First Steps Toward Image Phylogeny

Zanoni Dias, Anderson Rocha, Siome Goldenstein

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

***Abstract*—In this paper, we introduce and formally define a new problem, Image Phylogeny Tree (IPT): to find the structure of transformations, and their parameters, that generate a given set of near duplicate images. This problem has direct applications in security, forensics, and copyright enforcement. We devise a method for calculating an asymmetric dissimilarity matrix from a set of near duplicate images. We also describe a new algorithm to build an IPT. We also analyze our algorithm's computational complexity. Finally, we perform experiments that show near-perfect reconstructed IPT results when using an appropriate dissimilarity function.**

I. Introduction

Nowadays, digital content is easily redistributable, either through lawful or unlawful means. There are situations where each replication can cause slight modifications of the media, such as gamma-correction or resampling on images and codec re-encoding on videos. There are situations when we need to detect all the copies and/or versions of a given document cohabiting in the wild.

The literature has named this family of problems as *near duplicate detection & recognition* (NDDR) of a document. The problem of *detection* verifies if two documents are near copies of each other. The problem of *recognition* finds, in a large collection of documents, all member documents that are near copies of a given query.

There are several applications for NDDR techniques: (1) reducing the number of versions of a document – for storage and management purposes; (2) tracking the legal distribution and spread of a document on the Internet; (3) copyright and intellectual property protection; and (4) illegal material detection and apprehension. According to Maret [1], the more documents we have, the higher the need for efficient duplicate detection and recognition techniques.

While the detection of exact duplicates is relatively easy, this is not the case for slightly modified duplicates, or near-duplicates [1]–[4].

A far a more challenging task arises when we want to identify, among a set of near duplications, which document is the *original*, and the structure of generation of each near duplication. We would like to be able to tell the history of the transformations that gave rise to the duplications.

In this paper, we present the first steps toward the solution of this problem applied to the particular case of images. This rises several immediate applications.

- **Security:** the modification graph of a set of documents provides information of suspects' behavior, and points out the directions of content distribution.
- **Forensics:** better results if the analysis is performed in the original document instead of in a near duplicate [5].
- **Copyright enforcement:** traitor tracing without the requirement of source control techniques such as watermarking or fingerprinting.

We peek into biology to get an inspiration on how to identify the document's generating structure of modifications. In Biology, we can look at the process of evolution as a branching process, whereby populations change over time and may speciate into separate branches. We can visualize the branching process as a phylogenetic tree [6].

Just as organisms evolve in Biology, a document can change over time to slightly different versions of itself where each of these versions can generate other versions. Sometimes, the changes are unintentional such as those resulting from copying errors inserted during the process of preservation or duplication of manuscripts, such as the famous case of the *Book of Sogya* [7]. In other situations, the changes are intentional such as when a forger modifies a document with the intent of deceiving the viewer [8].

Given a set of near duplicate documents, we would like to identify their causal relationships and the transformations that lead from one to the other. This structure tells the evolving history of the document, even when some pieces, or connections, are missing.

Under proper assumptions, we could approach this problem with *watermarking* and *fingerprinting* techniques. We would be able to the document's markings to recover its history in case the document leaks into the internet, potentially having multiple variations. This is known in the literature as *traitor tracing*.

WIFS'2010, December 12-15, 2010, Seattle, WA, USA. 978-1-4244-9080-6/10/$26.00

However, traitor tracing and watermarking solutions are not always possible: (1) some transformations on the document can destroy its markings; (2) an operator aware of the presence of the marking and its mechanisms can take measures to remove it or change it; (3) watermarks only work for documents reproduced after their adoption, leaving all copies before its adoption unidentifiable; (4) in some cases, it is not possible to assume we have knowledge about the ownership of the source.

In this paper, we present a first solution to the problem of finding, from a set of image duplicates, the structure and parameters of a group of duplications. To accomplish such an objective, we propose and validate, an algorithm as a first step to construct what we call an *Image Phylogeny Tree* (IPT) based on a modified *Minimum Spanning Tree* algorithm.

Section II presents some related work in the literature of near duplicate detection and recognition, they all stop after finding the near duplicate set, and do not find the structure of connections. Section III defines the problem of image phylogeny and its properties. Section IV introduces our first solution toward image phylogeny, and Section V presents the experiments we performed and the methodology we used to validate our method. Finally, Section VI wraps up our paper and discusses possible future work.

