGMENTATION*. It takes the set of video segments of an MDS as an input, and produces the set of video clusters. Each video segment in the set is

represented in the video cluster form in Step 0. Each pair of video clusters is examined in Step 1 wrt. the condition specified in Lemma 3 to determine whether to merge two clusters or not.

Algorithm *MERGE_CLUSTERS*
***Input*:** video clusters VC_{IN1}, VC_{IN2}
***Output*:** video cluster VC_{OUT}
Step 0: /* Merge two video clusters */
$VC_{OUT}.cid \leftarrow NewCID()$
for each item $VS_a.sid$ in $VC_{IN1}.slist$
$VC_{OUT}.slist \leftarrow AddItem(VS_a.sid)$
end for
for each item $VS_b.sid$ in $VC_{IN2}.slist$
$VC_{OUT}.slist \leftarrow AddItem(VS_b.sid)$
end for
$VC_{OUT}.HR \leftarrow VS_{IN1}.HR \oplus VS_{IN2}.HR$
Step 1: **return** VC_{OUT}

Figure 8. Algorithm *MERGE_CLUSTERS*

Algorithm *VIDEO_CLUSTERING*
***Input*:** set of video segments $\boldsymbol{VS_i}$ for an *MDS* S_i
***Output*:** set of video clusters $\boldsymbol{VC_i}$
Step 0: /* Initialization */
$\boldsymbol{VC_i} \leftarrow \phi$
for each item VS_r in $\boldsymbol{VS_i}$
$VC_r \leftarrow VC(VS_r)$
$\boldsymbol{VC_i} \leftarrow \boldsymbol{VC_i} \cup \{VC_r\}$
end for
Step 1: /* Video cluster generation */
for each pair (VC_a, VC_b) in $\boldsymbol{VC_i}$
if $Vol(VC_a.HR \oplus VC_b.HR) \le Vol(VC_a.HR) + Vol(VC_b.HR)$ **then**
$VC_c \leftarrow$ ***MERGE_CLUSTERS***(VC_a, VC_b)
$\boldsymbol{VC_i} \leftarrow \boldsymbol{VC_i} - \{VC_a, VC_b\}$
$\boldsymbol{VC_i} \leftarrow \boldsymbol{VC_i} \cup \{VC_c\}$
end if
end for
Step 2: **return** set $\boldsymbol{VC_i}$

Figure 9. Algorithm *VIDEO_CLUSTERING*

5.3 Overall algorithm and complexity

By consolidating algorithms, *VIDEO_SEGMENTATION* and *VIDEO_CLUSTERING* that are described in Figure 6 and 9 respectively, we present the overall algorithm in Figure 10 to cluster a data set of MDS's. Except the data set itself, the algorithm takes only one input parameter *minPts* to determine outliers, and returns a set of video clusters and a set of outliers. Even though it returns a set of video clusters excluding video segments, the information on the video segments is not lost since each video cluster holds a list of them by Definition 7.

Algorithm *CLUSTER_VIDEODATASET*
***Input*:** data set of MDS's, *minPts*
***Output*:** set of video cluster VC, set of outliers O
Step 0: /* Initialization */
$VC \leftarrow \phi$, $O \leftarrow \phi$
Step 1: /* Video clustering process */
for each MDS S_i in the data set ($1 \le i \le N$)
VS_i, $O_i \leftarrow$ ***VIDEO_SEGMENTATION***(S_i, *minPts*)
$O \leftarrow O \cup O_i$
$VC_i \leftarrow$ ***VIDEO_CLUSTERING***(VS_i)
$VC \leftarrow VC \cup VC_i$
end for
Step 2: **return** set VC , set O

Figure 10. Algorithm *CLUSTER_VIDEODATASET*

By algorithm *CLUSTER_VIDEODATASET* in Figure 10, it is clear that its complexity is $O(N)$ wrt. the number of sequences N in a database. As for the number of points K in each sequence, we observe in algorithm *VIDEO_SEGMENTATION* of Figure 6 that the complexity is the order of K in Step 0 for the computation of threshold values, and also the order of K for the *for*-loop of Step 1. We do not consider the complexity wrt. the number of video segments or clusters since it is negligibly small compared to K and the value of it varies inside the algorithm. Consequently, the complexity of the overall algorithm is $O(NK)$, which is linear wrt. N and K.

