Video Phylogeny: Recovering Near-Duplicate Video Relationships

Zanoni Dias, Anderson Rocha, and Siome Goldenstein

Institute of Computing, University of Campinas (UNICAMP)
Av. Albert Einstein, 1251, Cidade Universitária
13083-852, Campinas, SP – Brazil
{zanoni,rocha,siome}@ic.unicamp.br

***Abstract*—To keep pace with the increasing popularity of image and video sharing services, several research groups have focused on the development of systems to identify similar copies images and videos on the internet. Although these techniques allow us to identify the set of near-duplicates of a document, they do not give any information about the structure of generation of the near-duplicates. In this paper, we are interested in the history of the transformations that generated a given a set of near-duplicate videos. Given a set of near-duplicate videos, our objective is to identify their causal/phylogenetic relationships. Solutions to this problem have several applications such as in security, forensics, copyright enforcement, and news tracking services.**

I. Introduction

Images and video content can change as they spread out and such changes are not always authorized. After one posts a video on the internet, other users can copy, resize and/or re-encode it and repost their versions. This process creates similar, but not identical, copies. Video-sharing services, such as Youtube, are interested in this problem due to its correlation to copyright infringement and illegal content distribution.

With the increasing popularity of image and video sharing services, several research groups have presented solutions for the detection and recognition of near-duplicate (NDDR) images [1], [2] and videos [3], [4] in the last decade. NDDR techniques represent the first step for several applications such as reducing document versions, and for copyright and intellectual property protection.

Although NDDR techniques allow us to identify the set of near-duplicates of a document, they do not give any information about the dependency relationship or structure of generation of the near-duplicates. This challenging task arises when we want to identify which document is the original within a set of near-duplicates, and also the structure of generation of the near-duplicates.

Only recently there were initial attempts to identify the structure of relationships within a set of near-duplicates [5]–[7], however, limited to the context of images. In this paper, we are interested in the history of the transformations that generated a given set of near-duplicate videos. Solutions to this problem have applications in: **Security** (hints about the directions of content distribution); **Forensics** (better analysis when working with less-damaged documents); **Copyright enforcement** (traitor tracing without watermarking or fingerprinting methods); and **News tracking services** (analysis of the opinion forming process across time and space).

Given a set of near-duplicate videos, our objective is to to identify their causal relationships and transformations. We formerly dealt with this problem in the context of near-duplicate images and formally defined it as an Image Phylogeny problem. In this paper, we extend our prior work to deal with videos naming this problem 'Video Phylogeny'. The phylogeny tree tells the evolving history of the video. With a Video Phylogeny Tree (VPT) our objective is to find the structure of transformations, and possibly their parameters, representing the phylogenetic relationships within a given set of near-duplicate videos.

Watermarking and *fingerprinting* techniques [8] represent one way to approach this problem. However, their use is not always possible. Sometimes we can not assume any knowledge about the ownership of the source. In other cases, the transformations on the documents can destroy its markings.

To build a video phylogeny tree with respect to a set of near-duplicate videos, we need to find a robust and informative dissimilarity function to compare the video duplicates and also devise a proper algorithm capable of creating such a tree from the dissimilarities available to it. In this paper, we extend the method for calculating an asymmetric dissimilarity matrix from a set of near-duplicate images we introduced in [5] and introduce five approaches to find the phylogenetic relationships of the videos based on the calculated dissimilarities. The first four approaches calculate the final video phylogeny tree based on operations performed directly over the dissimilarity matrices obtained with the video frames. The fifth approach calculates the final video phylogeny tree through operations performed over the trees obtained with the video frames. To validate the approaches, we conduct experiments with Youtube videos and use the evaluation methodology introduced in [5].

II. Related Work

A duplicate is a pairwise equivalence relationship linking the original document to its variations through a transformation (e.g., cropping, compression, color correction) [9].

The near-duplicate detection and recognition problem can be approached by two different phylosophies: by *watermarking*-

WIFS'2011, November 29th-December 2nd, 2011, Foz do Iguaçu, Brazil.
978-1-4577-1019-3/11/$26.00

and *fingerprinting-based* and by *content-based* approaches. Watermarking and fingerprinting methods rely on the embedding of a signature within the original document before its dissemination [8] while content-based methods rely on the analysis of the document's content in order to extract relevant visual features. Regardless of the general philosophy, in the past decade we have seen some progress toward the development of effective systems to identify the cohabiting versions of images [1], [2] and videos [3], [4] in the wild. However, only recently there were the first attempts to go beyond NDDR with techniques to identify the structure of relationships within a set of near-duplicates [5]–[7].

