IMAGE PHYLOGENY THROUGH DISSIMILARITY METRICS FUSION

A. Melloni[♯], P. Bestagini[♯], S. Milani[♯], M. Tagliasacchi[♯], A. Rocha[♭], S. Tubaro[♯]

[♯]Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133, Milano, Italy
[♭] Institute of Computing, University of Campinas (UNICAMP)
Av. Albert Einstein, 1251, Cidade Universitária, 13083-852, Campinas, SP - Brazil

ABSTRACT

Nowadays, multimedia objects can be easily modified, shared, and distributed, thus determining the widespread diffusion of multiple near-duplicate versions, i.e., objects obtained applying a set of processing operations to original content. This is the case of images downloaded from sharing platforms, modified (e.g., by performing color correction, splicing, etc.) and re-distributed. The evolution of a group of near-duplicate images (i.e., their phylogeny) is a powerful clue to determine both image authenticity and its origin. For this reason, the forensics community has proposed a set of possible solutions to perform phylogenetic analyses based on image dissimilarity computation. Here, we compare different image dissimilarity metrics, and propose a set of original strategies for image phylogeny tree reconstruction. The validation of the proposed methods is performed on a dataset of image phylogeny trees. Depending on the used evaluation metrics, some approaches are preferable to others according to the results. Hence, an analyst can choose the appropriate method according to its needs.

Index Terms— Image phylogeny, near-duplicates, image dissimilarity, phylogeny tree.

1. INTRODUCTION

With the widespread diffusion of cheap image capturing devices (e.g., compact cameras, smartphones, etc.), image sharing platforms such as Picasa and Flickr, are gaining the trend. Anyway, a significant part of the uploaded content is created by reusing already existing images. As an example, it is customary to download a picture, modify it (e.g., by performing color correction, object insertion, resize, etc.), and re-upload it to a website. Such duplicate versions are typically referred to as near-duplicates, since they are not perfect copies, instead they are generated applying a set of processing steps to an original object.

The project REWIND acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number:268478.

Such near-duplicate images may be created either maliciously or not, and their diffusion may have serious social consequences. In other words, a user may choose to modify a popular photo, such as the portrait of a famous person, or related to an important event, to simply create a funny picture to share with friends, and gain some clicks on his blog. However, newspapers may end up publishing such forged images (e.g., related to critical events), with the far more serious consequence of conveying biased information.

In order to prevent similar situations, the multimedia forensics community has proposed over the years a set of tools to blindly uncover the past history of images [1], videos [2] and audio [3]. These tools typically exploit specific traces left by forgery operations, to detect the use of a particular kind of manipulation (e.g., image resizing [4], multiple image compression [5], object insertion in videos [6], audio recapture [7], etc.).

However, the aforementioned methods do not take into account the possibility of comparing digital objects within a set of available near-duplicates, which is a common scenario when dealing with user-generated content. In order to take advantage of this possibility, the forensics community has then started to study techniques to understand and reconstruct the evolution of groups of near-duplicate objects. These include a few algorithms to deal with images [8, 9, 10], audio [11], and video [12, 13]. Thanks to these methods, it is possible to infer the causal relationship between near-duplicate objects in a set, to understand which one has been used to generate the others, eventually identifying the original object at the root of the phylogeny.

Focusing on still images, the proposed approaches typically tackle the problem in two steps. The first, consists in evaluating the dissimilarity between every pair of images in a set. The second, in using dissimilarity measurements obtained at the previous step to reconstruct the phylogeny tree, i.e., to establish causal relationships between pairs of similar images, in order to infer which image has possibly been used to generate the others. State-of-the-art methods differ in the image dissimilarity metrics, and tree reconstruction algorithms [10, 14, 15].

Fig. 1: Example of near-duplicate images found on the Web.

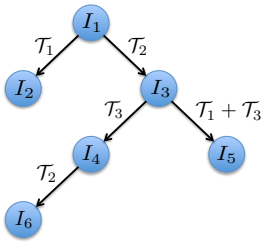

Fig. 2: Example of IPT. Each node represents an image, each branch a transformation, or a combination of transformations. The same transformation can be applied several times.

