

Large-Scale Image Phylogeny: Tracing Image Ancestral Relationships

Zanoni Dias, Siome Goldenstein, and Anderson Rocha
University of Campinas, Brazil

Three proposed solutions improve the efficiency and effectiveness of tracing the history and evolutionary process of digital objects, by analyzing near-duplicate images in large-scale scenarios.

Individual users reportedly generated more data in 2009 than in the entire history of mankind through 2008.^{1,2} A January 2012 report by the World Economic Forum in Davos, Switzerland, declared data “a new class of economic asset, like currency or gold.”³ However, how can we make sense of such a massive amount of data and transform it into information and, ultimately, into knowledge? Surely sophisticated algorithms are paramount for gleaning knowledge and insights from Big Data.

Parallel to its recognized importance, Big Data also has numerous problems because most of it is not typically ready for traditional databases or data mining. It is common, for instance, that multiple documents are simply duplicates or near-duplicates of one another. Although it is straightforward to find exact duplicates among available media, that is not true for many types of media objects when they undergo slight modifications that change them without interfering on their semantic meaning, leading to near-duplicate media objects.

The detection and recognition of near-duplicate media objects have received much attention from researchers in the last few years, especially for digital images and videos.^{4–7} This family of problems is named near-duplicate detection and recognition (NDDR) of a document. Detection aims at determining whether two documents are near copies of each other, and recognition seeks all member documents that are near copies of a given query in a large collection of documents. Although NDDR techniques are usually targeted at improving storage and search efficiency, other applications include tracking the legal distribution and spread of a document on the Internet.

Neglected until recently, a more challenging task arises when we attempt to identify an original document within a set of near duplicates or determine the generation structure of each near duplicate by pointing out the ancestral relationships for all the documents in a set. Only recently have researchers attempted to go beyond NDDR and identify the structure of relationships within a set of near duplicates. This new area, called *multimedia phylogeny*,⁸ explores the history and evolutionary process of digital objects, looking for causal and ancestry document relationships, the source of modifications, and the order and transformations that originally created the set of near duplicates. This area has several applications:

- *Security.* The transformation history of a set of documents can describe the flow of content distribution.
- *Forensics.* We can identify an original document out of a near-duplicate set to perform document forensic analysis.
- *Copyright enforcement.* We can devise traitor tracing methods without watermarking.
- *News tracking services.* Near-duplicate relationships can explain key elements about the opinion forming process across time and space.
- *Content-based retrieval.* We can design advanced information retrieval systems.

There are mainly two perspectives to consider when finding the structure of relationships within a set of near-duplicate images

Related Work in Near-Duplicate Detection

A near duplicate is normally seen as a transformed version of a document that preserves its semantics. Alexis Joly and his colleagues formally proposed a definition of a duplicate based on the notion of tolerated transformations.¹ 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 a set of tolerated transformations. \mathcal{D} is the original document or the root of the document evolution tree. A family of transformations $1T$ can contain several combinations of transformations such as $\mathcal{D}_3 = T_3 \circ T_2 \circ T_1(\mathcal{D})$, $T_{\beta=1,2,3} \in \mathcal{T}$.

An image duplicate links the original document to any of its variations through a transformation, such as warping and cropping, compression, or color and intensity correction and adjustments. The formal definitions in Joly¹ are not always followed in image and video near-duplicate literature. Typical NDDR methods normally tag any pair of images depicting similar content as “duplicates.” This might be a problem in a forensic scenario, for instance, when we need to probe whether a suspect indeed manipulated a protected photography.

In spite of that, in the past decade, we have seen some exceptional progress on the development of efficient and effective systems to identify document near duplicates in the wild for text, videos, and images.^{1,2–5} However, only recently were the first attempts to go beyond the detection of near duplicates, with attempts to identify the structure of relationships within a set of near duplicates. Lyndon Kennedy and Shih-Fu Chang first addressed the problem of parent-child relationships between pairs of images and termed it “document archaeology.”⁶ Alessia De Rosa and her colleagues proposed detecting image dependencies by considering the images’ mutual information calculated using content similarities and image noise acquisition telltales.⁷

Zanoni Dias and his colleagues defined the problem of image phylogeny tree reconstruction⁸ and later extended it to deal with more than one tree (user interaction needed)⁹ and for videos.¹⁰

John Kender and his colleagues studied content-based relationships among YouTube video clips related to the same event and illustrated the construction of a content-dependency graph representing how videos evolve regarding mutations, crossover, and other operators.¹¹

None of these approaches, however, are suitable for large-scale analyses. Here, our objective is to determine the structure of relationships within a set of near-duplicate images in a large-scale scenario using partial matrices.

References

1. A. Joly, O. Buisson, and C. Frélicot, “Content-Based Copy Retrieval Using Distortion-Based Probabilistic Similarity Search,” *IEEE Trans. Multimedia*, vol. 9, no. 2, 2007, pp. 293–306.
2. C. Xiao et al., “Efficient Similarity Joins for Near-Duplicate Detection,” *ACM Trans. Database Systems*, vol. 36, no. 3, 2011, article no. 15.
3. H. Shen et al., “Near-Duplicate Video Retrieval: Current Research and Future Trends,” *IEEE Multimedia*, 2013, to appear.
4. Y. Maret, “Efficient Duplicate Detection Based on Image Analysis,” doctoral thesis, École Polytechnique Fédérale de Lausanne, Switzerland, 2007.
5. W. Puh and M. Herzinger, “Detecting Duplicate and Near-Duplicate Files,” US patent 6658423, US Patent and Trademark Office, Dec. 2003.
6. L. Kennedy and S.-F. Chang, “Internet Image Archaeology: Automatically Tracing the Manipulation History of Photographs on the Web,” *Proc. ACM Conf. Multimedia (MM)*, ACM, 2008, pp. 349–358.
7. A. De Rosa et al., “Exploring Image Dependencies: a New Challenge in Image Forensics,” *Proc. Media Forensics and Security II*, SPIE, 2010, pp. X1–X12.
8. Z. Dias, A. Rocha, and S. Goldenstein, “First Steps Toward Image Phylogeny,” *Proc. IEEE Workshop Information Forensics and Security*, IEEE, 2010, pp. 1–6.
9. Z. Dias, A. Rocha, and S. Goldenstein, “Image Phylogeny by Minimal Spanning Trees,” *IEEE Trans. Image Forensics and Security*, vol. 7, no. 2, 2012, pp. 774–788.
10. Z. Dias, A. Rocha, and S. Goldenstein, “Video Phylogeny: Recovering Near-Duplicate Video Relationships,” *Proc. IEEE Workshop Information Forensics and Security*, IEEE, 2011, pp. 1–6.
11. J. Kender et al., “Video Genetics: A Case Study from YouTube,” *Proc. ACM Conf. Multimedia*, ACM 2010, pp. 1253–1258.

or videos.^{8–12} First, we need to find an informative dissimilarity function to compare the near duplicates. Second, we need to devise a proper algorithm capable of creating a tree of relationships from the dissimilarity measurements. Such a dissimilarity matrix correlating two near duplicates is normally expensive to compute, and early investigations were constrained

to the case of image and video near duplicates in small scenarios, with no more than 50 or 60 documents for each analysis. (See the “Related Work in Near-Duplicate Detection” sidebar for more details on previous work.)

