Efficient Video Similarity Measurement with Video Signature

*Sen-ching S. Cheung and Avideh Zakhor
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720 USA
email: {*cheungsc avz*}*@eecs.berkeley.edu*

Abstract

The proliferation of video content on the web makes similarity detection an indispensable tool in web data management, searching, and navigation. In this paper, we propose a number of algorithms to efficiently measure video similarity. We define video as a set of frames, which are represented as high dimensional vectors in a feature space. Our goal is to measure Ideal Video Similarity (IVS), defined as the percentage of clusters of similar frames shared between two video sequences. Since IVS is too complex to be deployed in large database applications, we approximate it with Voronoi Video Similarity (VVS), defined as the volume of the intersection between Voronoi Cells of similar clusters. We propose a class of randomized algorithms to estimate VVS by first summarizing each video with a small set of its sampled frames, called the Video Signature (ViSig), and then calculating the distances between corresponding frames from the two ViSig's. By generating samples with a probability distribution that describes the video statistics, and ranking them based upon their likelihood of making an error in the estimation, we show analytically that ViSig can provide an unbiased estimate of IVS. Experimental results on a large dataset of web video and a set of MPEG-7 test sequences with artificially generated similar versions are provided to demonstrate the retrieval performance of our proposed techniques.

1 Introduction

The amount of information on the world wide web has grown enormously since its creation in 1990. By February 2000, the web had over one billion uniquely indexed pages and 30 million audio, video and image links [1]. Since there is no central management on the web, duplication of content is inevitable. A study done in 1998 estimated that about 46% of all the text documents on the web have at least one

*The authors thank Professor A. Sinclair, Dr. W.-t. Tan, and anonymous reviewers for providing them with valuable comments on this paper.

"near-duplicate" - document which is identical except for low level details such as formatting [2]. The problem is likely to be more severe for web video clips as they are often stored in multiple locations, compressed with different algorithms and bitrates to facilitate downloading and streaming. Similar versions, in part or as a whole, of the same video can also be found on the web when some web users modify and combine original content with their own productions. Identifying these similar contents is beneficial to many web video applications:

- As users typically do not view beyond the first result screen from a search engine [3], it is detrimental to have all "near-duplicate" entries cluttering the top retrievals. Rather, it is advantageous to group together similar entries before presenting the retrieval results to users.
- When a particular web video becomes unavailable or suffers from slow network transmission, users can opt for a more accessible version among similar video content identified by the video search engine.
- Similarity detection algorithms can also be used for content identification when conventional techniques such as watermarking are not applicable. For example, multimedia content brokers may use similarity detection to check for copyright violation as they have no right to insert watermarks into original material.

One definition of a video similarity measure suitable for these applications is in terms of the percentage of visually similar frames between two video sequences. This is the video similarity measure used in this paper. An analogous measure, called the Tanimoto measure, is commonly used in comparing text documents where the similarity is defined as the percentage of words common to the two documents [2, 4]. There are other similarity measures proposed in the literature and some of them are reviewed in this paper. No matter which measure is used, the major challenge is how to efficiently perform the measurement. As video is a complex data type, many of the proposed similarity measures are computationally intensive. On the other hand, for every new video added to the database or a query video presented by a user, similarity measurements need to be performed with possibly millions of entries in the database. Thus, it is imperative to develop fast methods in computing similarity measurements for database applications.

Finding visually similar content is the central theme in the area of Content-Based Information Retrieval (CBIR). In the past decade, numerous algorithms have been proposed to identify visual content similar in color, texture, shape, motion and many other attributes. These algorithms typically

identify a particular attribute of an image or a video shot as a high-dimensional feature vector. Visual similarity between two feature vectors can then be measured by a metric defined on the feature vector space. The basic premise of this process is that the visual content being analyzed is homogeneous in the attribute of interest. Since a full-length video is typically a collection of video shots with vastly different content, it must be modeled as a set or a time-series of feature vectors, usually with one vector per video shot. When extending the similarity measurement to video, the first challenge is to define a single measurement to gauge the similarity between two video sequences. Multiple proposals can be found in the literature: in [5, 6, 7], warping distance is used to measure the temporal edit differences between video sequences. Hausdorff distance is proposed in [8] to measure the maximal dissimilarity between shots. Template matching of shot change duration is used by Indyk et al. [9] to identify the overlap between video sequences. A common step shared by all the above schemes is to match similar feature vectors between two video sequences. This usually requires searching through part of or the entire video. The full computations of these measurements thus require the storage of the entire video, and time complexity that is at least linear in the length of the video. Applying such computations in finding similar contents within a database of millions of video sequences may be too complex in practice.

On the other hand, the precise value of a similarity measurement is typically not required. As feature vectors are idealistic models and do not entirely capture the process of how similarity is judged in the human visual system [10], many CBIR applications only require an approximation of the underlying similarity value. As such, it is unnecessary to maintain full fidelity of the feature vector representations, and approximation schemes can be used to alleviate high computational complexity. For example, in a video similarity search system, each video in the database can be first summarized into a compact fixed-size representation. Then, the similarity between two video sequences can be approximated by comparing their corresponding representations.

There are two types of summarization techniques for similarity approximation: the higher-order and the first-order techniques. The higher-order techniques summarize all feature vectors in a video as a statistical distribution. These techniques are useful in classification and semantic retrieval as they are highly adaptive and robust against small perturbation. Nonetheless, they typically assume a restricted form of density models such as Gaussian, or mixtures of Gaussian, and require computationally intensive method like Expectation-Maximization for parameter estimation [11, 12, 13]. As a result, they may not be applicable for matching the enormous amount of and extremely diverse video content on

the web. The first-order techniques summarize a video by a small set of representative feature vectors. One approach is to compute the "optimal" representative vectors by minimizing the distance between the original video and its representation. If the metric is finite-dimensional Euclidean and the distance is the sum of squared metric, the well-known k-means method can be used [14]. For general metric spaces, we can use the k-medoid method which identifies k feature vectors within the video to minimize the distance [15, 8]. Both of these algorithms are iterative with each iteration running at $O(l)$ time for k-means, and $O(l^2)$ for k-medoids, where l represents the length of the video. To summarize long video sequences such as feature-length movies or documentaries, such methods are clearly too complex to be used in large databases.

In this paper, we propose a randomized first-order video summarization technique called the Video Signature (ViSig) method. The ViSig method can be applied to any metric space. Unlike the k-means or k-medoid methods, it is a single-pass $O(l)$ algorithm in which each video is represented by a set of "randomly" selected frames called the ViSig. In this paper, we show analytically and experimentally that we can obtain a reliable estimate of the underlying video similarity by using a very small ViSig to represent each video in the database. Based on a ground-truth set extracted from a large database of web video clips, we show that the ViSig method is able to achieve almost the same retrieval performance as the k-medoid of the same size, without the $O(l^2)$ complexity of k-medoid.

This paper is organized as follows: we describe the ViSig method and show a number of analytical results in Section 2. Experimental results on a large dataset of web video and a set of MPEG-7 test sequences with simulated similar versions are used in Section 3 to demonstrate the retrieval performance of our proposed algorithms. We conclude this paper in Section 4 by discussing related research. The proofs for all the propositions can be found in the appendices. The following is a list of acronyms and notations used in this paper:

Acronyms

NVS Naïve Video Similarity

IVS Ideal Video Similarity

VVS Voronoi Video Similarity

ViSig Video Signature

SV Seed Vector

VSS$_b$ Basic ViSig Similarity

PDF Probability Density Function

VG Voronoi Gap

VSS$_r$ Ranked ViSig Similarity

Notations

$(F, d(\cdot,\cdot))$ Feature vector space F with metric $d(\cdot,\cdot)$

x, y Video frames, represented as feature vectors

X, Y Video sequences, represented as sets of feature vectors

ϵ Frame Similarity Threshold

1_X Indicator function

$|X|$ Cardinality of set X

$\text{nvs}(X, Y; \epsilon)$ NVS between X and Y

$[X]_\epsilon$ Collection of clusters in X

$[X \cup Y]_\epsilon$ Clustered union between X and Y

$\text{ivs}(X, Y; \epsilon)$ IVS between X and Y

$V(X)$ Voronoi Diagram of video X

$V_X(x)$ Voronoi Cell of $x \in X$

$V_X(C)$ Voronoi Cell of a cluster $C \in [X]_\epsilon$

$g_X(s)$ The frame in X closest to s

$R(X, Y; \epsilon)$ Similar Voronoi Region

$\text{Vol}(A)$ Volume of a region A

$\text{Prob}(A)$ Probability of event A

$\text{vvs}(X, Y; \epsilon)$ VVS between X and Y

$\overrightarrow{X_S}$ ViSig of X with respect to the SV set S

$\text{vss}_b(\overrightarrow{X_S}, \overrightarrow{Y_S}; \epsilon, m)$ VSS_b between $\overrightarrow{X_S}$ and $\overrightarrow{Y_S}$

$f(u; X \cup Y)$ PDF that assigns equal probability to the Voronoi Cell of each cluster in $[X \cup Y]_\epsilon$

$G(X, Y; \epsilon)$ VG between X and Y

$Q(g_X(s))$ Ranking function for the ViSig frame $g_X(s)$

$\text{vss}_r(\overrightarrow{X_S}, \overrightarrow{Y_S}; \epsilon, m)$ VSS_r between $\overrightarrow{X_S}$ and $\overrightarrow{Y_S}$

$d_q(x, y), d'_q(x, y)$ l_1 and modified l_1 color histogram distances

$d(x, y), d'(x, y)$ Quadrant $d_q(x, y)$ and $d'_q(x, y)$

ρ Dominant Color Threshold

$\text{rel}(X)$ The set of video sequences that are subjectively similar (relevant) to video X

$\text{ret}(X, \epsilon)$ The set of video sequences that are declared to be similar to X by the ViSig method at ϵ level

$\text{Recall}(\epsilon)$ The recall in retrieving the ground-truth by the ViSig method at ϵ level

$\text{Precision}(\epsilon)$ The precision in retrieving the ground-truth by the ViSig method at ϵ level

2 Measuring Video Similarity

This section defines the video similarity models used in this paper, and describes how they can be efficiently estimated by our proposed algorithms. We assume that individual frames in a video are represented by high-dimensional feature vectors from a metric space $(F, d(\cdot,\cdot))$[1]. In order to be robust against editing changes in temporal domain, we define a video X as a finite set of feature vectors and ignore any temporal ordering. For the remainder of this paper, we make no distinction between a video frame and its corresponding feature vector. The metric $d(x, y)$ measures the visual dissimilarity between frames x and y. We assume that frames x and y are visually similar to each other if and only if $d(x, y) \leq \epsilon$ for an $\epsilon > 0$ independent of x and y. We call ϵ the Frame Similarity Threshold.

In Section 2.1, we define our target measure, called the Ideal Video Similarity (IVS), used in this

[1] For all x, y in a metric space F, the function $d(x, y)$ is a metric if a) $d(x, y) \geq 0$; b) $d(x, y) = 0 \Leftrightarrow x = y$; c) $d(x, y) = d(y, x)$; d) $d(x, y) \leq d(x, z) + d(z, y)$, for all z.

paper to gauge the visual similarity between two video sequences. As we explain in the section, this similarity measure is complex to compute exactly, and requires a significant number of frames to represent each video. An alternative form of video similarity, called VVS, is proposed in Section 2.2. This particular form of similarity leads directly to an efficient technique for representation and estimation called the ViSig method which is described in Section 2.3. From Sections 2.4 through 2.7, we analyze the scenarios where IVS cannot be reliably estimated by our proposed algorithm, and propose a number of heuristics to rectify the problems.