In this work, we perform an analysis of different Image Phylogeny Tree (IPT) reconstruction algorithms. More specifically, we compare a well known state-of-the-art image dissimilarity metric [14], and a novel modified version based on an image de-noising step, inspired by the work in [10]. Then, we propose tree reconstruction algorithms that exploit the aforementioned dissimilarity computation, and compare the achieved results with the algorithm proposed in [14]. We finally show that it is possible to merge dissimilarity measurements in order to reconstruct IPTs with different proposed techniques.

The validation of the analyzed and proposed methods is performed on a dataset of 1,000 near-duplicate trees, containing up to 20 nodes each. The robustness of the methods is evaluated considering the scenario of trees (17,000 in this case) with missing nodes, i.e., phylogeny reconstruction when we do not have the full set of images to rely upon. Results show that, depending on the used evaluation metrics, some approaches can be preferable to others. Therefore, an analyst can choose the appropriate method according to its needs.

The rest of the paper is organized as follows. In Section 2 we formulate the IPT reconstruction problem. Section 3 outlines the IPT reconstruction algorithm steps, detailing every different proposed technique as well as the state-of-the-art method used as reference. Section 4 reports the conducted experiments and the achieved results. Finally, in Section 5, we draw some conclusion remarks.

2. PROBLEM FORMULATION

In this paper, we adopt a common near-duplicate definition already used in other state-of-the-art works [10, 14, 16]. That is, two images are near-duplicates, if they are generated applying a set of tolerated transformations to an *originating* image. An example of near-duplicate images according to this definition is shown in Fig. 1.

To provide a better insight into this definition, let us consider an *originating* image I_1. From this image, we can obtain several *descendant* images applying one or more transformations $\mathcal{T}_j(I_1)$, $j \in \{1, ..., J\}$, where J is the number of all possible admissible transformations. Iterating this process, we can generate an Image Phylogeny Tree (IPT), whereby every node represents a picture, and every branch a transformation, or a combination of transformations (see Fig. 2). All the images in this tree are considered near-duplicates.

This tree structure, intrinsically embeds all the information concerning the causal relationship between all image pairs. The *originating* image, used to generate all the others, is the root of the tree. Leaves represent *descendant* images. The depth of a node in the tree expresses the number of transformations applied, starting from the root, to that node. Nodes linked with a branch are images in a parent-child relationship.

Notice that the phylogeny tree is, by definition, a directional and acyclic graph. Directionality explains the causal relationship between pairs of parent-child images. The acyclic property constrains every image to only have one parent sequence.

The algorithms we present aim at reconstructing the IPT from a set of near-duplicate images. In a real-world scenario, the *originating* image could be the photo of an important event officially released by a news agency. The *descendants* are all the modified versions distributed online. The reconstruction of the phylogeny tree allows to detect which image was actually the original one, i.e., the root of the tree. Moreover, it enables to shed very interesting insights on the way content is reused. As an example, it is possible to reveal how the original image was edited by different users, and how it was spread over time.

3. RECONSTRUCTION ALGORITHM

The algorithm we propose for IPT reconstruction from a set of near-duplicate images I_m, $m \in \{1, ..., M\}$, is based on the same pipeline followed by [14]. The rationale behind the algorithm is that images are pairwise compared, to find which one was used to generate any other image. After this comparison step, the phylogeny tree can be estimated.

Fig. 3 depicts the pipeline of the method. The main steps can be summarized as follows: i) image registration is performed between every image pairs; ii) a dissimilarity value is computed between every image pairs, taking into account results obtained at the previous step; iii) all the dissimilarity values are analyzed to estimate the phylogeny tree.

Image registration (i.e., step one) follows exactly the method proposed in [14]. On the other hand, we propose an alternative image dissimilarity measure (i.e., step two), and

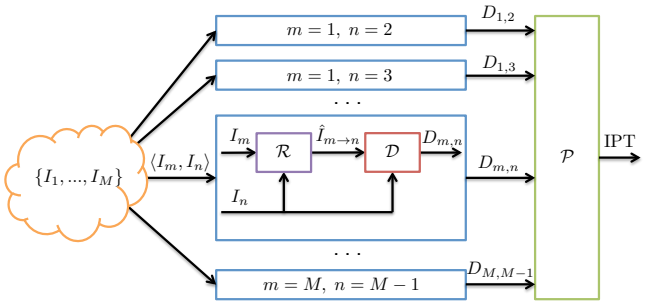

Fig. 3: Block diagram of the tree reconstruction algorithm. Starting from a set of M near-duplicate images, the block $\mathcal{R}$ performs registration, $\mathcal{D}$ dissimilarity computation, and $\mathcal{P}$ estimates the IPT from dissimilarity measures $D_{m,n}$.

different approaches to reconstruct the tree (i.e., step three). In the following, we show a detailed description of every step.