Here, we go beyond prior work and aim at finding the structure of relationships in a large-scale context. This article focuses on

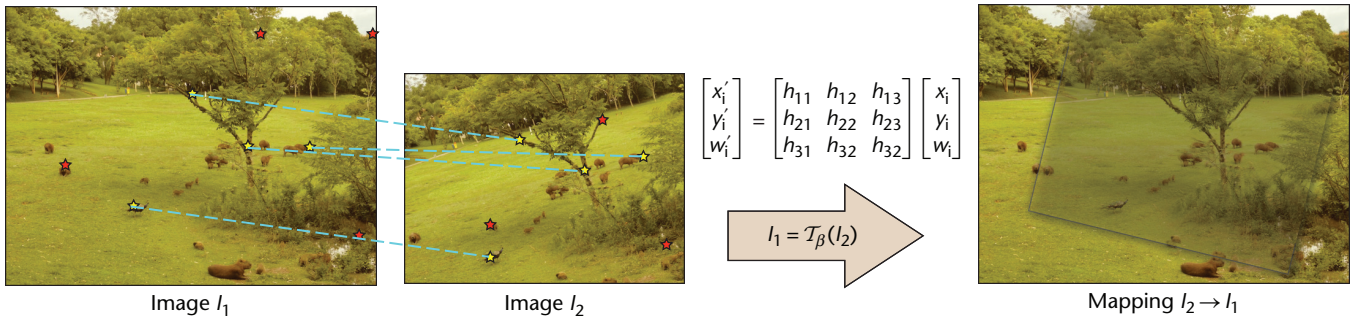


Figure 1. Image phylogeny tree construction process. To calculate image dissimilarities between a pair of images I_1 and I_2 , we find robust points of interest in both images, and for those which are good matches (yellow stars), we calculate an homography matrix representing the necessary parameters to transform one image to another's domain. Once we perform the mapping, we can compare both images pixel by pixel within the region of interest they overlap.

digital images, but the ideas also apply to videos and text. We devise methods for dealing with partially constructed dissimilarity matrices, which can actively request new entries on the fly. This means that we can start the procedure by calculating only a subset of the entries of the dissimilarity matrix and the methods we propose will take care of finding additional entries while optimizing the search in order to use as few entries as possible. We perform experiments with more than 1 million test cases and show that our solutions represent a step forward in efficiently and effectively determining ancestral relationships of digital images.

Image Phylogeny Formalization

Finding the structure of relationships in a set of near-duplicate images normally requires two steps: a dissimilarity function d responsible for calculating how different each pair of images is and how likely it is they are parent and child on the tree and a tree reconstruction algorithm that operates on this matrix. An image phylogeny final product is an *image phylogeny tree* (IPT), which connects images according to their ancestral/descendant relations.⁸ Formally, let $T_{\vec{\beta}}$ be an image transformation from a family \mathcal{T} . We can devise a dissimilarity function between two images \mathcal{I}_A and \mathcal{I}_B as the minimum

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

for all possible values of $\vec{\beta}$ that parameterizes \mathcal{T} . Equation 1 measures the amount of residual between the best transformation of \mathcal{I}_A to \mathcal{I}_B , according to the family of operations \mathcal{T} and

\mathcal{I}_B itself. We can perform such a comparison using any point-wise method \mathcal{L} , such as pixel-wise minimum squared error.

With a set of n near-duplicate images, the first task for creating an IPT is to calculate the dissimilarity between every pair of such images. For that, we need a reasonable set of possible image transformations, \mathcal{T} , from which one image can generate an offspring.^{8,11}

Building an Image Phylogeny Tree

Dias and his colleagues proposed an approach for calculating image dissimilarities and finding IPTs.⁸ For the dissimilarity calculations, the authors define these steps:

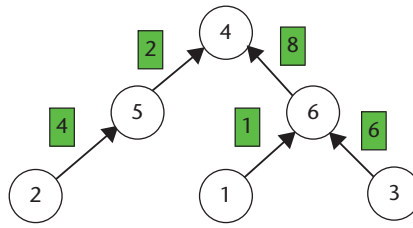
1. Find interest points in each pair of images (using SURF¹³) to estimate warping and cropping parameters robustly using RANSAC.¹⁴
2. Calculate pixel color normalization parameters by mapping the color channels of one image onto the color channels of the other image.
3. Compress one of the images with the same compression parameters as the other.
4. Uncompress both images and compare them pixel-by-pixel according to the minimum squared error metric.

Figure 1 depicts the process. The last step consists of finding the actual phylogeny tree associated with the calculated dissimilarity matrix. The Oriented-Kruskal algorithm,⁸ for example, finds the root of the tree and builds the

Dissimilarity matrix

M	1	2	3	4	5	6
1	-	21	47	27	35	39
2	21	-	23	13	19	22
3	41	31	-	32	27	28
4	6	26	18	-	5	17
5	25	8	44	20	-	44
6	2	30	12	50	9	-