6. Experiments

In order to evaluate the effectiveness of our proposed method, we have conducted experiments on data sets of various real-world videos such as TV news, dramas, and animation films. Our experiment focuses on showing the clustering quality of the method with respect to predefined measurements mentioned in Section 3.4. In this section, we describe our preparation for the experiment and give the results with brief analyses.

6.1. Experimental Preparation

For the experiment, we generated video clips of different lengths from various video data sources and extracted RGB color features from each frame of the video clips. According to the RGB feature values, a single frame is mapped to a point in a multidimensional space, and thus, each video clip is represented by an MDS. Experiments were conducted by using 3-dimensional data sets for convenience, but our method does not restrict the dimensionality of data sets.

Figure 11 depicts a sample sequence in the $[0,1]^3$ unit space. It is generated from a video clip with 3025 frames which is a part of a TV news program. Each frame of the clip in the figure is represented by a point with connected lines to adjacent points according to the temporal relationship. We executed the segmentation algorithm to get a set of video segments and a set of

outliers for each sequence in the data sets. From the video segments identified in this process, video clusters are produced by the video clustering algorithm. Figure 12 depicts the result of the clustering process, showing video clusters overlapped with the sequence in Figure 11. In the figure, a point that is not contained in any hexahedron is regarded as an outlier.