2.1 Ideal Video Similarity (IVS)

As mentioned in Section 1, we are interested in defining a video similarity measure that is based on the percentage of visually similar frames between two sequences. A naive way to compute such a measure is to first find the total number of frames from each video sequence that have at least one visually similar frame in the other sequence, and then compute the ratio of this number to the overall total number of frames. We call this measure the Naïve Video Similarity (NVS):

Definition 2.1 Naïve Video Similarity
Let X and Y be two video sequences. The number of frames in video X that have at least one visually similar frame in Y is represented by $\sum_{x\in X} 1_{\{y\in Y:\, d(x,y)\le\epsilon\}}$, where 1_A is the indicator function with $1_A = 1$ if A is not empty, and zero otherwise. The Naïve Video Similarity between X and Y, $\mathrm{nvs}(X, Y; \epsilon)$*, can thus be defined as follows:*

$$\mathrm{nvs}(X, Y; \epsilon) \triangleq \frac{\sum_{x\in X} 1_{\{y\in Y:\, d(x,y)\le\epsilon\}} + \sum_{y\in Y} 1_{\{x\in X:\, d(y,x)\le\epsilon\}}}{|X| + |Y|}, \tag{1}$$

where $|\cdot|$ denotes the cardinality of a set, or in our case the number of frames in a given video.

If every frame in video X has a similar match in Y and vice versa, $\mathrm{nvs}(X, Y; \epsilon) = 1$. If X and Y share no similar frames at all, $\mathrm{nvs}(X, Y; \epsilon) = 0$.

Unfortunately, NVS does not always reflect our intuition of video similarity. Most real-life video sequences can be temporally separated into video shots, within which the frames are visually similar. Among all possible versions of the same video, the number of frames in the same shot can be quite different. For instance, different coding schemes modify the frame rates for different playback capabilities, and video summarization algorithms use a single keyframe to represent an entire shot. As NVS is based solely on frame counts, its value is highly sensitive to these kinds of manipulations. To illustrate this with a pathological example, consider the following: given a video X, create a video Y by repeating one single frame in X for a great many times. If $|Y| \gg |X|$, $\mathrm{nvs}(X, Y; \epsilon) \approx 1$ even though X and Y share one common frame. It is possible to rectify the problem by using shots as the

fundamental unit for similarity measurement. Since we model a video as a set and ignore all temporal ordering, we instead group all visually similar frames in a video together into non-intersecting units called clusters.

A cluster should ideally contain only similar frames, and no other frames similar to the frames in a cluster should be found in the rest of the video. Mathematically, we can express these two properties as follows: for all pairs of frames x_i and x_j in X, $d(x_i, x_j) \leq \epsilon$ if and only if x_i and x_j belong to the same cluster. Unfortunately, such a clustering structure may not exist for an arbitrary video X. Specifically, if $d(x_i, x_j) \leq \epsilon$ and $d(x_j, x_k) \leq \epsilon$, there is no guarantee that $d(x_i, x_k) \leq \epsilon$. If $d(x_i, x_k) > \epsilon$, there is no consistent way to group all the three frames into clusters.

In order to arrive at a general framework for video similarity, we adopt a relatively relaxed clustering structure by only requiring the forward condition, i.e. $d(x_i, x_j) \leq \epsilon$ implies that x_i and x_j are in the same cluster. A cluster is simply one of the connected components [16, ch.5] of a graph in which each node represents a frame in the video, and every pair of frames within ϵ from each other are connected by an edge. We denote the collection of all clusters in video X as $[X]_\epsilon$. It is possible for such a definition to produce chain-like clusters where one end of a cluster is very far from the other end. Nonetheless, given an appropriate feature vector and a reasonably small ϵ, we have empirically found most clusters in real video sequences to be compact, i.e. all frames in a cluster are similar to each other. We call a cluster ϵ-compact if all its frames are within ϵ from each other. The clustering structure of a video can be computed by a simple hierarchical clustering algorithm called the single-link algorithm [17].

In order to define a similarity measure based on the visually similar portion shared between two video sequences X and Y, we consider the clustered union $[X \cup Y]_\epsilon$. If a cluster in $[X \cup Y]_\epsilon$ contains frames from both sequences, these frames are likely to be visually similar to each other. Thus, we call such a cluster Similar Cluster and consider it as part of the visually similar portion. The ratio between the number of Similar Clusters and the total number of clusters in $[X \cup Y]_\epsilon$ forms a reasonable similarity measure between X and Y. We call this measure the Ideal Video Similarity (IVS):

Definition 2.2 Ideal Video Similarity, IVS
Let X and Y be two video sequences. The IVS between X and Y, or $\mathrm{ivs}(X, Y; \epsilon)$*, is defined to be the fraction of clusters in $[X \cup Y]_\epsilon$ that contain frames from both sequences, i.e. $C \in [X \cup Y]_\epsilon$ with $1_{C \cap X} \cdot 1_{C \cap Y} = 1$. Specifically,* $\mathrm{ivs}(X, Y; \epsilon)$ *can be expressed by the following equation:*

$$\mathrm{ivs}(X, Y; \epsilon) \triangleq \frac{\sum_{C \in [X \cup Y]_\epsilon} 1_{C \cap X} \cdot 1_{C \cap Y}}{|[X \cup Y]_\epsilon|} \tag{2}$$

The main theme of this paper is to develop efficient algorithms to estimate the IVS between a pair of video sequences. A simple pictorial example to demonstrate the use of IVS is shown in Figure 1(a). The feature space is represented as a 2D square. Dots and crosses signify frames from two different video sequences, and frames closer than ϵ are connected by dotted lines. There are altogether three clusters in the clustered union, and only one cluster has frames from both sequences. The IVS is thus 1/3.

It is complex to precisely compute the IVS. The clustering used in IVS depends on the distances between frames from the two sequences. This implies that for two l-frame video sequences, one needs to first compute the distance between l^2 pairs of frames before running the clustering algorithm and computing the IVS. In addition, the computation requires the entire video to be stored. The complex computation and large storage requirements are clearly undesirable for large database applications. As the exact similarity value is often not required in many applications, sampling techniques can be used to estimate IVS. Consider the following simple sampling scheme: let each video sequence in the database be represented by m randomly selected frames. We estimate the IVS between two sequences by counting the number of similar pairs of frames W_m between their respective sets of sampled frames. As long as the desired level of precision is satisfied, m should be chosen as small as possible to achieve low complexity. Nonetheless, even in the case when the IVS is as large as one, we show in the following proposition that we need a large m to find even one pair of similar frames among the sampled frames.

Proposition 2.1 *Let X and Y be two l-frame video sequences. Assume for every frame x in X, Y has exactly one frame y similar to it, i.e. $d(x, y) \leq \epsilon$. We also assume the same for every frame in Y. Clearly,* $\text{ivs}(X, Y; \epsilon) = 1$. *The expectation of the number of similar frame pairs W_m found between m randomly selected frames from X and from Y is given below:*

$$E(W_m) = \frac{m^2}{l}. \tag{3}$$

Despite the fact that the IVS between the video sequences is one, Equation (3) shows that we need, on average, $m = \sqrt{l}$ sample frames from each video to find just one similar pair. Furthermore, comparing two sets of $\sqrt{l}$ frames requires l high-dimensional metric computations. A better random sampling scheme should use a fixed-size record to represent each video, and require far fewer frames to identify highly similar video sequences. Our proposed ViSig method is precisely such a scheme and is the topic of the following section.

2.2 Voronoi Video Similarity

As described in the previous section, the simple sampling scheme requires a large number of frames sampled from each video to estimate IVS. The problem lies in the fact that since we sample frames from two video sequences independently, the probability that we simultaneously sample a pair of similar frames from them is rather small. Rather than independent sampling, the ViSig method introduces dependence by selecting frames in each video that are similar to a set of predefined random feature vectors common to all video sequences. As a result, the ViSig method takes far fewer sampled frames to find a pair of similar frames from two video sequences. The number of pairs of similar frames found by the ViSig method depends strongly on the IVS, but does not have a one-to-one relationship with it. We call the form of similarity estimated by the ViSig method the Voronoi Video Similarity (VVS). In this section, we explain VVS and in Section 2.3, we discuss how it is estimated by the ViSig method. The discrepancies between VVS and IVS, and how they can be rectified by modifying the ViSig method are addressed in Sections 2.4 and 2.5.

The term "Voronoi" in VVS is borrowed from a geometrical concept called the Voronoi Diagram. Given a l-frame video $X = \{x_t : t = 1, \ldots, l\}$, the Voronoi Diagram $V(X)$ of X is a partition of the feature space F into l Voronoi Cells $V_X(x_t)$. By definition, the Voronoi Cell $V_X(x_t)$ contains all the vectors in F closer to $x_t \in X$ than to any other frames in X, i.e. $V_X(x_t) \stackrel{\Delta}{=} \{s \in F : g_X(s) = x_t \text{ and } x_t \in X\}$, where $g_X(s)$ denotes the frame in X closest[2] to s. A simple Voronoi Diagram of a 3-frame video is shown in Figure 1(b). We can extend the idea of the Voronoi Diagram to video clusters by merging Voronoi Cells of all the frames belonging to the same cluster. In other words, for $C \in [X]_\epsilon$, $V_X(C) \stackrel{\Delta}{=} \bigcup_{x \in C} V_X(x)$. Given two video sequences X and Y and their corresponding Voronoi Diagrams, we define the Similar Voronoi Region $R(X, Y; \epsilon)$ as the union of all the intersection between the Voronoi Cells of those $x \in X$ and $y \in Y$ where $d(x, y) \leq \epsilon$:

$$R(X, Y; \epsilon) \stackrel{\Delta}{=} \bigcup_{d(x,y) \leq \epsilon} V_X(x) \cap V_Y(y). \tag{4}$$

It is easy to see that if x and y are close to each other, their corresponding Voronoi Cells are very likely to intersect in the neighborhood of x and y. The larger number of frames from X and from Y

[2] If there are multiple x's in X that are equidistant to s, we choose $g_X(s)$ to be the one closest to a predefined vector in the feature space such as the origin. If there are still multiple candidates, more predefined vectors can be used until a unique $g_X(s)$ is obtained. Such an assignment strategy ensures that $g_X(s)$ depends only on X and s but not some arbitrary random choices. This is important to the ViSig method which uses $g_X(s)$ as part of a summary of X with respect to a randomly selected s. Since $g_X(s)$ depends only on X and s, sequences identical to X produce the same summary frame with respect to s.

that are close to each other, the larger the resulting $R(X, Y; \epsilon)$ becomes. A simple pictorial example of two video sequences with their Voronoi Diagrams is shown in Figure 1(c): dots and crosses represent the frames of the two sequences; the solid and broken lines are the boundary between the two Voronoi Cells of the two sequences represented by dots and crosses respectively. The shaded region shows the Similar Voronoi Region between these two sequences. Similar Voronoi Region is the target region whose volume defines VVS. Before providing a definition of VVS, we need to first clarify what we mean by the volume of a region in the feature space.

We define volume function $\text{Vol} : \Omega \rightarrow \mathbb{R}$ to be the Lebesgue measure over the set, Ω, of all the measurable subsets in feature space F[18]. For example, if F is the real line and the subset is an interval, the volume function of the subset is just the length of the interval. We assume all the Voronoi Cells considered in our examples to be measurable. We further assume that F is compact in the sense that $\text{Vol}(F)$ is finite. As we are going to normalize all volume measurements by $\text{Vol}(F)$, we simply assume that $\text{Vol}(F) = 1$. To compute the volume of the Similar Voronoi Region $R(X, Y; \epsilon)$ between two video sequences X and Y, we first notice that individual terms inside the union in Equation (4) are disjoint from each other. By the basic properties of Lebesgue measure, we have

$$\text{Vol}(R(X, Y; \epsilon)) = \text{Vol}(\bigcup_{d(x,y) \leq \epsilon} V_X(x) \cap V_Y(y)) = \sum_{d(x,y) \leq \epsilon} \text{Vol}(V_X(x) \cap V_Y(y)).$$

Thus, we define the VVS, $\text{vvs}(X, Y; \epsilon)$, between two video sequences X and Y as,

$$\text{vvs}(X, Y; \epsilon) \stackrel{\Delta}{=} \sum_{d(x,y) \leq \epsilon} \text{Vol}(V_X(x) \cap V_Y(y)) \tag{5}$$

The VVS of the two sequences shown in Figure 1(c) is the area of the shaded region, which is about 1/3 of the area of the entire feature space. Notice that for this example, the IVS is also 1/3. VVS and IVS are close to each other because the Voronoi Cell for each cluster in the cluster union has roughly the same volume (area). In general, when the clusters are not uniformly distributed over the feature space, there can be a large variation among the volumes of the corresponding Voronoi Cells. Consequently, VVS can be quite different from IVS. In the following section, we ignore the differences between VVS and IVS and introduce the Basic ViSig method to estimate either similarity measure.