Kennedy et al. [7] addressed the problem of parent-child relationships between images pairs. They proposed specialized detectors (e.g., scaling, cropping, color processing) to detect plausible parent-child relationships within a set of images. As the method relies only on directional processing information of image pairs, it is useful to point out possible candidates of either *highly-manipulated* or *original images*, but not appropriate to infer the phylogeny of a given set of near-duplicates.

Different from the Kennedy et al.'s approach [7], recently two research groups have concurrently focused on the discovery of the structure of dependencies within a set of images and on the underlying conditions of such dependencies [5], [6].

De Rosa et al. [6] aim at exploring image relationships based on the conditional probability of dependency between two images. For that, they use the images' content and their content-independent counterparts.

Since a real forensic case [10], our group have been investigating the use of graph theory approaches to identify the images' generating structure of modifications. In our previous work [5], we devised a method for calculating an asymmetric dissimilarity matrix from a set of near-duplicate images and introduced an algorithm to find the phylogenetic relationships of the images based on the calculated dissimilarities which we called Oriented Kruskal. This approach has been extended to deal with several other possible image transformations and scenarios where some pieces, or connections, are missing[1]. Different from all previous approaches, in this paper, we deal with the context of videos, as we present in Section IV.

III. Video Phylogeny

Extending the concepts presented in [5] to the context of videos, a *Video Phylogeny Tree* describes the structure of transformations and the evolution of near-duplicate videos.

Let $T_{\vec{\beta}}$ be a video transformation from a family $\mathcal{T}$. We define a dissimilarity function between two videos $\mathcal{V}_A$ and $\mathcal{V}_B$ as the minimum

$$d_{\mathcal{V}_A,\mathcal{V}_B} = \left|\mathcal{V}_B - T_{\vec{\beta}}(\mathcal{V}_A)\right|_{\text{comparison method } \mathcal{L},} \quad (1)$$

for all possible values of $\vec{\beta}$ that parameterizes $\mathcal{T}$. Eq. 1 measures the amount of residual between the best transformation of $\mathcal{V}_A$ to $\mathcal{V}_B$, according to the family of operations $\mathcal{T}$, and $\mathcal{V}_B$ itself. The comparison is made possible using a comparison method $\mathcal{L}$ in the end which can be employed frame-wise.

[1]Such an extension is now under consideration of a journal.

IV. Our approach for Video Phylogeny

To reconstruct the VPT from a set of near-duplicate videos, we need to consider the tree-building algorithm and the dissimilarity function.

A. Overview of the Dissimilarity Matrix Construction

Our first task to build a VPT of n near-duplicate videos, is to calculate the dissimilarity between every pair of such videos. We need to consider a good set of possible videos transformations, $\mathcal{T}$, from which one video can generate a descendant. A video can undergo countless possible transformations to create a near-duplicate of itself. The family of video transformations $\mathcal{T}$ that we consider in this paper consists of:

1) **Resampling**: the video can be re-sampled (up or down);
2) **Cropping:** the video can be cropped;
3) **Brightness, Contrast, and Gamma**: the video pixels can be color-adjusted through brightness and contrast operations;

The operations are equally performed across the video frames. All the videos are compressed using the standard H.264 compression algorithm.

Given any pair of videos $\mathcal{V}_A$ and $\mathcal{V}_B$, our first task consists of estimating the possible values of $\vec{\beta}$ that parameterizes $\mathcal{T}$ that best approximates $\mathcal{V}_A$ onto $\mathcal{V}_B$ through the dissimilarity function $|\mathcal{V}_B - T_{\vec{\beta}}(\mathcal{V}_A)|_{\mathcal{L}}$ (Eq. 1), according to the chosen comparison method $\mathcal{L}$.