II. Related Work

A near duplicate is a transformed version of a document that remains recognizable. Joly et al. [4] formally proposes a definition of what a duplicate is, based on the notion of *tolerated transformations*. According to the authors, a document $\mathcal{D}_1$ is a near duplicate of a document $\mathcal{D}$, if $\mathcal{D}_1 = T(\mathcal{D})$, $T \in \mathcal{T}$, where $\mathcal{T}$ is is a set of tolerated transformations. $\mathcal{D}$ is called the original document, patient zero or the root of the document evolution tree.

A family of transformations $\mathcal{T}$ can contain several combinations of transformations such as $\mathcal{D}_3 = T_3 \circ T_2 \circ T_1(\mathcal{D})$, $T_{\beta=1,2,3} \in \mathcal{T}$. Given an original document $\mathcal{D}$, we can construct a tree of all its near duplications as we show in Figure 1.

A duplicate is a pairwise equivalence relationship. It links the original document to any of its variations through a transformation (e.g., compression, brightness & contrast adjustment, and cropping) [1]. If a document $\mathcal{D}$ has a direct duplicate $\mathcal{D}_1$ and $\mathcal{D}_1$ has a direct duplicate $\mathcal{D}_2$, then document $\mathcal{D}_2$ is in turn a duplicate of document $\mathcal{D}$.

Roughly speaking, there exists two different near duplicate detection philosophies: *watermarking-* and *fingerprinting-based* and *content-based* approaches. Watermarking and fingerprinting methods rely on the embedding of a signature within the original document before its dissemination. With such methods, we can detect the original artwork by checking the signature's presence and modifications patterns within documents. In contrast, content-based methods rely on the analysis of the document's content in order to extract relevant visual features. These methods identify when a set of features are close to those of the original document.

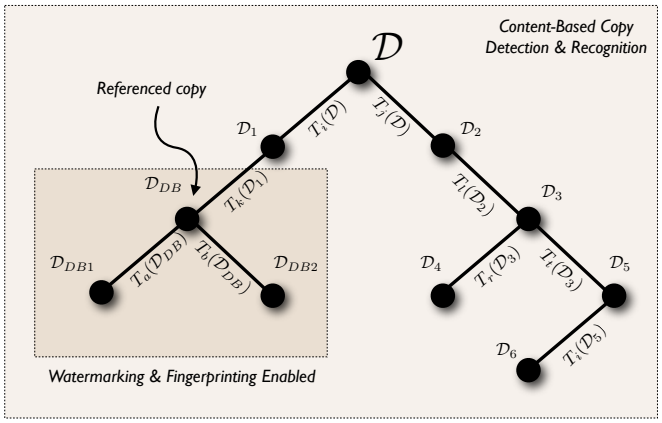


Fig. 1. Near duplicates tree of a document $\mathcal{D}$ and its transformations according to Joly et al [4]. If we can embed a marking on a given document, we can track its transformations easily. On the other hand, when no markings are available or possible, we can use content-based copy detection & recognition methods.

The term *near duplicate* or *copy* refers to any document of the tree. Figure 1 shows that *watermarking* and *fingerprinting* methods for the detection of copyright infringement only allows the detection of the copies of the referenced object. On the other hand, content-based approaches enable the detection of all the copies of the original object.

Regardless the general philosophy, several near-duplicates detection and retrieval methods (NDDR) have been introduced in the last years for a variety of applications. NDDR methods have been used for consumer photograph collections organization [9], [10], multimedia linking [11], copyright infringement detection in images or videos [12]–[16], and forged image detection [8], [17], [18].

Work in the literature is focused in the identification of near duplicates, and no one has yet targeted the identification of the structure of the modifications, or which transformations were used.

In this paper, we solve a a far a more challenging task than NDDR: from a set of near of images known to be near duplications, we identify structure of generation of each near duplication, as well as their transformations — the tree in Figure 1, and the parameters in its edges. We want to be able to tell the history of the duplications, pointing out which document generated the other and so on without making use of any watermarking or fingerprinting method.