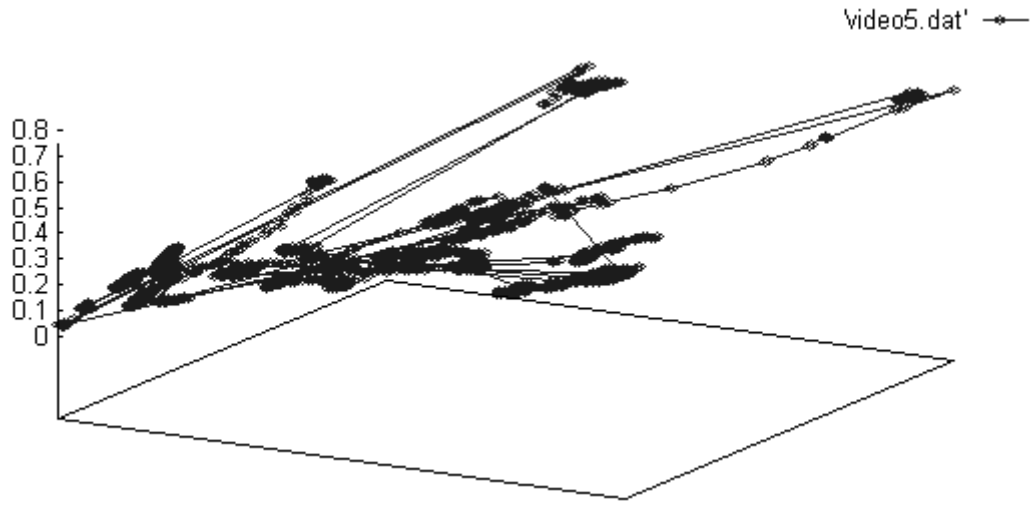


Figure 11. A 3-dimensional sequence generated from a video clip

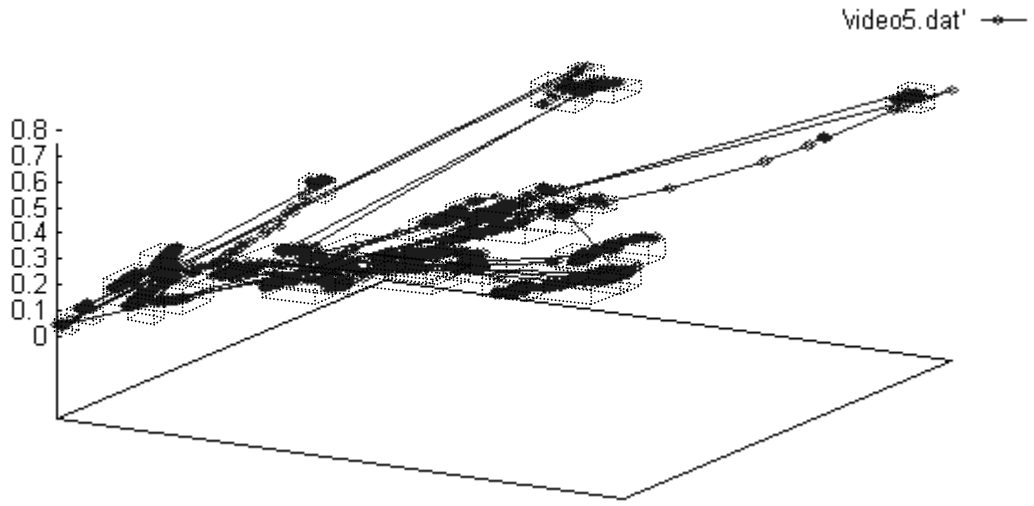


Figure 12. A 3-dimensional sequence with video clusters

Our experimental data sets consist of 3-dimensional sequences of different lengths, from 30 frames to 9000 frames (video clips of one second to five minutes length). Total number of clips used in the experiment is 6,984. Table 2 summarizes these data sets.

6.2. Experimental Results

6.2.1. Evaluation Parameters

We discussed two clustering factors, geometric and semantic factors, that should be considered for clustering MDS's. To measure the clustering quality on the geometric characteristics of video

Table 2. Test data sets used in the experiment

Data set name	Length of video clips (#frames)	Number of video clips
v1	30 < L ≤ 500	1,527
v2	500 < L ≤ 1000	1,216
v3	1000 < L ≤ 2000	1,032
v4	2000 < L ≤ 3000	836
v5	3000 < L ≤ 4000	645
v6	4000 < L ≤ 5000	508
v7	5000 < L ≤ 6000	415
v8	6000 < L ≤ 7000	336
v9	7000 < L ≤ 8000	262
v10	8000 < L ≤ 9000	207
Total		6,984

segments and clusters, we use VPP, EPP, and PPC defined in Equation 8 as evaluation parameters. In addition to the geometric characteristics, we measure the semantic relationship among points in the video segment by using the mean distance *MeanDist* between consecutive points in segments. Let us consider a video segment *VS* with *k* points, P_j for $j = 1, 2, \ldots, k$. Then, the summation of Euclidean distances between consecutive points for *VS*, $DistSum_{VS}$, will be:

$$DistSum_{VS} = \sum_{1 \le j \le k-1} dist(P_j, P_{j+1})$$

When an MDS *S* is composed of *p* video segments, VS_h for $h = 1, 2, \ldots, p$, the *MeanDist* for *S* is defined as follows:

$$MeanDist_S = \frac{\sum_{1 \le h \le p} DistSum_{VS_h}}{\sum_{1 \le h \le p} (VS_h.HR.k - 1)} \tag{23}$$

6.2.2. Results and Analyses

We compared our method to the MCOST algorithm proposed in [6] since other related algorithms [10, 11] are based on it with slight modifications. The experimental results are shown in Figure 13-17. In the figures, the results by the segmentation and the clustering of our method are denoted as *V_SEG* and *V_CL* respectively, while that by the MCOST algorithm is denoted as *MCOST*.

Volume per point: The volume of the hyper-rectangle per point (VPP) generated by each algorithm is depicted in Figure 13 over different lengths of video data sets. In the figure, we can observe that VPP's of *V_SEG* and *V_CL* decrease more sharply than that of *MCOST* as the length of the video clip increases. For long sequences (say v7 through v10), VPP of *V_CL* is 29% to 42% of that of *MCOST*. It is because our method reflects characteristics of an individual sequence during the segmentation process, including the length of the sequence. That is, a unit hyper-cube, which is used as the geometric bounding condition for segmentation, is likely to be dense as the