To estimate the $\vec{\beta}$ that approximates video $\mathcal{V}_A$ onto $\mathcal{V}_B$, we:

1) Select f frames from both videos assuming they are temporal coherent.
2) Calculate corresponding points between video frames $\mathcal{V}_A$ and $\mathcal{V}_B$ using the SURF algorithm [11]. Each frame $f_A^i \in \mathcal{V}_A$ is directly compared to frame $f_B^i \in \mathcal{V}_B$.
3) Robustly estimate the affine warping transformation parameters which includes color correction, cropping, and resampling operations for video $\mathcal{V}_A$ with respect to $\mathcal{V}_B$ taking the corresponding points into consideration and using Random Sampling Consensus (RANSAC) algorithm [12] for each pair of frames. The color channels of a frame $f_A^i \in \mathcal{V}_A$ are normalized according to the mean and variance of the corresponding frame $f_B^i \in \mathcal{V}_B$.
4) Calculate the dissimilarity between both videos considering frame point-wise dissimilarity using the standard *Minimum Squared Error* (MSE) as the technique.

Step 1 gives candidate points to estimate the resampling and cropping operations for frames $f_A^i \in \mathcal{V}_A$ with respect to the frames $f_B^i \in \mathcal{V}_B$. However, these points are likely unstable and we need to robustly filter them in Step 2. At this point, we are able to resample and crop frame $f_A^i \in \mathcal{V}_A$ with an affine warp robustly estimated with RANSAC and the local feature descriptors calculated in Step 1. In Step 3, we perform pixel intensity normalization of frame $f_A^i \in \mathcal{V}_A$ according to the f_B^i color channels' mean and variance. Considering all possible pairs of videos in a near-duplicate set, we end up with f dissimilarity matrices where each entry represents the dissimilarity of $\mathcal{V}_A$ and $\mathcal{V}_B$ for a given pair of frames $f_A^i \in \mathcal{V}_A$ and $f_B^i \in \mathcal{V}_B$. The dissimilarity calculation for each pair of

frames is the same as introduced in [5]. For the estimation of the dissimilarity matrix, we use *OpenCV*[2] algorithms.

Our task is to find the final VPT based on the f frame-wise calculated dissimilarity matrices for n near-duplicate videos.

B. Approaches for Building the Final Video Phylogeny Tree

In Section IV-A, we explained how to create f dissimilarity matrices with respect to the selected f frames of each of the n near-duplicate videos taking into consideration a family of transformations. Before introducing five approaches to produce the final video phylogeny tree, we would like to recall how to create a phylogeny tree for a single set of n images [5].

Given a dissimilarity matrix M built upon a set of n near-duplicate frames (f_a^b where $a \in [1 \dots f]$ is the a-th selected frame of video $\mathcal{V}_b$ where $b \in [1 \dots n]$), the Oriented Kruskal algorithm [5] starts with a forest in which each node (near-duplicate frame) is the root of a tree with just one element. Next, the algorithm sorts all positions (i, j) of M from the lowest to the highest dissimilarity values. The algorithm then analyzes each position (i, j) according to the sorted order, joining different trees and checking whether or not the endpoints of the analyzed position is a root. The algorithm stops when there are $n - 1$ edges in the tree.

Fig. 1 depicts the execution of the algorithm introduced in [5] for a toy example with $n = 6$ near-duplicate images. The first step of the algorithm initializes a forest with $n = 6$ roots and sorts all the positions (i, j) in the dissimilarity matrix M. The algorithm starts the construction of the IPT taking each position (i, j) at a time and performing the required tests to ensure it can safely insert the tested position as an oriented edge $(j \rightarrow i)$ in the final answer. The algorithm first selects the position $(6, 1)$ with the lowest dissimilarity value $M[6, 1] = 12$. Since this position connects two disjoint trees (Test I) and the endpoint 1 is a root (Test II), it is selected. The same happens with the position (4,5). The algorithm then tests position (4,1) with dissimilarity 16. Since the endpoint 1 is not a root (it belongs to a tree with root 6), it is discarded. The algorithm continues until Step 8, where it tests the position (4,6) with dissimilarity 27.

Single-Frame Expectation. With f dissimilarity matrices of $n \times n$ entries for a set of n near-duplicate videos, we wonder what to expect about the video phylogeny reconstruction process. To answer, we can calculate an image phylogeny tree for each frame using Oriental Kruskal as explained above and illustrated in Fig. 1. With f image phylogeny trees, we calculate the evaluation metrics as devised in [5]. With this approach we have some sort of expectation of the final video phylogeny results but we do not have the actual final video phylogeny tree. This measure also refers to what we would expect to obtain when analyzing a single random frame from the videos.

