Video Linkage: Group Based Copied Video Detection

Hung-sik Kim Jeongkyu Lee† Haibin Liu Dongwon Lee*
The Pennsylvania State University, USA †University of Bridgeport, USA
{hungsik, haibin, dongwon}@psu.edu, jelee@bridgeport.edu

ABSTRACT

Sites to share user-created video clips such as YouTube and Yahoo Video have become greatly popular in recent years. One of the challenges of such sites is, however, to prevent video clips that violate copyrights by illegally copying and editing scenes from other videos. Due to the sheer number of clips uploaded every day, automatic methods to detect (illegally) copied video clips in a large collection are desirable. Toward this problem, in this paper, we present a novel framework, termed as Video Linkage, that is based on the record linkage techniques. Our proposal is based on the observations that: (1) a video clip can be represented as a "group" of key frames, (2) two video clips are deemed to be similar if two groups of key frames are similar as a whole – i.e., the similarity of two video clips can be measured by means of graph-based similarity measures such as maximal cardinality bipartite matching, and (3) if a video clip v_a is copied to v_b, then v_a and v_b must be somehow similar, but not all similar video clips are illegally copied ones – i.e., *similar* videos can be used as a filter for fast detection of *copied* videos. The validity of our observations and Video Linkage technique is thoroughly evaluated using both real and synthetic data sets – i.e., on average, our proposals achieved 0.94 as precision and 0.93 as recall across 10 genres and 6 editing patterns.

Categories and Subject Descriptors: H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval – Information filtering, Retrieval models

General Terms: Algorithms, Design, Performance.

Keywords: Video, Copyright, Linkage

1. INTRODUCTION

Due to the surge of Web 2.0 in recent years, user-created contents (UCC) such as blogs, photos, and videos, are everywhere. In particular, sites such as YouTube[1] and Yahoo Video[2] where users upload video clips for sharing have become very popular, partly thanks to the advancement of video recording/editing, hardware/software and people's desire for interactive multimedia contents on the Web. One of the challenges of such sites is, however, to prevent video clips that violate copyrights by illegally copying and editing scenes from other videos. This is because people often upload movie trailers or TV shows to such sites without proper authorization, often knowing its illegal nature. For instance, in 2007, Viacom[3], the owner of MTV, asked the removal of their copyrighted contents from YouTube that have been viewed more than 1.5 billion times. Therefore, it only gets more important for companies to be able to detect and remove such illegal contents in their collections to avoid serious legal and financial responsibilities.

Due to the sheer number of video clips uploaded/viewed every day, however, manual operations to detect and remove pirated videos cannot keep up with demand. According

Figure 1: Key frames of an original video scene (a) from the movie "Forrest Gump", three copied/altered ones (b)-(d), and a similar one (e).

*Partially supported by IBM and Microsoft gifts.

[1] http://www.youtube.com/

[2] http://video.yahoo.com/

[3] http://www.viacom.com/

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CIVR'08, July 7–9, 2008, Niagara Falls, Ontario, Canada.
Copyright 2008 ACM 978-1-60558-070-8/08/07 ...$5.00.

to comScore Video Metrix report[4], 134 million Americans viewed more than 9 billion online videos, 2.4 million of which are from YouTube, during July 2007. The situation will get only exacerbated as time passes. Naturally, therefore, automatic methods to detect illegally copied video clips *fast* and *accurately* in a large collection are greatly desirable. However, in general, detecting copied videos is a much harder problem than detecting similar videos, as illustrated in the following motivating example.

Example 1. Figure 1 shows selected key frames of an original video scene, (a), from the movie "Forrest Gump", three other scenes, (b)-(d), that are possibly copied and altered from (a), and one similar but non-copied scene, (e). We obtained all video scenes except (a) from YouTube. It is observed that the copied videos, (b)-(d), are edited by some basic operations such as integrating several shots, changing format, and inserting title, transitions, and/or credit. For example, Figure 1(b) is edited by adding a credit scene, changing resolution, and cropping the original scene, while Figure 1(c) by adding a transition scene and changing the size of video. In particular, Figure 1(d) is (supposedly) illegally captured by a camcorder, which causes a lot of noise and unexpected modification like in the last key frame. However, on the other hand, Figure 1(e) is a similar video of Figure 1(a), but not copied one since it is an animation parodying "Forrest Gump". Therefore, a good system for copied video detection should conclude that only Figure 1(b)-(d) are copies of Figure 1(a) by utilizing editing methods. □

Toward this **Copied Video Detection (CVD)** problem, in this paper, we present a novel solution, termed as Video Linkage, that is based on the record linkage techniques – i.e., to determine if two entities represented as relational records are approximately the same or not (see the related work in Section 3). Informally, given a video v_q and a collection of videos V, we aim at detecting all videos from V that are copies of v_q. For the detection, the Video Linkage technique is based on the following observations. First, a video can be represented as a "group" of *shots* and in turn a shot as a "group" of image *frames*. Furthermore, inherently, there is a temporal ordering among frames of a video. Second, two videos are deemed to be similar if two groups of frames are similar, and the notion of "groups" can be well captured by graphs. Therefore, we can measure the similarity between two videos by means of graph-based similarity measures such as maximal cardinality or weighted bipartite matching. Lastly, if a video v_a is illegally copied from a video v_b, then v_a and v_b must be somehow similar (having similar but altered shots). Therefore, we can prune dissimilar videos out using a simpler similarity for fast detection of copied videos.

Our contributions are as follows:

- To enable group based matching idea, we propose a method to transform a video to a group of frames. A set of shots are first identified from a video, and in turn a small number of key frames are extracted from each shot. For the frame-to-frame similarity, we extract a feature set $\mathcal{F}$ from each frame that includes HSV color histogram (f_H), YCbCr color layout (f_Y), and motion vector (f_M). In addition, we compute a binary image signature $\mathcal{S}$ for the efficient processing of CVD.

[4] http://www.comscore.com/press/

Notation	Description
V	a set of videos (or shots)
v	a video (or a video shot)
f	a frame
$sim_v(v_1, v_2)$	video-to-video similarity
$sim_f(f_1, f_2)$	frame-to-frame similarity
θ / ρ	threshold for $sim_v(v_1, v_2)$ / $sim_f(f_1, f_2)$
$\mathcal{F}$ / $\mathcal{S}$	feature set / signature for a frame

Table 1: Notations used.

- Once videos are captured as a group of frames, we propose five efficient group based video similarity measures: (1) two exact measures, VL and $NCVL$, based on the maximum weight bipartite matching and maximum weight non-crossing bipartite matching, (2) one greedy measure, gVL, and (3) two approximate measures, aVL and $aNCVL$. Further, we show the partial order relationship among five measures and their utility (in the filtered Video Linkage algorithm).
- Based on the five new measures, we propose two algorithmic frameworks for the copied video detection problem: (1) *standalone* Video Linkage algorithm and (2) *pipelined* Video Linkage algorithm.

2. OVERVIEW OF PROBLEM & SOLUTION

Throughout the paper, we use notations in Table 1.

Problem Overview. The copied video detection problem can be modeled as both *selection* problem (i.e., select top-k copied videos) as well as *threshold* problem (i.e., find all copied videos above a threshold). To make the presentation simple, hereafter, we use the threshold version of the problem. Informally, the copied video detection problem in our setting is defined as follows:

> **Copied Video Detection (CVD).** Given a set of query videos V_q and a collection of source videos V_s, for each query video v_q ($\in V_q$), detect all copied videos, V_c ($\subseteq V_s$) that contain either duplicated or altered video shots from v_q.