2.3 Video Signature Method

It is straightforward to estimate $\text{vvs}(X, Y; \epsilon)$ by random sampling: First, generate a set S of m independent uniformly distributed random vectors $s_1, \ldots, s_m$, which we call Seed Vectors (SV). By

uniform distribution, we mean for every measurable subset A in F, the probability of generating a vector from A is $\mathrm{Vol}(A)$. Second, for each random vector $s \in S$, determine if s is inside $R(X, Y; \epsilon)$. By definition, s is inside $R(X, Y; \epsilon)$ if and only if s belongs to some Voronoi Cells $V_X(x)$ and $V_Y(y)$ with $d(x, y) \leq \epsilon$. Since s must be inside the Voronoi Cell of the frame closest to s in the entire video sequence, i.e. $g_X(s)$ in X and $g_Y(s)$ in Y, an equivalent condition for $s \in R(X, Y; \epsilon)$ is $d(g_X(s), g_Y(s)) \leq \epsilon$. Since we only require $g_X(s)$ and $g_Y(s)$ to determine if each SV s belongs to $R(X, Y; \epsilon)$, we can summarize video X by the m-tuple $\overrightarrow{X_S} \triangleq (g_X(s_1), \ldots, g_X(s_m))$ and Y by $\overrightarrow{Y_S}$. We call $\overrightarrow{X_S}$ and $\overrightarrow{Y_S}$ the Video Signature (ViSig) with respect to S of video sequences X and Y respectively. In the final step, we compute the percentage of ViSig frame pairs $g_X(s)$ and $g_Y(s)$ with distances less than or equal to ϵ to obtain:

$$\mathrm{vss}_b(\overrightarrow{X_S}, \overrightarrow{Y_S}; \epsilon, m) \triangleq \frac{\sum_{i=1}^{m} 1_{\{d(g_X(s_i), g_Y(s_i)) \leq \epsilon\}}}{m}. \tag{6}$$

We call $\mathrm{vss}_b(\overrightarrow{X_S}, \overrightarrow{Y_S}; \epsilon, m)$ the Basic ViSig Similarity (VSS_b) between ViSig's $\overrightarrow{X_S}$ and $\overrightarrow{Y_S}$. As every SV $s \in S$ in the above algorithm is chosen to be uniformly distributed, the probability of s being inside $R(X, Y; \epsilon)$ is $\mathrm{Vol}(R(X, Y; \epsilon)) = \mathrm{vvs}(X, Y; \epsilon)$. Thus, $\mathrm{vss}_b(\overrightarrow{X_S}, \overrightarrow{Y_S}; \epsilon, m)$ forms an unbiased estimator of the VVS between X and Y. We refer to this approach of generating ViSig and computing VSS_b the Basic ViSig method. In order to apply the Basic ViSig method to a large number of video sequences, we must use the same SV set S to generate all the ViSig's in order to compute VSS_b between an arbitrary pair of video sequences.

The number of SV's in S, m, is an important parameter. On one hand, m represents the number of samples used to estimate the underlying VVS and thus, the larger m is, the more accurate the estimation becomes. On the other hand, the complexity of the Basic ViSig method directly depends on m. If a video has l frames, it takes l metric computations to generate a single ViSig frame. The number of metric computations required to compute the entire ViSig is thus $m \cdot l$. Also, computing the VSS_b between two ViSig's requires m metric computations. It is, therefore, important to determine an appropriate value of m that can satisfy both the desired fidelity of estimation and the computational resource of a particular application. The following proposition provides an analytical bound on m in terms of the maximum error in estimating the VVS between any pair of video sequences in a database:

Proposition 2.2 *Assume we are given a database Λ with n video sequences and a set S of m random SV's. Define the error probability $P_{err}(m)$ to be the probability that any pair of video sequences in Λ*

has their m-frame VSS_b *different from the true VVS value by more than a given* $\gamma \in (0, 1]$*, i.e.*

$$P_{err}(m) \triangleq \mathrm{Prob}\Big(\bigcup_{X,Y \in \Lambda} \{|\mathrm{vvs}(X, Y; \epsilon) - \mathrm{vss}_b(\overrightarrow{X_S}, \overrightarrow{Y_S}; \epsilon, m)| > \gamma\}\Big) \tag{7}$$

A sufficient condition to achieve $P_{err}(m) \leq \delta$ *for a given* $\delta \in (0, 1]$ *is as follows:*

$$m \quad \geq \quad \frac{2 \ln n - \ln \delta}{2\gamma^2} \tag{8}$$

It should be noted that the bound (8) in Proposition 2.2 only provides a sufficient condition and does not necessarily represent the tightest bound possible. Nonetheless, we can use such a bound to understand the dependencies of m on various factors. First, unlike the random sampling described in Section 2.1, m does not depend on the length of individual video sequences. This implies that it takes far fewer frames for the ViSig method to estimate the similarity between long video sequences than random frame sampling. Second, we notice that the bound on m increases with the natural logarithm of n, the size of the database. The ViSig size depends on n because it has to be large enough to simultaneously minimize the error of all possible pairs of comparisons, which is a function of n. Fortunately, the slow-growing logarithm makes the ViSig size rather insensitive to the database size, making it suitable for very large databases. The contribution of the term $\ln \delta$ is also quite insignificant. Comparatively, m is most sensitive to the choice of γ. A small γ means an accurate approximation of the similarity, but usually at the expense of a large number of sample frames m to represent each video. The choice of γ should depend on the particular application at hand.

2.4 Seed Vector Generation

We have shown in the previous section that the VVS between two video sequences can be efficiently estimated by the Basic ViSig method. Unfortunately, the estimated VVS does not necessarily reflect the target measure of IVS as defined in Equation (2). For example, consider the two pairs of sequences in Figure 2(a) and Figure 2(b). As in Figure 1(c), dots and crosses are frames from the two sequences, whose Voronoi Diagrams are indicated by solid and broken lines respectively. The IVS's in both cases are 1/3. Nonetheless, the VVS in Figure 2(a) is much smaller than 1/3, while that of Figure 2(b) is much larger. Intuitively, as mentioned in Section 2.2, IVS and VVS are the same if clusters in the clustered union are uniformly distributed in the feature space. In the above examples, all the clusters are clumped in one small area of the feature space, making one Voronoi Cell significantly larger than the other. If the Similar Cluster happens to reside in the smaller Voronoi Cells, as in the case of

Figure 2(a), the VVS is smaller than the IVS. On the other hand, if the Similar Cluster is in the larger Voronoi Cell, the VVS becomes larger. This discrepancy between IVS and VVS implies that VSS_b, which is an unbiased estimator of VVS, can only be used as an estimator of IVS when IVS and VVS is close. Our goal in this section and the next is to modify the Basic ViSig method so that we can still use this method to estimate IVS even in the case when VVS and IVS are different.

As the Basic ViSig method estimates IVS based on uniformly-distributed SV's, the variation in the sizes of Voronoi Cells affects the accuracy of the estimation. One possible method to amend the Basic ViSig method is to generate SV's based on a probability distribution such that the probability of a SV being in a Voronoi Cell is independent of the size of the Cell. Specifically, for two video sequences X and Y, we can define the Probability Density Function (PDF) based on the distribution of Voronoi Cells in $[X \cup Y]_\epsilon$ at an arbitrary feature vector u as follows:

$$f(u; X \cup Y) \stackrel{\Delta}{=} \frac{1}{|[X \cup Y]_\epsilon|} \cdot \frac{1}{\text{Vol}(V_{X \cup Y}(C))} \tag{9}$$

where C is the cluster in $[X \cup Y]_\epsilon$ with $u \in V_{X \cup Y}(C)$. $f(u; X \cup Y)$ is constant within the Voronoi Cell of each cluster, with the value inversely proportional to the volume of that Cell. Under this PDF, the probability of a random vector u inside the Voronoi Cell $V_{X \cup Y}(C)$ for an arbitrary cluster $C \in [X \cup Y]_\epsilon$ is given by $\int_{V_{X \cup Y}(C)} f(u; X \cup Y)\, du = 1/|[X \cup Y]_\epsilon|$. This probability does not depend on C, and thus, it is equally likely for u to be inside the Voronoi Cell of any cluster in $[X \cup Y]_\epsilon$.

Recall that if we use uniform distribution to generate random SV's, VSS_b forms an unbiased estimate of the VVS defined in Equation (5). If we use $f(u; X \cup Y)$ to generate SV's instead, VSS_b now becomes an estimate of the following general form of VVS:

$$\sum_{d(x,y) \le \epsilon} \int_{V_X(x) \cap V_Y(y)} f(u; X \cup Y)\, du. \tag{10}$$

Equation (10) reduces to Equation (5) when $f(u; X \cup Y)$ is replaced by the uniform distribution, i.e. $f(u; X \cup Y) = 1$. It turns out, as shown by the following proposition, that this general form of VVS is equivalent to the IVS under certain conditions.

Proposition 2.3 *Assume we are given two video sequences X and Y. Assume clusters in $[X]_\epsilon$ and clusters in $[Y]_\epsilon$ either are identical, or share no frames that are within ϵ from each other. Then, the following relation holds:*

$$\text{ivs}(X, Y; \epsilon) = \sum_{d(x,y) \le \epsilon} \int_{V_X(x) \cap V_Y(y)} f(u; X \cup Y)\, du. \tag{11}$$

The significance of this proposition is that if we can generate SV's with $f(u; X \cup Y)$, it is possible to estimate IVS using VSS_b. The condition that all clusters in X are Y are either identical or far away from each other is to avoid the formation of a special region in the feature space called a Voronoi Gap (VG). The concept of VG is expounded in Section2.5.

In practice, it is impossible to use $f(u; X \cup Y)$ to estimate the IVS between X and Y. This is because $f(u; X \cup Y)$ is specific to the two video sequences being compared, while the Basic ViSig method requires the same set of SV's to be used by all video sequences in the database. A heuristic approach for SV generation is to first select a set Ψ of training video sequences that resemble video sequences in the target database. Denote $T \stackrel{\Delta}{=} \bigcup_{Z \in \Psi} Z$. We can then generate SV based on the PDF $f(u; T)$, which ideally resembles the target $f(u; X \cup Y)$ for an arbitrary pair of X and Y in the database.

To generate a random SV s based on $f(u; T)$, we follow a four-step algorithm, called the SV Generation method, as follows:

1. Given a particular value of ϵ_{sv}, identify all the clusters in $[T]_{\epsilon_{sv}}$ using the single-link algorithm.

2. As $f(u; T)$ assigns equal probability to the Voronoi Cell of each cluster in $[T]_{\epsilon_{sv}}$, randomly select a cluster C' from $[T]_{\epsilon_{sv}}$ so that we can generate the SV s within $V_T(C')$.

3. As $f(u; T)$ is constant over $V_T(C')$, we should ideally generate s as a uniformly-distributed random vector over $V_T(C')$. Unless $V_T(C')$ can be easily parameterized, the only way to achieve this is to repeatedly generate uniform sample vectors over the entire feature space until a vector is found inside $V_T(C')$. This procedure may take an exceedingly long time if $V_T(C')$ is small. To simplify the generation, we select one of the frames in C' at random and output it as the next SV s.

4. Repeat the above process until the required number of SV's has been selected.

In Section 3, we compare performance of this algorithm against uniformly distributed SV generation in retrieving real video sequences.