III. Image Phylogeny

An *Image Phylogeny Tree* describes the structure of transformations and the evolution of near duplicate images. That requires the prior knowledge of the set of near duplicates, and also a *dissimilarity function* $d(\cdot, \cdot)$ that yields small values for similar images, and large values for distinct images, those that have suffered more significant transformations. The dissimilarity generalizes the concept of distance, but does not constitute a metric space in the *image space* [19].

Let $T_{\vec{\beta}}$ be an image transformation from a family $\mathcal{T}$. We define a dissimilarity function between two images $\mathcal{I}_A$ and $\mathcal{I}_B$ by estimating $\vec{\beta}$ such that

$$d_{\mathcal{I}_A,\mathcal{I}_B} = \left|\mathcal{I}_B - T_{\vec{\beta}}(\mathcal{I}_A)\right|_{\text{image metric}}, \quad (1)$$

has a minimum value. The value of $d_{\mathcal{I}_A,\mathcal{I}_B}$ is a metric of the residual of the transformation that best approximates $\mathcal{I}_A$ into $\mathcal{I}_B$ according to the family of operations described by $\mathcal{T}$.

The dissimilarity function $d_{\mathcal{I}_A,\mathcal{I}_B}$ is not a distance, and does not form a metric space along the images — it does not satisfy the symmetry property. For example, if the family of transformations $\mathcal{T}$ represents only the spatial rescaling operation in images, reducing the spatial resolution (downsampling) is different than increasing the spatial resolution (upsampling). When $\mathcal{I}_A$ and $\mathcal{I}_B$ have different dimensions, $d_{\mathcal{I}_A,\mathcal{I}_B}$ would calculate the image metric of the residual on $\mathcal{I}_B$'s image dimensions, while $d_{\mathcal{I}_B,\mathcal{I}_A}$ would calculate the image metric of the residual on $\mathcal{I}_A$'s image dimensions.

Since d does not represent a metric, our problem is different than the biological phylogenetic tree formulation [6], where the strength of the metric [6], [20] is what eventually defines the quality of the final results.

The input of our problem is a directed graph, and we want to construct a directed tree that satisfies a series of conditions (see Section IV). We are not only interested in finding the structure of transformations — the topology of the tree — but, if possible, to recover the set of parameters $\vec{\beta}$ that generated a child $\mathcal{I}_B$ from its parent $\mathcal{I}_A$ through $\mathcal{I}_B = T_{\vec{\beta}}(\mathcal{I}_A)$.

IV. Oriented Kruskal Algorithm for Image Phylogeny

In this paper, we propose a solution for the construction of the *Image Phylogeny Tree* based on a modified Kruskal's *Minimum Spanning Tree* algorithm [21], summarized on Algorithm 1 which we call *Oriented Kruskal Algorithm*.

The algorithm requires as input a dissimilarity matrix M with respect to a set of n near duplicate images. Data manipulation is different for several reasons, and requires special care. Although our interpretation of the tree is that the transformation goes from the parent to its child representing the changes of the images, in the Oriented Kruskal the data structure keeps the opposite relationship, pointing the edges from child to parent.

A. Algorithm

Given a dissimilarity matrix M built upon a set of n near duplicate images, Lines 1-3 of Algorithm 1 initialize the parent vector (which represents the tree) and create n initial trees, each one containing a vertex representing an image. Each position `Parent[i]` identifies the parent of a node with `id = i`. The **for** loop in lines 6-16 examines matrix positions in order of dissimilarity, from lowest to highest. The *for* loop checks, for each position (i, j), if the endpoints i and j do not belong to the same tree (Test I) and if j is the root of a tree (Test II). If so, we can add the oriented edge $(j \rightarrow i)$ to the forest. Otherwise, we discard such a position.

In the end, all the nodes are connected by $n_{edges} = n-1$ edges forming a tree representing the structure of modifications of the original document with respect to its near duplications.

