Phylogeny of JPEG images by ancestors estimation using missing markers on image pairs

Noé Le Philippe[1], William Puech [1] and Christophe Fiorio[1]
[1] LIRMM Laboratory, UMR 5506
CNRS, University of Montpellier
Montpellier, France
e-mail: {noe.lephilippe, william.puech, christophe.fiorio}@lirmm.fr

***Abstract*— Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt**

***Keywords*— Image phylogeny, JPEG compression, Social networks, Image phylogeny tree, near-duplicates**

I. Introduction

It has never been so easy to share ideas and image content thanks to social networks. Every time content is shared however, its information can be altered, often not on purpose, as an image being recompressed to fit low bandwith networks would. But not always, there can also be malicious alteration that aim to modify the content meaning. To trust an image, it is mandatory to be able to tell whether content has been tampered with.

After every transformation, a new image is created. This new image is called a near-duplicate of its parent if, as defined by Joly *et al.* [1], it is a transformed version of a document that remains recognizable. It is formally defined as follows: $I_{n+1} = T(I_n), T \in \mathcal{T}$ where I_n is the parent image at generation n, I_{n+1} is the image at the next generation $n+1$, the child image, and $\mathcal{T}$ is a set of tolerated transformations, I_{n+1} are I_n near-duplicate images. The set $\mathcal{T}$ can contain any transformations, for instance, $\mathcal{T} = \{$*resampling, cropping, affine warping, color changing, lossy compression*$\}$. Authors used the words "tolerated transformation", their meaning is twofold. It limits the transformations to those present in the set $\mathcal{T}$ and sets an arbitrary limit to the strength of any given transformation. For example an image cropped by more than 20% can be considered as a new image and not a near-duplicate.

In this paper, our goal is to identify the parent-child relationships in a set of near-duplicate images and extract the phylogeny of such relationships. The phylogeny is represented by a tree. The root is most likely the original image, or as close as possible to the original, unprocessed image. The leaves, on the other end of the spectrum, are the one most tampered with. We limit ourselves to the study of $\mathcal{T} = \{$*JPEG compression*$\}$, and specifically when the child image quality factor is smaller than its parent to focus on two main fields: image phylogeny, and forensics and multiple JPEG compression detection.

Most state of the art methods for image phylogeny tree estimation [2]–[5] have a two step method. First, they compute a dissimilarity matrix, an assymetric matrix, using a dissimilarity function. The dissimilarity function, given Equation 1, is a function that, given two images, returns small values for similar images.

$$d(I_m, I_n) = \min_{T_{\overrightarrow{\beta}}} \left| I_n - T_{\overrightarrow{\beta}}(I_m) \right|_{comparison\ method}, \quad (1)$$

where I_m and I_n are the images being compared, $T_{\overrightarrow{\beta}} \in \mathcal{T}$ is the transformation being estimated and $\overrightarrow{\beta}$ is a vector of all possible parameters of T. For JPEG compression, $\overrightarrow{\beta} = \{1..100\}$. The tree is then constructed using minimum spanning tree algorithms on the dissimilarity matrix.

JPEG compression, and in particular the detection of double JPEG compression, with the same quantization table [6] or not [7], or with aligned grid or not [8] has been largely studied. The detection is however often limited to at most two or three compressions and unless $Q_f = 100$ [9], we are not able to estimate the number of compressions the image underwent. In an image phylogeny tree, where JPEG compression is the only tolerated transformation, there will be a lot more than two compressions and traditionnal methods will not be very effective. The problem is rather to know whether an image is the result of the compression of another one, and which one, and whether an image has been compressed more times than another one.

Section II presents our method, the theorems and definitions we introduced, the ancestor estimation and the tree reconstruction. Section III shows our results and we discuss how and why we obtained them and finally section IV concludes our work and opens a few perspectives.