2.5 Voronoi Gap

We show in Proposition 2.3 that the general form of VVS using an appropriate PDF is identical to the IVS, provided that all clusters between the two sequences are either identical or far away from each other. As feature vectors are not perfect in modeling human visual system, visually similar

clusters may result in feature vectors that are close but not identical to each other. Let us consider the example in Figure 2(c) where frames in Similar Clusters between two video sequences are not identical but within ϵ from each other. Clearly, the IVS is one. Consider the Voronoi Diagrams of the two sequences. Since the boundaries of the two Voronoi Diagrams do not exactly coincide with each other, the Similar Voronoi Region, as indicated by the shaded area, does not occupy the entire feature space. As the general form of VVS defined in Equation (10) is the weighted volume of the Similar Voronoi Region, it is strictly less than the IVS. The difference between the two similarities is due to the unshaded region in Figure 2(c). The larger the unshaded region is, the larger the difference between the two similarities. If a SV s falls within the unshaded region in Figure 2(c), we can make two observations about the corresponding ViSig frames $g_X(s)$ and $g_Y(s)$ from the two sequences X and Y: (1) they are far apart from each other, i.e. $d(g_X(s), g_Y(s)) > \epsilon$; (2) they both have similar frames in the other video, i.e. there exists $x \in X$ and $y \in Y$ such that $d(x, g_Y(s)) \leq \epsilon$ and $d(y, g_X(s)) \leq \epsilon$. These observations define a unique characteristic of a particular region, which we refer to as Voronoi Gap (VG). Intuitively, any SV in VG between two sequences produces a pair of dissimilar ViSig frames, even though both ViSig frames have a similar match in the other video. More formally, we define VG as follows:

Definition 2.3 Voronoi Gap (VG)
Let X and Y be two video sequences. The VG $G(X, Y; \epsilon)$ of X and Y is defined by all $s \in F$ that satisfy the following criteria:

1. $d(g_X(s), g_Y(s)) > \epsilon$,
2. *there exists* $x \in X$ *such that* $d(x, g_Y(s)) \leq \epsilon$,
3. *there exists* $y \in Y$ *such that* $d(y, g_X(s)) \leq \epsilon$.

In section 2.6, we show that the volume of VG is non-trivial by computing its volume for a particular feature space, namely the Hamming Cube. This implies that it is quite possible to find some of the SV's used in the Basic ViSig method to fall inside the VG. As a result, the performance of the Basic ViSig method in estimating IVS may be adversely affected. In section 2.7, we introduce the Ranked ViSig method, which mitigates this problem by avoiding SV's that are likely to be inside a VG.

2.6 Voronoi Gap in Hamming Cube

The example in Figure 2(c) seems to suggest that VG is small if ϵ is small. An important question is how small ϵ should be before we can ignore the contribution of the VG. It is obvious that the precise

value of the volume of a VG depends on the frame distributions of the two video sequences, and the geometry of the feature space. It is, in general, difficult to compute even a bound on this volume without assuming a particular feature space geometry and frame distributions. In order to get a rough idea of how large VG is, we compute a simple example using a h-dimensional hamming cube. A hamming cube is the set containing all the h-bit binary numbers. The distance between two vectors is simply the number of bit-flips to change the binary representation of one vector to the other. Since it is a finite space, the volume function is simply the cardinality of the subset divided by 2^h. We choose the hamming cube because it is easy to analyze, and some commonly used metrics such as l_1 and l_2 can be embedded inside the hamming cube with low distortion [19].

To simplify the calculations, we only consider two-frame video sequences in the h-dimensional hamming cube H. Let $X = \{x_1, x_2\}$ be a video in H. Let the distance between x_1 and x_2 be a positive integer k. We assume the two frames in X are not similar, i.e. the distance between them is much larger than ϵ. In particular, we assume that $k > 2\epsilon$. We want to compute the "gap volume", i.e. the probability of choosing a SV s that is inside the VG formed between X and some video sequence in H. Based on the definition of VG, if a two-frame video sequence Y has a non-empty VG with X, Y must have a frame similar to each frame in X. In other words, the IVS between X and Y must be one. Let Γ be the set of all two-frame sequences whose IVS with X is one. The gap volume is thus the volume of the union of the VG formed between X and each video in Γ. As shown by the following proposition, this gap probability can be calculated using the binomial distribution.

Proposition 2.4 *Let $X = \{x_1, x_2\}$ be a two-frame video in the Hamming cube H, and Γ be the set of all two-frame sequences whose IVS with X is one. Define A to be the union of the VG formed between X and every video in Γ, i.e.*

$$A \stackrel{\Delta}{=} \bigcup_{Y \in \Gamma} G(X, Y; \epsilon).$$

Then, if $k = d(x_1, x_2)$ is an even number larger than 2ϵ, the volume of A can be computed as follows:

$$\begin{aligned} \mathrm{Vol}(A) &= \mathrm{Prob}(k/2 - \epsilon \leq R < k/2 + \epsilon) \\ &= \frac{1}{2^k} \sum_{r=k/2-\epsilon}^{k/2+\epsilon-1} \binom{k}{r} \end{aligned} \tag{12}$$

where R is a random variable that follows a binomial distribution with parameters k and 1/2.

We compute $\mathrm{Vol}(A)$ numerically by using the right hand side of Equation (12). The resulting plot of $\mathrm{Vol}(A)$ versus the distance k between the frames in X for $\epsilon = 1, 5, 10$ is shown in Figure 3(a). $\mathrm{Vol}(A)$

decreases as k increases and as ϵ decreases, but it is hardly insignificant even when k is substantially larger than ϵ. For example, at $k = 500$ and $\epsilon = 5$, $\text{Vol}(A) \approx 0.34$. It is unclear whether the same phenomenon occurs for other feature spaces. Nonetheless, rather than assuming that all VG's are insignificant and using any random SV, intuitively it makes sense to identify those SV's that are inside the VG, and discard them when we estimate the corresponding video similarity.

2.7 Ranked ViSig Method

Consider again the example in Figure 2(c). Assume that we generate m random SV's to compute VSS_b. If n out of m SV's are inside the unshaded VG, we can reject these n SV's and use the remaining $(m - n)$ SV's for the computation. The resulting VSS_b becomes one which exactly matches the IVS in this example. The only caveat in this approach is that we need an efficient algorithm to determine whether a SV is inside the VG. Direct application of Definition 2.3 is not very practical, because conditions (2) and (3) in the definition require computing the distances between a ViSig frame of one video and all the frames in the other video. Not only does the time complexity of comparing two ViSig's is significantly larger than the VSS_b, it defeats the very purpose of using a compact ViSig to represent a video. A more efficient algorithm is thus needed to identify if a SV is inside the VG.

In this section, we propose an algorithm, applied after generating the ViSig, that can identify those SV's which are more likely to be inside the VG. In Figure 2(c), we observe that the two sequences have a pair of dissimilar frames that are roughly equidistant from an arbitrary vector s in the VG: x and $g_X(s)$ in the "dot" sequence, and y and $g_Y(s)$ in the "cross" sequence. They are not similar as both $d(x, g_X(s))$ and $d(y, g_Y(s))$ are clearly larger than ϵ. Intuitively, since vectors such as s inside the VG are close to the boundaries between Voronoi Cells in both sequences, it is not surprising to find dissimilar frames such as x and $g_X(s)$ that are on either side of the boundaries to be roughly equidistant to s. This "equidistant" condition is refined in the following proposition to upper-bound the difference between distance of s and x, and distance of s and $g_X(s)$ by 2ϵ:

Proposition 2.5 *Let X and Y be two video sequences. Assume all clusters in $[X \cup Y]_\epsilon$ are ϵ-compact. If a SV $s \in G(X, Y; \epsilon)$, there exists a frame $x \in X$ such that*

1. *x is not similar to $g_X(s)$, the ViSig frame in X with respect to s, i.e. $d(x, g_X(s)) > \epsilon$.*
2. *x and $g_X(s)$ are roughly equidistant to s. Specifically, $d(x, s) - d(g_X(s), s) \leq 2\epsilon$.*

Similarly, we can find a $y \in Y$ that share the same properties with $g_Y(s)$.

The significance of Proposition 2.5 is that it provides a test for determining whether a SV s can ever be inside the VG between a particular video X and any other arbitrary sequence. Specifically, if there are no frames x in X such that x is dissimilar to $g_X(s)$ and $d(x,s)$ is within 2ϵ from $d(s, g_X(s))$, we can guarantee that s will never be inside the VG formed between X and any other sequence. The condition that all Similar Clusters must be ϵ-compact is to avoid pathological chain-like clusters as discussed in Section 2.1. Such an assumption is not unrealistic for real-life video sequences.

To apply Proposition 2.5 in practice, we first define a Ranking Function $Q(\cdot)$ for the ViSig frame $g_X(s)$,

$$Q(g_X(s)) \stackrel{\Delta}{=} \min_{x \in X,\, d(x, g_X(s)) > \epsilon} d(x,s) - d(g_X(s), s). \tag{13}$$

An example of $Q()$ as a function of a 2-D SV s is shown as a contour plot in Figure 3(b). The three crosses represent the frames of a video. Lighter color regions correspond to the area with larger $Q()$ values, and thus farther away from the boundaries between Voronoi Cells. By Proposition 2.5, if $Q(g_X(s)) > 2\epsilon$, s cannot be inside the VG formed between X and any other sequence. In practice, however, this condition might be too restrictive in that it might not allow us to find any SV. Recall that Proposition 2.5 only provides a sufficient and not a necessary condition for a SV to be inside VG. Thus, even if $Q(g_X(s)) \leq 2\epsilon$, it does not necessarily imply that s will be inside the VG between X and any particular sequence.

Intuitively, in order to minimize the chances of being inside any VG, it makes sense to use a SV s with as large of a $Q(g_X(s))$ value as possible. As a result, rather than using only the ViSig frames with $Q(g_X(s)) > 2\epsilon$, we generate a large number of ViSig frames for each ViSig, and use the few ViSig frames with the largest $Q(g_X(s))$ for similarity measurements. Let $m' > m$ be the number of frames in each ViSig. After we generate the ViSig $\overrightarrow{X_S}$ by using a set S of m' SV's, we compute and rank $Q(g_X(s))$ for all $g_X(s)$ in $\overrightarrow{X_S}$. Analogous to VSS_b defined in Equation (6), we define the Ranked ViSig Similarity (VSS_r) between two ViSig's $\overrightarrow{X_S}$ and $\overrightarrow{Y_S}$ based on their top-ranked ViSig frames:

$$\begin{aligned} \text{vss}_r(\overrightarrow{X_S}, \overrightarrow{Y_S}; \epsilon, m) \;\; \stackrel{\Delta}{=} \;\; & \frac{1}{m}\Big(\sum_{i=1}^{m/2} 1_{\{d(g_X(s_{j[i]}), g_Y(s_{j[i]})) \leq \epsilon\}} + \\ & \sum_{i=1}^{m/2} 1_{\{d(g_X(s_{k[i]}), g_Y(s_{k[i]})) \leq \epsilon\}}\Big) \end{aligned} \tag{14}$$

$j[1], \ldots, j[m']$ and $k[1], \ldots, k[m']$'s denote the rankings of the ViSig frames in $\overrightarrow{X_S}$ and $\overrightarrow{Y_S}$ respectively, i.e. $Q(g_X(s_{j[1]})) \geq \ldots \geq Q(g_X(s_{j[m']}))$ and $Q(g_Y(s_{k[1]})) \geq \ldots \geq Q(g_Y(s_{k[m']}))$. We call this method of generating ViSig and computing VSS_r the Ranked ViSig method. Notice that the right hand side

of Equation (14), the first term uses the top-ranked $m/2$ ViSig frames from $\overrightarrow{X_S}$ to compare with the corresponding ViSig frames in $\overrightarrow{Y_S}$, and the second term uses the top-ranked $m/2$ frames from $\overrightarrow{Y_S}$. Computing VSS_r thus requires m metric computations, the same as computing VSS_b. This provides an equal footing in complexity to compare the retrieval performances between these two methods in Section 3.

3 Experimental Results

In this section, we present experimental results to demonstrate the performance of the ViSig method. All experiments use color histograms as video frame features described in more detail in Section 3.1. Two sets of experiments are performed. Results of a number of controlled simulations are presented in Section 3.2 to demonstrate the heuristics proposed in Section 2. In Section 3.3, we apply the ViSig method to a large set of real-life web video sequences.

3.1 Image Feature

In our experiments, we use four 178-bin color histograms on the Hue-Saturation-Value (HSV) color space to represent each individual frame in a video. Color histogram is one of the most commonly used image features in content-based retrieval system. The quantization of the color space used in the histogram is shown in Figure 4. This quantization is similar to the one used in [20]. The saturation (radial) dimension is uniformly quantized into 3.5 bins, with the half bin at the origin. The hue (angular) dimension is uniformly quantized at 20°-step size, resulting in 18 sectors. The quantization for the value dimension depends on the saturation value. For those colors with the saturation values near zero, a finer quantizer of 16 bins is used to better differentiate between gray-scale colors. For the rest of the color space, the value dimension is uniformly quantized into three bins. The histogram is normalized such that the sum of all the bins equals one. In order to incorporate spatial information into the image feature, the image is partitioned into four quadrants, with each quadrant having its own color histogram. As a result, the total dimension of a single feature vector becomes 712.