Algorithm 1 Oriented Kruskal

Require: a dissimilarity matrix M
1: **for** $i \in [1..n]$ **do** ▷ Initialization
2: $Parent[i] \leftarrow i$
3: **end for**
4: $Sorted \leftarrow$ sort positions (i, j) of M into nondecreasing order
5: $n_{edges} \leftarrow 0$ ▷ Controls stopping criterium
6: **for** each position $(i, j) \in Sorted$ **do**
7: **if** ($\texttt{Root}(i) \neq \texttt{Root}(j)$) **then** ▷ **Test I:** joins different trees
8: **if** ($Root(j) = j$) **then** ▷ **Test II:** endpoint must be a root
9: $Parent[j] \leftarrow i$
10: $n_{edges} \leftarrow n_{edges} + 1$
11: **end if**
12: **end if**
13: **if** ($n_{edges} = n - 1$) **then** ▷ The IPT has already n-1 edges
14: **return** $Parent$ ▷ Returning the final IPT
15: **end if**
16: **end for**

The running time depends on how we implement the `Root` function. If we use a *disjoint-set-forest* with the *union-by-rank* and *path-compression heuristics*, we can implement such a function very efficiently [22].

Lines 1-3 take $O(n)$ time to initialize the parent vector. Line 4 takes $O(n^2 \log n)$ to sort the matrix positions since we have $(n^2 - n)$ relevant positions when analyzing n near duplicate images. Lines 6-16 take $O(n^2\alpha(n))$ to check any position, where $\alpha(n)$ is the inverse of *Ackermann function*. The amortized cost for finding the root using an implementation with *union-by-rank* and *path-compression* heuristic is $\alpha(n)$ [22]. Finally, since $\alpha(n) = O(\log n)$, the total running time of our algorithm is $O(n^2 \log n)$, similar to the well-known minimum spanning tree algorithm proposed by Kruskal [21].

B. Simulation of the Algorithm for one IPT

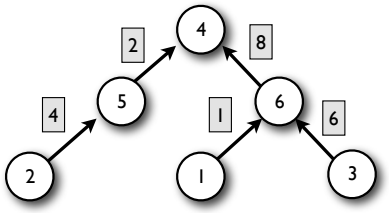
Figure 2 shows the execution of the proposed algorithm for a toy example of $n = 6$ image duplicates. The algorithm initially receives a dissimilarity matrix M that captures the dissimilarities between each pair of duplicates.

The first step of the algorithm initializes a forest with $n = 6$ roots, one for each duplicate and sorts all the positions (i, j) in the dissimilarity matrix M according to their dissimilarity value. Thereafter, the algorithm starts the construction of the Image Phylogeny Tree 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 tree. 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

Dissimilarity Matrix

M	1	2	3	4	5	6
1	-	31	57	37	45	49
2	31	-	33	23	29	32
3	51	41	-	42	37	38
4	16	36	28	-	15	27
5	35	18	54	30	-	54
6	12	40	22	60	19	-

Reconstructed Tree [6, 5, 6, 4, 4, 4]



Algorithm Steps

✔	1	M[6,1] =	12	Select Edge (1 → 6)
✔	2	M[4,5] =	15	Select Edge (5 → 4)
×	3	M[4,1] =	16	**Test II:** Root(1) = 6
✔	4	M[5,2] =	18	Select Edge (2 → 5)
×	5	M[6,5] =	19	**Test II:** Root(5) = 4
✔	6	M[6,3] =	22	Select Edge (3 → 6)
×	7	M[2,4] =	23	**Test I:** Root(2) = Root(4)
✔	8	M[4,6] =	27	Select Edge (6 → 4)

Fig. 2. Simulation of the algorithm to construct an Image Phylogeny Tree from a Dissimilarity Matrix.

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 proceeds with the selection of the position (5,2) as an oriented edge and discarding the position (6,5) because the endpoint 5 is not a root of any tree. After discarding position (6,5), the algorithm selects the position (6,3) with dissimilarity 22 as an oriented edge and discards the position (2,4) because it joins two nodes belonging to the same tree. Finally, the algorithm selects the last oriented edge to form the final tree in Step 8 when testing the position (4,6) with dissimilarity 27.

C. The Importance of the Dissimilarity Matrix

There are two important and independent factors in the process of reconstructing the IPT: the algorithm and the dissimilarity function necessary to create the dissimilarity matrix.

An exceptional dissimilarity function can make a mediocre method shine, while not even the best technique will be able to properly work with an inadequate dissimilarity function.

In this paper, we present a new algorithm, the Oriented Kruskal, for the reconstruction of the IPT. For its proper methodological evaluation, we need an appropriate dissimilarity function, so we can discard it as a potential source of errors during the reconstruction experiments. In Section V, we carefully design a good dissimilarity function for the set of experiments that we use to evaluate our algorithm.