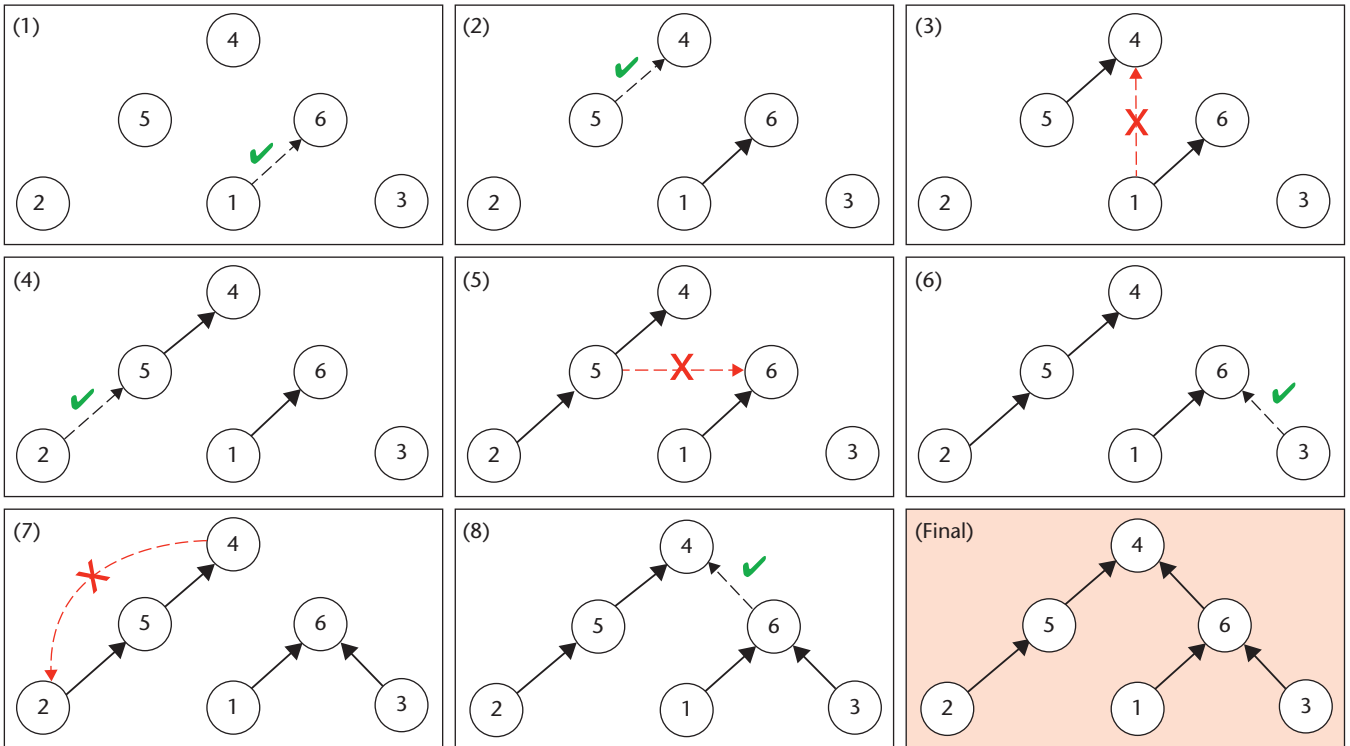
Reconstructed tree [6, 5, 6, 4, 4, 4]



Algorithm steps

1	$M[6,1] = 2$	✓	Select edge (1, 6)
2	$M[4,5] = 5$	✓	Select edge (5, 4)
3	$M[4,1] = 6$	✗	Test II: Root(1) = 6
4	$M[5,2] = 8$	✓	Select edge (2, 5)
5	$M[6,5] = 9$	✗	Test II: Root(5) = 4
6	$M[6,3] = 12$	✓	Select edge (3, 6)
7	$M[2,4] = 13$	✗	Test I: Root(2) = Root(4)
8	$M[4,6] = 17$	✓	Select edge (6, 4)

Construction breakdown



oriented tree representing image parenthood relationships with complexity $O(n^2 \log n)$, for n available near-duplicate images.

Given a dissimilarity matrix M of n near-duplicate images, the Oriented-Kruskal algorithm starts with each image near duplicate as the root of a tree. Next, the algorithm sorts all positions (i, j) of M and then analyzes each position (i, j) according to the sorted order, joining different trees and checking whether each of the endpoints of the analyzed position is a root.

Figure 2 depicts a step-by-step toy example execution of such an algorithm for $n = 6$ near-duplicate images. The algorithm starts with a forest with $n = 6$ roots and sorts all the positions (i, j) in M according to their dissimilarity value.

The most expensive task in image phylogeny is the dissimilarity matrix calculation. For $n = 1,000$ images, we need to calculate $n^2 - n$ entries. (The matrix is not symmetric and the diagonal is not calculated.) Each entry in this matrix requires the mapping of one image onto another image's domain, which requires estimating the necessary transformation parameters.

According to the Oriented-Kruskal algorithm,⁸ we need approximately one second to calculate each entry of a matrix using a conventional 2.1 GHz two-core machine with 4 Gbytes of memory. For an $n = 1,000$ matrix, this scales up to 11.5 processing days. For comparison, the reconstruction process once the dissimilarity matrix is calculated takes less than one minute. Although parallelization techniques could be used to speed up each dissimilarity calculus,

Figure 2. Step-by-step toy example simulation. Using the Oriented-Kruskal algorithm,⁸ we construct an image phylogeny tree from a dissimilarity matrix.

even with such improvements, methods for finding the phylogeny tree from only a fraction of the matrix are paramount because they will drastically reduce the number of computations required.

Proposed Methodology

The most expensive task in reconstructing an IPT is the dissimilarity matrix calculation. Often we need to deal with hundreds or thousands of near-duplicate images or videos each time, and 11.5 computing days to calculate the dissimilarity matrix before using an IPT reconstruction algorithm for $n = 1,000$ images is prohibitive.

Here, we expand on the best-known image phylogeny solution for dealing with large-scale image phylogeny problems: the Oriented-Kruskal algorithm.⁸ We introduce three methods able to reconstruct an image phylogeny tree with a partially complete dissimilarity matrix.

We start the procedure by calculating only a subset (randomly chosen) of elements of the matrix, and if necessary, these methods automatically demand the calculation of missing entries. The whole problem in this article is that it is prohibitive to calculate the entire dissimilarity matrix to seek the phylogeny tree. Instead, we propose solutions to deal with partial matrices (sparse), which demand new calculations as necessary. Only a small percentage of new entries are calculated, lowering the final number of dissimilarity calculations.

Grandparent Heuristic

The first method we propose to deal with a partially complete dissimilarity matrix M is called the *grandparent heuristic*:

1. Calculate the initial IPT using the partially complete dissimilarity matrix and the Oriented-Kruskal algorithm.
2. For each node x in the reconstructed IPT, check if its dissimilarity to its grandparent y is known. If not, calculate the dissimilarity $d(x, y)$ and $d(y, x)$.
3. Reconstruct the IPT from the updated matrix including the new calculated entries.
4. Repeat steps 2 and 3 until convergence or until the amount of new calculation

solicitations exceeds t percent of the total entries in the matrix (here, $t = 10$ percent).

The word “convergence” means that any of the proposed heuristics will stop if the calculated tree at a given iteration t is the same as the one calculated in the step $t - 1$. (The tree does not change with more entries in the matrix.) This algorithm checks $O(n)$ entries at each iteration in the worst case (for example, a degenerate tree in the form of a linked list). The rationale is that knowing the dissimilarity between each triplet (grandchild, parent, grandparent) for the current IPT gives the algorithm a broader overview of the reconstruction process, allowing it to better decide where each triplet is consistent and to decide new possible triplets.

Figure 3a depicts one algorithm’s round for $n = 16$ near duplicates. The tree depicted is the result of a reconstruction from a sparse dissimilarity matrix. After the first IPT construction, the algorithm checks for each node if it knows the dissimilarity of that node to its grandparent. We need to perform 13 grandparent tests in this iteration. The grandparent of node 13 is node 5, whereas node 2 does not have a grandparent in this tree.