We use two distance measurements in comparing color histograms: the l_1 metric and a modified version of the l_1 distance with dominant color first removed. The l_1 metric on color histogram was first used in [21] for image retrieval. It is defined by the sum of the absolute difference between each bin of the two histograms. We denote the l_1 metric between two feature vectors x and y as $d(x, y)$,

with its precise definition stated below:

$$d(x,y) \triangleq \sum_{i=1}^{4} d_q(x_i, y_i) \text{ where } d_q(x_i, y_i) \triangleq \sum_{j=1}^{178} |x_i[j] - y_i[j]| \tag{15}$$

where x_i and y_i for $i \in \{1,2,3,4\}$ represent the quadrant color histograms from the two image feature vectors. A small $d()$ value usually indicates visual similarity, except when two images share the same background color. In those cases, the metric $d()$ does not correctly reflect the differences in the foreground as it is overwhelmed by the dominant background color. Such scenarios are quite common among the videos found on the web. Examples include those video sequences composed of presentation slides or graphical plots in scientific experiments. To mitigate this problem, we develop a new distance measurement which first removes the dominant color, then computes the l_1 metric for the rest of the color bins, and finally re-normalizes the result to the proper dynamic range. Specifically, this new distance measurement $d'(x,y)$ between two feature vectors x and y can be defined as follows:

$$d'(x,y) \triangleq \sum_{i=1}^{4} d'_q(x_i, y_i)$$

$$\text{where } d'_q(x_i, y_i) \triangleq \begin{cases} \frac{2}{2 - x_i[c] - y_i[c]} \sum_{j=1, j\neq c}^{178} |x_i[j] - y_i[j]| & \text{if } x_i[c] > \rho \text{ and } y_i[c] > \rho \\ \sum_{j=1}^{178} |x_i[j] - y_i[j]| & \text{otherwise.} \end{cases} \tag{16}$$

In Equation (16), the dominant color is defined to be the color c with bin value exceeding the Dominant Color Threshold ρ. ρ has to be larger than or equal to 0.5 to guarantee a single dominant color. We set $\rho = 0.5$ in our experiments. When the two feature vectors share no common dominant color, $d'()$ reduces to $d()$.

Notice that the modified l_1 distance $d'()$ is not a metric. Specifically, it does not satisfy the triangle inequality. Thus, it cannot be directly applied to measuring video similarity. The l_1 metric $d()$ is used in most of the experiments described in Sections 3.2 and 3.3. We only use $d'()$ as a post-processing step to improve the retrieval performance in Section 3.3. The process is as follows: first, given a query video X, we declare a video Y in a large database to be similar to X if the VSS, either the Basic or Ranked version, between the corresponding ViSig $\overrightarrow{X_S}$ and $\overrightarrow{Y_S}$ exceeds a certain threshold λ. The l_1 metric $d()$ is first used in computing all the VSS's. Denote the set of all similar video sequences Y in the database as $\text{ret}(X, \epsilon)$. Due to the limitation of l_1 metric, it is possible that some of the video sequences in Π may share the same background as the query video but are visually very different. In

the second step, we compute the $d'()$-based VSS between $\overrightarrow{X_S}$ and all the ViSig's in $\text{ret}(X, \epsilon)$. Only those ViSig's whose VSS with $\overrightarrow{X_S}$ larger than λ are retrained in $\text{ret}(X, \epsilon)$, and returned to the user as the final retrieval results.

3.2 Simulation Experiments

In this section, we present experimental results to verify the heuristics proposed in Section 2. In the first experiment, we demonstrate the effect of the choice of SV's on approximating the IVS by the ViSig method. We perform the experiment on a set of 15 video sequences selected from the MPEG-7 video data set [22][3]. This set includes a wide variety of video content including documentaries, cartoons, and television drama, etc. The average length of the test sequences is 30 minutes. We randomly drop frames from each sequence to artificially create similar versions at different levels of IVS. ViSig's with respect to two different sets of SV's are created for all the sequences and their similar versions. The first set of SV's are independent random vectors, uniformly distributed on the high-dimensional histogram space. To generate such random vectors, we follow the algorithm described in [23]. The second set of SV's are randomly selected from a set of images in the Corel Stock Photo Collection. These images represent a diverse set of real-life images, and thus provide a reasonably good approximation to the feature vector distribution of the test sequences. We randomly choose around 4000 images from the Corel collection, and generate the required SV's using the SV Generation Method, with ϵ_{sv} set to 2.0, as described in Section 2.4. Table 1 shows the VSS_b with $m = 100$ SV's per ViSig at IVS levels of 0.8, 0.6, 0.4 and 0.2. VSS_b based on Corel images are closer to the underlying IVS than those based on random vectors. In addition, the fluctuations in the estimates, as indicated by the standard deviations, are far smaller with the Corel images. The experiment thus shows that it is advantageous to use SV's that approximate the feature vector distribution of the target data.

In the second experiment, we compare the Basic ViSig method with the Ranked ViSig method in identifying sequences with IVS one under the presence of small feature vector displacements. As described in Section 2.5, when two frames from two video sequences are separated by a small ϵ, the Basic ViSig method may underestimate the IVS because of the presence of VG. To combat such a problem, we propose the Ranked ViSig method in Section 2.5. In this experiment, we create similar video by adding noise to individual frames. Most of the real-life noise processes such as compression are highly video dependent, and cannot provide a wide-range of controlled noise levels for our experiment.

[3] The test set includes video sequences from MPEG-7 video CD's v1, v3, v4, v5, v6, v7, v8, and v9. We denote each test sequence by the CD they are in, followed by a number such as v8_1 if there are multiple sequences in the same CD.

As such, we introduce artificial noise that directly corresponds to the different noise levels as measured by our frame metric. As shown in [21], the l_1 metric used in histograms is equal to twice the percentage of the pixels between two images that are of different colors. For example, if the l_1 metric between two histograms is 0.4, it implies that 20% of the pixels in the images have different color. Thus, to inject a particular ϵ noise level to a frame, we determine the fraction of the pixels that need to have different colors and randomly assign colors to them. The color assignment is performed in such a way that ϵ noise level is achieved exactly. Five ϵ levels are tested in our experiments: 0.2, 0.4, 0.8, 1.2 and 1.6. In our definition of a feature vector, four histograms are used per frame. This means that an ϵ of, for example, 1.6 results in an average noise level of 0.4 for each histogram. After injecting noise to create the similar video, a 100-frame Basic ViSig ($m = 100$) and a 500-frame Ranked ViSig ($m' = 500$) are generated for each video. All SV's are randomly sampled from the Corel dataset. To ensure the same computational complexity between the two methods, the top $m/2 = 50$ Ranked ViSig frames are used in computing VSS_r. The results are shown in Table 2. The averages and standard deviations over the entire set are shown in the last two rows. Since the IVS is fixed at one, the closer the measured similarity is to one, the better the approximation is. Even though both methods show a general trend of increasing error as the noise level increase, as expected, VSS_r measurements are much closer to one than VSS_b.

3.3 Web Video Experiments

To further demonstrate how the ViSig method can be applied to a realistic application, we test our algorithms on a large dataset of web video and measure their retrieval performance using a ground-truth set. Its retrieval performance is further compared with another summarization technique called the k-medoid. k-medoid was first proposed in [15] as a clustering method used in general metric spaces. One version of k-medoid was used by Chang et al. for extracting a predefined number of representative frames from a video [8]. Given a l-frame video $X = \{x_t : t = 1, \ldots, l\}$, the k-medoid of X is defined to be a set of k frames $x_{t_1}, \ldots, x_{t_k}$ in X that minimize the following cost function:

$$\sum_{t=1}^{l} \min_{j=1,\ldots,k} d(x_t, x_{t_j}) \tag{17}$$

Due to the large number of possible choices in selecting k frames from a set, it is computationally impractical to precisely solve this minimization problem. In our experiments, we use the PAM algorithm proposed in [15] to compute an approximation to the k-medoid. This is an iterative algorithm and the time complexity for each iteration is on the order of l^2. After computing the k-medoid for each

video, we declare two video sequences to be similar if the minimum distance between frames in their corresponding k-medoids is less than or equal to ϵ.

The dataset for the experiments is a collection of 46,356 video sequences, crawled from the web between August and December, 1999. The URL addresses of these video sequences are obtained by sending dictionary entries as text queries to the AltaVista video search engine [24]. Details about our data collection process can be found in [25]. The statistics of the four most abundant formats of video collected are shown in Table 3. The ground-truth is a set of manually identified clusters of almost-identical video sequences. We adopt a best-effort approach to obtain such a ground-truth. This approach is similar to the pooling method [26] commonly used in text retrieval systems. The basic idea of pooling is to send the same queries to different automatic retrieval systems, whose top-ranked results are pooled together and examined by human experts to identify the truly relevant ones. For our system, the first step is to use meta-data terms to identify the initial ground-truth clusters. Meta-data terms are extracted from the URL addresses and other auxiliary information for each video in the dataset [27]. All video sequences containing at least one of the top 1000 most frequently used meta-data terms are manually examined and grouped into clusters of similar video. Clusters which are significantly larger than others are removed to prevent bias. We obtain 107 clusters which form the initial ground-truth clusters. This method, however, may not be able to identify all the video clips in the dataset that are similar to those already in the ground-truth clusters. We further examine those video sequences in the dataset that share at least one meta-data term with the ground-truth video, and add any similar video to the corresponding clusters. In addition to meta-data, k-medoid is also used as an alternative visual similarity scheme to enlarge the ground-truth. A 7-frame k-medoid is generated for each video. For each video X in the ground-truth, we identify 100 video sequences in the dataset that are closest to X in terms of the minimum distance between their k-medoids, and manually examine them to search for any sequence that is visually similar to X. As a result, we obtain a ground-truth set consisting of 443 video sequences in 107 clusters. The cluster size ranges from 2 to 20, with average size equal to 4.1. The ground-truth clusters serve as the subjective truth for comparison against those video sequences identified as similar by the ViSig method.

When using the ViSig method to identify similar video sequences, we declare two sequences to be similar if their VSS_b or VSS_r is larger than a certain threshold $\lambda \in [0, 1]$. In the experiments, we fix λ at 0.5 and report the retrieval results for different numbers of ViSig frames, m, and the Frame Similarity Threshold, ϵ. Our choice of fixing λ at 0.5 is based on the following reason: as the dataset

consists of extremely heterogeneous contents, it is rare to find partially similar video sequences. We notice that most video sequences in our dataset are either very similar to each other, or not similar at all. If ϵ is appropriately chosen to match subjective similarity, and m is large enough to keep sampling error small, we would expect the VSS for an arbitrary pair of ViSig's to be close to either one or zero, corresponding to either similar or dissimilar video sequences in the dataset. We thus fix λ at 0.5 to balance the possible false-positive and false-negative errors, and vary ϵ to trace the whole spectrum of retrieval performance. To accommodate such a testing strategy, we make a minor modification in the Ranked ViSig method: recall that we use the Ranking Function $Q()$ as defined in Equation (13) to rank all frames in a ViSig. Since $Q()$ depends on ϵ and its computation requires the entire video sequence, it is cumbersome to recompute it whenever a different ϵ is used. ϵ is used in the $Q()$ function to identify the clustering structure within a single video. Since most video sequences are compactly clustered, we notice that their $Q()$ values remain roughly constant for a large range of ϵ. As a result, we a priori fix ϵ to be 2.0 to compute $Q()$, and do not recompute them even when we modify ϵ to obtain different retrieval results.