A naive solution to the CVD problem can be conceptually broken into two levels of quadratic computation. First, to compute $sim_v(v_a, v_b)$ using the proposed bipartite graph matching idea, one needs to compute all pair-wise frame-to-frame similarities, $sim_f(f_i, f_j)$, as follows:

```
for each frame f_i ∈ v_a
  for each frame f_j ∈ v_b
    G[i, j] ← sim_f(f_i, f_j);
compute a graph matching M on G[i, j];
sim_v(v_a, v_b) ← some similarity defined on M;
```

However, often, the number of frames per video tends to be extremely large – e.g., 18,000 frames for only 10 minutes video clip. Therefore, one needs to devise an efficient way to avoid the excessive computations of $sim_f(f_i, f_j)$. Second, once a similarity between two videos is determined, one still needs to have nested-loop style computation among two collections of videos as follows:

```
for each query video v_q ∈ V_q
  for each source video v_s ∈ V_s
    if sim_v(v_q, v_s) ≥ θ, return v_s;
```

Both have costly time complexities of $O(|v_a||v_b|)$ and $O(|V_q||V_s|)$, respectively, making the naive solution prohibitively expensive. Therefore, the objective of our proposal is to find the efficient solutions for both sim_v and sim_f.

Since we focus on the efficient methods of sim_v and sim_f using the record linkage context, the issue of multimedia indexing for fast copied video detection is not discussed in this paper. Instead, we will filter out non-similar video clips using a signature ($\mathcal{S}$) before Video Linkage algorithms are applied. Based on the proposed group based similarity measures, we plan to develop such an indexing technique in future work.

Solution Overview. In order to address the problem, we first investigate frame-to-frame similarity measure $sim_f()$ by extracting features of frames. For the video-to-video similarity measure $sim_v()$, we propose a variety of novel group based solutions. Then, we study the algorithmic improvement therein. The basic flow of our proposed solution is as follows:

- (Sections 4.1 and 4.2) In order to avoid the quadratic computation of $sim_f()$ per video, we extract various features from video frames, detect video shots using those features, and identify "key" frames per video. At the end, each video is captured as a group of extracted key frames. Since every frame in a shot contains almost similar contents, we consider video shots as primary processing units in the CVD problem[5].

- (Section 4.3) For the efficient processing of CVD, non-similar video clips are filtered out using a binary image signature.

- (Sections 4.4 and 4.5) Since a video clip is a substantially more complex entity than a textual relational record in the record linkage problem, we investigate novel similarity measures that take advantage of the notion of "group" and the constraint of temporal order among frames. In particular, we propose several group similarity measures based on the maximum weight bipartite matching and the maximum weight non-crossing bipartite matching, respectively, and then extend them into faster measures to address the issue of sim_v in the above algorithm.

- (Section 4.6) Finally, the proposed five similarity measures are used in two Video Linkage algorithms.

3. RELATED WORK

The general data linkage problem has been known as various names – record linkage, entity resolution, object matching, merge-purge, etc (e.g., [1, 5, 16]). Readers are referred to excellent survey papers (e.g., [18]) for the latest development of the linkage problem. Two most relevant record linkage techniques that we exploit in this paper is *group record linkage* and *blocking*. Group record linkage [14] is a novel record linkage that authors have developed to match data objects that have a group of elements in them. In this paper, we extend the technique further to accommodate temporal ordering among frames of videos. Blocking technique was first proposed by [7], and has been studied extensively [12] where initial rough but fast clustering is followed by more exhaustive record matching step. We apply the blocking idea in the pipelined Video Linkage algorithm in Section 4.6.

The techniques in the CVD problem can be classified into two main approaches: (1) *watermarking based approach* [10, 20, 15] utilizes invisible information embedded into the media to identify the ownership, and (2) *content-based approach* [9, 17, 8, 6, 19, 4] extracts persistent features to determine copied or non-copied media by distance or similarity measures.

In the watermarking based approach, non-visible information (e.g., multimedia fingerprinting) is embedded into the media by modifying a video, and detected to identify the ownership by the detection system. For instance, Lin et al. [10] utilized a block matching algorithm for embedding and detecting the key. In [15], an active clustering approach was proposed to achieve efficiency.

Since the watermarking based approach cannot be feasible to old media without watermarks, there have been more research studies investigating the content based approach that requires the persistent features and distance (or similarity) measures. Depending on the editing methods to generate copied videos, each algorithm proposed different features and different (dis-)similarity measures. In [8], a spatio-temporal sequence matching is proposed to handle a wide range of modifications in videos, while [6] proposed statistical similarity search for the CVD problem based on an approximate search paradigm. A survey of comparative study regarding to features and distance measures used in the CVD problem can be found in several literatures [9, 4].

Almost all aforementioned approaches concentrated along the transformations of media, such as contrast, blur, zoom in and out, brightness and histogram equalization [9, 4]. However, the CVD problem based on the transformations is very limited to capture the characteristics of copied videos in popular video uploading sites, such as YouTube and Yahoo Video. Based on our extensive observation over more than 12,000 video clips in YouTube, most common and popular editing methods are simple operations such as integrating several shots, changing format of videos, and inserting title, transitions, or credit. Moreover, expensive processing costs are required to keep the robustness over the various transformations, which prevents fast processing over very large video databases. [13] proposed a fast shot-based matching strategy that employs a hash table to index the signatures extracted from shots. However, it is limited to retrieve duplicated shots, i.e., the proposed approach is close to a video retrieval system. Therefore, unlike existing works, we present a novel solution, termed as Video Linkage, that is based on the record linkage techniques to detect copied video from a very large collection of videos efficiently.

4. MAIN PROPOSAL: VIDEO LINKAGE

The main idea of our Video Linkage proposal is based on the premise that when a video v_c is copied from a video v_q, (1) there exist major editing operations for the copy, such as integrating shot, adding title, transitions or(and) credit, changing format, and resizing of the original videos, and (2) v_q and v_c must be somehow similar (having similar but altered shots). We propose to capture the intuitions as the notion of "group" in this section.

[5]To make our presentation simple, we use a term "video" instead of "video shot".

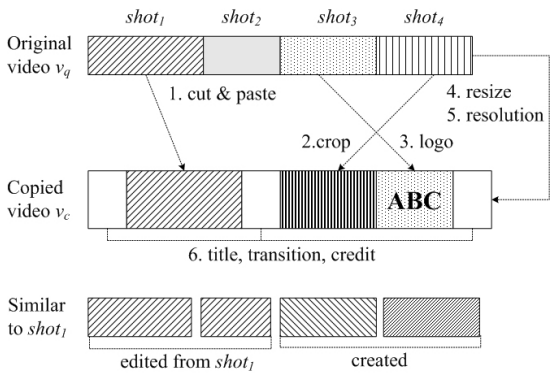


Figure 2: Copied vs. similar videos.

4.1 Characterization of Copied Videos

Generally, a copied video detection (CVD) system is different from a content-based video retrieval (CBVR) system. The result of CVD is a set of copied videos that are illegally edited videos from copyright protected videos, while that of CBVR is a set of similar videos that are edited or created videos with similar contents. Therefore, a CVD system should utilize both the editing methods as well as similar contents.

Figure 2 illustrates the difference between copied and similar videos. As shown in the figure, a copied video v_c in sites such as YouTube and Yahoo Video is typically created by editing the original video v_q using "cut and paste," "cropping," "adding logo/text," "resizing," "changing video format/resolution," and "adding title, transition, or(and) credit." When copied videos are captured as "groups" of frames (to be discussed), we argue that the detection of edit patterns be easier. On the other hand, a similar video can be either an edited video or a non-copied videos with similar contents.

In order to capture the characteristics of copied videos discussed, we need a feature set $\mathcal{F}$ that can describe each frame appropriately. For the feature set, we extracted three features from each frame including HSV color histogram (λ_H), YCbCr color layout (λ_Y), and motion vector histogram (λ_M).