II. Method

In this section, we present a new approach to build an image phylogeny tree from a set of near-duplicate images. Unlike state of the art methods that mainly focus on complex algorithm that try their best to approximate the tree from a

dissimilarity matrix, we try to take a binary decision between two images: "Can this image be an ancestor of that other one?". It allows us to focus more on the images themselves and their characteristics and to have a simple tree reconstruction algorithm. Figure 1 shows the diagram of our method.

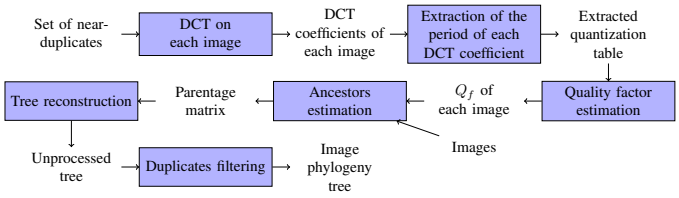


Fig. 1: Diagram of our method.

In this section, we will first present the theorem we based designed our method on, then we explain how we extract the period of each DCT coefficient to extract a quantization table. We also explain how we compute the quality factor of an image and estimate the ancestors. We then present our tree reconstruction algorithm that will eventually output an image phylogeny tree.

A. Theorems

For every image pair in the set, we take a binary decision that will allow us to build a parentage matrix. We try to decide whether an image is an ancestor of another one. We set up a framework to solve this problem.

Definition 1: A marker is a local or global feature extracted from the image. This marker shows that a particular transformation was applied to the image. This marker is passed on the image children, and will be used to prove that an image is not an ancestor of another one. For instance a grayscale image cannot be an ancestor of a color image because the "color" marker is missing.

Definition 1: Let $f(I_m, I_n)$ be a function that for every image pair (I_m, I_n) detects every time there is one a marker visible in the image I_m and not in its potential child I_n, thus proving that I_m is not an ancestor of I_n. This function is called a negation function. This is an abstract function, and that is where lies the difficulty.

Theorem 1: For all image pair (I_m, I_n) in a set of near-duplicate images, if there is no marker proving that I_m is not an ancestor of I_n then there is a parent-child relationship between I_m and I_n, $I_m \rightarrow I_n$, with $m < n$.

Proof: If $f(I_m, I_n)$ cannot find a marker that proves that I_m is not an ancestor of I_n then this marker does not exist, thus I_m is an ancestor of I_n, with $m < n$. ■

B. Quality factor estimation

We work with images that underwent at least one JPEG compression. These images, traveling through social networks, may have had their format changed, and converted to raw format for instance. The quality factor Q_f of the last JPEG compression, mandatory for our method, is then lost. We propose to estimate the quality factor of images that did not undergo any lossless compression other than JPEG.

1) Quantization table extraction: when a signal is quantized, its values gather around multiples of the quantization step. The gap between these values, the period, is what we are going to compute for the first 35 DCT coefficients of the image (in zigzag order). We only use the first 35 because further coefficients are often too quantized and have many null values to be of any use: we may badly estimate their period, and hinder the table extraction process. To get these periods, we compute the auto-correlation of a signal, as shown Fig. 2. We only keep the first few peaks above a threshold. It allows us to filter out the noise and not have missing peaks because of the threshold, as we would have Fig. 2 with peaks alternating above and below the threshold. We repeat this process for every DCT coefficient selected, it outputs a partial quantization table, named $\widehat{q}(u, v)$, with $u, v \in \{0, ..., 7\}$. Next we compute Q_f from $\widehat{q}(u, v)$.

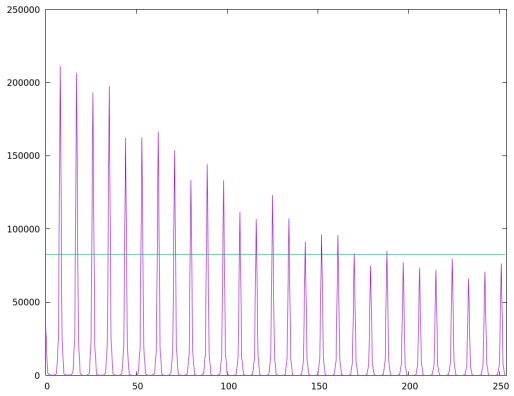


Fig. 2: Auto-correlation of the DC coefficients and the threshold in green.