3.1. Image registration

The first step of the algorithm consists in performing the registration between every pair of images. This means, given a reference image I_n, and an image to register I_m, searching for the transformation that possibly map I_m to I_n. To this purpose, as described in [14], we take into account the effect of three possible processing operations: i) color correction; ii) affine transformations (possibly with cropping); iii) JPEG compression.

As first step, we perform color equalization of I_m in the RGB space. This is done by forcing the average and the standard deviation of pixel values in the red, blue, and green component of I_m, to be the same as I_n. More formally, let us define the red, blue, and green components of an image I as I^c, $c \in \{R, G, B\}$. The equalized version of I_m becomes $\bar{I}_{m\to n}$, which is composed by the three components

$$\bar{I}^c_{m\to n} = (I^c_m - \mu^c_m)\,\frac{\sigma^c_n}{\sigma^c_m} + \mu^c_n, \quad c \in \{R, G, B\}, \qquad (1)$$

where μ^c is the average of the c-th component, σ^c the standard deviation, and the whole operation is applied pixel-wise.

Then, we estimate the affine transformation that projects $\bar{I}_{m\to n}$ to I_n. The transformation is estimated by matching key-points computed using the Speeded-Up Robust Features (SURF) algorithm [17]. After warping $\bar{I}_{m\to n}$ to I_n, images are cropped to the same dimension (if needed). Finally, we apply JPEG compression to the warped, and cropped version of $\bar{I}_{m\to n}$, using the quantization step adopted in I_n (available from its bitstream).

After performing all these operations, we obtain the image $\hat{I}_{m\to n}$, which is an estimate of I_n obtained from I_m. If I_n was actually obtained from I_m, then $\hat{I}_{m\to n} \simeq I_n$. Otherwise, $\hat{I}_{m\to n}$ and I_n differ.

3.2. Image dissimilarity

In order to detect whether an image I_n could have possibly been generated from I_m, we need a dissimilarity measure $D_{m,n}$ to compare each registered image $\hat{I}_{m\to n}$ with the reference image I_n. If $\hat{I}_{m\to n}$ and I_n are slightly dissimilar, then I_m could be the parent of I_n.

In [14] the dissimilarity measure is computed as

$$D'_{m,n} = \|I_n - \hat{I}_{m\to n}\|, \qquad (2)$$

which is the l_2-norm of the pixel-wise difference between the two images.

Starting from this definition, we propose a different dissimilarity measure. More specifically, according to [10], an image can be described as the composition of two separable and independent parts: i) a *component* part, conveying information about the content of the depicted scene; ii) a *randomness* part, related to the peculiarities of the process that produced the image. These two parts can be obtained as the result of a de-noising operation. The *component* part corresponds to the de-noised image, whereas the *randomness* part corresponds to the noise, obtained subtracting the de-noised image from the input image. In our implementation, we adopted the wavelet-based de-noising approach proposed in [18].

The intuition is that, the *component* part contains only information about the content, thus being very similar for many near-duplicate images. On the other hand, *randomness* contains traces left by the processing operations applied to the image. Therefore, it can be used to characterize near-duplicates.

To exploit this intuition, the dissimilarity measure we propose is

$$D''_{m,n} = \|I''_n - \hat{I}''_{m\to n}\|, \qquad (3)$$

where I'' indicates the *randomness* part of an image. Notice that this is different from what is proposed in [10], where dissimilarity measure is based on correlation rather than l_2-norm.

The obtained D' or D'' are known as dissimilarity matrices. Notice that dissimilarity is computed only for $m \neq n$, since self-dissimilarity (i.e., dissimilarity of an image with itself), is set by default to zero.

Notice that a dissimilarity matrix (either D' or D'') can be analyzed column-wise to infer information about parent-child relationships between image pairs. As an example, the n-th column of D collects dissimilarities between every image I_m and the n-th one, i.e., how plausible is that I_n is child of each I_m. This is exploited by the tree reconstruction algorithm.

3.3. Tree reconstruction

A tree reconstruction algorithm takes as input a dissimilarity matrix (or two in our proposed methods) and returns an estimate of the IPT. For the sake of notation, we refer to a generic dissimilarity matrix D from now on. This means that either D' or D'' can be used.