Direct Ancestry Heuristic

The second method we propose to deal with a partially complete dissimilarity matrix M is called the *direct heuristic*:

1. Calculate the initial IPT using the partial dissimilarity matrix and the Oriented-Kruskal algorithm.
2. For each node x in the reconstructed IPT, check if its dissimilarity to each of its direct ancestors y_i is known. If not, calculate the dissimilarity $d(x, y_i)$ and $d(y_i, x)$.
3. Reconstruct the IPT from the updated matrix including the new calculated entries.
4. Repeat steps 2 and 3 until convergence or until the amount of new calculation solicitations exceeds t percent of the total entries in the matrix (here, $t = 10$ percent).

This algorithm checks $O(n^2)$ entries at each iteration in the worst case (for example, a degenerate tree in the form of a linked list). This is a generalization of the grandparent heuristic.

The rationale is that knowing the entire ancestral line of node x for the current IPT gives the algorithm the ability to check the consistency of all possible triplets (grandchild, parent, grandparent), quadruplets (great-grandchild, parent, grandparent, great-grandparent), and X -plets (more than four elements) until its first ancestor, the root of the tree. This knowledge gives a broader overview of the tree and allows the algorithm to better decide about node dependencies.

Figure 3b depicts one algorithm's round for $n = 16$ near duplicates. After the first IPT construction, the algorithm checks, for each node x , if it knows the dissimilarity of that node to each of its direct ancestors. We needed to perform 26 direct ancestry tests in this iteration. For instance, node 13 has three direct ancestors besides its parent (nodes 5, 2, and 1), whereas node 6 has one and node 12 has two.

(In)Direct Ancestry Heuristic

The third method we propose to deal with a partially complete dissimilarity matrix M is called the *(in)direct heuristic*:

1. Calculate the initial IPT using the partially complete dissimilarity matrix and the Oriented-Kruskal algorithm.
- 2a. For each node x in the reconstructed IPT, check if its dissimilarity to each of its direct ancestors y_i is known. If not, calculate the dissimilarity $d(x, y_i)$ and $d(y_i, x)$.
- 2b. If no dissimilarity was calculated in step 2a, then for each node x in the reconstructed IPT, check if its dissimilarity to each of the direct descendants y_i of its direct ancestors z_i (that is, x 's siblings, uncles, granduncles, and so on) is known. If not, calculate the dissimilarity $d(x, y_i)$ and $d(y_i, x)$.
3. Reconstruct the IPT from the updated matrix including the new calculated entries.
4. Repeat steps 2 and 3 until convergence or until the amount of new calculation solicitations exceeds t percent of the total entries in the matrix (here, $t = 10$ percent).

This algorithm checks $O(n^2)$ entries at each iteration in the worst case (for example, for a

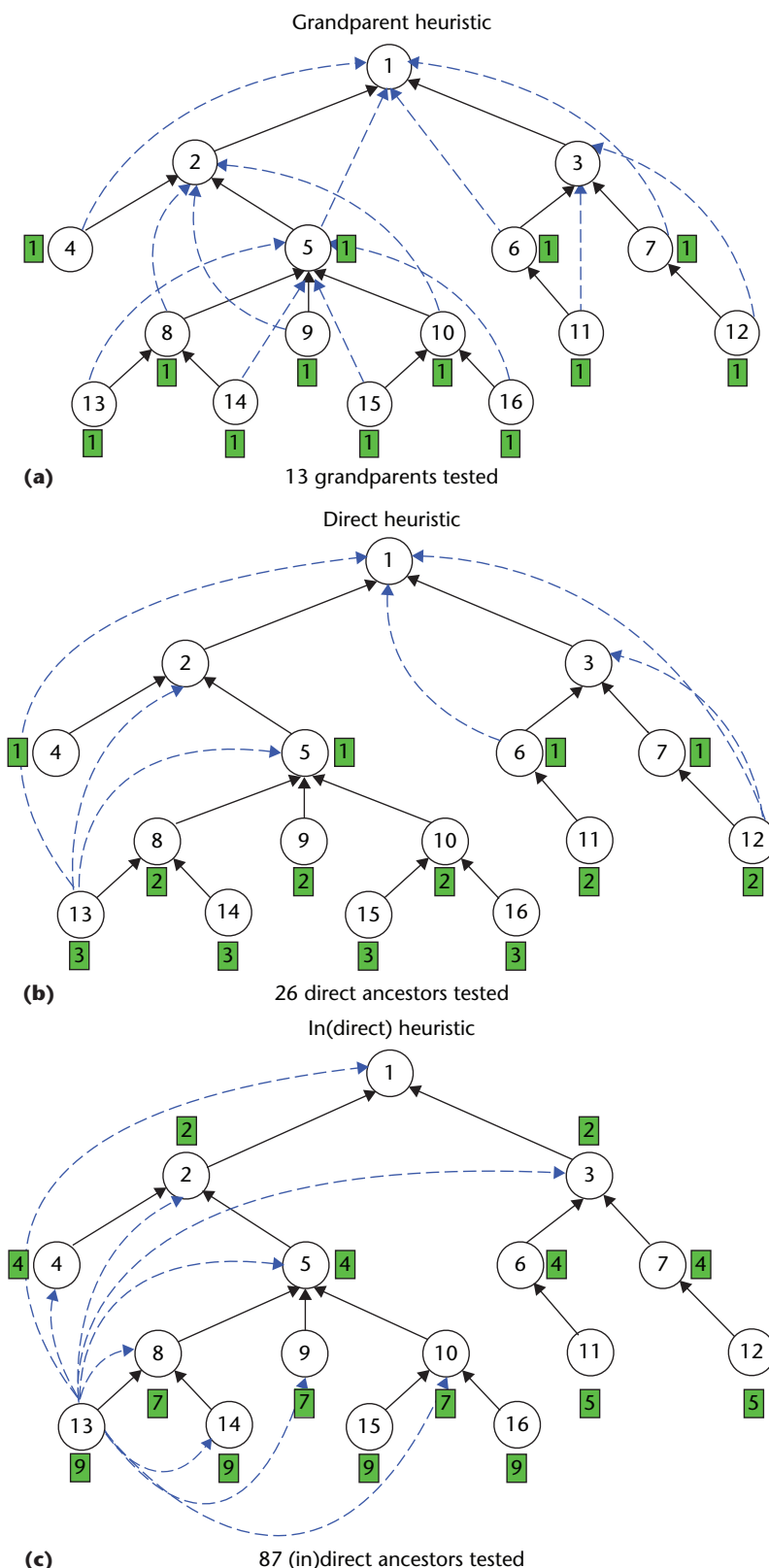


Figure 3. Improving the efficiency and effectiveness of the Oriented-Kruskal algorithm⁸ to construct an image phylogeny tree from a dissimilarity matrix using the (a) grandparent, (b) direct, and (c) *in(direct)* heuristics.