2) Quality factor estimation: we have two choices to compute the quality factor from $\widehat{q}(u, v)$. We can either check against every single quantization table from $Q_f = 1$ to $Q_f = 100$ which one is closest or use equations 2 and 3 from the JPEG standard:

$$\begin{aligned} &if \quad Q_f < 50 \quad Q_s = 5000/Q_f \\ &else \quad Q_s = 200 - (Q_f \times 2) \end{aligned} \tag{2}$$

$$q(u, v) = \frac{(base(u, v) \times Q_s) - 50}{100} \quad with\ 1 \leq q(u, v) \leq 255 \tag{3}$$

where $base(u, v)$ is the reference JPEG quantization table for $Q_f = 50$. The main advantage of checking against every table is that it outputs a Q_f. The Q_f is n the index of the minimal distance between $q_n(u, v)$ and $\widehat{q}(u, v)$, $n \in \{1..100\}$. This method however is computation heavy and lacks accuracy.

Using Equations 2 and 3, even if it is light and fast, has one main issue: it cannot compute Q_f. It is possible to get Q_s from Equation 3, but it is not possible to get Q_f from Q_s in equation 2 without knowing whether Q_s is greater than 50.

We chose to use both methods. We get a first estimation of Q_f by finding which table from $q_n(u,v)$ is most similar to $\widehat{q}(u,v)$, we call this step primary estimation. With this primary estimation of Q_f, we refine our estimation by using equations 2 and 3. Thus, a Q_f* is computed for each coefficient in the partial quantization table $\widehat{q}(u,v)$. The actual Q_f is the mean of all Q_f*. This step is called secondary estimation. We should add that even if after the primary estimation, Q_f is incorrectly estimated above 50, the difference between both cases in equation 2 is small for values of Q_f close to 50. The estimated Q_f is not always exactly the real Q_f anyway, we address this problem in the ancestor estimation.

C. Ancestors estimation

The quality factor Q_f used to compress the image is mandatory to estimate the ancestors. Not only is it a very effective marker to filter out images that cannot be ancestors, the quality factor also allows us to normalize images to better compare them.

We assume that JPEG compression is a deterministic operation. That is, for the same implementation the same input will always output the same result. This implies that all compressions of the same image will be identical, for any number of compression the image underwent beforehand. As explained before, we limit ourselves to JPEG compression, it means that to obtain a child image from its parent, the parent was JPEG compressed. An ancestor is called the parent if it is one transformation away from its child, one compression away in our case.

From this, we can accurately estimate the parent:

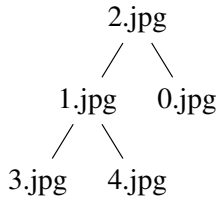
Let I_m and I_n be two images and Q_{f_m} and Q_{f_n} their quality factors, and $C(I, Q_f)$ a JPEG compression operator, with I an image and Q_f a quality factor.

If $I_n = C(I_m, Q_{f_n})$, then I_m is the parent of I_n.

As we stated section II-B, our Q_f is not always the right Q_f, but can be too big or too small. We solve that testing Q_f and its neighbors as compression candidates until a ancestor is found. The running time of our algorithm is closely related with the accuracy of the estimation of our Q_f's. This method, even if it yields excellent results, can only find the direct parent and not further ancestors. It is indeed not easy to take a binary decision when the bounds of the values changes so much with the input data.

D. Tree reconstruction

The ancestor estimation process output a binary matrix of size $n \times n$, n being the number of images in the set. This matrix is called a parentage matrix. A 1 value at index (i, j) in this matrix means that image I_i is an ancestor of image I_j. A 0 value tells us they are not related. Figure 3 shows an example of a tree with its parentage matrix.