There is a strong parallel between the dissimilarity function of the Oriented Kruskal and the metric for phylogenetic trees and their algorithms, where properties on the metric can guarantee better results [6]. Metric spaces and distances are also important in clustering [20], and on kernel methods, such as SVMs [23].

V. Experiments and Methodology

A. Evaluation of Results

In this section, we describe four quantitative metrics we devised to evaluate a reconstructed tree namely *Root Evaluation*, *Edges Evaluation*, *Leaves Evaluation*, and *Ancestry Evaluation*.

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

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

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

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

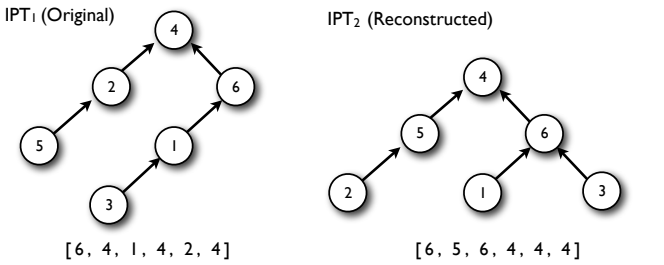
Each of these metrics evaluates a different set of properties of the results, and together allow the practitioner to have a picture of the overall behavior of reconstruction algorithm. As in any designed experiment, we can only calculate them if we do have the real *Ground Truth* to compare our estimated result. In the next section, we report the methodological approach we used in this paper for obtaining this set of controlled experiments and show the results for several images, transformations, and parameters.

In order to describe the above evaluation metrics, we use the illustrative example of Figure 3 of an original tree and its reconstructed version. Suppose an original image phylogeny tree IPT_1, and the resulting tree of the Oriented Kruskal Algorithm, IPT_2.

Considering the *Root Evaluation* metric, the reconstructed tree has the correct root node. According to the *Edges Evaluation* metric, the reconstructed tree has $E = \frac{|\{(1\to6),(6\to4)\}|}{5} = \frac{2}{5} = 40\%$ correct edges. In addition, the reconstructed tree has $L = \frac{|\{3\}|}{|\{1,2,3,5\}|} = \frac{1}{4} = 25\%$ of correct leaves. Finally, the algorithm has found $A = \frac{6}{11} \cong 55\%$ correct ancestors, where common ancestry are $\{(4,2), (6,1), (4,1), (6,3), (4,3), (4,6)\}$.

B. Experiments

We study the potential of our tree construction algorithm in a limited scenario containing only two possible image transformations: scaling and JPEG compression. Therefore, each image is subject to a transformation of scaling, recompression or both each time.



Trees (node points to its parent)

$\text{IPT}_1 = [6, 4, 1, 4, 2, 4]$ $\quad$ $\text{IPT}_2 = [6, 5, 6, 4, 4, 4]$

Root

$\text{Root}(\text{IPT}_1) = 4$ $\quad$ $\text{Root}(\text{IPT}_2) = 4$

Edges (oriented edges of children to parents)

$E_1 = \{(1 \to 6), (2 \to 4), (3 \to 1), (5 \to 2), (6 \to 4)\}$

$E_2 = \{(1 \to 6), (2 \to 5), (3 \to 6), (5 \to 4), (6 \to 4)\}$

Leaves

$L_1 = \{5, 3\}$ $\quad$ $L_2 = \{1, 2, 3\}$

Ancestry

$A_1 = \{(2,5), (4,5), (4,2), (1,3), (6,3), (4,3), (6,1), (4,1), (4,6)\}$

$A_2 = \{(5,2), (4,2), (4,5), (6,1), (4,1), (6,3), (4,3), (4,6)\}$

Fig. 3. On the left, an example of an original tree encompassing an original image and its transformations to create near duplicates. On the right, the reconstructed phylogeny tree of transformations and near duplications. On the bottom, a table detailing the data structures necessary to run and evaluate the Oriented Kruskal Algorithm.

We ran experiments for trees with 10 and 20 nodes. In other words, a tree with 10 nodes represents an original image with its nine near duplicates.