Let $\mathcal{F} = \{\lambda_H, \lambda_Y, \lambda_M\}$ be a set of features extracted from each frame. Then, each feature in $\mathcal{F}$ and its similarity measure between two frames, f_1 and f_2, are defined as follows:

1. λ_H represents a scalable color of the frame based on a HSV color histogram. The color histogram, CH, has 256 bins (L) in total including 16 values in H (hue), and 4 values in S (saturation) and V (value), i.e., $L = 16 \times 4 \times 4 = 256$ bins. A similarity measure for HSV color histogram, $sim_{\lambda_H}()$, is defined as:

$$sim_{\lambda_H}(f_1, f_2) = \frac{\sum_{l=0}^{L-1} min(CH_{f_1}(l), CH_{f_2}(l))}{\sum_{l=0}^{L-1} CH_{f_1}(l)}$$

2. λ_Y describes a color layout based on YCbCr color domain. This is for the spatial distribution of the colors, i.e., first divide each frame into 16×16 pixels block, second average YCbCr values for each block with $YCbCr_{avg} = 2/3 \cdot Y + 1/6 \cdot Cb + 1/6 \cdot Cr$, and lastly compute the similarity using *cosine distance*. Let $\{v_{1m}\}$ and $\{v_{2m}\}$ for $m = 1, 2, \ldots, M$ be the feature vector of f_1 and f_2 for average YCbCr, respectively. M is the number of blocks in one frame. A similarity measure for average $YCbCr$, $sim_{\lambda_Y}()$, is defined as:

$$sim_{\lambda_Y}(f_1, f_2) = \frac{\sum_{m=1}^{M} v_{1m} \cdot v_{2m}}{\sqrt{\sum_{m=1}^{M} v_{1m}^2 \cdot \sum_{m=1}^{M} v_{2m}^2}}$$

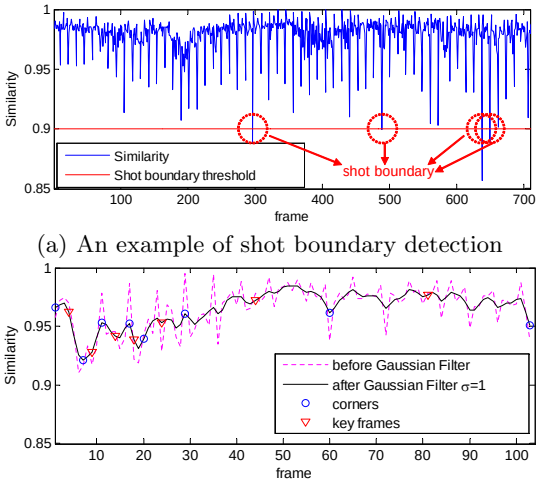
3. λ_M represents a moving pattern of the frame based on a motion vector histogram (MH). First, we find a motion vector for each 16×16 block over two consecutive frames. Each motion vector then falls into one of 9 bins ($N = 9$), i.e., (0,0), (1,0), (1,1), (0,1), (-1,0), (0,-1), (-1,-1), (1,-1), and (-1,1). A similarity measure for motion vector histogram, $sim_{\lambda_M}()$, is defined as:

$$sim_{\lambda_M}(f_1, f_2) = \frac{\sum_{n=0}^{N-1} min(MH_{f_1}(n), MH_{f_2}(n))}{\sum_{n=0}^{N-1} MH_{f_1}(n)}$$

Each feature mentioned above characterizes a color scale, a color layout, and a motion in a copied video, respectively. Now, we define the frame-to-frame similarity measure, $sim_f()$, with the extracted feature set, $\mathcal{F}$, as follows:

$$sim_f(f_1, f_2) = \sum_{\lambda \in \mathcal{F}} w_\lambda \cdot sim_\lambda(f_1, f_2) \quad (1)$$

where w_λ is a weight of a feature λ such that $\sum_{\lambda \in \mathcal{F}} w_\lambda = 1$, and $sim_\lambda()$ is a similarity of two frames with respect to the feature λ. The frame-to-frame similarity measure, $sim_f()$ will be used for any comparison between two frames in this paper.



(a) An example of shot boundary detection

(b) An example of key frames selection

Figure 3: Shot boundary and key frame selection.

4.2 Group Formation

Once features are extracted from each frame, in this section, we discuss how to transform a video (shot) into a "group" of important frames (i.e., key frames) so that we can detect copied videos using the notion of "group" later on.

First, a *shot* is defined as a sequence of frames taken by one camera operation. In other words, all frames in the same shot tend to contain similar contents. Due to this

nature of shot, we select shots as a primary processing unit to detect copied videos. We use $sim_f()$ with $\mathcal{F} = \{\lambda_H\}$ in Equation (1) to detect shot boundaries, since λ_H is robust enough to detect them with the less computational cost. For the detection, we compute the similarity of two consecutive frames, f_i and f_{i+1}, and if $sim_f(f_i, f_{i+1})$ is more than a certain threshold (ρ_{shot}), then f_i and f_{i+1} are considered as a shot boundary. Otherwise, both f_i, f_{i+1} belong to the same shot. Figure 3(a) illustrates how four shot boundaries are detected when $\rho_{shot} = 0.9$.

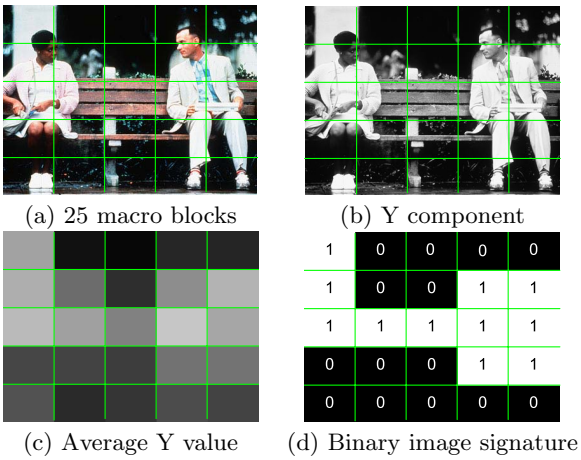
Second, the *key frame(s)* is(are) one or more selected frames that can represent the semantics of a video well. By using a small number of key frames per shot, we aim at reducing the computational cost of video processing drastically. To select the key frames, we consider three variations – *uniform*, *dynamic*, and *hybrid* key selection strategies. In the uniform strategy, a frame is selected as a key frame with uniform gap (e.g., every 50 frames in a shot). In the dynamic strategy, frame similarity values obtained in detecting shot boundaries are re-used to construct a similarity curve that shows how the contents change over an entire video. The high curvature indicates a significant change around the frames while the flat indicates no change. Those frames in high curvatures of a similarity curve are, thus, selected as key frames. To detect the high curvature points, we employ the algorithm proposed by [2]. In order to avoid noise, the similarity curve is smoothed by applying convolution with Gaussian filter ($\sigma = 1$). Finally, in the hybrid strategy, we first apply the dynamic strategy and if no key frame is selected for k consecutive frames by the dynamic strategy, $(k+1)$-th frame from the previously selected key frame is selected as a key frame.

Figure 3(b) shows the example including an original similarity curve, a smoothed curve, and detected key frames (marked as circle). In practice, both shot boundary detection and key frame extraction steps can be done in a single scan of a video sequence.

4.3 Binary Image Signature

A binary image signature is computed for each key frame used for efficient processing of CVD. The purpose of binary image signature is to filter non-similar video shots out very fast by a simple boolean operation. Therefore, the same or similar key frame should have the same binary signature. For this requirement, extreme quantization is applied to transform complex image data to a series of binary information. A binary image signature of a key frame can be computed as shown in Figure 4. First, a key frame is divided to 25 macro blocks (Figure 4(a)). Second, we average Y component in YCbCr color layout for each macro blocks (Figure 4(b)). Third, we find a median value among 25 Y average values (Figure 4(c)). Finally, extreme quantization is applied, i.e., if a value is less than the median, it becomes 0. Otherwise, 1. In this case, a binary image signature of a given image $\mathcal{S}$ is '1000010011111110001100000'.