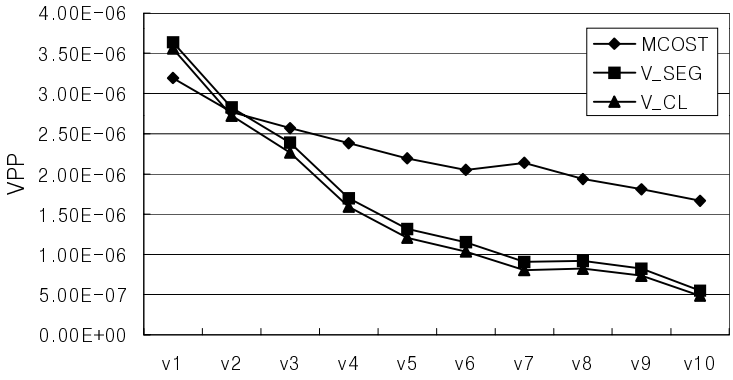


Figure 13. Volume per point comparison

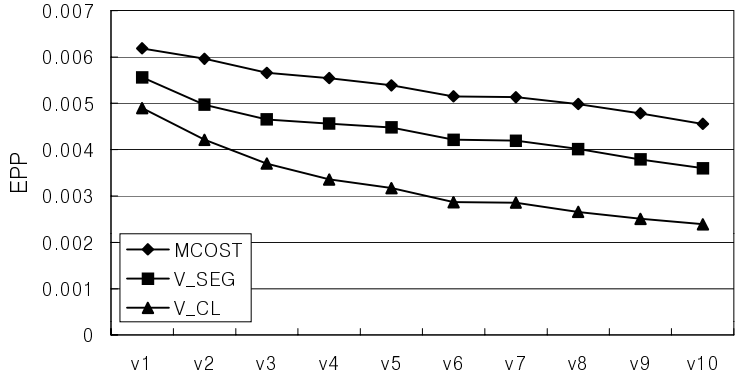


Figure 14. Edge per point comparison

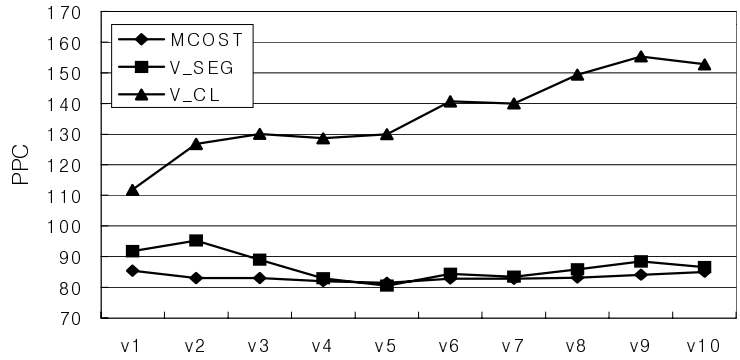


Figure 15. Number of points per cluster

length of a sequence increases. It is based on the phenomenon that a video clip has higher probability to have similar segments, as it becomes longer. For instance, a long news video clip may have many similar scenes in which the same anchor appears. It is natural that a video clip has dense segments when the unit hyper-cube of it becomes dense, since its volume is used as a bounding threshold for segmentation. In addition, our method is able to handle outliers properly,

which also contributes to generating dense segments.

Other interesting observation is that for short sequences (say, v1) VPP's of *V_SEG* and *V_CL* are larger than that of *MCOST*. The unit hyper-cube of the clip with a few segments (say 2-4 segments) may occupy large space if the segments are located far away from each other in the space. Clearly, a sparse unit-cube causes the video clip to have sparse segments by the segmentation algorithm. From the viewpoint of the volume factor, we observe that our method is more efficient than *MCOST* for video clips longer than approximately 500 frames. In reality, most of video clips are longer than 500 frames.

On the other hand, VPP of *V_CL* shows a steady enhancement compared to that of *V_SEG*, since video clusters are generated based on Lemma 3 in Section 5.

Edge per point: As for the edge of the hyper-rectangle per point (EPP), Figure 14 shows considerable differences among *MCOST*, *V_SEG*, and *V_CL*. EPP of *V_CL* is 52% to 79% of that of *MCOST*, and EPP of *V_SEG* is 78% to 89% of that of *MCOST*. This enhancement results from the fact that our method considers the edge as an important clustering factor as well as the volume. The capability of handling outliers may also contribute to the enhancement. EPP's of *V_SEG* and *V_CL* decrease as the length of a video clip increases, which is caused by the same reason stated in VPP's case. EPP of *V_CL* shows 66% to 88% of that of *V_SEG*, while VPP of *V_CL* shows a small enhancement compared to that of *V_SEG*. It illustrates that merging video segments in the video clustering process improves EPP much rather than VPP.

Number of points per cluster: Related to VPP and EPP discussed above, the number of points per cluster (PPC) is shown in Figure 15. As we can observe in the figure, PPC's of *MCOST* and *V_SEG* show a little difference. PPC of *V_SEG* shows a slight enhancement over all video data sets, compared to *MCOST*. However, the clustering process that merges multiple segments into a cluster enhances PPC greatly. PPC of *V_CL* is 1.30 to 1.85 times better than that of *MCOST*.

From experimental results regarding VPP, EPP, and PPC, we conclude that our method generates denser clusters than *MCOST* wrt. the volume and edge, and the densities of clusters become higher as video clips become longer.