Table 1. Transformations and their operational ranges for creating the dataset following the protocol proposed in Dias.⁸

Transformation	Operational range
<i>Geometry</i>	
(1) Resampling (up/down)	
Global scaling	[90%, 110%]
(2) Rigid transformation	
Global scaling	[90%, 110%]
Rotation	[−5°, 5°]
(3) Generic affine transformation	
Scaling by axis	[90%, 110%]
Rotation	[−5°, 5°]
Off-diagonal correction	[0.95, 1.05]
<i>Cropping</i>	
(4) Cropping	[0%, 5%]
<i>Color</i>	
(5) Brightness adjustment	[−10%, 10%]
(6) Contrast adjustment	[−10%, 10%]
(7) Gamma correction	[0.9, 1.1]
<i>Compression</i>	
(8) Recompression	[50%, 100%]

degenerate tree in the form of a linked list or for a star-shaped tree in which one node is the parent of all other nodes). We can see it as a generalization of the direct heuristic method, which is good at finding roots, and improves upon it to better find ancestors and node connections (edges). The rationale is that knowing all direct relatives of a node x for the current IPT gives the algorithm freedom to change any necessary weak relationship in the tree.

Figure 3c depicts one algorithm's round for $n = 16$ near duplicates. After the first IPT construction, the algorithm checks, for each node x , if it knows the dissimilarity of that node to all descendants of a direct ancestor of x . We needed to perform 87 (in)direct ancestry tests in this iteration. For instance, node 13 has node 14 as a direct descendant of node 8. Nodes 8, 9, and 10 are direct descendants of node 13's direct ancestor, node 5. Also, nodes 4 and 5 are direct descendants of node 13's direct ancestor, node 2. Finally, nodes 2 and 3 are direct descendants of node 13's direct ancestor, node 1. Therefore, node 13 has nine (in)direct ancestors.

Experiments and Validation

Now we turn to the methodology we use to validate the techniques we discuss here.

We follow the protocol proposed in Dias,⁸ which uses a controlled dataset of near duplicates. We also provide experiments with Web images to show the effectiveness of the three proposed methods.

Evaluation Metrics

We used four quantitative measures of success to analyze results: root, edges, leaves, and ancestry for scenarios for which we have ground truth.⁸ We calculate the four metrics according to

$$\text{metric}(\text{IPT}_1, \text{IPT}_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} \quad (2)$$

where “metric” is the evaluation metric of interest (for example, ancestry), IPT_i are the calculated tree and the one used as a reference (the tree ground truth), S_1 is the set of elements in the first tree corresponding to the metric (for instance, the set of ancestors for all nodes of the first tree in the case of calculating the metric “ancestry”), and S_2 is the equivalent for the reference tree. These metrics evaluate complementary properties of the results providing a big picture of the reconstruction algorithm's behavior.

Setup of the Controlled Experiments

We considered image phylogeny trees with $n = 1,000$ image near duplicates. When generating a near duplicate, an image can undergo several possible transformations. Here, we select typical transformations an image can undergo, such as the ones in Table 1. We use the Uncompressed Color Image Database (UCID) dataset,¹⁵ which consists of 512×384 pixel resolution images without compression artifacts.

We validated six different scenarios, each one with 190,000 test cases totaling 1,140,000 test cases. Each test case represents a known percentage of entries of the dissimilarity matrix for reconstructing a phylogeny tree, a tree topology, a set of transformation parameters, and a different original image. In all cases, we considered trees with 1,000 nodes (an original image with its 999 near duplicates) from different tree topologies (forms of the tree) and a combination of parameters from Table 1.

Baseline with Different Sparse Dissimilarity Matrices

We used the Oriented-Kruskal algorithm⁸ as a baseline. We designed three experiments

(cases) to evaluate this algorithm for reconstructing IPTs with different percentages of known dissimilarity matrix entries. Case 1 evaluates the algorithm when knowing p percent of the entries but with no symmetry warranty (knowing $d(x, y)$ but not necessarily $d(y, x)$), whereas case 2 imposes this constraint. Finally, case 3 evaluates the algorithm when knowing p percent of the entries, but for each entry, we have its corresponding row and column known.

Figure 4 presents the best result out of this three experiments: case 2, knowing $d(x, y)$ and $d(y, x)$ for p percent of the dissimilarity matrix. Each point in the plot represents 100 tests with different tree topologies and parameters and images totaling 190,000 test cases. We do not show the results for the other two experiments (380,000 test cases) because their results are worse than the one Figure 4 depicts.

For a complete matrix, the baseline algorithm finds the root in nearly all cases and finds 54 percent of the nodes' correct ancestors. However, the algorithm performs poorly when it knows only a percentage of the full matrix. For instance, for $p = 30$ percent, it correctly finds the root in only 34 percent of the cases and roughly 12 percent of correct nodes' ancestors.

Grandparent, Direct, and (In)Direct Ancestry Heuristics

We have seen that the state-of-the-art Oriented-Kruskal algorithm⁸ is not appropriate for dealing with partially complete dissimilarity matrices and demonstrated a huge performance loss when comparing to a full matrix.

This section presents the results for the three methods we designed to demand additional calculations to a partially complete matrix in order to improve the results. Figures 5a and 5b present the results for grandparent and direct heuristics, and Figures 5c and 5d presents results for the (in)direct heuristic. Each point in the plot is associated with the grid mark just before it—that is, point $p = 30$ percent for the direct heuristic means the algorithm started with 30 percent of known entries, solicited a few more to be calculated, and converged with 35 percent. We discuss the overhead for new calculations later on.

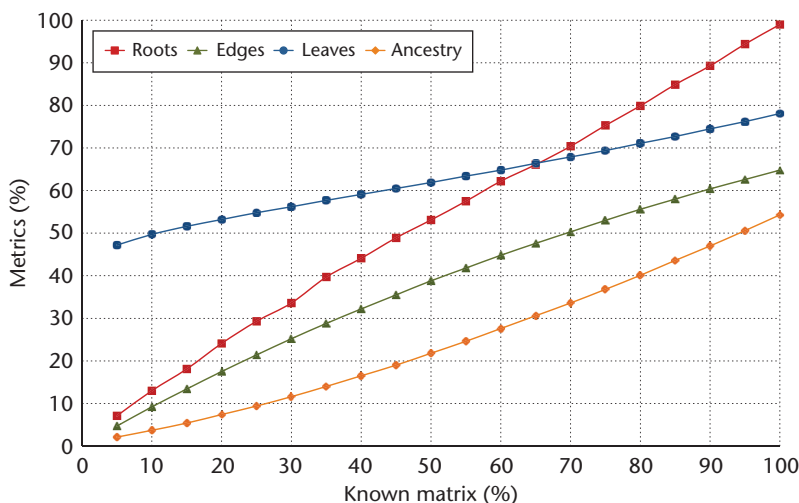


Figure 4. Oriented-Kruskal⁸ baseline for the different known percentages of the dissimilarity matrices for case 2 (knowing $d(x, y)$ but not necessarily $d(y, x)$). The algorithm performs poorly when it knows only a percentage of the full matrix.