If a video is a copy of another, then there should be at least one frame that has a common image signature. In addition, similar images should have the same image signature because it is computed by extreme quantization. We assume that no extreme distortion can be applied to all frames in a copied video. Therefore, we can use the signature to filter non-similar video clips out for efficient processing. In other words, if we find the same signature in two different videos,



(a) 25 macro blocks (b) Y component (c) Average Y value (d) Binary image signature

Figure 4: Steps to compute a binary image signature by extreme quantization.

one of Video Linkage algorithms will be applied to detect actual copied videos. Otherwise, we just ignore it.

4.4 Video Linkage Measures

Once a video is captured by key frames as shown in Section 4.2, now, we are ready to discuss how to measure the similarity between two videos utilizing the "group" information and video characteristics. In essence, we significantly extend the group based record linkage techniques developed by authors in [14] in order to exploit the temporal order among frames, and propose five Video Linkage measures and two Video Linkage algorithms.

Definition 1 (Video as Group) *A video v is captured as a group of frames: g=$\{f_1, \ldots, f_m\}$.* □

Given two groups g_1 and g_2, one of the simplest and most intuitive similarity measures is the *Jaccard* similarity, defined as $\frac{|g_1 \cap g_2|}{|g_1 \cup g_2|}$. By generalizing the Jaccard similarity to be able to handle approximate matching between two frames, we can use the bipartite graph idea.

Definition 2 (Weighted Bipartite Graph for Groups) *Given two groups of image frames, $g_1 = \{f_{11}, f_{12}, \ldots, f_{1m_1}\}$ and $g_2 = \{f_{21}, f_{22}, \ldots, f_{2m_2}\}$, a weighted bipartite graph is a bipartite graph $G = \{N, E, \Omega\}$, where $N = g_1 \cup g_2$, $E = g_1 \times g_2$, and Ω=$\{\omega(i,j) | \omega(i,j) = sim_f(f_{1i}, f_{2j})\}$* □

Definition 3 (Maximum Weight Bipartite Matching) *A* matching *is a set of pairwise non-adjacent edges in E. A* maximum weight bipartite matching *M is a matching M such that $\sum_{(f_{1i}, f_{1j}) \in M}(\omega(i,j))$ is the maximum.* □

Conceptually, the numerator and denominator of the Jaccard similarity are equivalent to the sum of weights in M and the number of nodes in N, offset by the number of edges in M, respectively. Based on this observations, now, we propose our group based video similarity measure as follows:

Definition 4 (Video Linkage) *For the bipartite group G = $\{N, E, \Omega\}$ over two groups of image frames, $g_1 = \{f_{11}, f_{12}, \ldots, f_{1m_1}\}$ and $g_2 = \{f_{21}, f_{22}, \ldots, f_{2m_2}\}$, the* Video Linkage *measure, $VL_{\omega,\rho}$, is the normalized weight of the the*

maximum weight bipartite matching M_1:

$$VL_{\omega,\rho}(g_1, g_2) = \frac{\Sigma_{(f_{1i}, f_{2j}) \in M_1}(\omega(i,j))}{m_1 + m_2 - |M_1|}$$

such that $\omega(i,j) \geq \rho$, where ρ is a user-set minimum threshold for edge similarity. □

Note that the denominator of $VL_{\omega,\rho}$ adds up the number of edges in the matching, M_1, and the number of "unmatched" frames in each of g_1 (i.e., $m_1 - |M_1|$) and g_2 (i.e., $m_2 - |M_1|$). When the numerator is large, it capture the intuition that two videos have "many" similar frames. Similarly, when the denominator is small, it captures the intuition that a large fraction of frames in the two groups are similar. Note also that we do *not* consider all pair-wise edges between two groups. Rather, we prune away those edges whose ω is substantially low (i.e., $\omega(i,j) < \rho$). Not only this early pruning helps improve the accuracy of Video Linkage technique, it speeds up computation significantly since all subsequent algorithms work faster on a "sparse" bipartite graph.

In addition to $VL_{\omega,\rho}$, we also observe that there might be the inherent temporal order among frames. Suppose a pirate copies a portion of video scenes, say 10 seconds with 300 frames, into his own video and alters them (e.g., adding logo and subtitle and changing contrast and brightness). Although the visual effects and characteristics of 300 frames might have changed, temporal order among 300 frames is still intact. Although it is possible to change temporal order among copied frames, we believe such cases are rare. Therefore, we extend $VL_{\omega,\rho}$ to take advantage of the order among elements of groups.

Definition 5 (Non-Crossing Bipartite Matching) *Consider an "ordered" bipartite graph G={N, E, Ω} over groups g_1 and g_2, where nodes in each group are numbered in increasing order from top to bottom. Two edges between nodes, $e_1 = (i,j)$ and $e_2 = (p,q)$, are said "crossing" iff ($i \leq p$ and $j \geq q$) or ($i \geq p$ and $j \leq q$). Then, a* non-crossing matching *is a subset of edges M_2 ($\in E$) such that no two edges of M_2 cross, including crossing at nodes. A* maximum weight non-crossing bipartite matching *is a non-crossing matching such that $\sum_{(f_{1i}, f_{1j}) \in M_2}(\omega(i,j))$ is the maximum.* □

When applied to the problem of matching two videos, v_1 and v_2, a non-crossing bipartite matching captures the intuition that once a frame f_{1i} ($\in v_1$) and a frame f_{2j} ($\in v_2$) match each other, no crossing matching can occur (i.e., the sequential order among frames must be followed). By capitalizing on this intuition, then we define our second group based video linkage measure as follows:

Definition 6 (Non-Crossing Video Linkage) *For the "ordered" bipartite graph G={N, E, Ω} over two groups g_1={f_{11}, f_{12}, …, f_{1m_1}} and g_2={f_{21}, f_{12}, …, f_{2m_2}}, the non-crossing* Video Linkage *measure, $NCVL_{\omega,\rho}$, is the the normalized weight of the maximum weight "non-crossing" bipartite matching M_2:*

$$NCVL_{\omega,\rho}(g_1, g_2) = \frac{\Sigma_{(f_{1i}, f_{2j}) \in M_2}(\omega(i,j))}{m_1 + m_2 - |M_2|}$$

such that $\omega(i,j) \geq \rho$, where ρ is given. □

Both $VL_{\omega,\rho}$ and $NCVL_{\omega,\rho}$ are guaranteed to be between 0 and 1. Furthermore, from the definitions, the following follows.

Lemma 1. *For two groups g_1 and g_2:*

$$NCVL_{\omega,\rho}(g_1, g_2) \leq VL_{\omega,\rho}(g_1, g_2)$$

where ρ is given. ■

Proof. Let us compare $VL_{\omega,\rho}$ of Definition 4 and $NCVL_{\omega,\rho}$ of Definition 6. Then, (1) the numerator of $VL_{\omega,\rho}$ is at least as large as the numerator of $NCVL_{\omega,\rho}$ since M_2 can have only equal or lower weighted edges than M_1 has due to the "non-crossing" constraint: i.e., $\Sigma_{(f_{1i}, f_{2j}) \in M_2}(\omega(i,j)) \leq \Sigma_{(f_{1i}, f_{2j}) \in M_1}(\omega(i,j))$, and (2) the denominator of $VL_{\omega,\rho}$ is no larger than the denominator of $NCVL_{\omega,\rho}$ since $|M_1| \geq |M_2|$: i.e., $m_1 + m_2 - |M_1| \leq m_1 + m_2 - |M_2|$. Since the numerator is larger and the denominator is smaller, $VL_{\omega,\rho}$ is at least as large as $NCVL_{\omega,\rho}$. (q.e.d)

4.5 Faster Video Linkage Measures

Both $VL_{\omega,\rho}$ and $NCVL_{\omega,\rho}$ capture the intuitions of two matching videos very well. However, both measures are computationally costly because of the requirement that "no node in the bipartite graph can have more than one edge incident on it." The known algorithms to find maximum weight bipartite matching and maximum weight non-crossing bipartite matching have time complexities of $O(N^2E)$ [3] (e.g., Hungarian or Bellman-Ford) and $O(N^2)$ [11], respectively. In search of faster video linkage measures, therefore, we relax this requirement of the bipartite matching using the greedy strategy.