Minimum and Average VPT. Two initial approaches to calculate the final video phylogeny tree from f dissimilarity matrices consist on performing operations directly on the matrices. The first approach, called `min`, consists of creating a dissimilarity matrix M' where each entry m_{ij} is the minimum value across the corresponding entries in all f matrices. The second approach, called `avg`, consists of creating a dissimilarity matrix M' where each entry m_{ij} is the average value across the corresponding entries in all f matrices. Once we create the derived dissimilarity matrix, we create the final VPT using Oriented Kruskal as introduced in [5].

Normalized Minimum and Average VPT. Before calculating the final VPT using `min` or `avg` approach, we can normalize all the matrices to the 0-1 interval and be more robust to avoid the result of one dissimilarity matrix having a high weight in the total computed value. For instance, considering `min` without normalization, if we have one frame with no content (e.g., black frame), its dissimilarity matrix will have small entries which will be selected even if the majority of the other dissimilarity matrices contain more compatible entries. Considering `avg` without normalization, high values on a matrix will highly impact the average calculations.

We can improve `min` and `avg` by first normalizing each dissimilarity matrix with respect to its highest value. With all dissimilarity matrices in the 0-1 interval, we can calculate `min` and `avg` as before. We call these approaches `min-norm` and `avg-norm`. With the derived dissimilarity matrix, we create the final VPT using Oriented Kruskal as introduced in [5].

Tree Reconciliation VPT. Our final approach to construct the video phylogeny tree for a set of n near-duplicate videos consists of dealing with the f frame phylogeny trees directly instead of performing operations on the dissimilarity matrices. For that, we introduce an algorithm called `tree reconciliation`. The idea is to take the f trees and reconcile them toward the final VPT. The algorithm builds the final VPT by adding, at each step, the most used edges across the f individual trees.

The first step consists of creating the f frame phylogeny trees using Oriented Kruskal algorithm [5]. Then we create a matrix upon which we will create the final VPT. In this matrix, each entry represents the number of times a node i is identified as a child of j across the f phylogeny trees.

Algorithm 1 presents the process for creating the parenthood matrix P. The algorithm receives three parameters: the number of near-duplicate videos n, the number of selected frames f, and a vector of trees, t, with the f phylogeny trees previously calculated. Following [5], we represent a tree as in `[1, 1, 2]` which refers to a tree with $n = 3$ vertices such that vertex 1 is the root of the tree and also the parent of vertex 2, which in turn, is the parent of vertex 3.

The position `t[i]` represents the tree for the i-th selected frame, `t[i][j]` represents j's parent in tree i. After initialization (Lines 1–5), the algorithm computes the number of times a node i is identified as a child of j across the f trees (Lines 6–10). This algorithm has complexity $O(n(n + f))$ given the initialization costs $O(n^2)$ and the cost to fill matrix P depends on traversing f trees of size n nodes or $O(nf)$.

With the matrix P, we can create the final VPT using Algorithm 2. The algorithm consists of starting with n disconnected nodes and incrementally connecting them until we have the final VPT. However, we connect the nodes in order of importance according to matrix P. The algorithm receives

[2]http://sourceforge.net/projects/opencvlibrary/

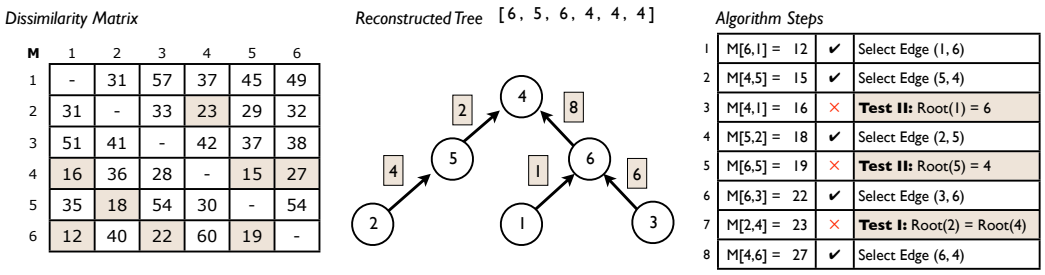

Figure 1. Simulation of the Oriented Kruskal algorithm to construct an Image Phylogeny Tree from a Dissimilarity Matrix as presented in [5].

Algorithm 1 Reconciliation Matrix.