(a) Phylogeny tree

-	I_0	I_1	I_2	I_3	I_4
I_0	-	0	0	0	0
I_1	0	-	0	1	1
I_2	1	1	-	1	1
I_3	0	0	0	-	0
I_4	0	0	0	0	-

(b) Parentage matrix

Fig. 3: An example of a phylogeny tree and its parentage matrix.

From this example, we can notice several things. A column with only 0 values does not have any ancestor, it is then the root of the tree, which, by definition, does not have ancestors. A row with only 0 values is the parent of no one: it is a leaf. We can generalize and say that the less a column has 0 values, the less ancestors it has, the closest it is to the root. Algorithm 1 presents our tree reconstruction algorithm.

Data: M a $n \times n$ parentage matrix
Result: the root of the tree

```
1  nextRoot ← row with min sum of elements;
2  treeRoot ← nextRoot;
3  forall rows row of M do
4      root ← nextRoot;
5      mark root as done;
6      for i ← 0 to n do
7          row[i] ← 0;
8          if sum of elements of row = 0 then
9              add i as child of root;
10         end
11         if row has the smallest sum of elements and is
            not marked as done then
12             nextRoot ← i;
13         end
14     end
15 end
16 return treeRoot
```

Algorithm 1: Tree reconstruction algorithm.

At each iteration, an image is selected as the root (lines 1 and 11) and named $root$. $root$ is removed (line 7) from the ancestors of the other images. If these other images do not have any ancestor (line 8), it means that $root$ was the direct parent of the image being processed. This image is added as a child of $root$ (line 9). Line 5 prevents images from being a potential root twice. This algorithm runs in $O(n^2)$. It has two nested loops and if the sums are computed once at the beginning and updated every time a parent is removed, there is no extra loop increasing complexity.

III. Experiments and results

In this section, we presents the results we obtained with our method. We start by describing how our datasets were generated and then explain our results and how and why they were obtained.

A. Dataset generation

We created three datasets, one with trees with 15 images, another one with 25 images and a last one with 50 images. It allows us to see how things like the distance to the root or the number of compressions affect our tree estimation method.

From a seed images (never lossy compressed), a first image is compressed with $85 < Q_f < 99$, it is the root image. This image is added to the images pool. While the number of images in the pool is smaller than the desired number, an image is randomly chosen in the pool, compressed with $Q_{f_{parent}} - 15 < Q_{f_{child}} < Q_{f_{parent}} - 1$ and added to the pool. The quality factor cannot be smaller than 30, further than that, the image is both too deteriorated and does not contain meaningful information and not a realistic use case on social networks. We used the six base images of BOWS 2 [10], where each image was used to create 15 trees for each tree size, a total of 90 trees per dataset, or 8100 images.

B. Details on a tree

Before giving the results of our method on our datasets, we unfold our algorithms on a 15 images tree as an example for the reader to better visualize how our algorithms work. We use one image of this dataset to show how its quality factor is estimated and then expand to the rest of the tree. The chosen image has the following compression history : $H_{Q_f} = \{86, 85, 73\}$. A partial quantization table $\widehat{q}(u, v)$ is computed using auto-correlation and peak detection techniques. The table $\widehat{q}(u, v)$ for this image is available Fig. 4a. The next step is the primary estimation, where $\widehat{q}(u, v)$ is checked against every quantization table from $Q_f = 1$ to $Q_f = 100$. Figure 5 shows the distance between $\widehat{q}(u, v)$ and every other quantization tables. The primary estimation returns $Q_f = 71$, the quantization table of $Q_f = 71$ is available Fig. 4b for comparison as well as the table of $Q_f = 73$ Fig. 4c, the actual quality factor. A negative value in $\widehat{q}(u, v)$ means that no peak was detected and that no period was estimated, a 0 value means that the coefficient outside of the 35 coefficients range.