With $p \cong 10$ percent known entries, the direct heuristic (Figure 5b) successfully finds the root of the tree in nearly 100 percent of the cases, which represents a huge implication for forensics or copyright infringement lawsuits, for instance, when we need to trace the original source of a digital document and study how it was spread out. The improvement when finding the nodes ancestors is also remarkable. For $p \cong 30$ percent of the entries known, the heuristics successfully finds 30 percent of the ancestors against 10 percent of the baseline, as discussed in earlier. The overhead for the grandparent heuristic is lower than that of the direct ancestry, but in both cases, the heuristics, on average, did not request more than 5 to 7 percent new entry calculations

Figure 5c shows the results for the (in)direct ancestry heuristic method with respect to the ground truth, and Figure 5d compares it with the baseline.⁸ We can see it performs better than both the direct and grandparent heuristics. Also, when the (in)direct method starts with 5 percent of known matrix entries, it automatically solicits a few more calculations and, with $p \cong 15$ percent, successfully finds the root in about 100 percent of the cases. With the $p = 5$ percent of known matrix entries, the baseline only finds the root of the trees in 13 percent of the cases. Thus, the (in)direct method saves 85 percent calculations and thus improves the efficiency over the baseline

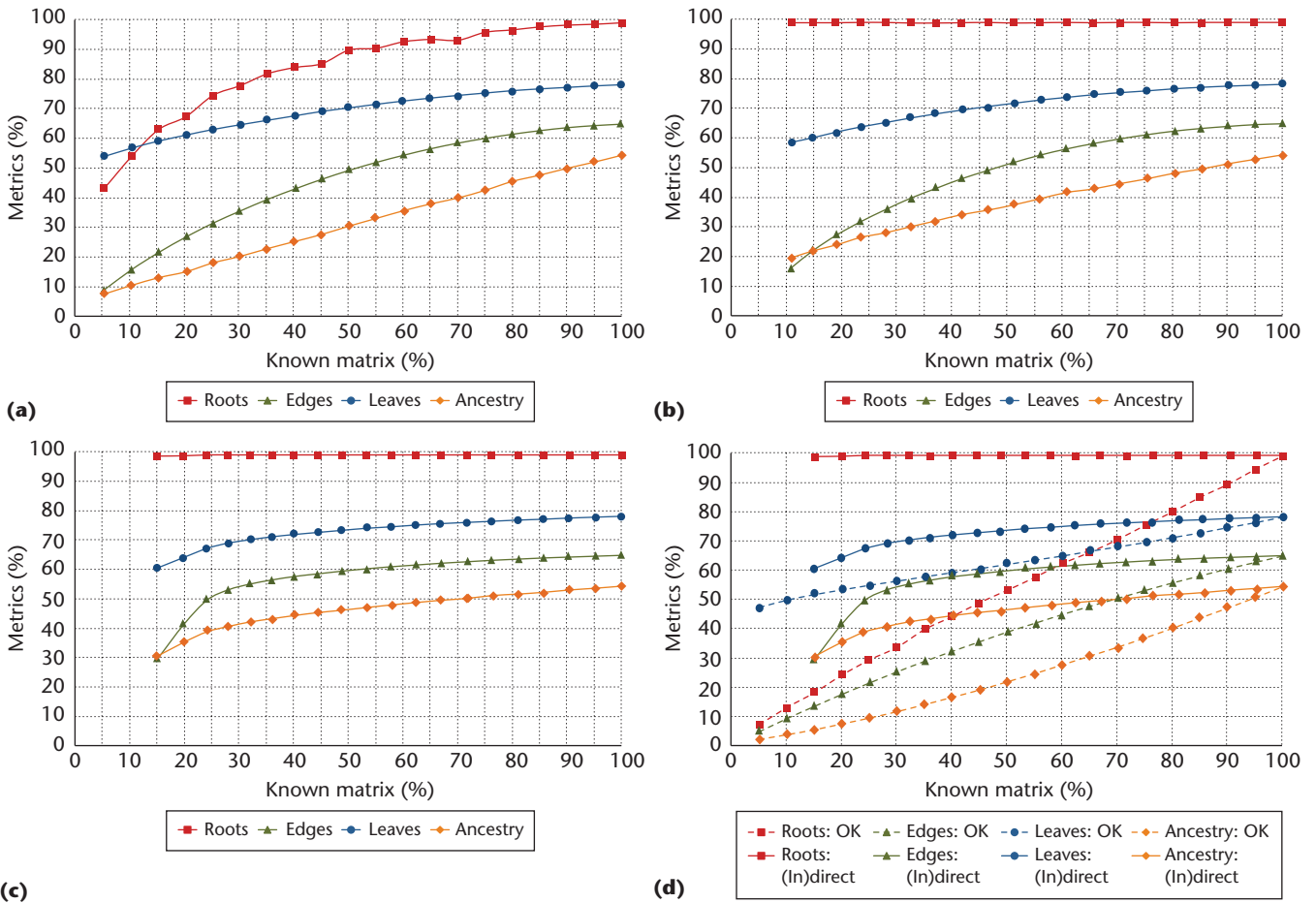


Figure 5. Proposed method comparison. We compared the (a) grandparent, (b) direct, and (c) (in)direct heuristics for different known percentages of the dissimilarity matrices with the ground truth. (d) We also compared the (in)direct heuristic with the case 2 for the baseline Oriented-Kruskal (OK) algorithm.⁸

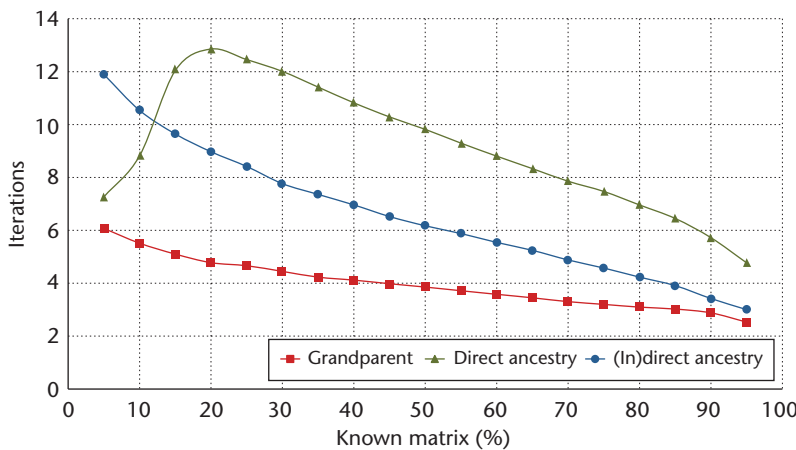


Figure 6. Number of required algorithm iterations. Regardless the number of iterations, none of the three methods actually calculates more than 10 percent new entries.

method by more than six times. The difference for the other metrics is also remarkable with the proposed method soliciting, at most, 10 percent new entries while improving the effectiveness by several percentage points.