Require: number of near-duplicate videos, n
Require: number of selected frames, f
Require: 2-d vector, t, with the f phylogeny trees previously calculated
1: **for** $i \in [1..n]$ **do** ▷ Initialization
2: **for** $j \in [1..n]$ **do**
3: $P[i, j] \leftarrow 0$
4: **end for**
5: **end for**
6: **for** $i \in [1..f]$ **do** ▷ Creating the matrix P
7: **for** $j \in [1..n]$ **do**
8: $P[j, t[i][j]] = P[j, t[i][j]] + 1$
9: **end for**
10: **end for**
11: **return** P ▷ Returning the parenthood matrix P

Algorithm 2 Tree Reconciliation.

Require: number of near-duplicate videos, n
Require: matrix, P, from Algorithm 1
1: **for** $i \in [1..n]$ **do** ▷ Tree initialization
2: $tree[i] \leftarrow i$
3: **end for**
4: $sorted \leftarrow$ sort positions (i, j) of P into nonincreasing order
▷ List of edges sorted from the most to the least common
5: $r \leftarrow 0$ ▷ Initially, the final root r is not defined
6: $n_{edges} \leftarrow 0$
7: **for each** position $(i, j) \in sorted$ **do** ▷ Testing each edge in order
8: **if** $r = 0$ and $i = j$ **then** ▷ Defining the root of the tree
9: $r \leftarrow i$
10: **end if**
11: **if** $i \neq r$ **then** ▷ If i is not the root of the tree
12: **if** $\texttt{Root}(i) \neq \texttt{Root}(j)$ **then**
13: **if** $\texttt{Root}(j) = j$ **then**
14: $tree[j] \leftarrow i$
15: $n_{edges} \leftarrow n_{edges} + 1$
16: **if** $n_{edges} = n - 1$ **then** ▷ If the tree is complete
17: **return** $tree$ ▷ Returning the final VPT
18: **end if**
19: **end if**
20: **end if**
21: **end if**
22: **end for**

two parameters as input: the number of near-duplicate videos n and the $n \times n$ matrix P computed using the Algorithm 1.

Lines 1–3 initialize the final VPT. Line 4 sorts the entries (edges) in P from the most to the least common. Line 5–6 initialize the root and the number of edges counter. Lines 7–22 tests each edge (i, j) in order to insert it into the tree.

The running time depends on how we implement the `Root` function whose role is to determine the root of a given tree or sub-tree. If we use a *disjoint-set-forest* with the *union-by-rank* and *path-compression heuristics*, we can implement such a function efficiently [13]. Using such implementation, the final complexity of the algorithm is $O(n^2 \log n)$.

Running Example. The VPT algorithms `min`, `avg`, `min-norm`, and `avg-norm` perform operations over the $n \times n$ f dissimilarity matrices available for n near-duplicate video and, in the end, construct the VPT using the Oriented Kruskal algorithm as illustrated in Fig. 1. On the other hand, the `tree reconciliation` algorithm constructs the final VPT based on the f frame phylogeny trees.

Fig. 2 shows a toy example example for $n = 6$ near-duplicate videos and $f = 6$ selected frames for each video. Using Oriental Kruskal algorithm as introduced in [5] and illustrated in Fig. 1, we create one tree per selected frame. Using Algorithm 1, we create the parenthood matrix P for the six trees. Recall that the entries in P represent the number of times a node i appears as a child in all trees. For instance, $M[1, 1] = 4$ because the node 1 appears four times as a child of itself (root of the tree) in P. $M[5, 3] = 2$ because node 5 appears as a child of node 3 twice.

Using the parenthood matrix P, we use Algorithm 2 to create the final VPT. The first step consists of sorting the edges in P in non-increasing order. Then the algorithm proceeds selecting one edge at a time and testing whether or not it is safe to put it into the final VPT. We first select entry $P[1, 1]$ as it is the one with the highest value. As this entry means an edge of node 1 to itself, it is the root of the VPT. In the second step, we select edge (2,1) which means node 2 is now a child of node 1. The procedure continues until we select the edge (1,4) which is discarded since it would create a loop in the tree. The next edges (2,4), (3,2), (4,1), and (5,3) are reject as well. Finally, we select the edge (6,1) which completes the tree ($n - 1$ edges).

V. Experiments and Methods

To validate the techniques we propose in this paper for reconstructing the video phylogeny tree from n near-duplicate videos, we create a data set comprising several transformations that a video can undergo to generate a near-duplicate. We analyze the results using several quantitative measures of success as introduced to the image phylogeny case in [5].