The performance measurements used in our experiments are recall and precision as defined below. Let Λ be the web video dataset and Φ be the ground-truth set. For a video $X \in \Phi$, we define the Relevant Set to X, $\text{rel}(X)$, to be the ground-truth cluster that contains X, minus X itself. Also recall the definition of the Return Set to X, $\text{ret}(X, \epsilon)$, from Section 3.1, as the set of video sequences in the database which are declared to be similar to X by the ViSig method, i.e. $\text{ret}(X, \epsilon) \stackrel{\Delta}{=} \{Y \in \Lambda : \text{vss}(\overrightarrow{X_S}, \overrightarrow{Y_S}) \leq \lambda\} \setminus \{X\}$. $\text{vss}(\cdot)$ can be either VSS_b or VSS_r. By comparing the Return and Relevant Sets of the entire ground-truth, we can define the recall and precision as follows:

$$\text{Recall}(\epsilon) \stackrel{\Delta}{=} \frac{\sum_{X \in \Phi} |\text{rel}(X) \cap \text{ret}(X, \epsilon)|}{\sum_{X \in \Phi} |\text{rel}(X)|} \text{ and } \text{Precision}(\epsilon) \stackrel{\Delta}{=} \frac{\sum_{X \in \Phi} |\text{rel}(X) \cap \text{ret}(X, \epsilon)|}{\sum_{X \in \Phi} |\text{ret}(X, \epsilon)|}.$$

Thus, recall computes the fraction of all ground-truth video sequences that can be retrieved by the algorithm. Precision measures the fraction retrieved by the algorithm that are relevant. By varying ϵ, we can measure the retrieval performance of the ViSig methods for a wide range of recall values.

The goal of the first experiment is to compare the retrieval performance between the Basic and the Ranked ViSig methods at different ViSig sizes. The modified l_1 distance on the color histogram is used in this experiment. SV's are randomly selected by the SV Generation Method, with ϵ_{sv} set to 2.0, from a set of keyframes representing the video sequences in the dataset. These keyframes are extracted by the AltaVista search engine and captured during data collection process; each video is

represented by a single keyframe. For the Ranked ViSig method, $m' = 100$ keyframes are randomly selected from the keyframe set to produce the SV set which is used for all ViSig sizes, m. For each ViSig size in the Basic ViSig method, we average the results of four independent sets of randomly selected keyframes in order to smooth out the statistical variation due to the limited ViSig sizes. The plots in Figure 5(a) show the precision versus recall curves for four different ViSig sizes: $m = 2$, 6, 10, and 14. The Ranked ViSig method outperforms the Basic ViSig method in all four cases. Figure 5(b) shows the Ranked method's results across different ViSig sizes. There is a substantial gain in performance when the ViSig size is increased from two to six. Further increase in ViSig size does not produce much significant gain. The precision-recall curves all decline sharply once they reach beyond 75% for recall and 90% for precision. Thus, six frames per ViSig is adequate in retrieving ground-truth from the dataset.

In the second experiment, we test the difference between using the modified l_1 distance and the l_1 metric on the color histogram. The same Ranked ViSig method with six ViSig frames is used. Figure 5(c) shows that the modified l_1 distance significantly outperforms the straightforward l_1 metric. Finally, we compare the retrieval performance between k-medoid and the Ranked ViSig method. Each video in the database is represented by seven medoids generated by the PAM algorithm. We plot the precision-recall curves for k-medoid and the six-frame Ranked ViSig method in Figure 5(d). The k-medoid technique provides a slightly better retrieval performance. The advantage seems to be small considering the complexity advantage of the ViSig method over the PAM algorithm – First, computing VSS_r needs six metric computations but comparing two 7-medoid representations requires 49. Second, the ViSig method generates ViSig's in $O(l)$ time with l being the number of frames in a video, while the PAM algorithm is an iterative $O(l^2)$ algorithm.

4 Concluding Remarks

In this paper, we have proposed a class of techniques named ViSig which are based on summarizing a video sequence by extracting the frames closest to a set of randomly selected SV's. By comparing the ViSig frames between two video sequences, we have obtained an unbiased estimate of their VVS. In applying the ViSig method to a large database, we have shown that the size of a ViSig depends on the desired fidelity of the measurements and the logarithm of the database size. In order to reconcile the difference between VVS and IVS, the SV's used must resemble the frame statistics of video in the target database. In addition, ViSig frames whose SV's are inside the VG should be avoided when

comparing two ViSig's. We have proposed a ranking method to identify those SV's that are least likely to be inside the VG. By experimenting with a set of MPEG-7 test sequences and their artificially-generated similar versions, we have demonstrated that IVS can be better approximated by using (a) SV's based on real images than uniformly random generation, and (b) the ranking method than the basic method. We have further characterized the retrieval performance of different ViSig methods based on a groundtruth set from a large set of web video.

The basic premise of our work is on the importance of IVS as a similarity measurement. IVS defines a general similarity measurement between two sets of objects endowed with a metric function. By using the ViSig method, we have demonstrated one particular application of IVS, which is to identify highly similar video sequences found on the web. As such, we are currently investigating the use of IVS on other types of pattern matching and retrieval problems. We have also considered other aspects of the ViSig method in conjunction with its use on large databases. In our recent work [28], we have proposed a novel dimension reduction technique on signature data for fast similarity search, and a clustering algorithm on a database of signatures for improving retrieval performance.

References

[1] Inktomi Corp., "Inktomi webmap," http://www2.inktomi.com/webmap, January 2000.

[2] N. Shivakumar and H. Garcia-Molina, "Finding near-replicas of documents on the web," in *World Wide Web and Databases. International Workshop WebDB'98*, Valencia, Spain, Mar. 1998, pp. 204–12.

[3] C. Silverstein, M. Henzinger, J. Marais, and Michael Moricz, "Analysis of a very large altavista query log," Tech. Rep. SRC-Technical Note 1998-014, Compaq Systems Research Center, 1998.

[4] A.Z. Broder, S.C. Glassman, M.S. Manasse, and G. Zweig, "Syntactic clustering of the web," in *Sixth International World Wide Web Conference*, Sept. 1997, vol. 29, no.8-13 of *Computer Networks and ISDN Systems*, pp. 1157–66.

[5] M.R. Naphade, R. Wang, and T.S. Huang, "Multimodal pattern matching for audio-visual query and retrieval," in *Proceedings of the Storage and Retrieval for Media Datbases 2001*, San Jose, USA, jan 2001, vol. 4315, pp. 188–195.

[6] D. Adjeroh, I. King, and M.C. Lee, "A distance measure for video sequence similarity matching," in *Proceedings International Workshop on Multi-Media Database Management Systems*, Dayton, OH, USA, Aug. 1998, pp. 72–9.

[7] R. Lienhart, W. Effelsberg, and R. Jain, "VisualGREP: A systematic method to compare and retrieve video sequences," in *Proceedings of storage and retrieval for image and video databases VI*. SPIE, Jan. 1998, vol. 3312, pp. 271–82.

[8] H.S. Chang, S. Sull, and S.U. Lee, "Efficient video indexing scheme for content-based retrieval," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 8, pp. 1269–79, Dec 1999.

[9] P. Indyk, G. Iyengar, and N. Shivakumar, "Finding pirated video sequences on the internet," Tech. Rep., Stanford Infolab, Feb. 1999.

[10] S. Santini and R. Jain, "Similarity measures," *IEEE Tran. on Pattern Analysis and Machine Intelligence*, vol. 21, no. 9, pp. 871–83, Sept 1999.

[11] H. Greenspan, J. Goldberger, and A. Mayer, "A probabilistic framework for spatio-temporal video representation and indexing," in *Proc. of the 7th European Conference on Computer Vision, Part IV*, 2002, pp. 461–75.

[12] G. Iyengar and A.B. Lippman, "Distributional clustering for efficient content-based retrieval of images and video," in *Proceedings 1998 International Conference on Image Processing*, Vancouver, B.C., Canada, 2000, vol. III, pp. 81–4.

[13] N. Vasconcelos, "On the complexity of probabilistic image retrieval," in *Proceedings Eighth IEEE International Conference on Computer Vision*, Vancouver, B.C., Canada, 2001, vol. 2, pp. 400–407.

[14] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *5th Berkeley Symposium on Mathematical Statistics*, 1967, vol. 1, pp. 281–97.

[15] L. Kaufman and P.J. Rousseeuw, *Finding Groups in Data*, John Wiley & Sons, New York, 1990.

[16] T. Cormen, C. Leiserson, and R. Riverst, *Introduction to Algorithms*, The MIT Press, Cambridge, Massachusetts, 2nd edition, 2001.

[17] R. Sibson, "Slink: An optimally efficient algorithm for the single-link cluster method," *The Computer Journal*, vol. 16, no. 1, pp. 30–4, 1973.

[18] H.L. Royden, *Real Analysis*, Macmillan Publishing Company, 1988.

[19] P. Indyk, *High-dimensional computational geometry*, Ph.D. thesis, Stanford University, 2000.

[20] J.R. Smith, *Integrated Spatial and Feature Image Systems: Retrieval, Analysis and Compression*, Ph.D. thesis, Columbia University, 1997.

[21] M.J. Swain and D.H. Ballard, "Color indexing," *International Journal of Computer Vision*, vol. 7, no. 1, pp. 11–32, November 1991.

[22] MPEG-7 Requirements Group, "Description of mpeg-7 content set," Tech. Rep. N2467, ISO/IEC JTC1/SC29/WG11, 1998.

[23] A. Woronow, "Generating random numbers on a simplex," *Computers and Geosciences*, vol. 19, no. 1, pp. 81–88, 1993.

[24] AltaVista, http://www.altavista.com, *AltaVista Image, Audio and Video search.*

[25] S.-C. Cheung and A. Zakhor, "Estimation of web video multiplicity," in *Proceedings of the SPIE – Internet Imaging*, San Jose, California, Jan. 2000, vol. 3964, pp. 34–6.

[26] K. Sparck Jones and C. van Rijsbergen, "Report on the need for and provision of an "ideal" information retrieval test collection," Tech. Rep. British Library Research and Development Report 5266, Computer Laboratory, University of Cambridge, 1975.

[27] S.-C. Cheung and A. Zakhor, "Efficient video similarity measurement and search," in *Proceedings of 7th IEEE International Conference on Image Processing*, Vancouver, British Columbia, Sept. 2000, vol. 1, pp. 85–88.

[28] S.-C. Cheung and A. Zakhor, "Towards building a similar video search engine for the world-wide-web," *Submitted to IEEE Transactions on Multimedia*, 2002.

[29] G.R. Grimmett and D.R. Stirzaker, *Probability and Random Processes*, Oxford Science Publications, 1992.

Appendices

A Proof of Proposition 2.1

Without loss of generality, let $X = \{x_1, x_2, \ldots, x_l\}$ and $Y = \{y_1, y_2, \ldots, y_l\}$ with $d(x_i, y_i) \leq \epsilon$ for $i = 1, \ldots, l$. Let Z_i be a binary random variable such that $Z_i = 1$ if both x_i and y_i are chosen as sampled frames, and 0 otherwise. Since W_m is the total number of similar pairs between the two set of sampled frames, it can be computed by summing all the Z_i's:

$$\begin{aligned} W_m &= \sum_{i=1}^{l} Z_i \\ E(W_m) &= \sum_{i=1}^{l} E(Z_i) = \sum_{i=1}^{l} \text{Prob}(Z_i = 1) \end{aligned}$$

Since we independently sample m frames from each sequence, the probability that $Z_i = 1$ for any i is $(m/l)^2$. This implies that $E(W_m) = m^2/l$.