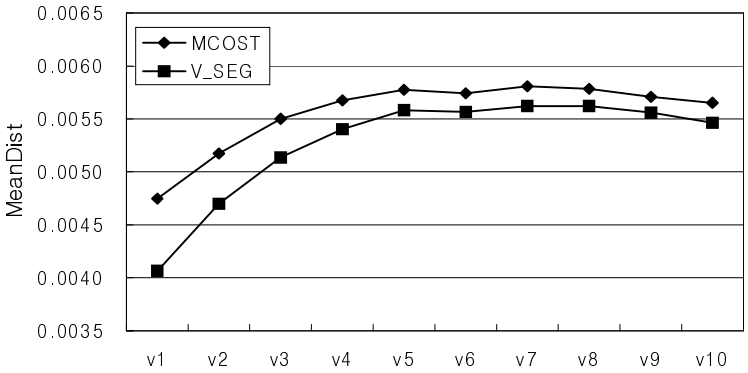


Figure 16. Mean distance between consecutive points in segments

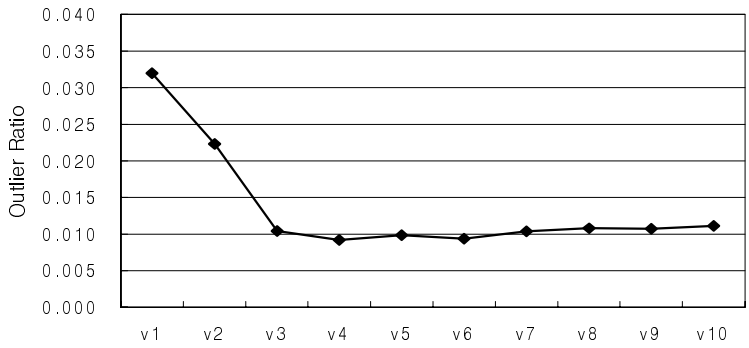


Figure 17. Ratio of outliers with respect to the total points in sequences

Mean distance between points in segments: As an indicator to show the semantic relationship between points in the segment, the *MeanDist* is evaluated as shown in Figure 16. Apparently, the small value of the *MeanDist* indicates that the elements in segments are semantically closer than the large value. The *MeanDist* of *V_SEG* is 85% to 97% of that of *MCOST* over all video data sets, which illustrates that video segments produced by our method have a closer semantic relationship among their elements than those produced by *MCOST*. It is because our method considers the semantic bounding condition described in Definition 6 as an important clustering factor, while *MCOST* does not.

Ratio of outliers: Figure 17 shows the ratio of the number of outliers versus the number of total points in sequences. To determine outliers, *minPts* is set to 5. We believe this choice is reasonable in the video application domain, since it takes approximately 1/6 sec. to play a 5-frame video and the segments with frames less than 5 are nearly meaningless in reality. A 1/6 sec. playing time may not be perceived well by the human visual system. Starting from 3.2% for short sequences, it shows an almost steady rate of 1.0% for sequences longer than 1000 frames. These outliers are treated separately from video clusters for similarity search processing.

7. Conclusions

The retrieval of video data sets is one of the great potential areas in database applications, even though it has not been widely studied. For an efficient retrieval of video data sets, the clustering process is essential as a foundational work for representing, indexing, and storing video data sets. In this paper, we have investigated the segmentation and clustering of large video data sets. To solve the problem, we have first discussed clustering factors considering geometric and semantic characteristics of clusters, and defined the measures to evaluate the clustering quality. Based on these clustering factors and measures, we proposed an effective clustering method that has the following desirable properties:

- It maintains the temporal and semantic relationship among elements within a video cluster.

- It generates dense clusters wrt. the volume and the edge which satisfy the predefined criteria of clustering quality.
- It identifies outliers properly in order to deal with them differently from video clusters in the subsequent retrieval process.

Another important property of our method is that most of parameter values for clustering are determined using the characteristics of the video clip, not supplied by the user. It can be of benefit from the following two aspects: First, our method needs only one parameter *minPts* to determine outliers as an input. It is desirable to minimize the number of input parameters since it is not easy to get the domain knowledge in advance in order to set input values, such as the number of clusters to be generated or the minimum distance between clusters. Second, our method generates clusters considering the properties of a video clip, such as the type of a video. For example, the scenes of a news video are not frequently changed while those of a cartoon or an animation film are frequently changed.