Definition 7 (Greedy Video Linkage) *Consider the bipartite graph G={N, E, Ω} over two groups g_1={f_{11}, f_{12}, …, f_{1m_1}} and g_2={f_{21}, f_{12}, …, f_{2m_2}}.*

- *For each frame $f_i \in g_1$, find a frame $f_j \in g_2$ with the highest ω ($\geq \rho$) and denote the set of all such frame pairs as $S1$.*
- *Symmetrically, for each frame $f_j \in g_2$, find a frame $f_i \in g_1$ with the highest ω ($\geq \rho$) and denote the set of all such frame pairs as $S2$.*

Then, a greedy Video Linkage *measure, $gVL_{\omega,\rho}$, is:*

$$gVL_{\omega,\rho}(g_1, g_2) = \frac{\Sigma_{(f_{1i}, f_{2j}) \in S1 \cup S2}(\omega(i,j))}{m_1 + m_2 - |S1 \cup S2|}$$

such that $\omega(i,j) \geq \rho$, where ρ is given. □

Note that neither $S1$ nor $S2$ may be a matching. In $S1$, the same frame in g_2 may be the target of more than one frame in g_1 (thus violating the definition of "matching"). Similarly, in $S2$, the same frame in g_1 may be the target of more than one frame in g_2.

Lemma 2. *The greedy video linkage measure, $gVL_{\omega,\rho}(g_1, g_2)$, can be computed in $O(N + E \log E)$ time on the bipartite graph G={N, E, Ω} over two groups g_1 and g_2.* ■

The usefulness of the greedy group based video linkage measure, $gVL_{\omega,\rho}$, lies on the fact that its similarity value is always an over-estimation of true similarity value of $VL_{\omega,\rho}$. Therefore, the value of $gVL_{\omega,\rho}$ is not bounded between 0 and 1.

Lemma 3. *For two groups g_1 and g_2:*

$$VL_{\omega,\rho}(g_1, g_2) \leq gVL_{\omega,\rho}(g_1, g_2)$$

where ρ is given. ■

Measure	Time Complexity
$VL_{\omega,\rho}(g_1, g_2)$	$O(N^2E)$
$NCVL_{\omega,\rho}(g_1, g_2)$	$O(N^2)$
$gVL_{\omega,\rho}(g_1, g_2)$	$O(N + E \log E)$
$aVL_{\omega,\rho}(g_1, g_2)$	$O(E \log E)$
$aNCVL_{\omega,\rho}(g_1, g_2)$	$O(E^2)$

Table 2: Time complexities of five measures.

PROOF. Let us compare $VL_{\omega,\rho}$ of Definition 4 and $gVL_{\omega,\rho}$ of Definition 7. Then, (1) the numerator of $gVL_{\omega,\rho}$ is at least as large as the numerator of $VL_{\omega,\rho}$: i.e., $\Sigma_{(f_{1i},f_{2j})\in M_1}(\omega(i,j)) \leq \Sigma_{(f_{1i},f_{2j})\in S1\cup S2}(\omega(i,j))$, and (2) the denominator of $gVL_{\omega,\rho}$ is no larger than the denominator of $VL_{\omega,\rho}$: i.e., $m_1 + m_2 - |S1 \cup S2| \leq m_1 + m_2 - |M_1|$. Since the numerator is larger and the denominator is smaller, $gVL_{\omega,\rho}$ is at least as large as $VL_{\omega,\rho}$. (q.e.d)

Next, we propose two heuristics based approximations of the bipartite matching.

Definition 8 (Approximate Video Linkage) *Consider the bipartite graph G=\{N, E, Ω\} over two groups g_1=\{f_{11}, f_{12}, ..., f_{1m_1}\} and g_2=\{f_{21}, f_{12}, ..., f_{2m_2}\}. For two empty sets, $R1$ and $R2$,*

- *Sort all edges ($\in E$) by ω in decreasing order.*
- *For each edge e_{ij}=(f_{1i},f_{2j}) in order, if neither f_{1i} nor f_{2j} is visited, add e_{ij} into $R1$ and mark f_{1i},f_{2j} as "visited" (initially, all nodes are set as "unvisited").*
- *For each edge e_{ij}=(f_{1i},f_{2j}) in order, if e_{ij} does not "cross" any edges from $R2$, add e_{ij} into $R2$.*

Then, two approximate Video Linkage *measures are:*

$$aVL_{\omega,\rho}(g_1, g_2) = \frac{\Sigma_{(f_{1i},f_{2j})\in R1}(\omega(i,j))}{m_1 + m_2 - |R1|}$$

$$aNCVL_{\omega,\rho}(g_1, g_2) = \frac{\Sigma_{(f_{1i},f_{2j})\in R2}(\omega(i,j))}{m_1 + m_2 - |R2|}$$

such that $\omega(i,j) \geq \rho$, where ρ is given. □

Note that unlike $S1$ and $S2$, both $R1$ and $R2$ are still a matching since no nodes participate more than once. However, they may *not* be a maximum matching. Therefore, the following holds.

Lemma 4. *For two groups g_1 and g_2:*

$$aNCVL_{\omega,\rho}(g_1, g_2) \leq aVL_{\omega,\rho}(g_1, g_2) \leq VL_{\omega,\rho}(g_1, g_2)$$

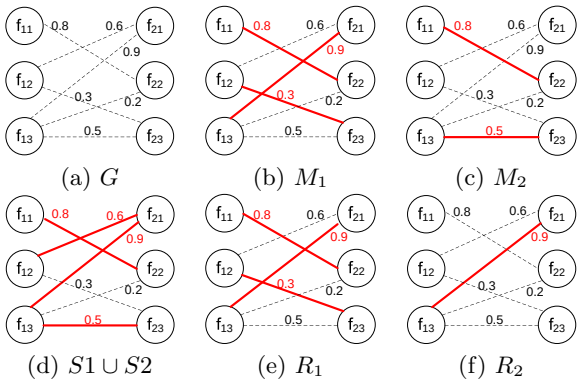
$$aNCVL_{\omega,\rho}(g_1, g_2) \leq NCVL_{\omega,\rho}(g_1, g_2)$$

where ρ is given. ■

PROOF. Omitted. (q.e.d)

Since the bipartite graph that we deal with is often very sparse (i.e., $N \gg E$) due to the early pruning from the constraint of $\omega(i,j) \geq \rho$, these two approximate video linkage measures can be computed faster than their corresponding exact video linkage measures.

Lemma 5. *On the bipartite graph G=\{N, E, Ω\} over two groups g_1 and g_2, both $aVL_{\omega,\rho}(g_1, g_2)$ and $aNCVL_{\omega,\rho}(g_1, g_2)$ can be computed in $O(E \log E)$ and $O(E^2)$ times, respectively.* ■



(a) G (b) M_1 (c) M_2

(d) $S1 \cup S2$ (e) R_1 (f) R_2

Figure 5: An illustration of Example 2.

The time complexities of five Video Linkage measures are summarized in Table 2.