All images in the set undergo the same process, followed by the secondary estimation. Table 1 shows how the secondary estimation improves the primary estimation. This secondary estimation is used to compress images to normalize and compare them. Table 2 shows the parentage matrix returned by our algorithm. We can see that there is at most one 1 value per line, it means that only the parent was found and no other ancestor. Finally, Fig. 6 shows both the original tree and the reconstructed one. The trees are identical, all ancestry relationships are correct.

9	6	5	9	13	22	22	31
6	6	8	10	14	16	28	0
8	7	9	13	20	-1	0	0
8	9	12	19	-1	-1	0	0
10	12	-1	31	30	0	0	0
12	13	28	30	0	0	0	0

(a) Quantization table returned by periods estimation, $\widehat{q}(u, v)$.

9	7	8	8	10	14	28	42
6	7	8	10	13	20	37	53
6	8	9	13	21	32	45	55
9	11	14	17	32	37	50	57
14	15	23	30	39	47	60	65
23	34	33	50	63	60	70	58

(b) Best quantization table found during the primary estimation where the distance between $\widehat{q}(u, v)$ and table(i) is minimal, $i = 71$.

9	6	8	8	10	13	26	39
6	6	7	9	12	19	35	50
5	8	9	12	20	30	42	51
9	10	13	16	30	35	47	53
13	14	22	28	37	44	56	60
22	31	31	47	59	56	65	54

(c) Quantization table of the actual Q_f, $i = 73$.

Fig. 4: Example of quantization tables used during the tree reconstruction.

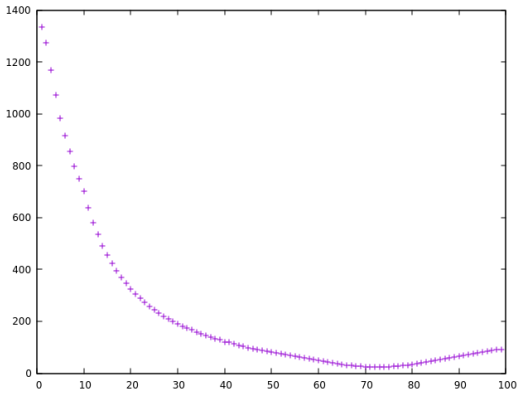


Fig. 5: Distances between $\widehat{q}(u, v)$ and table(i).

C. Experimental results

We use seven metrics to measure our results, three metrics measuring the accuracy of the estimation of the quality factor, the four other, given by Dias *et al.* [2] are

Root: $R(IPT_1, IPT_2) = \begin{cases} 1 & if \texttt{ Root (IPT}_1\texttt{)} = \texttt{Root (IPT}_2\texttt{)} \\ 0 & Otherwise \end{cases}$

Edges: $E(IPT_1, IPT_2) = \frac{|E_1 \cap E_2|}{n-1}$

Leaves: $L(IPT_1, IPT_2) = \frac{|L_1 \cap L_2|}{|L_1 \cup L_2|}$

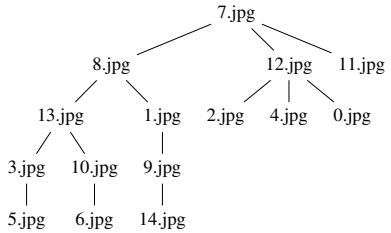
Ancestry: $A(IPT_1, IPT_2) = \frac{|A_1 \cap A_2|}{|A_1 \cup A_2|}$

Image	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}
Primary	71	81	71	58	78	46	61	85	81	79	65	74	84	71	74
Secondary	71	81	71	58	78	46	61	87	82	80	65	74	85	71	75
Actual value	73	81	71	57	79	44	62	86	82	80	65	74	85	71	75

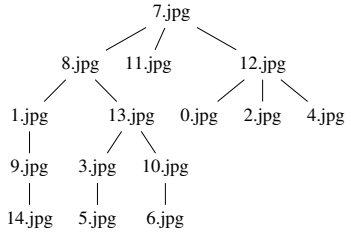
Table 1: Primary and secondary estimation of Q_f for every image in the set.