A. Evaluation Metrics

We use four quantitative metrics (*Root*, *Edges*, *Leaves*, and *Ancestry*) to evaluate a reconstructed tree in scenarios where

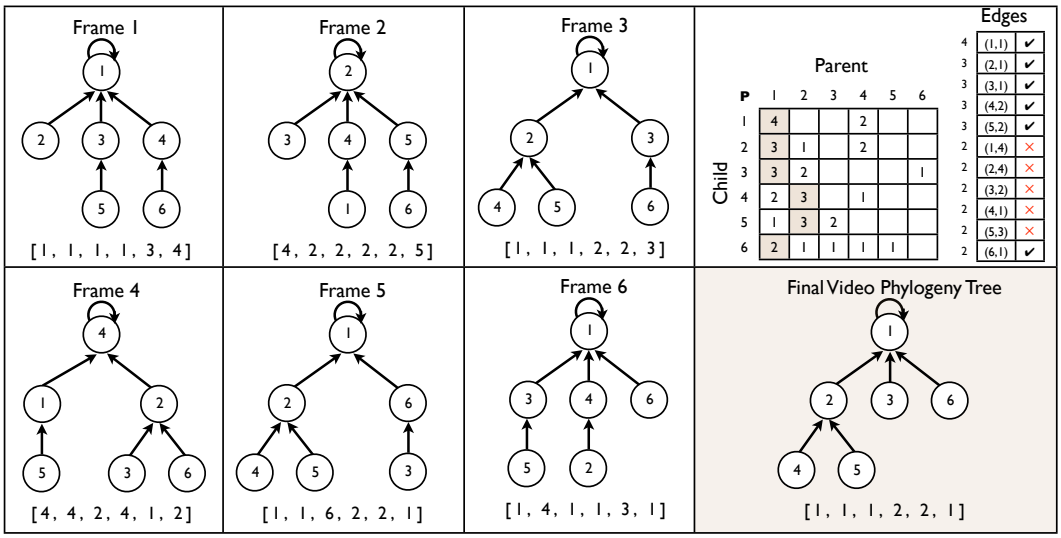

Figure 2. Simulation of the Tree Reconciliation algorithm (TR) to construct a Video Phylogeny Tree from $f = 6$ image (or frame) phylogeny trees.

we have Ground Truth as introduced in [5]. In this case, we want to compare a reconstructed VPT with its ground truth tree. For this intent, we use the metrics from [5]:

Root: $R(\text{VPT}_1, \text{VPT}_2) = \begin{cases} 1, & \text{If } \texttt{Root(VPT}_1\texttt{)} = \texttt{Root(VPT}_2\texttt{)} \\ 0, & \text{Otherwise} \end{cases}$

Edges: $E(\text{VPT}_1, \text{VPT}_2) = \frac{|E_1 \cap E_2|}{n-1}$

Leaves: $L(\text{VPT}_1, \text{VPT}_2) = \frac{|L_1 \cap L_2|}{|L_1 \cup L_2|}$

Ancestry: $A(\text{VPT}_1, \text{VPT}_2) = \frac{|A_1 \cap A_2|}{|A_1 \cup A_2|}$

The `Root` metric evaluates if we find the correct root of the VPT while the `Edges` metric evaluates the amount of correct connections (edges) we find in the VPT. The `Leaves` metric accounts for the number of correct leaves we find and the `Ancestry` metric accounts for the portion of correctly identified ancestry information across the tree. E_i, L_i, and A_i mean the sets of edges, leaves, and ancestry in a tree i, respectively.

Another form of evaluating the effectiveness of a VPT reconstruction approach is to assess the average `Depth` of the tree in which it finds the correct root (the lower the average depth the better). If an algorithm finds the correct root at depth zero, it means it correctly identified the root of the tree.

As in any designed experiment, we can only calculate the metrics if we do have the real *Ground Truth* to compare the estimated result. Section V-B shows the methodological approach used for obtaining this set of controlled experiments.

B. Data Set

To generate a near-duplicate a video can undergo several possible transformations. However, these transformation must not destroy the overall meaning of the video otherwise it would not be considered a near-duplicate. In this paper, we select typical transformations a video can undergo such as: different scales for horizontal and vertical axis resampling, contrast and brightness adjustment, non-linear gamma correction, and cropping. Here, we do not consider temporal cropping. Tab. I shows the transformations and their operational ranges for creating the data set.

Table I
TRANSFORMATIONS AND THEIR OPERATIONAL RANGES FOR CREATING THE CONTROLLED DATA SET.