Example 2. Consider a sparse bipartite graph G=\{N, E, Ω\} where Ω= \{$\omega(f_{11}, f_{22}) = 0.8$, $\omega(f_{12}, f_{21}) = 0.6$, $\omega(f_{12}, f_{23}) = 0.3$, $\omega(f_{13}, f_{21}) = 0.9$, $\omega(f_{13}, f_{22}) = 0.2$, $\omega(f_{13}, f_{23}) = 0.5$\}. Then, five video linkage measures are computed as follows:

- Since $M_1 = \{(f_{11}, f_{22}), (f_{12}, f_{23}), (f_{13}, f_{21})\}$, $VL = \frac{0.8+0.9+0.3}{3+3-3} = 0.67$.
- Since $M_2 = \{(f_{11}, f_{22}), (f_{13}, f_{23})\}$, $NCVL = \frac{0.8+0.4}{3+3-2} = 0.3$.
- Since $S1$=\{(f_{11}, f_{22}), (f_{12},f_{21}), (f_{13},f_{21})\}, $S2$ = \{(f_{13}, f_{21}), (f_{12}, f_{21}), (f_{13}, f_{23})\}, and $S1 \cup S2$ = \{(f_{11},f_{22}), (f_{12},f_{21}), (f_{13},f_{21}), (f_{13},f_{23})\}, $gVL = \frac{0.8+0.6+0.9+0.4}{3+3-4} = 1.35$.
- Since $R1 = \{(f_{13}, f_{21}), (f_{11}, f_{22}), (f_{12}, f_{23})\}$, $aVL = \frac{0.9+0.8+0.3}{3+3-3} = 0.67$.
- Since $R2 = \{(f_{13}, f_{21})\}$, $aNCVL = \frac{0.9}{3+3-1} = 0.18$.

Figure 5 illustrates the example. □

Lemma 6. *There is no bounding between $aVL_{\omega,\rho}(g_1, g_2)$ and $NCVL_{\omega,\rho}(g_1, g_2)$.* ■

PROOF. For two bipartite graphs G=\{N, E, Ω\} where Ω= \{$\omega(f_{11}, f_{21}) = 0.5$, $\omega(f_{11}, f_{22}) = 0.6$, $\omega(f_{12}, f_{22}) = 0.5$\}. Since $M_2 = \{(f_{11}, f_{21}), (f_{12}, f_{22})\}$, $NCVL = \frac{0.5+0.5}{2+2-2}$=0.5. However, since $R1 = \{(f_{11}, f_{21})\}$, $aVL = \frac{0.6}{2+2-1} = 0.2$. This leads to $aVL < NCVL$. However, Example 2 illustrates another case where $aVL > NCVL$ holds. (q.e.d)

Finally, combining Lemmas 1, 3, 4, and 6, we get the partial order among five Video Linkage measures.

Theorem 1. *The following partial order exists among five video linkage measures:*

$$aNCVL \leq NCVL, aVL \leq VL \leq gVL$$

That is, VL and $NCVL$ are bounded by $aNCVL$ (lower bound) and gVL (upper bound). ■

The advantage of these partial order is that both $gVL_{\omega,\rho}$ (g_1, g_2) and $aNCVL_{\omega,\rho}(g_1, g_2)$ can be computed faster than both $VL_{\omega,\rho}(g_1, g_2)$ and $NCVL_{\omega,\rho}$ (g_1, g_2), respectively. Therefore, quickly computing both $gVL_{\omega,\rho}(g_1, g_2)$ and $aNCVL_{\omega,\rho}$

Input : A query video v_q and a source video set V_s
Output : A copied video set V_c ($\subseteq V_s$)
linkage $\leftarrow$ {VL, $NCVL$, gVL, aVL, $aNCVL$};
$V_c \leftarrow \emptyset$;
foreach v_s ($\in V_s$) **do**
  if *linkage*$(v_q, v_s) \geq \theta$ **then** $V_c \leftarrow V_c \cup v_s$;
return V_c;

Algorithm 1: Standalone Video Linkage.

Input : A query video v_q and a source video set V_s
Output : A copied video set V_c ($\subseteq V_s$)
slow-linkage $\leftarrow$ {VL, $NCVL$};
$V_c \leftarrow \emptyset$;
foreach v_s ($\in V_s$) **do**
  if $gVL(v_q, v_s) < \theta$ **then** continue;
  if $aNCVL(v_q, v_s) \geq \theta$ **then** $V_c \leftarrow V_c \cup v_s$; continue;
  if *slow-linkage*$(v_q, v_s) \geq \theta$ **then** $V_c \leftarrow V_c \cup v_s$;
return V_c;

Algorithm 2: Pipelined Video Linkage.

(g_1, g_2) can help us efficiently address our video linkage problem. For instance, imagine that we want to check if two videos v_1 and v_2 have a similarity above θ or not. Then, since $gVL_{\omega,\rho}$ is an upper bound of $VL_{\omega,\rho}$, if $gVL_{\omega,\rho}(g_1, g_2) < \theta$, then it must be the case that $VL_{\omega,\rho}(g_1, g_2) < \theta$. Hence, (g_1, g_2) is guaranteed to *not* be part of the answer and can be pruned away. Reversely, using $aNCVL_{\omega,\rho}(g_1, g_2)$ as the lower bound of $VL_{\omega,\rho}$, if $aNCVL_{\omega,\rho}(g_1, g_2) \geq \theta$, then (g_1, g_2) is guaranteed to be part of the answer since $VL_{\omega,\rho}(g_1, g_2) \geq \theta$ is true.

Proposition 1. *To speed up computation, both $gVL_{\omega,\rho}$ (g_1, g_2) and $aNCVL_{\omega,\rho}(g_1, g_2)$ can be used in filtering for both $VL_{\omega,\rho}$ (g_1, g_2) and $NCVL_{\omega,\rho}(g_1, g_2)$, respectively. It is guaranteed that this filtering does not incur any false negatives.*

4.6 Putting All Pieces Together

Based on the findings in Section 4.4 and 4.5, in this section, we propose two different Video Linkage techniques – *Standalone* Video Linkage and *Pipelined* Video Linkage. Given two videos v_1 and v_2 that are passed through the filtering step in Section 4.3, suppose that we want to determine if their similarity is above θ or not. Then,

- *Standalone* Video Linkage algorithm computes one of five video linkage measures {VL, $NCVL$, gVL, aVL, $aNCVL$} to determine if $sim_v(v_1, v_2) \geq \theta$ or not.
- *Pipelined* Video Linkage algorithm exploits Proposition 1. That is, to determine if $sim_v(v_1, v_2) \geq \theta$ or not, we first check the fast (but approximate) greedy measure $gVL_{\omega,\rho}(v_1, v_2) < \theta$. If so, we conclude $sim_v(v_1, v_2) \not\geq \theta$. Else, we next check another fast (but approximate) measure $aNCVL_{\omega,\rho}(v_1, v_2) \geq \theta$. If so, we conclude $sim_v(v_1, v_2) \geq \theta$. Else, finally, we resort back to standalone Video Linkage and check $sim_v(v_1, v_2) \geq \theta$ using one of slow but exact video linkage measures {VL, $NCVL$} for $sim_v()$ function.

The details of two Video Linkage algorithms are illustrated in Algorithms 1 and 2, respectively.

5. EXPERIMENTAL VALIDATION

To validate our proposals, we have performed experiments with real videos downloaded from YouTube (with respect to the following aspects): (1) Whether the proposed Video Linkage measures are robust over various editing methods, genres, and thresholds, and (2) Whether the proposed Video Linkage measures provide enough performance to detect copied video in terms of speed, scalability, and accuracy. All proposed algorithms are implemented in Java and JMF 2.1e, and executed in a desktop with Intel Celeron 3.20GHz, 2GB RAM. To extract the features from **.flv* video format used in YouTube videos, Java wrapper for ffmpeg, i.e., fobs4jmf, is applied for our implementation.