Oriented Kruskal (OK-D). In [14], the authors proposed the Oriented Kruskal algorithm, whose input is only a dissimilarity matrix D. In a nutshell, the algorithm initializes a completely disconnected graph, where each node represents an image. At every step, D is inspected and the minimum value $D_{m,n}$ is detected (not considering the trivial case of self-dissimilarity $m = n$). Image I_n is then directionally connected as child of I_m, if this does not create loops. Iteratively, the IPT is built, skipping all the links that create cycles in the graph.

Ordered Oriented Kruskal (OOK-D^A-D^B). OK as proposed in [14] uses only dissimilarity as criterion for linking nodes. This means that the first images to be linked are those whose dissimilarity is the lowest. However, it is possible that an image I_n has low dissimilarity with more than one image. This means that there is a high probability that I_n is son of more than one image, which is clearly not possible. Our algorithm aims at penalizing this behavior, including as first nodes in the IPT, those images whose parent-child relationship is unambiguous, i.e., images slightly dissimilar only to one parent.

To this purpose, we introduce a ranking value that defines in which order the dissimilarity matrix is scanned. This allows us to also merge information from two different dissimilarity matrices: i) D^A is used to compute the ranking; ii) D^B is scanned and used to estimate the IPT. Notice that D^A and D^B can be either D' or D'' in every possible combination.

The ranking value is defined for every column of D^A (i.e., for each child) as

$$r_n = \frac{\min_1(D^A_{m,n})}{\min_2(D^A_{m,n})}, \tag{4}$$

where $\min_1$ and $\min_2$ are the first and second minimum values computed on the n-th column of D^A, respectively (discarding the case $m = n$). Low values of r_n indicate images I_n that are clearly little dissimilar to only one parent (i.e., $\min_2 >> \min_1$).

The algorithm then scans D^B, and selects the column n with the lowest r_n value. Image I_n is connected as child of the image with the lowest dissimilarity with it, if this does not create loops. Iteratively the whole IPT is built.

A detailed description is provided in Algorithm 1. In this algorithm, IPT is defined as a vector of M elements, whose n-th value is the index of the parent associated to I_n. The function `descendant`(n) returns the indexes of all the images that descend from I_n, while `argsort(.)` returns the indexes of sorted elements in ascending order. Notice, that the last step of the algorithm is needed to avoid loops, i.e., once I_m is established as the parent of I_n, none of the progenies of I_n can be the parent of I_m.

Recursively-Ordered Oriented Kruskal (ROOK-D^A-D^B). We also propose another slightly different algorithm, which comes from a modification of the presented OOK, and that makes use of two dissimilarity matrices D^A and D^B as well. In this version, we re-compute the vector r_n at every iteration. This allows to update r_n to take into account links that would cause loops.

Algorithm 1: Ordered Oriented Kruskal

Data: Disimilarity matrices D^A and D^B
Result: IPT

IPT(1:M) = 0 ; // init tree structure
compute $r_{1:M}$ according to (4);
r_idx = `argsort`($r_{1:M}$) ; // idx of sorted vector
for $i = 1 : M$ **do**
  n = r_idx(i) ; // n-th column has the highest r_n
  D_idx = `argsort`($D^B_{1:M,n}$) ; // idx of sorted column
  m = D_idx(2) ; // 1st less dissimilar is n itself
  IPT(n) = m ; // I_m is parent of I_n
  loop_idx = `descendant`(n) ; // idx of I_n progenies
  $D_{m,\text{loop\_idx}} = \infty$; // avoid loops at next iteration
end

3.4. Fusion

Since the proposed algorithms work iteratively, tracing one link at a time, it is possible to switch from one algorithm to another one, after a few iterations.

As an example, it is possible to reconstruct a few nodes of the IPT using ROOK-D'-D'' as already described. After a few iterations, it is possible to stop using the ROOK-D'-D'' algorithm, and continue the IPT reconstruction procedure switching to OK-D', simply removing from D' the contribute given by the nodes already reconstructed. We refer to this case as ROOK-D'-D'' + OK-D'.

Fusion is useful since some algorithms prove to be more accurate in reconstructing the first nodes of the IPT, while others perform better during the last iterations. This fusion method allows to exploit the best characteristics of two different algorithms. This effect will be clarified during the experimental validation in Section 4.