Transformation	Oper. Range
(1) Global Resampling/Scaling (Up/Down)	$[90\%, 110\%]$
(2) Scaling by axis	$[90\%, 110\%]$
(3) Cropping	$[0\%, 5\%]$
(4) Brightness Adjustment	$[-10\%, 10\%]$
(5) Contrast Adjustment	$[-10\%, 10\%]$
(6) Gamma Correction	$[0.9, 1.1]$

The transformations can be combined in any form to create a near-duplicate. In addition, the color transformations can be performed either linearly or non-linearly across the color channels. All the near-duplicate generation process is performed using the algorithms implemented in MEncoder library[3].

To create the data set, we selected 16 of the most watched video commercials for the 2011 Super Bowl[4]. All the videos were originally in HD resolution and each video contains, at least, 30 seconds of content.

For each video, we have created 16 near-duplicate trees of size 10 (the original and nine near-duplicate videos). The final data set contains 256 test cases. In all the experiments, we selected one frame per second to create the VPT. We also considered two and three frames per second but one frame per second represents the best efficiency/effectiveness tradeoff.

C. Experiments

In this section, we show the experiments for the different methods we introduce for reconstructing a video phylogeny tree from n near-duplicate videos.

Tab. II first shows the results for the minimum expectation we have when building the video phylogeny trees.

[3]http://www.mplayerhq.hu/

[4]http://www.ic.unicamp.br/~rocha/pub/wifs-2011-super-bowl-videos.html

For that, we calculate an image phylogeny tree for each of the f selected frames using Oriental Kruskal [5]. We then compute the metrics as discussed in Section V-A. The table also shows the results of the proposed methods based on operations over dissimilarity matrices `min`, `avg`, `min-norm`, and `avg-norm`. Finally, the table shows the results for the `tree reconciliation` method which creates the final VPT by reconciling f phylogeny trees.

At first, we notice the `min-based` methods are indeed worse than the general expectation for the reconstruction. The metrics show that `single frame expectation` results generally are better than `min` and `min-norm` results. The `avg-based` methods perform better than the single frame expectation and finds the correct root of the tree in about 85% of the cases and have an average root depth of 0.203. The `avg-based` methods find 62-64% of the correct ancestry information. As expected, the normalization plays an important role in the VPT reconstruction improving the quality of the results for the `min` and `avg` methods.

The `tree reconciliation` algorithm obtains the best-performing results. It finds the correct root in 91.0% of the test cases which is an improvement of 18.9% over the single-frame expectation or 5.9% over the `avg-norm` method. However, the average root depth for the `tree reconciliation` algorithm is only 0.098 or 74.3% more accurate than the `single frame expectation`. This result means that if the method does not correctly find the root it only misses it by a small fraction. Creating the final VPT from reconciling the actual phylogeny trees is more advantageous than performing operations of the dissimilarity matrices.

Tab. III presents the breakdown results for the `tree reconciliation` algorithm per video. Each row represents the average result of 16 different trees over near-duplicate sets with the considered video in that row. We see that some videos are more difficult than others. However, the low standard deviations suggest that the method is reasonably stable across different videos.

Table II
AVERAGE RESULTS FOR THE 16×16 TEST CASES UNDER CONSIDERATION FOR THE PROPOSED VPT METHODS.

Method	Root	Depth	Edges	Leaves	Ancestry
(E) Single Frame	76.5%	0.382	54.2%	67.7%	58.6%
(1) Min	59.0%	0.926	49.6%	64.1%	50.8%
(2) Min-Norm	68.0%	0.605	51.3%	66.4%	54.2%
(3) Avg	85.6%	0.215	56.6%	70.3%	62.0%
(4) Avg-Norm	85.9%	0.203	58.0%	72.4%	64.5%
(5) Reconc. Tree	91.0%	0.098	65.8%	77.7%	70.4%
(5)/(E) Boost	**18.9%**	**74.3%**	**21.4%**	**14.7%**	**20.1%**

VI. CONCLUSIONS

In this paper, we tackled the problem of identifying the video relationships within a set of near-duplicate videos. The solution extends our prior work for image phylogeny [5].

We presented five different ways of calculating final video phylogeny from a set of selected frames for the n near-duplicate videos of interest. The methods differ in their nature of calculating the final VPT. Four of them are based on operations on the available dissimilarity matrices while the fifth and best-performing one builds the final VPT based on reconciling the actual frame phylogeny trees available.