Number of Iterations

Figure 6 depicts the number of iterations for the three methods. Each iteration refers to the number of times the method needs to check its ancestors and request new entry calculations. The three heuristics stop either when they converge or when the amount of new entry calculations extrapolates 10 percent of the matrix.

Figure 6 shows that more known entries in the matrix lead to fewer iterations.

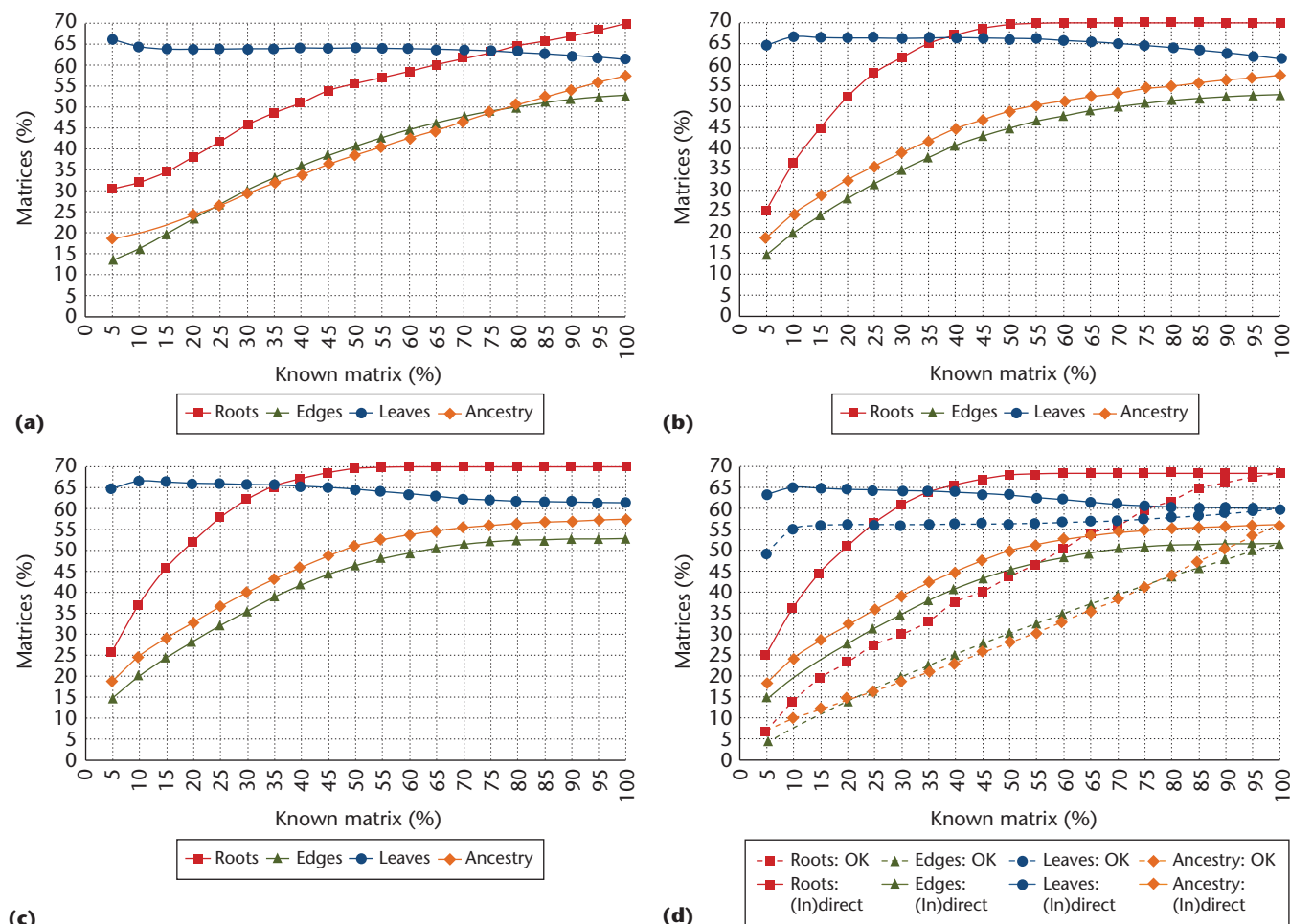


Figure 8. Using images collected from the Web, the (a) grandparent, (b) direct, and (c) (in)direct heuristics for different known percentages of the dissimilarity matrices compared with respect to the ground truth. (d) We also compared the best heuristic, (in)direct, with the case 2 for the baseline Oriented-Kruskal (OK) algorithm (light dashed line).⁸

created five modified images (near duplicates) for each original image, leading to sets with 96 to 120 images. Although we did not know the correct ancestry and relationships of the original images, we do know the parents of the modified generated images (and that they are all leaves). Figure 7a depicts examples of the analyzed near-duplicate sets.

For the experiments with the proposed heuristics, we limited the number of new entries in 10 percent of the matrix. Also, as leaves are more than 85 percent of the tree, this metric shall be analyzed with caution because it creates a bias for that particular topological structure. Nonetheless, it is an important metric to assess. For instance, leaves might be of particular interest because normally they are the most modified images in

a phylogeny tree; they play an important role in some applications, such as sentiment analysis and news tracking.

As Figure 8 depicts, the heuristics improve the raw reconstruction. For example, with 1/4 of the matrix, the (in)direct ancestry heuristic finds the root twice as often as the Oriented-Kruskal algorithm, with similar results for edges and ancestry metrics. Compared with the synthetic validation, the (in)direct ancestry heuristic now has only a slight advantage over the direct ancestry heuristic.

Scenario 2. Here we want to exemplify the actual evolutionary tree of a famous 2011 photograph: “The Situation Room.” Taken by White House photographer Pete Souza on 1 May 2011. This photograph depicts the

US president and his national security team receiving live updates of the Operation Neptune Spear, which led to the killing of Osama Bin Laden. After its online release, different communication channels, social networks, activists, and so forth widely reproduced this image.

For this experiment, we collected the original image released by the White House along with 98 near-duplicate images through Google Images. A manual analysis of the set shows at least nine different patterns of modifications in the images on the Web. Along with the original image, which we labeled ID 0000, Figure 7b shows the regular near-duplicate images (ID 0*) as well as cases of inserting Italian soccer player Mario Balotelli (ID a*), text overlay (ID b*), watermarking (ID c*), face swap (ID d*), inserting elements such as a joystick (ID e*), inserting other people (ID f*), and inserting hats (ID g*).