4. EXPERIMENTS AND RESULTS

In order to validate the analyzed and proposed methods, we considered the pool of near-duplicate images generated from the *Uncompressed Color Image Database* (UCID) [19]: this database contains a huge variety of uncompressed images with 512×384 pixel resolution. We randomly selected 10 images, and for each image we created trees of 10 and 20 nodes, using 10 different tree topologies. Each tree with M nodes contains an original image and $M - 1$ near-duplicates. Each near-duplicate is generated using a set of different transformations: JPEG compression with different quality factors, global scaling, different scaling for horizontal and vertical image axes, rotation, cropping, contrast adjustment, brightness adjustment, and nonlinear gamma correction. Table 1 shows the parameters ranges used for these transformations.

Table 1: Parameters of transformations implemented using ImageMagick[1].

Transformation	Range
JPEG compression	[50%,100%]
Global scaling	[90%,110%]
Scaling by axis	[90%,110%]
Rotation	[-5°,5°]
Cropping	[0%,5%]
Contrast adjustment	[-10%,10%]
Brightness adjustment	[-10%,10%]
Gamma correction	[0.9,1.1]

The experimental dataset has a total of 1000 test trees (i.e., 500 trees of 10 nodes, and 500 trees of 20 nodes).

The metrics used to compare the different strategies relies on Dias *et al.* [14] quantitative metrics: i) *edges* - correct orientation of children-parents relationships; ii) *leaves* - correct identification of the furthest sons in the tree; iii) *ancestry* - correct reconstruction of all children-relatives relationships, from root to the most distant sons. In [14], another metric is also defined, i.e., *root*, which measures the correct identification of the tree root. However, we do not report here *root* results, since we were able to identify the correct root for almost all the cases, thus making non informative the comparison of this metric for the proposed algorithms. Instead, we introduced another metric, *tree*, which evaluates the percentage of nodes correctly placed into the IPT vector. More formally, these metrics can be defined as

$$\textbf{\textit{Edges}}\textbf{:}\ E(\text{IPT}_1,\text{IPT}_2)=\frac{|E_1\cap E_2|}{N-1}$$

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

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

$$\textbf{\textit{Tree}}\textbf{:}\ T(\text{IPT}_1,\text{IPT}_2)=\frac{|\text{IPT}_1=\text{IPT}_2|}{M}$$

To better understand these definitions, let us consider the tree depicted in Fig. 2. In this case, computing the aforementioned quantities according to the definition given in [14] leads to

$$\begin{aligned}\text{IPT}_1 &= \{1,1,1,3,3,4\},\\ E_1 &= \{(2\to 1),(3\to 1),(4\to 3),(5\to 3),\\ &\quad (6\to 4)\},\\ L_1 &= \{2,5,6\},\\ A_1 &= \{(1,2),(1,3),(1,4),(1,5),(1,6),(3,4),\\ &\quad (3,5),(3,6),(4,6)\}.\end{aligned} \tag{5}$$

First, we considered the case of IPT reconstruction, when the analyst analyzes all the images that compose the tree under analysis. Fusion methods clearly achieve different results according to the number of iteration steps performed before switching from the first, to the second fused algorithm. An example is given in Fig. 4, where the fusion was adopted switching from OOK-D'-D'' to OK-D'. Depending on the metric, the analyst must apply a different number of iterations before switching, to achieve the best result.

[1] http://www.imagemagick.org/

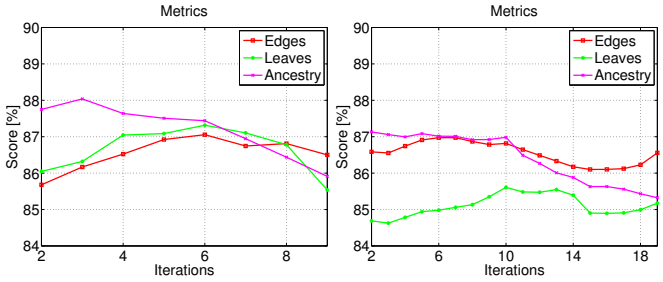

Fig. 4: Metrics scores [%] for fusion technique OOK-D''-D' + OK-D'. Left: 10 nodes trees. Right: 20 nodes trees.