Table III
RESULTS FOR THE TREE RECONCILIATION APPROACH USING 16 DIFFERENT TREES.

Video	Root	Depth	Edges	Leaves	Ancestry
V_{01}	100.0%	0.000	68.1%	77.9%	73.4%
V_{02}	87.5%	0.125	66.9%	76.0%	68.8%
V_{03}	75.0%	0.312	56.9%	73.7%	57.8%
V_{04}	81.2%	0.188	57.5%	68.2%	60.7%
V_{05}	93.8%	0.062	69.4%	81.3%	73.9%
V_{06}	93.8%	0.125	66.2%	77.7%	72.7%
V_{07}	100.0%	0.000	73.1%	83.2%	79.5%
V_{08}	93.8%	0.062	59.4%	75.0%	66.1%
V_{09}	100.0%	0.000	70.6%	80.2%	73.1%
V_{10}	100.0%	0.000	65.6%	75.9%	72.3%
V_{11}	81.2%	0.188	64.4%	80.0%	69.8%
V_{12}	100.0%	0.000	68.7%	80.2%	76.4%
V_{13}	87.5%	0.125	75.0%	82.5%	77.7%
V_{14}	100.0%	0.000	69.4%	78.1%	72.5%
V_{15}	81.2%	0.188	56.9%	72.8%	64.2%
V_{16}	81.2%	0.188	65.0%	80.2%	67.2%
Average	91.0%	0.098	65.8%	77.7%	70.4%
Std Dev	8.8%	0.097	5.6%	4.0%	6.0%

To create a realistic set of experiments, we accounted for common transformations a video undergoes when generating an offspring, namely resampling, cropping, and channel-wise pixel normalization. We validated the approaches on a controlled environment with 16 videos from the 2011 Super Bowl commercials.

Our future work includes the research for approaches to deal with temporal cropping, missing links [5] as well as multiple trees in the set of n near-duplicate videos.

ACKNOWLEDGMENT

We thank FAPESP, CNPq, and Microsoft for the financial support.

REFERENCES

[1] Y. Maret, "Efficient Duplicate Detection Based on Image Analysis," PhD Thesis, EPFL, Lausanne, Switzerland, 2007.
[2] H. Kim, H.-W. Chang, J. Lee, and D. Lee, "BASIL: Effective Near-Duplicate Image Detection using Gene Sequence Alignment," in *ECIR*. Springer, 2010, pp. 229–240.
[3] A. Jaimes, S.-F. Chang, and A. Loui, "Duplicate detection in consumer photography and news video," in *ACMMM*, 2002, pp. 423–424.
[4] Z. Huang, H. T. Shen, J. Shao, B. Cui, and X. Zhou, "Practical online near-duplicate subsequence detection for continuous video streams," *IEEE TMM*, vol. 12, no. 5, pp. 386–398, August 2010.
[5] Z. Dias, A. Rocha, and S. Goldenstein, "First Steps Toward Image Phylogeny," in *IEEE WIFS*. IEEE, 2010, pp. 1–6.
[6] A. D. Rosa, F. Uccheddua, A. Costanzo, A. Piva, and M. Barni, "Exploring Image Dependencies: a New Challenge in Image Forensics," in *Media Forensics and Security II*. SPIE, 2010, pp. X1–X12.
[7] L. Kennedy and S.-F. Chang, "Internet Image Archaeology: Automatically Tracing the Manipulation History of Photographs on the Web," in *ACMMM*. ACM, 2008, pp. 349–358.
[8] I. Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker, *Digital Watermarking and Steganograph*, 2nd ed. Morgan Kaufmann, 2007.
[9] A. Joly, O. Buisson, and C. Frélicot, "Content-Based Copy Retrieval Using Distortion-Based Probabilistic Similarity Search," *IEEE TMM*, vol. 9, no. 2, pp. 293–306, 2007.
[10] S. Goldenstein and A. Rocha, "High-Profile Forensic Analysis of Images," in *ICDP*, 2009, pp. 1–6.
[11] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Elsevier CVIU*, vol. 110, no. 3, pp. 346–359, 2008.
[12] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," in *Comm. of the ACM*, vol. 24(6), 1981, pp. 381–395.
[13] R. E. Tarjan, "Efficiency of a good but not linear set union algorithm," *ACM JACM*, vol. 22, no. 2, pp. 215–225, 1975.