In each of these cases, we have randomly generated 10 different tree topologies, and for each topology we have created five different set of parameters also randomly selected. The range selection for scaling was 90% to 110% with respect to the input image. The range selection for re-compression was 50% to 100% in image quality. The experiment has a total of 100 test trees (50 trees of size 10, and 50 trees of size 20).

We evaluated the method on the *Uncompressed Color Image Database* (UCID) [24][1] which contains a wide variety of images with 512×384-pixel resolution, without compression artifacts.

We tested all trees on 50 images of the UCID data set. For reproducibility, we used the images with `id` $= i \times 25$ where $i \in [1..50]$. Our experiments showed that the reconstruction algorithm, most of the times, is not sensitive to the choice of the image.

After performing all the transformations, as pointed out by the trees used as *Ground Truth*, we calculate the dissimilarity matrix according to Equation 1 with the standard Minimum Squared Error (MSE) technique as the image metric. Later, as a fully independent process, we feed our Oriented Kruskal algorithm using the dissimilarity matrix as input to obtain a reconstructed tree.

[1] `http://www.staff.lboro.ac.uk/~cogs/datasets/UCID/ucid.html`

TABLE I

SUMMARY OF THE RESULTS FOR EXPERIMENTS WITH 20 NEAR DUPLICATES. EACH ROW REPRESENTS THE AVERAGE RESULT ON 250 TESTS COMPRISING 50 DIFFERENT IMAGES AND FIVE DIFFERENT SETS OF TRANSFORMATION PARAMETERS.

	Evaluation Metrics (Average Results)			
Topology	Root	Edges	Leafs	Ancestry
1	100 %	100 %	100 %	100 %
2	100 %	99 %	95.56 %	96.10 %
3	100 %	100 %	100 %	100 %
4	100 %	100 %	100 %	100 %
5	100 %	99.40 %	95.55 %	99.83 %
6	100 %	100 %	100 %	100 %
7	100 %	99.96 %	99.95 %	99.88 %
8	99.20 %	99.74 %	99.87 %	99.45 %
9	100 %	99.96 %	99.89 %	99.99 %
10	100 %	100 %	100 %	100 %
Total	**99.92%**	**99.81%**	**99.48%**	**99.53%**

In the experiments, all the trees with 10 nodes had perfect reconstruction. Table I shows the near-perfect experiment results on trees with 20 nodes. Each table row represents a different topology with 250 tests comprising 50 different images and five different sets of transformation parameters.

We implemented our algorithms using `python` scripts and `ImageMagick` library[2], both for transformation and JPEG encoding as well as the all tree-related operations (construction and evaluation).

The full experiment took approximately 168 computer-hours on current 2.4GHz 64-bit Intel processors with 8GB of memory. As in many similar applications, most of this time is spent calculating the dissimilarity matrix M.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented, for the first time, the problem of image phylogeny reconstruction. Differently than state-of-the-art near duplicate detection and recognition solutions, we were interested in finding the causal relationships and the transformations that lead one image to the other in a set of near duplicates rather than just finding such near duplicates.

Once we have defined this problem, we have presented a solution to calculate a dissimilarity matrix that estimates possible transformations an image from a set of near duplicates might have undergone and reconstruct its phylogeny tree. Our reconstruction technique is based on a modified Kruskal minimum spanning tree algorithm. In addition, we have devised four evaluation metrics to assess the quality of the reconstructed phylogeny trees with respect to their *Ground Truth* trees.

[2] `http://www.imagemagick.org/`

Our new solution have yielded near-perfect results when using an appropriate dissimilarity function, and has a lot of potential for a first approach to the image phylogeny reconstruction problem. We intend to pursue further investigations regarding different dissimilarity functions.

Future work includes exploring optimizations in the calculations of the dissimilarity matrix and extending the family of transformations $\mathcal{T}$. Finally, we are also interested in investigating the behavior of our solution when we have a scenario with missing links.

ACKNOWLEDGMENTS

The authors thank the financial support of CNPq (Grants 483177/2009-1, 200815/2010-5, 309254/2007-8, 551007/2007-9, 551623/2009-8, and 200717/2010-3) and FAPESP (Grant 2009/18438-7).

REFERENCES

[1] Y. Maret, "Efficient Duplicate Detection Based on Image Analysis," PhD Thesis, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2007.

[2] E. Valle, "Local-descriptor matching for image identification systems," PhD Thesis, Universit de Cergy-Pontoise, Cergy-Pontoise, France, 2008.