Figure 7b also depicts the resulting tree using the best proposed algorithm, (in)direct heuristic, using only 25 percent of the dissimilarity matrix, which was four times faster than the Oriented-Kruskal algorithm.⁸ As expected, the algorithm correctly finds the root of the tree and puts simple near-duplicate images close to it (ID 0*, red ellipses). It automatically groups most of the cases we discussed. For instance, there are explicitly two subtrees containing the Balotelli case (blue ellipses). Also, as watermarking normally is inserted by different people, it is reasonable to expect the images not forming a single dense subtree and that is exactly what happens here (ID c*, purple ellipses).

Although real-world cases do not have ground truth, the phylogeny algorithms we present here allow us to focus on different aspects of near-duplicate evolution. We concentrate our attention on the analyses of images on the top of the tree, which supposedly have fewer modifications (important for forensics). We can identify the most modified images in the set (leaves) and group related modifications on images (for better content retrieval).

Conclusion

Investigating the history of objects with just a few clues is a challenging research topic in many science fields. For instance, in paleontology, we can analyze fossil fragments and their changes overtime to determine their

The proposed heuristics can, on the fly, request the computation of new (dis)similarities, saving time on the most expensive phylogeny analysis task.

age and origin. In life sciences, phenotypic and genomic studies enable us to trace back the evolution of living species. Similarly, multimedia phylogeny aims to collect image telltales to study the evolutionary processes to which multimedia objects are subject overtime.

Most of the changes related to near-duplicate multimedia objects are natural and not necessarily harmful. However, the distribution may result in copyright infringement or represent a criminal action,^{8,16} and the spreading pattern itself can help companies understand the demographics and effectiveness of an ad campaign or a product.

Determining the structure of modifications of a set of near duplicates is now an important problem. Determining modification trees is computationally expensive and the current solutions are not designed to deal with large-scale analyses. Here, we have extended our prior work to handle large-scale scenarios and presented three methods to select extra entries of an incomplete dissimilarity matrix on the fly that will lead to improvements in the reconstruction results.

The main advantages of the proposed heuristics are that they can, on the fly, request the computation of new (dis)similarities saving time on the most expensive task of the phylogeny analysis. The heuristics do so while preserving the accuracy of tree root and ancestral relationship estimations. The three proposed heuristics only require 10 percent, at most, extra fast-to-compute calculations.

We are now exploring faster ways to compute the dissimilarities and studying extensions to deal with other types of media, such as videos and text.

MM

Acknowledgments

This work was partially supported by São Paulo Research Foundation – FAPESP (grant 2010/05647-4), National Counsel of Technological and Scientific Development – CNPq (grants 307018/2010-5, 304352/2012-8, 306730/2012-0, and 477692/2012-5), Microsoft, and the European Union through the Rewind project. The Rewind project acknowledges the financial support of the Future and Emerging Technologies (FET) program within the Seventh Framework Program for Research of the European Commission (under FETOpen grant 268478).


References

1. S. Lohr, "The Age of Big Data," *New York Times*, 12 Feb. 2012.
2. A. Weigend, "The Social Data Revolution(s)," *Harvard Business Rev.*, 20 May 2009.
3. "Big Data, Big Impact: New Possibilities for International Development," tech report, World Economic Forum, 2012.
4. C. Xiao et al., "Efficient Similarity Joins for Near-Duplicate Detection," *ACM Trans. Database Systems*, vol. 36, no. 3, 2011, article no. 15.
5. H. Shen et al., "Near-Duplicate Video Retrieval: Current Research and Future Trends," *IEEE Multimedia*, 2013, to appear.
6. Y. Maret, "Efficient Duplicate Detection Based on Image Analysis," doctoral thesis, École Polytechnique Fédérale de Lausanne, Switzerland, 2007.
7. A. Joly, O. Buisson, and C. Frélicot, "Content-Based Copy Retrieval Using Distortion-Based Probabilistic Similarity Search," *IEEE Trans. Multimedia*, vol. 9, no. 2, 2007, pp. 293–306.
8. Z. Dias, A. Rocha, and S. Goldenstein, "Image Phylogeny by Minimal Spanning Trees," *IEEE Trans. Image Forensics and Security*, vol. 7, no. 2, 2012, pp. 774–788.
9. L. Kennedy and S.-F. Chang, "Internet Image Archaeology: Automatically Tracing the Manipulation History of Photographs on the Web," *Proc. ACM Conf. Multimedia (MM)*, ACM, 2008, pp. 349–358.
10. Z. Dias, A. Rocha, and S. Goldenstein, "First Steps Toward Image Phylogeny," *Proc. IEEE Workshop Information Forensics and Security*, IEEE, 2010, pp. 1–6.
11. A.D. Rosa et al., "Exploring Image Dependencies: a New Challenge in Image Forensics," *Proc. Media Forensics and Security II*, SPIE, 2010, pp. X1–X12.
12. Z. Dias, A. Rocha, and S. Goldenstein, "Video Phylogeny: Recovering Near-Duplicate Video Relationships," *Proc. IEEE Workshop Information Forensics and Security*, IEEE, 2011, pp. 1–6.
13. H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded Up Robust Features," *Proc. European Conf. Computer Vision*, Springer-Verlag, 2006, pp. 1–14.
14. M. Fischler and R. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Comm. ACM*, vol. 24, no. 6, 1981, pp. 381–395.
15. G. Schaefer and M. Stich, "UCID: An Uncompressed Colour Image Database," *Proc. SPIE Storage and Retrieval Methods and Applications for Multimedia*, SPIE, 2004.
16. A. Rocha et al., "Vision of the Unseen: Current Trends and Challenges in Digital Image and Video Forensics," *ACM Computing Surveys*, vol. 43, no. 4, 2011, article no. 26.

Zanoni Dias is an assistant professor in the Institute of Computing at the University of Campinas (Unicamp), Brazil. His research interests include theoretical computer science, bioinformatics, and computational molecular biology. Dias has a PhD in computer science from Unicamp. Contact him at zanoni@ic.unicamp.br.

Siome Goldenstein is an associate professor in the Institute of Computing at the University of Campinas, (Unicamp), Brazil. His research interests include computer vision, computer graphics, forensics, and machine learning. Goldenstein has a PhD in computer and information science from the University of Pennsylvania. His is a senior member of IEEE. Contact him at siome@ic.unicamp.br.

Anderson de Rezende Rocha is a Microsoft Research faculty fellow and an assistant professor in the Institute of Computing at the University of Campinas (Unicamp), Brazil. His research interests include digital forensics, pattern analysis, machine learning, and computer vision. Rocha has a PhD in computer science from Unicamp. He is a member of IEEE. Contact him at anderson.rocha@ic.unicamp.br.

 Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.