B Proof of Proposition 2.2

To simplify the notation, let $\rho(X,Y) = \text{vvs}(X,Y;\epsilon)$ and $\hat{\rho}(X,Y) = \text{vss}_b(\overrightarrow{X_S},\overrightarrow{Y_S};\epsilon,m)$. For an arbitrary pair of X and Y, we can bound the probability of the event $|\rho(X,Y) - \hat{\rho}(X,Y)| > \gamma$ by the Hoeffding Inequality [29]:

$$\text{Prob}(|\rho(X,Y) - \hat{\rho}(X,Y)| > \gamma) \quad \leq \quad 2\exp(-2\gamma^2 m) \tag{18}$$

To find an upper bound for $P_{err}(m)$, we can combine (18) and the union bound as follows:

$$\begin{aligned} P_{err}(m) &= \text{Prob}\Big(\bigcup_{X,Y\in\Lambda} |\rho(X,Y) - \hat{\rho}(X,Y)| > \gamma\Big) \\ &\leq \sum_{X,Y\in\Lambda} \text{Prob}(|\rho(X,Y) - \hat{\rho}(X,Y)| > \gamma) \\ &\leq \frac{n^2}{2} \cdot 2\exp(-2\gamma^2 m) \end{aligned}$$

A sufficient condition for $P_{err}(m) \leq \delta$ is thus

$$\begin{aligned} \frac{n^2}{2} \cdot 2\exp(-2\gamma^2 m) &\leq \delta \\ m &\geq \frac{2\ln n - \ln\delta}{2\gamma^2} \end{aligned}$$

C Proof of Proposition 2.3

For each term inside the summation on the right hand side of Equation (11), $d(x,y)$ must be smaller than or equal to ϵ. If $d(x,y) \leq \epsilon$, our assumption implies that both x and y must be in the same cluster C belonging to both $[X]_\epsilon$ and $[Y]_\epsilon$. As a result, we can rewrite the right hand side of Equation (11) based only on clusters in $[X]_\epsilon \cap [Y]_\epsilon$:

$$\sum_{d(x,y)\leq\epsilon} \int_{V_X(x)\cap V_Y(y)} f(u; X\cup Y)\, du. = \sum_{C\in[X]_\epsilon\cap[Y]_\epsilon} \sum_{z\in C} \int_{V_X(z)\cap V_Y(z)} f(u; X\cup Y)\, du \tag{19}$$

Based on the definition of a Voronoi Cell, it is easy to see that $V_X(z) \cap V_Y(z) = V_{X\cup Y}(z)$ for all $z \in C$ with $C \in [X]_\epsilon \cap [Y]_\epsilon$. Substituting this relationship into Equation (19), we obtain:

$$\begin{aligned} \sum_{d(x,y)\leq\epsilon} \int_{V_X(x)\cap V_Y(y)} f(u; X\cup Y)\, du &= \sum_{C\in[X]_\epsilon\cap[Y]_\epsilon} \int_{V_{X\cup Y}(C)} f(u; X\cup Y)\, du \\ &= \sum_{C\in[X]_\epsilon\cap[Y]_\epsilon} \int_{V_{X\cup Y}(C)} \frac{1}{|[X\cup Y]_\epsilon| \cdot \text{Vol}(V_{X\cup Y}(C))}\, du \\ &= \frac{1}{|[X\cup Y]_\epsilon|} \sum_{C\in[X]_\epsilon\cap[Y]_\epsilon} \frac{\int_{V_{X\cup Y}(C)} du}{\text{Vol}(V_{X\cup Y}(C))} \\ &= |[X]_\epsilon \cap [Y]_\epsilon| / |[X\cup Y]_\epsilon| \end{aligned}$$

Finally, we note that $[X]_\epsilon \cap [Y]_\epsilon$ is in fact the set of all Similar Clusters in $[X\cup Y]_\epsilon$, and thus the last expression equals to the IVS. The reason is that for any Similar Cluster C in $[X\cup Y]_\epsilon$, C must have at least one $x \in X$ and one $y \in Y$ such that $d(x,y) \leq \epsilon$. By our assumption, C must be in both $[X]_\epsilon$ and $[Y]_\epsilon$.

D Proof of Proposition 2.4

Without loss of generality, we assume that x_1 is at the origin with all zeros, and x_2 has k 1's in the rightmost positions. Clearly, $d(x_1,x_2) = k$. Throughout this proof, when we mention a particular sequence $Y \in \Gamma$, we adopt the convention that $Y = \{y_1, y_2\}$ with $d(x_1,y_1) \leq \epsilon$ and $d(x_2,y_2) \leq \epsilon$.

We first divide the region A into two partitions based on the proximity to the frames in X:

$$A_1 \triangleq \{s \in A : g_X(s) = x_1\} \text{ and } A_2 \triangleq \{s \in A : g_X(s) = x_2\}$$

We adopt the convention that if there are multiple frames in a video Z that are equidistant to a random vector s, $g_Z(s)$ is defined to be the frame furthest away from the origin. This implies that all vectors equidistant to both frames in X are elements of A_2. Let s be an arbitrary vector in H, and R be the random variable that denotes the number of 1's in the rightmost k bit positions of s. The probability that R equals to a particular r with $r \leq k$ is as follows:

$$\text{Prob}(R = r) = \frac{1}{2^k}\binom{k}{r}$$

Thus, R follows a binomial distribution of parameters k and $1/2$. In this proof, we show the following relationship between A_2 and R:

$$\text{Vol}(A_2) = \text{Prob}(k/2 \leq R < k/2 + \epsilon) \tag{20}$$

With an almost identical argument, we can show the following:

$$\text{Vol}(A_1) = \text{Prob}(k/2 - \epsilon \leq R < k/2) \tag{21}$$

Since $\text{Vol}(A) = \text{Vol}(A_1) + \text{Vol}(A_2)$, the desired result follows.

To prove Equation (20), we first show if $k/2 \leq R < k/2 + \epsilon$, then $s \in A_2$. Be the definitions of A and A_2, we need to show two things: (1) $g_X(s) = x_2$; (2) there exists a $Y \in \Gamma$ such that $s \in G(X, Y; \epsilon)$, or equivalently, $g_Y(s) = y_1$. To show (1), we rewrite $R = k/2 + N$ where $0 \leq N < \epsilon$ and let the number of 1's in s be L. Consider the distances between s and x_1, and between s and x_2. Since x_1 is all zeros, $d(s, x_1) = L$. As x_2 has all its 1's in the rightmost k position, $d(s, x_2) = (L - R) + (k - R) = L + k - 2R$. Thus,

$$\begin{aligned} d(s, x_1) - d(s, x_2) &= L - (L + k - 2R) \\ &= 2R - k \\ &= 2N \geq 0, \end{aligned}$$

which implies that $g_X(s) = x_2$. To show (2), we define y_1 to be a h-bit binary number with all zeros, except for ϵ 1's in the positions which are randomly chosen from the R 1's in the rightmost k bits of s. We can do that because $R \geq k/2 \geq \epsilon$. Clearly, $d(x_1, y_1) = \epsilon$ and $d(s, y_1) = L - \epsilon$. Next, we define y_2 by toggling ϵ out of k 1's in x_2. The positions we toggle are randomly chosen from the same R 1's bits in s. As a result, $d(x_2, y_2) = \epsilon$ and $d(s, y_2) = (L - R) + (k - R + \epsilon) = L + \epsilon - 2N$. Clearly, $Y \triangleq \{y_1, y_2\}$ belongs to Γ. Since

$$\begin{aligned} d(s, y_2) - d(s, y_1) &= (L + \epsilon - 2N) - (L - \epsilon) \\ &= 2(\epsilon - N) > 0, \end{aligned}$$

$g_Y(s) = y_1$ and consequently, $s \in G(X, Y; \epsilon)$.

Now we show the other direction: if $s \in A_2$, then $k/2 \leq R < k/2 + \epsilon$. Since $s \in A_2$, we have $g_X(s) = x_2$ which implies that $L = d(s, x_1) \geq d(s, x_2) = L + k - 2R$ or $k/2 \leq R$. Also, there exists a $Y \in \Gamma$ with $s \in G(X, Y; \epsilon)$. This implies $g_Y(s) = y_1$, or equivalently, $d(s, y_1) < d(s, y_2)$. This inequality is strict as equality will force $g_Y(s) = y_2$ by the convention we adopt for $g_Y(\cdot)$. The terms on both sides of the inequality can be bounded using the triangle inequality: $d(s, y_1) \geq d(s, x_1) - d(x_1, y_1) = L - \epsilon$ and $d(s, y_2) \leq d(s, x_2) + d(x_2, y_2) = L + k - 2R + \epsilon$. Combining both bounds, we have

$$L - \epsilon < L + k - 2R + \epsilon \Rightarrow R < k/2 + \epsilon$$

This completes the proof for Equation (20). The proof of Equation (21) follows the same argument with the roles of x_1 and x_2 reversed. Combining the two equations, we obtain the desired result.

E Proof of Proposition 2.5

We prove the case for video X and the proof is identical for Y. Since $s \in G(X, Y; \epsilon)$, we have $d(g_X(s), g_Y(s)) > \epsilon$ and there exists $x \in X$ with $d(x, g_Y(s)) \leq \epsilon$. Since all Similar Clusters in $[X \cup Y]_\epsilon$ are ϵ-compact, $g_X(s)$ cannot be in the same cluster with x and $g_Y(s)$. Thus, we have $d(g_X(s), x) > \epsilon$. It remains to show that $d(x, s) - d(g_X(s), s) \leq 2\epsilon$. Using the triangle inequality, we have

$$\begin{aligned} d(x,s) - d(g_X(s),s) &\leq d(x, g_Y(s)) + d(g_Y(s), s) - d(g_X(s), s) \\ &\leq \epsilon + d(g_Y(s), s) - d(g_X(s), s) \end{aligned} \tag{22}$$

$s \in G(X, Y; \epsilon)$ also implies that there exists $y \in Y$ such that $d(y, g_X(s)) \leq \epsilon$. By the definition of $g_Y(s)$, $d(g_Y(s), s) \leq d(y, s)$. Thus, we can replace $g_Y(s)$ with y in (22) and combine with the triangle inequality to obtain:

$$\begin{aligned} d(x,s) - d(g_X(s),s) &\leq \epsilon + d(y,s) - d(g_X(s),s) \\ &\leq \epsilon + d(y, g_X(s)) \\ &\leq 2\epsilon. \end{aligned}$$

Seed Vectors	Uniform Random				Corel Images			
IVS	0.8	0.6	0.4	0.2	0.8	0.6	0.4	0.2
v1_1	0.59	0.37	0.49	0.20	0.85	0.50	0.49	0.23
v1_2	0.56	0.38	0.31	0.05	0.82	0.63	0.41	0.18
v3	0.96	0.09	0.06	0.02	0.82	0.52	0.40	0.21
v4	0.82	0.75	0.55	0.24	0.92	0.44	0.48	0.25
v5_1	0.99	0.71	0.28	0.18	0.76	0.66	0.39	0.12
v5_2	0.84	0.35	0.17	0.29	0.81	0.68	0.36	0.10
v5_3	0.97	0.36	0.74	0.07	0.76	0.59	0.51	0.15
v6	1.00	0.00	0.00	0.00	0.79	0.61	0.46	0.25
v7	0.95	0.89	0.95	0.60	0.86	0.60	0.49	0.16
v8_1	0.72	0.70	0.47	0.17	0.88	0.69	0.38	0.20
v8_2	1.00	0.15	0.91	0.01	0.86	0.53	0.35	0.21
v9_1	0.95	0.85	0.54	0.15	0.93	0.56	0.44	0.18
v9_2	0.85	0.70	0.67	0.41	0.86	0.56	0.39	0.17
v9_3	0.90	0.51	0.30	0.10	0.78	0.70	0.45	0.15
v9_4	1.00	0.67	0.00	0.00	0.72	0.45	0.42	0.24
Average	0.873	0.499	0.429	0.166	0.828	0.581	0.428	0.187
Stddev	0.146	0.281	0.306	0.169	0.060	0.083	0.051	0.046

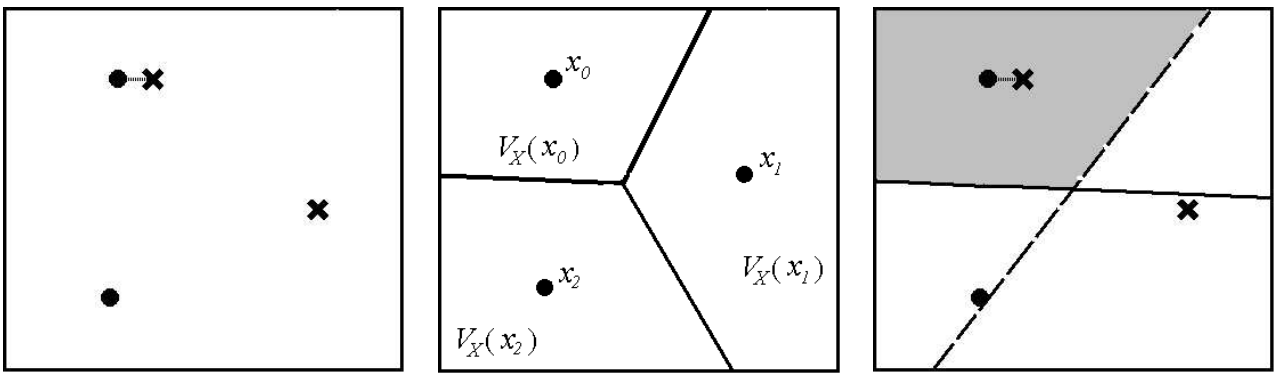
Table 1: *Comparison between using uniform random and corel image SV's. The second through fifth columns are the results of using uniform random SV's and the rest are the corel image SV's. Each row contains the results of a specific test video at IVS levels 0.8, 0.6, 0.4 and 0.2. The last two rows are the averages and standard deviations of the estimated IVSs over all the test sequences. The IVSs estimated with corel image SVs are closer to the ideal values in row two, and have much less variations than those estimated with random SVs.*