We have performed experiments with various real video data sets and examined the clustering quality. Our method has shown considerable effectiveness wrt. VPP, EPP, PPC, and *MeanDist* as shown in the experimental results. As a future work, we plan to study the similarity search method for large vedeo data sets based on the video segments and clusters proposed in this paper.

References

[1] C. C. Aggarwal, C. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park, Fart algorithms for projected clustering, in: Proceedings of ACM SIGMOD Int'l Conference on Management of Data, Pennsylvania, 1999, pp. 61-72.

[2] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, Automatic subspace clustering of high dimensional data for data mining applications, in: Proceedings of ACM SIGMOD Int'l Conference on Management of Data, Washington, 1998, pp. 94-105.

[3] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, The R*-tree: an efficient and robust access method for points and rectangles, in: Proceedings of ACM SIGMOD Int'l Conference on Management of Data, New Jersey, 1990, pp. 322-331.

[4] S. Berchtold, D. Keim, and H. Kriegel, The X-tree: an index structure for high-dimensional data, in: Proceedings of Int'l Conference on Very Large Data Bases, India, 1996, pp. 28-39.

[5] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Int'l Conference on Knowledge Discovery in Databases and Data Mining, Oregon, 1996, pp. 226-231.

[6] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, Fast subsequence matching in time-series databases, in: Proceedings of ACM SIGMOD Int'l Conference on Management of Data, Minnesota, 1994, pp. 419-429.

[7] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, Query by image and video content: the QBIC system, IEEE Computer, 28(1995), 23-32.

[8] S. Guha, R. Rastogi, and K. Shim, CURE: An efficient clustering algorithm for large databases, in: Proceedings of ACM SIGMOD Int'l Conference on Management of Data, Washington, 1998, pp. 73-84.

[9] A.Guttman, R-trees: a dynamic index structure for spatial searching, in: Proceedings of ACM SIGMOD Int'l Conference on Management of Data, Massachusetts, 1984, pp. 47-57.

[10] V. Kobla, D. Doermann, and C. Faloutsos, Video Trails: Representing and visualizing structure in video sequences, in: Proceedings of ACM Multimedia, Washington, 1997, pp. 335-346.

[11] S. L. Lee, S. J. Chun, D. H. Kim, J. H. Lee, and C. W. Chung, Similarity search for multidimensional data sequences, in: Proceedings of IEEE Int'l Conference on Data Engineering, California, 2000, pp. 599-608.

[12] S. L. Lee, and C. W. Chung, On the effective clustering of multidimensional data sequences, Technical Report CS-TR-2000-154, in http://cs.kaist.ac.kr/library/tr/, KAIST, 2000.

[13] R. T. Ng and J. Han, Efficient and effective clustering methods for spatial data mining, in: Proceedings of Int'l Conference on Very Large Data Bases, Chile, 1994, pp. 144-155.

[14] T. Sellis, N. Roussopoulos, and C. Faloutsos, The R+ tree: a dynamic index for multi-dimensional objects, in: Proceedings of Int'l Conference on Very Large Data Bases, England, 1987, pp. 507-518.

[15] T. Zhang, R. Ramakrishnan, and M. Livny, BIRCH: An efficient data clustering method for very large databases, in: Proceedings of ACM SIGMOD Int'l Conference on Management of Data, Canada, 1996, pp. 103-114.

Appendix A. Proofs of Lemmas

Lemma 1. *The clustering that satisfies the geometric bounding condition guarantees better clustering quality than the case of the uniform distribution, wrt. VPP and EPP.*

Proof. Suppose that the segmentation of MDS S generates p video segments and the set is $\boldsymbol{VS} = \{VS_1, \ldots, VS_p\}$. Let VS_j $(1 \le j \le p)$ be generated by merging a point u times starting with a single point, and the increments of volume and edge by the lth merging $(1 \le l \le u)$ be $\Delta Vol_{j,l}$ and $\Delta Edge_{j,l}$, respectively. Since $VS_j.HR.k = 1$, $Vol(VS_j.HR) = 0$, and $Edge(VS_j.HR) = 0$ in the initial state, after merging u times, we get: $VS_j.HR.k = 1 + u$, $Vol(VS_j.HR) = \sum_{1 \le l \le u} \Delta Vol_{j,l}$, and $Edge(VS_j.HR) = \sum_{1 \le l \le u} \Delta Edge_{j,l}$. Since every merging step satisfies Equation 11, the following holds: $\Delta Vol_{j,l} \le e^n$ and $\Delta Edge_{j,l} \le 2^{n-1} \cdot n \cdot e$. Thus, we derive:

$$Vol(VS_j.HR) = \sum_{1 \le l \le u} \Delta Vol_{j,l} \le \sum_{1 \le l \le u} e^n < (1+u) \cdot e^n = VS_j.HR.k \cdot e^n$$

$$Edge(VS_j.HR) = \sum_{1 \le l \le u} \Delta Edge_{j,l} \le \sum_{1 \le l \le u} (2^{n-1} \cdot n \cdot e) < (1+u) \cdot 2^{n-1} \cdot n \cdot e = VS_j.HR.k \cdot 2^{n-1} \cdot n \cdot e$$

Suppose that the volume and the edge per point wrt. MDS S after segmentation are VPP_S and EPP_S, and those of the uniform distribution are VPP_0 and EPP_0 respectively. Using Equation 8 and above equations, we conclude:

$$VPP_S = \frac{\sum_{1 \le j \le p} Vol(VS_j.HR)}{\sum_{1 \le j \le p} VS_j.HR.k} < \frac{\sum_{1 \le j \le p} (VS_j.HR.k) \cdot e^n}{\sum_{1 \le j \le p} VS_j.HR.k} = e^n = VPP_0$$

$$EPP_S = \frac{\sum_{1 \le j \le p} Edge(VS_j.HR)}{\sum_{1 \le j \le p} VS_j.HR.k} < \frac{\sum_{1 \le j \le p} (VS_j.HR.k) \cdot 2^{n-1} \cdot n \cdot e}{\sum_{1 \le j \le p} VS_j.HR.k} = 2^{n-1} \cdot n \cdot e = EPP_0 \quad \blacksquare$$

Lemma 2. *When two hyper-rectangles, HR_1 and HR_2, intersect, then the following always holds*:

$$Edge(HR_1 \oplus HR_2) \le Edge(HR_1) + Edge(HR_2) \tag{20}$$

Proof. Let $e_i(HR)$ be the edge length of HR in dimension i. When HR_1 and HR_2 intersect, the edges of HR_1 and HR_2 intersect in every dimension. It is clear that $e_i(HR_1 \oplus HR_2) \le e_i(HR_1) + e_i(HR_2)$ for $i = 1, 2, \ldots, n$. If we consider all dimensions, then:

$$\begin{aligned} Edge(HR_1 \oplus HR_2) &= \sum_{1 \le i \le n} e_i(HR_1 \oplus HR_2) \\ &\le \sum_{1 \le i \le n} (e_i(HR_1) + e_i(HR_2)) \\ &= \sum_{1 \le i \le n} e_i(HR_1) + \sum_{1 \le i \le n} e_i(HR_2) \\ &= Edge(HR_1) + Edge(HR_2). \end{aligned}$$

Therefore, Lemma 2 holds. ■

Lemma 3. *Merging two hyper-rectangles, HR_1 and HR_2, of video segments or video clusters guarantees better clustering quality wrt. VPP, EPP, and PPC than non-merging if the following condition holds:*

$$Vol(HR_1 \oplus HR_2) \le Vol(HR_1) + Vol(HR_2) \tag{22}$$

Proof. When the above condition holds, we derive the following:

$$VPP_m = \frac{Vol(HR_1 \oplus HR_2)}{HR_1.k + HR_2.k} \le \frac{Vol(HR_1) + Vol(HR_2)}{HR_1.k + HR_2.k} = VPP_n.$$

Since $Vol(HR_1 \oplus HR_2) \le Vol(HR_1) + Vol(HR_2)$ holds when HR_1 and HR_2 intersect, we know that $Edge(HR_1 \oplus HR_2) \le Edge(HR_1) + Edge(HR_2)$ always holds from Lemma 2. Thus, we derive:

$$EPP_m = \frac{Edge(HR_1 \oplus HR_2)}{HR_1.k + HR_2.k} \le \frac{Edge(HR_1) + Edge(HR_2)}{HR_1.k + HR_2.k} = EPP_n.$$

Let PPC_m be PPC of a merged hyper-rectangle and PPC_n for the case of non-merging. Then, it is clear by Equation 8 that PPC_n is $(HR_1.k + HR_2.k) / 2$ while PPC_m is $(HR_1.k + HR_2.k)$. Thus, $PPC_m > PPC_n$. Therefore, Lemma 3 holds. ■