Table 2 reports results for the first three metrics on the best performing algorithms and the state-of-the-art one [14]. Concerning fusion methods, we report only the best achievable results, i.e., those obtained after the optimal number of iterations. Notice that we report fusion results only considering the case in which the second algorithm is OK. This is due to the fact that this configuration led us to better results, since OOK and ROOK reconstruct well the nodes during the first iterations, while OK performs better during last iterations. From Table 2 it is possible to notice that the best results are always achieved using a fusion method.

Turning our attention to the metric *tree*, we also considered the case of possibly missing nodes, i.e., the analyst analyzes only a subset of the images that compose the tree. As an example, with reference to Fig. 2, the analyst might not have image I_4. In this case we expect that the best achievable result is obtained by linking I_6 to I_3. In this scenario, we kept fixed the root node, and studied the behavior of tree reconstruction algorithms with $K = 1, 2, 4, 6$ missing nodes, on trees of 20 nodes. We considered all the possible tree combinations, with only one node missing. In case of $K > 1$, we fixed five sets of possible deleted nodes combinations for each value of K. Therefore, the created test set included 17000 trees of up to 20 nodes each.

Table 3 reports results for metric *tree* in both scenarios of missing and non-missing nodes. Notice that the algorithm that performs better on this metric is ROOK-D''. The state-of-the-art OK-D' outperforms the others only in case of 6 missing nodes out of 20.

As a result, depending on the considered metric, the analyst should prefer an algorithm to another. Anyway, notice that the state-of-the-art algorithm proposed in [14] outperforms the proposed solutions only in one case.

5. CONCLUSIONS

In this paper we proposed a set of solutions to reconstruct an image phylogeny tree, starting from a set of near-duplicate images. The methods are validated using different metrics on

Table 2: Metric scores [%] for the best algorithms computed on 10 and 20 nodes trees. Baseline method is highlighted.

Metric			Edges		Leaves		Ancestry	
Number of nodes			10	20	10	20	10	20
OK-D' (baseline) [14]			85.01	86.50	85.07	84.63	87.25	86.91
OOK-D'			82.30	84.66	84.25	85.44	83.16	82.68
OOK-D''			86.77	87.74	85.28	86.53	85.86	85.48
ROOK-D'			82.95	84.32	83.88	83.22	83.84	82.74
ROOK-D''			87.34	86.76	85.97	84.08	86.71	84.29
OOK-D'-D'	+	OK-D'	85.59	86.88	86.11	85.82	87.95	86.83
OOK-D'-D''	+	OK-D''	85.96	87.55	86.37	**87.78**	84.30	85.93
OOK-D''-D'	+	OK-D'	87.05	86.97	**87.32**	85.61	**88.04**	**87.13**
OOK-D''-D''	+	OK-D''	87.19	86.90	85.79	85.51	85.70	85.39
ROOK-D'-D'	+	OK-D'	85.63	87.10	86.01	86.05	87.97	86.92
ROOK-D'-D''	+	OK-D''	86.70	**87.90**	86.64	87.75	85.04	86.46
ROOK-D''-D'	+	OK-D'	87.19	86.96	87.13	85.38	87.94	**87.13**
ROOK-D''-D''	+	OK-D''	**87.48**	87.12	85.71	85.38	86.27	84.47

Table 3: *Tree* scores [%] for all the algorithms, with and without missing nodes. Baseline method is highlighted.

Number of nodes			10	20				
Missing nodes			0	0	1	2	4	6
OK-D' (baseline) [14]			35.60	10.02	8.03	9.82	5.69	**5.73**
OOK-D'			26.60	9.22	7.27	8.45	5.61	3.85
OOK-D''			38.60	28.06	18.55	17.52	7.86	2.73
ROOK-D'			30.60	7.61	6.55	8.42	5.29	3.53
ROOK-D''			**43.60**	**30.86**	**19.90**	**16.91**	**7.98**	3.97
OOK-D'-D'	+	OK-D'	35.40	10.62	1.88	4.09	0.88	0.32
OOK-D'-D''	+	OK-D''	36.80	23.45	2.50	4.69	1.00	0.52
OOK-D''-D'	+	OK-D'	40.60	24.45	6.15	7.09	3.13	0.96
OOK-D''-D''	+	OK-D''	42.00	27.86	6.07	7.69	3.05	0.96
ROOK-D'-D'	+	OK-D'	35.40	10.62	1.84	4.21	0.88	0.32
ROOK-D'-D''	+	OK-D''	38.80	23.45	2.99	5.09	1.24	0.68
ROOK-D''-D'	+	OK-D'	41.00	23.05	4.83	6.65	3.05	1.04
ROOK-D''-D''	+	OK-D''	43.40	30.86	5.08	6.69	2.57	1.32