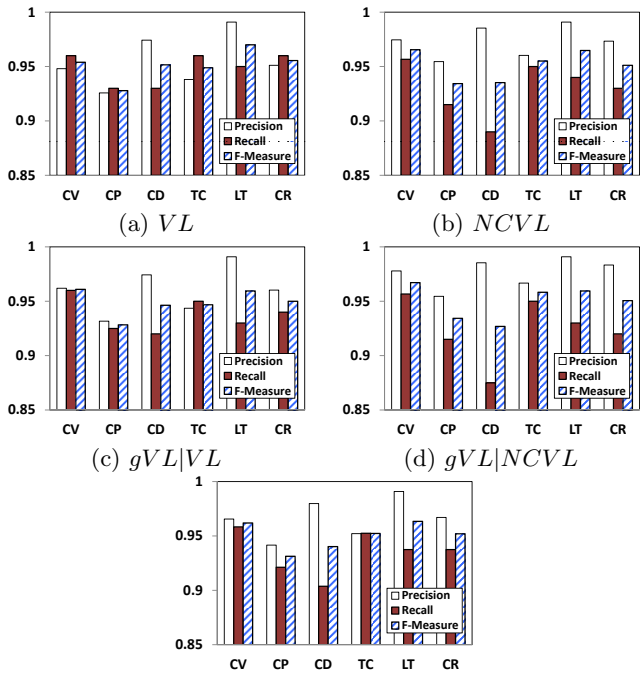
Genre	# of V_s	# of V_q	# of V_c	# of shots
AV	261	10	90	146,746
CO	284	10	90	131,713
ET	239	10	90	140,029
MU	230	10	90	160,278
HS	247	9	81	107,725
PB	440	10	90	270,115
NP	289	10	90	184,249
PA	394	9	81	126,097
TE	332	10	90	196,575
GA	216	10	90	117,653
	2,932	98	882	1,581,180

Table 3: Description of data set.

5.1 Set-Up

Data Sets. For a source video data set (V_s), we download 2,050 video clips from YouTube throughout all categories. Among the 12 categories in YouTube, 9 categories are selected: Autos & Vehicles (AV), Comedy (CO), Entertainment (ET), Howto & Style (HS), Music (MU), News & Politics (NP), People & Blogs (PB), Pets & Animals (PA), and Travel & Events (TE). In addition Game (GA) category is added for the purpose of diversity. In order to synthesize copied videos, we select 98 videos as original videos (V_q) from 10 genres. For each original video, 10 copied videos are generated by 'cut' in a video (3), 'cut and paste' in a video (2), 'cut and paste' from different videos (1), 'resize/resolution'(2), 'adding title and credit'(1), and 'adding logo/title' (1) ($(\cdot)$ indicates the number of copies). Therefore, 882 copied videos (V_c) are created in total. All copied videos are added to a source video set in the corresponding genre. As a result, we have 2,932 source videos for the data set. Since Video Linkage schemes use shots as a primary processing unit, each video in the data set is segmented into a number of shots using the shot boundary detection method mentioned in Section 4.2. Table 3 summarizes the statistics of the data sets. The second column (V_s) is the number of source videos, while the third and fourth columns are the number of original (V_q) and copied (V_c) videos, respectively. The last column indicates the total number of shots extracted from each genre.

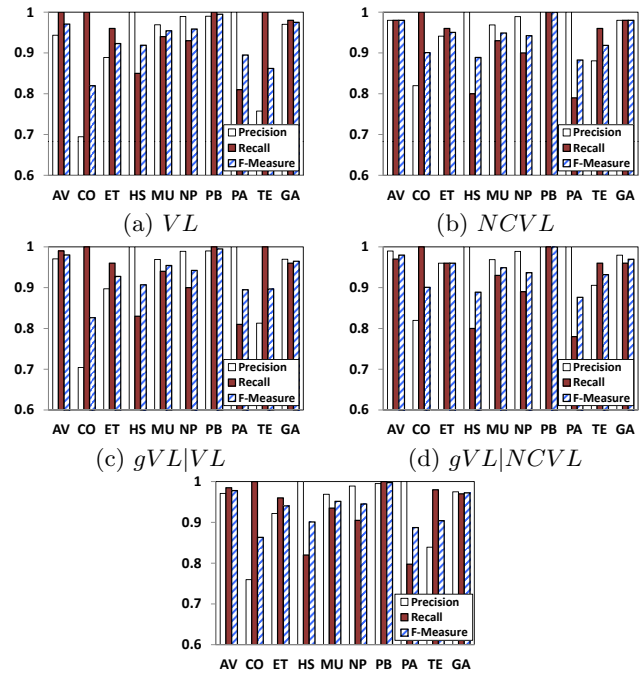
Schemes and Evaluation Metrics. Due to space constraint, in the following presentation, we will use two schemes from *Standalone* Video Linkage algorithm (i.e., VL and $NCVL$), and two others from *Pipelined* Video Linkage (i.e., $gVL|VL$ and $gVL|NCVL$). The notation "$gVL|VL$" means that two steps, gVL and VL, are applied in a row as in the pipeline. Two standpoints are considered as performance



(a) VL (b) $NCVL$ (c) $gVL|VL$ (d) $gVL|NCVL$

(e) Average performance of various editing methods

Figure 6: Performance of each scheme over 6 editing methods and average performance over all editing methods; cut in a video (CV), cut and paste in a video (CP), cut and paste from different videos (CD), adding title and credit (TC), adding logo/title (LT), and change resolution (CR).



(a) VL (b) $NCVL$ (c) $gVL|VL$ (d) $gVL|NCVL$

(e) Average performance of various genres

Figure 7: Performance of each scheme over 10 genres; Autos & Vehicles (AV), Comedy (CO), Entertainment (ET), Howto & Style (HS), Music (MU), News & Politics (NP), People & Blogs (PB), Pets & Animals (PA), Travel & Events (TE), and Game (GA).

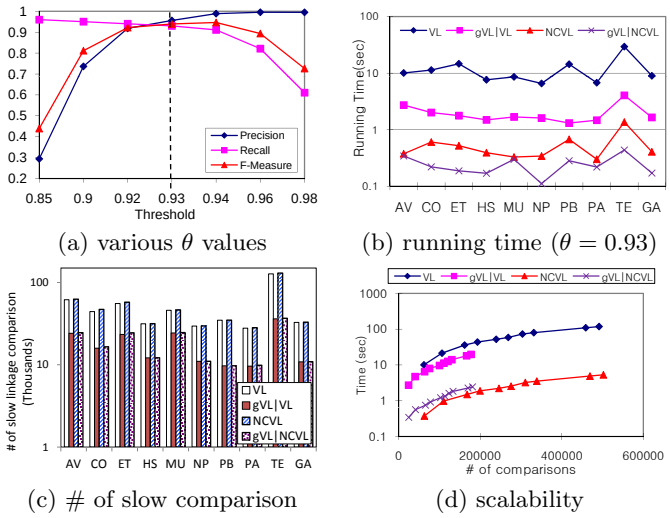
metrics: (1) **accuracy** in terms of precision ($\frac{N_{TruePositive}}{N_{AllPositive}}$), recall ($\frac{N_{TruePositive}}{N_{AllTrue}}$), and F-measure ($\frac{2\times(Precision\times Recall)}{Precision+Recall}$), and (2) **performance** in terms of *wall-clock running time*.

5.2 Robustness of Video Linkage

Editing Methods. As stated earlier, a copied video v_c in sites such as YouTube and Yahoo Video is typically created by altering the original video v_q using several basic editing methods. Therefore, a good measure for CVD should be robust against such editing methods including 'cut in a video (CV)', 'cut and paste in a video (CP)', 'cut and paste from different videos (CD)', 'adding title and credit (TC)', 'adding logo/title (LT)', and 'change resolution (CR)'. Figure 6 shows the performance of each scheme for 6 editing methods, and average performance over all editing methods. Figure 6(a)-(d) are the results of VL, $NCVL$, $gVL|VL$, and $gVL|NCVL$, respectively. In addition, Figure 6(e) shows the average performance of 3 Video Linkage measures. As shown in the figure, our proposed Video Linkage measures are robust over all editing methods, i.e., 0.93 in recall, 0.94 in precision and 0.935 in F-measure on average. Specifically, the high precision and recalls of Video Linkage measures indicate that our proposals go a good job in detecting copied videos accurately from data sets.