Algorithm	VSS_b					VSS_r				
ϵ	0.2	0.4	0.8	1.2	1.6	0.2	0.4	0.8	1.2	1.6
v1_1	0.89	0.76	0.62	0.54	0.36	1.00	1.00	0.90	0.87	0.74
v1_2	0.81	0.73	0.55	0.47	0.34	1.00	0.98	0.83	0.73	0.62
v3	0.90	0.76	0.70	0.42	0.36	1.00	1.00	0.96	0.87	0.72
v4	0.86	0.74	0.64	0.48	0.38	1.00	1.00	0.96	0.83	0.74
v5_1	0.90	0.77	0.64	0.45	0.41	1.00	1.00	0.98	0.79	0.86
v5_2	0.96	0.81	0.52	0.66	0.56	1.00	1.00	1.00	0.86	0.78
v5_3	0.88	0.83	0.59	0.42	0.39	1.00	1.00	0.90	0.83	0.74
v6	0.88	0.72	0.64	0.49	0.49	1.00	1.00	0.98	0.92	0.78
v7	0.89	0.84	0.68	0.46	0.43	1.00	1.00	1.00	0.91	0.78
v8_1	0.85	0.67	0.58	0.52	0.30	1.00	1.00	0.87	0.79	0.73
v8_2	0.90	0.80	0.72	0.59	0.56	1.00	1.00	0.99	0.95	0.86
v9_1	0.87	0.77	0.62	0.67	0.48	1.00	0.99	0.89	0.84	0.82
v9_2	0.82	0.70	0.55	0.50	0.37	1.00	1.00	0.90	0.78	0.59
v9_3	0.86	0.65	0.66	0.49	0.40	1.00	1.00	0.91	0.70	0.58
v9_4	0.92	0.86	0.71	0.61	0.53	1.00	1.00	0.93	0.89	0.82
Average	0.879	0.761	0.628	0.518	0.424	1.000	0.998	0.933	0.837	0.744
Stddev	0.038	0.061	0.061	0.080	0.082	0.000	0.006	0.052	0.070	0.088

Table 2: *Comparison between* VSS_b *and* VSS_r *under different levels of perturbation. The table follows the same format as in Table 1. The perturbation levels* ϵ *tested are 0.2, 0.4, 0.8, 1.2 and 1.6. The average* $VSS_r s$ *are much closer to the ideal IVS value of one than the average* $VSS_b s$.

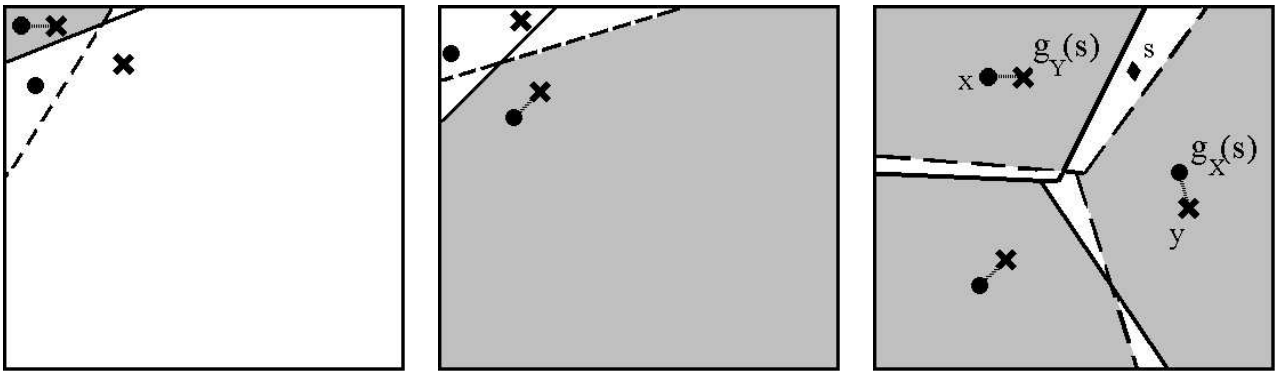
Video Type	% over all clips	Duration (mean ± std-dev in minutes)
MPEG	31	0.26 ± 0.7
QuickTime	30	0.51 ± 0.6
RealVideo	22	9.57 ± 18.5
AVI	16	0.16 ± 0.3

Table 3: *Statistics of collected web video sequences*



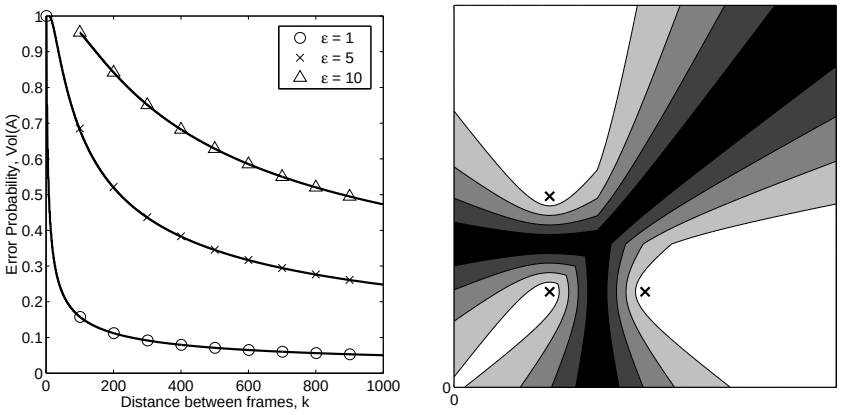
(a) (b) (c)

Figure 1: *(a) Two video sequences with IVS equal to 1/3. (b) The Voronoi Diagram of a 3-frame video* X. *(c) The shaded area, normalized by the area of the entire space, is equal to the VVS between the two sequences shown.*



(a) (b) (c)

Figure 2: *(a), (b) Two examples of sequences with identical IVSs but very different VVSs. (c) The VG for X and Y corresponds to the unshaded region in the figure. There are two characteristics to any SV s inside VG: 1) $g_X(s)$ and $g_Y(s)$ are not similar, 2) there exist $x \in X$ similar to $g_Y(s)$, and $y \in Y$ similar to $g_X(s)$.*



(a) (b)

Figure 3: *(a) The error probability for the hamming cube at different values of ϵ and distances k between the frames in the video. (b) Values of ranking function $Q(\cdot)$ for a three-frame video sequence. Lighter colors correspond to larger values.*

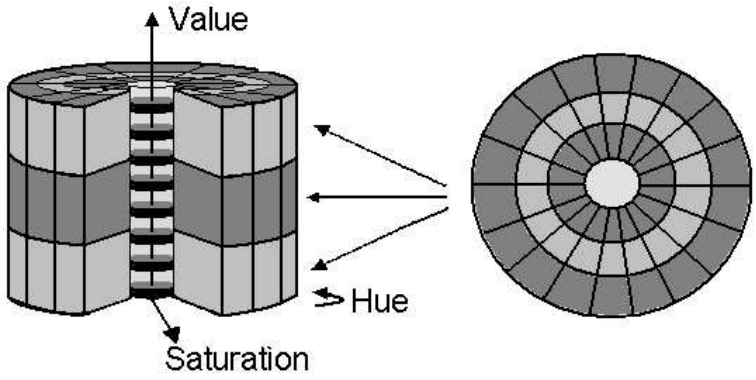


Figure 4: *Quantization of the HSV color space.*

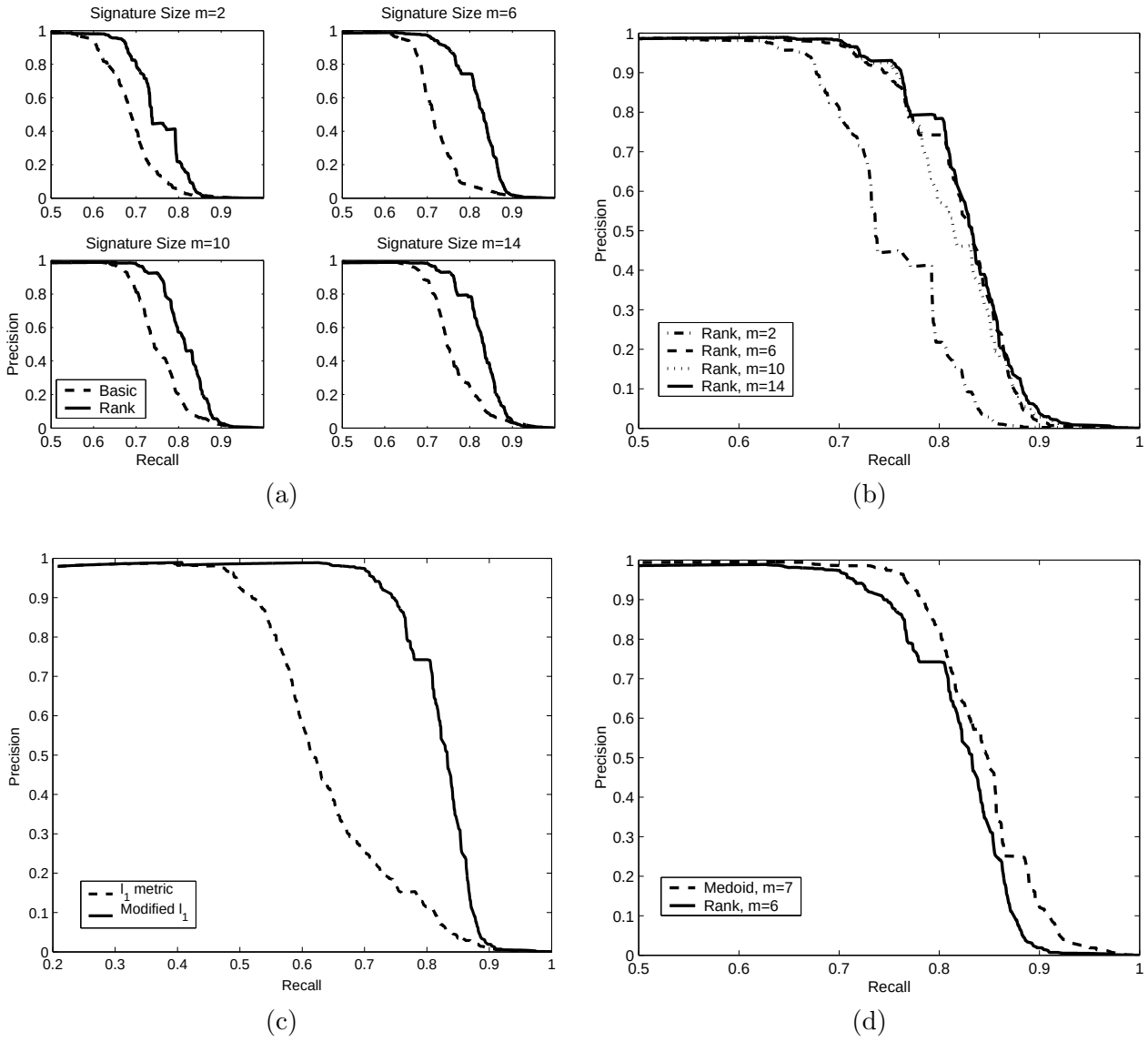


Figure 5: *Precision-recall plots for web video experiments: (a) Comparisons between the Basic (broken-line) and Ranked (solid) ViSig methods for four different ViSig sizes:* $m = 2, 6, 10, 14$*; (b) Ranked ViSig methods for the same set of ViSig sizes; (c) Ranked ViSig methods with* $m = 6$ *based on* l_1 *metric (broken) and modified* l_1 *distance (solid) on color histograms; (d) Comparison between the Ranked ViSig method with* $m = 6$ *(solid) and k-medoid with 7 representative frames (broken).*