-	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}
I_0	-	0	0	0	0	0	0	0	0	0	0	0	1	0	0
I_1	0	-	0	0	0	0	0	0	1	0	0	0	0	0	0
I_2	0	0	-	0	0	0	0	0	0	0	0	0	1	0	0
I_3	0	0	0	-	0	0	0	0	0	0	0	0	0	1	0
I_4	0	0	0	0	-	0	0	0	0	0	0	0	1	0	0
I_5	0	0	0	1	0	-	0	0	0	0	0	0	0	0	0
I_6	0	0	0	0	0	0	-	0	0	0	1	0	0	0	0
I_7	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0
I_8	0	0	0	0	0	0	0	1	-	0	0	0	0	0	0
I_9	0	1	0	0	0	0	0	0	0	-	0	0	0	0	0
I_{10}	0	0	0	0	0	0	0	0	0	0	-	0	0	1	0
I_{11}	0	0	0	0	0	0	0	1	0	0	0	-	0	0	0
I_{12}	0	0	0	0	0	0	0	1	0	0	0	0	-	0	0
I_{13}	0	0	0	0	0	0	0	0	1	0	0	0	0	-	0
I_{14}	0	0	0	0	0	0	0	0	0	1	0	0	0	0	-

Table 2: Parentage matrix.



(a) Original image phylogeny tree.



(b) Reconstructed image phylogeny tree.

Fig. 6: Example of an image phylogeny tree and its reconstruction.

$Root$ is trivial and returns 1 if the roots of both trees are identical, $Edge$ measures the ratio of nodes which have the correct direct parent, $Leaves$ is the ratio of correct leaves and $Ancestry$ is the ratio of correct ancestors up to the root.

The three metrics used to measure the accuracy of the estimation of the quality factor is the estimation mean error, the mean error on all the Q_f's on every image of the dataset. The mean overestimation and mean underestimation, which should be symmetrical, tells us whether our algorithm overestimates the Q_f's, and when. Figures 3 shows the results we got using our methods on the three datasets.

We can see that the smaller the dataset, the better the results. We should add that the best score for the three first metrics is 0, when all Q_f were correctly estimated, and the best score for the other is 100, when the two trees are identical.

Metric \ Dataset	**15 images**	**25 images**	**50 images**
Mean estimation error of Q_f	0.42	0.64	0.83
Mean overestimation of Q_f	1.61	1.86	1.94
Mean underestimation of Q_f	1.07	2.12	6.68
roots	95.83	88.88	84.72
edges	99.70	99.24	98.97
leaves	99.59	99.15	98.74
ancestry	99.44	96.88	96.96

Table 3: Results obtained from 8100 imgase, the first three metrics measure the accuracy of the estimation of Q_f, the others, which measure the quality of the tree reconstruction are given in percentage.

The mean errors, even if they are below 1 for the three datasets, and are very good, get worse when the number of images increases, in particular the overestimation is pretty high for 50 images. This is due to too much noise on the DCT coefficients for images compressed a lot of times. We cannot properly estimated the period of such coefficients and therefore cannot properly estimate Q_f. When no peak is detected, our algorithm actually assumes that the image was not quantized, or compressed with a Q_f very close to 100, since a signal not quantized will not show any peaks. A high number of compression is most likely to happen for 50 images, hence the worse results.

We assumed that JPEG compression was a deterministic operation. However, it appears not to be the case for small Q_f's (< 35) with a high number of compressions. It means that for some images, no parent can be found. We wanted to have a simple tree reconstruction algorithm, with only binary data, no other information is available during the reconstruction process. Thus, our algorithm cannot distinguish the root, which has no parent, and a problematic image, with no parents detected. The root is then chosen as the image with the smallest index. Again, this is most likely to occur on big datasets.