Genres. Since many existing methods of CVD are dependent on data domain, it is crucial for CVD to work with various genres of videos. In order to verify the robustness of Video Linkage measures against video contents, we test 4 measures on 10 genres data set mentioned in Table 3. Each genres has its unique characteristics of video contents. For example, a video in People & Blogs (PB) genre usually has very static images with relatively long shots, while that of Entertainment (ET) contains a lot of fast movements. Figure 7 shows the performance results of each scheme for 10 genres, and the overall average performance on all genres. Figure 7(a)-(d) are the results of VL, $NCVL$, $gVL|VL$, and $gVL|NCVL$, respectively. Also, the overall performance of Video Linkage can be found at Figure 7(e). As the graph shows, Video Linkage measures provide very consistent results over various genres in terms of recalls. However, the precisions and F-measure are relatively lower than recalls in some genres, such as Comedy (CO), Entertainment (ET), and Travel & Events (TE). This is caused by a lot of motion changes in a short time period. In order to increase the performance in terms of precision and F-measure in the specific genre, we can adjust the weight value of each feature (w_λ) to the contents. To keep the purpose of general solution of CVD, we use the same w_λ in this paper.

Thresholds. As mentioned in Section 2, we use the threshold version of the CVD problem. Therefore, we need an appropriate threshold value θ for all Video Linkage measures. However, predicting an optimal θ value is a challenging problem itself. To decide the threshold, we investigate the average recalls of all schemes over various values of θ, i.e., 0.85 to 0.98. Figure 8(a) shows the results of average recalls for $NCVL$. As observed in the figure, $NCVL$ has hight recall values with $\theta < 0.93$. Therefore, we used 0.93 as our default threshold value.



(a) various θ values (b) running time ($\theta = 0.93$)

(c) # of slow comparison (d) scalability

Figure 8: Various performance results.

5.3 Performance of Video Linkage

Next, we evaluate the performance of our Video Linkage methods with respect to the running time and # of comparison of Video Linkage schemes. Figure 8(b) first shows the running time of four methods, VL, $gVL|VL$, $NCVL$, and $gVL|NCVL$, over video sets in all genres. As expected, VL is the slowest method, regardless of genres, due to its high computational cost. Furthermore, the application of gVL as the "filter" in $gVL|VL$ speeds up the performance significantly (on average 5 times faster than without filter). In fact, $NCVL$ is faster than both VL and $gVL|VL$ because of different time complexities shown in Table 2, and $gVL|NCVL$ is the fastest among all 4 measures.

Figure 8(c) shows the performance of Video Linkage measures with respect to the number of computations. The number of computations during the copied video detection is the dominant component for the performance of overall CVD procedure. Thus, we count the number of computation for Video Linkage measure to evaluate the performance. We can observe that $gVL|VL$ requires the smallest number of computations during the CVD. Since it filters out a lot of non-similar videos, one can reduce the computation time. Figure 8(d) shows the scalability of Video Linkage measures over the number of shot comparisons. As expected, the total computational time of all Video Linkage measures are increasing monotonic over the number of shot comparisons. Specifically, $gVL|NCVL$ is the best algorithm in terms of scalability. That indicates the proposed scheme can be applied into a real life system to detect copied videos easily.

6. CONCLUSION AND FUTURE WORK

In this paper, we have presented the novel idea of group based copied video detection. In order to enable group based matching idea, we introduce a method to transform a video to a group of key frames. Once videos are captured as a group of frames, we propose five Video Linkage measures that are exhaustive, greedy, or approximate solution. Then, for efficient processing of Video Linkage solutions, non-similar videos are filtered out using a binary image signature. Finally, two algorithmic frameworks are proposed for the copied video detection problem: standalone and pipelined Video Linkage algorithms. Using videos downloaded from YouTube, our proposals are validated with respect of robustness and performance.

Video Linkage implementations and sample data sets used in this paper are available at:

`http://pike.psu.edu/download/civr08/`

7. REFERENCES

[1] I. P. Fellegi and A. B. Sunter. "A Theory for Record Linkage". *J. of the American Statistical Society*, 64:1183–1210, 1969.

[2] C. Gianluigi and S. Raimondo. "An Innovative Algorithm for Key frame Extraction in Video Summarization". *J Real-Time Image Proc*, 1:69–88, 2006.

[3] S. Guha, N. Koudas, A. Marathe, and D. Srivastava. "Merging the Results of Approximate Match Operations". In *VLDB*, pages 636–647, 2004.

[4] A. Hampapur, K.-H. Hyun, and R. Bolle. "Comparison of Sequence Matching Techniques for Video Copy Detection". In *SPIE, Storage and Retrieval for Media Databases*, pages 194–201, San Jose, CA, Jan 2002.

[5] M. A. Hernandez and S. J. Stolfo. "The Merge/Purge Problem for Large Databases". In *ACM SIGMOD*, 1995.

[6] A. Joly, O. Buisson, and C. Frelicot. Statistical similarity search applied to content-based video copy detection. In *ICDE Workshop*, page 1285, Tokyo, Japan, Apr 2005.

[7] R. P. Kelley. "Blocking Considerations for Record Linkage Under Conditions of Uncertainty". In *Proc. of Social Statistics Section*, pages 602–605, 1984.

[8] C. Kim and B. Vasudev. "Spatiotemporal Sequence Matching for Efficient Video Copy Detection". *IEEE TCSVT*, 15(1):127–132, 2005.

[9] J. Law-To, L. Chen, A. Joly, I. Laptev, O. Buisson, V. Gouet-Brunet, N. Boujemaa, and F. Stentiford. "Video copy detection: a comparative study". In *CIVR*, pages 371–378, Amsterdam, The Netherlands, July 2007.

[10] Y.-R. Lin, H.-Y. Huang, and W.-H. Hsu. "An Embedded Watermark Technique in Video for Copyright Protection". In *ICPR*, volume 4, pages 795–798, Hong Kong, China, Aug 2006.

[11] F. Malucelli, T. Ottmann, and D. Pretolani. "Efficient Labelling Algorithms for the Maximum Non Crossing Matching Problem". *Discrete Applied Mathematics*, 47(2):175–179, 1993.

[12] A. McCallum, K. Nigam, and L. H. Ungar. "Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching". In *ACM KDD*, Boston, MA, Aug. 2000.

[13] X. Naturel and P. Gros. "A Fast Shot Matching Strategy for Detecting Duplicate Sequences in a Television Stream". In *CVDB*, pages 21–27, Baltimore, MD, June 2005.

[14] B.-W. On, N. Koudas, D. Lee, and D. Srivastava. "Group Linkage". In *IEEE ICDE*, Istanbul, Turkey, Apr. 2007.

[15] S. Roy and E.-C. Chang. "Watermarking with Retrieval Systems". *Multimedia Systems*, 9(5):433–440, 2004.

[16] S. Sarawagi and A. Bhamidipaty. "Interactive Deduplication using Active Learning". In *ACM KDD*, 2002.

[17] H. T. Shen, X. Zhou, Z. Huang, J. Shao, and X. Zhou. "UQLIPS: A Real-time Near-duplicate Video Clip Detection System". In *VLDB*, pages 1374–1377, Vienna, Austria, Sep 2007.

[18] W. E. Winkler. "The State of Record Linkage and Current Research Problems". Technical report, US Bureau of the Census, Apr. 1999.

[19] D.-Q. Zhang and S.-F. Chang. "Detecting Image Near-Duplicate by Stochastic Attributed Relational Graph Matching with Learning". In *ACM Multimedia*, pages 877–884, New York, NY, USA, Oct 2004.

[20] J. Zobel and T. C. Hoad. "Detection of video sequences using compact signatures". *ACM TOIS*, 24(1):1–50, 2006.