[3] E. Valle, M. Cord, and S. Philipp-Foliguet, "High-dimensional descriptor indexing for large multimedia databases," in *Intl. Conference on Information and Knowledge Management (CIKM)*, 2008, pp. 739–748.

[4] A. Joly, O. Buisson, and C. Frlicot, "Content-Based Copy Retrieval Using Distortion-Based Probabilistic Similarity Search," *IEEE Trans. on Multimedia (TMM)*, vol. 9, no. 2, pp. 293–306, 2007.

[5] S. Goldenstein and A. Rocha, "High-Profile Forensic Analysis of Images," in *Intl. Conference on Crime Detection and Prevention (ICDP)*, 2009, pp. 1–6.

[6] B. Lewin, *Genes VI*, 6th ed. Oxford University Press, 1997.

[7] J. Reeds, *John Dee: Interdisciplinary studies in English Renaissance Thought*, 1st ed. Springer, 2006, ch. John Dee and the Magic Tables in the Book of Soyga.

[8] A. Rocha, W. Scheirer, T. E. Boult, and S. Goldenstein, "Vision of the Unseen: Current Trends and Challenges in Digital Image and Video Forensics," *ACM Computing Surveys (CSUR)*, In Press 2011.

[9] A. Jaimes, S. fu Chang, and A. Loui, "Duplicate detection in consumer photography and news video," in *ACM Multimedia (ACMMM)*, 2002, pp. 423–424.

[10] F. Schaffalitzky and A. Zisserman, "Multi-view matching for un-ordered image sets, or "how do I organize my holiday snaps?"," in *European Conference on Computer Vision (ECCV)*, 2002, pp. 414–431.

[11] D.-Q. Zhang and S. fu Chang, "Detecting image near-duplicate by stochastic attributed relational graph matching with learning," in *ACM Multimedia (ACMMM)*, 2004, pp. 877–884.

[12] S.-A. B. L. Amsaleg and P. Gros, "Robust content-based image searches for copyright protection," in *ACM Intl. Workshop on Multimedia Databases (MMDB)*, 2003, pp. 70–77.

[13] C.-S. Lu and C.-Y. Hsu, "Geometric distortion-resilient image hashing scheme and its applications on copy detection and authentication," *Multimedia Systems*, vol. 11, no. 2, pp. 159–173, December 2005.

[14] Z. Liu, T. Liu, D. Gibbon, and B. Shahraray, "Effective and Scalable Video Copy Detection," in *ACM Intl. Conference on Multimedia Information Retrieval (ICMR)*, 2010, pp. 119–128.

[15] X. Cheng and L.-T. Chia, "Stratification-based Keyframe Cliques for Removal of Near-Duplicates in Video Search Results," in *ACM Intl. Conference on Multimedia Information Retrieval (ICMR)*, 2010, pp. 313–322.

[16] Z. X. H. Ling, F. Zou, Z. Lu, and P. Li, "Robust Image Copy Detection Using Multi-resolution Histogram," in *ACM Intl. Conference on Multimedia Information Retrieval (ICMR)*, 2010, pp. 129–136.

[17] Y. Ke, R. Sukthankar, and L. Huston, "Efficient near-duplicate detection and sub-image retrieval," in *ACM Multimedia (ACMMM)*, 2004, pp. 869–876.

[18] J. Fridrich, D. Soukal, and J. Lukas, "Detection of Copy-Move Forgery in Digital Images," in *Digital Forensics Research Conference (DFRWS)*, 2003.

[19] M. O'Searcoid, *Metric Spaces*. Springer, 2006.

[20] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. Wiley-Interscience, 2000.

[21] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proc. of the American Mathematical Society (AMS)*, vol. 7, no. 1, pp. 48–50, 1956.

[22] R. E. Tarjan, "Efficiency of a good but not linear set union algorithm," *Journal of the ACM*, vol. 22, no. 2, pp. 215–225, 1975.

[23] B. Schlkopf and A. Smola, *Learning with Kernels*. MIT Press, 2002.

[24] G. Schaefer and M. Stich, "UCID – An Uncompressed Colour Image Database," in *SPIE Storage and Retrieval Methods and Applications for Multimedia*, 2004, pp. 472–480.