a huge dataset of near-duplicate images. Our analysis highlights the fact that an analyst, depending on the evaluation metric adopted, must carefully choose the appropriate strategy. In particular, methods using the proposed ranking vector (i.e., OOK and ROOK) tend to reconstruct very well the tree during the first iterations, justifying their use as first algorithms in fusion techniques. Future works will be devoted to the study of the proposed algorithms when more than two solutions are used for fusion, with the goal of automatically choosing the methods to be combined, depending on the analyzed scenario. Moreover, other strategies to compute the ranking vector used in OOK and ROOK will be investigated.

6. REFERENCES

[1] A. Piva, "An overview on image forensics," *ISRN Signal Processing*, vol. 2013, pp. 22, 2013.

[2] S. Milani, M. Fontani, P. Bestagini, M. Barni, A. Piva, M. Tagliasacchi, and S. Tubaro, "An overview on video forensics," *APSIPA Transactions on Signal and Information Processing*, vol. 1, pp. e2, 2012.

[3] R.C. Maher, "Overview of audio forensics," in *Intelligent Multimedia Analysis for Security Applications*. Springer Berlin Heidelberg, 2010.

[4] M. Kirchner, "Fast and reliable resampling detection by spectral analysis of fixed linear predictor residue," in *ACM Workshop on Multimedia and Security (MM&Sec)*, 2008.

[5] S. Milani, M. Tagliasacchi, and S. Tubaro, "Discriminating multiple jpeg compression using first digit features," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012.

[6] P. Bestagini, S. Milani, M. Tagliasacchi, and S. Tubaro, "Local tampering detection in video sequences," in *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, 2013.

[7] P. Bestagini, M. Zanoni, L. Albonico, A. Paganini, A. Sarti, and S. Tubaro, "Feature-based classification for audio bootlegs detection," in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2013.

[8] L. Kennedy and S.-F. Chang, "Internet image archaeology: automatically tracing the manipulation history of photographs on the web," in *ACM International Conference on Multimedia (ACM-MM)*, 2008.

[9] Z. Dias, A. Rocha, and S. Goldenstein, "First steps toward image phylogeny," in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2010.

[10] A. De Rosa, F. Uccheddu, A. Piva, M. Barni, and A. Costanzo, "Exploring image dependencies: A new challenge in image forensics," in *SPIE Conference on Media Forensics and Security (MFS)*, 2010.

[11] L. Nucci, M. Tagliasacchi, and S. Tubaro, "A phylogenetic analysis of near-duplicate audio tracks," in *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, 2013.

[12] Z. Dias, A. Rocha, and S. Goldenstein, "Video phylogeny: Recovering near-duplicate video relationships," in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2011.

[13] S. Lameri, P. Bestagini, A. Melloni, S. Milani, A. Rocha, M. Tagliasacchi, and S. Tubaro, "Who is my parent? reconstructing video sequences from partially matching shots," in *2014 IEEE International Conference on Image Processing (ICIP)*, 2014.

[14] Z. Dias, A. Rocha, and S. Goldenstein, "Image phylogeny by minimal spanning trees," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 7, pp. 774–788, 2012.

[15] Z. Dias, S. Goldenstein, and A. Rocha, "Large-scale image phylogeny: Tracing image ancestral relationships," *IEEE MultiMedia (MM)*, vol. 20, pp. 58–70, 2013.

[16] A. Joly, O. Buisson, and C. Frelicot, "Content-based copy retrieval using distortion-based probabilistic similarity search," *IEEE Transactions on Multimedia (TMM)*, vol. 9, pp. 293–306, 2007.

[17] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding (CVIU)*, vol. 110, pp. 346–359, 2008.

[18] M. K. Mihcak, I. Kozintsev, K. Ramchandran, and P. Moulin, "Low-complexity image denoising based on statistical modeling of wavelet coefficients," *IEEE Signal Processing Letters (SPL)*, vol. 6, pp. 300–303, 1999.

[19] G. Schaefer and M. Stich, "UCID - an uncompressed colour image database," in *SPIE Storage and Retrieval Methods and Applications for Multimedia (SPIE-SRMAM)*, 2004.