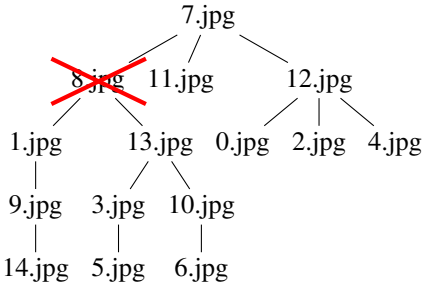
An other problem, which does not affect the root, but rather the rest of the tree, is block convergence, and image convergence, as shown by Lai and Bohme [11]. After a given number of compressions, an image can become identical to its parent, it then becomes impossible to tell which is which. This is caused by images with $Q_f = 30$, the lower bound of Q_f during our generation. If an image with $Q_f = 30$ has a child, this child will also have $Q_f = 30$, and so recursively.

The results given Table 3 are obtained on the whole dataset, where all images are available. Our method is very effective to

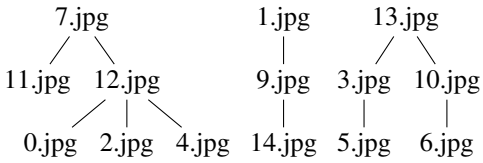
Metric \ Dataset	15 images	25 images	50 images
roots	69.60	33.33	49.01
edges	88.86	92.40	95.07
leaves	91.30	93.00	94.72
ancestry	78.06	82.22	88.21

Table 4: Results with one image missing in each tree.

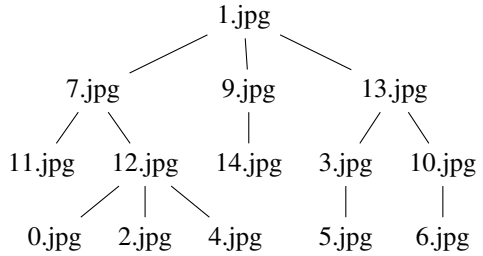
estimate the phylogeny tree when all images are present, when there is always a direct parent. Since we cannot estimate other ancestors, we wanted to try our methods on the same dictates, with one image missing to see if the results were still good. Figure 4 shows the results. We can see that the root metric suffers the most from the image missing. Indeed, our tree reconstruction algorithm cannot differentiate an image with no parents (the root) and an image with no parents detected. The root of the tree is in this case almost randomly chosen. Figure 7 shows how a tree with a missing node results can be obtained, giving an incorrect tree estimation. We actually have perfect subtrees, with the right parents, but we do not have the information on how to link those trees together.



(a) Original tree with node 8 removed.



(b) Three subtrees that we do not know how to link.



(c) Output of the tree reconstruction algorithm.

Fig. 7: Example of a tree with a missing node and a reconstruction attempt.

Even though our method was designed for grayscale images, we wanted to see if the luminance channel of a color image contained enough information for our method to estimate the tree. None of our algorithm were modified and we only gave color images instead of grayscale images as input. Table 8 shows our results. The results are not very different from

Metric \ Dataset	15 images	25 images	50 images
Mean estimation error of Q_f	1.15	1.29	1.42
Mean overestimation of Q_f	1.85	2.70	2.64
Mean underestimation of Q_f	2.97	3.79	4.94
roots	93.94	81.82	87.88
edges	99.35	98.61	99.38
leaves	99.62	98.66	99.78
ancestry	98.79	94.13	98.82

Fig. 8: Results table with color images.

grayscale images, slightly worse, but not significantly so. The estimation of Q_f is not as good, it is because of the rounding when going from RGB to YUV. We did not use any of the chrominance channel, the luminance channel seems to contain enough information and allows us to reliably estimate the tree.

IV. Conclusion

”Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?”

References

[1] A. Joly, O. Buisson, and C. Frélicot, “Content-based copy retrieval using distortion-based probabilistic similarity search,” *Multimedia, IEEE Transactions on*, vol. 9, no. 2, pp. 293–306, 2007.

[2] Z. Dias, A. Rocha, and S. Goldenstein, “First steps toward image phylogeny,” in *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on*, IEEE, 2010, pp. 1–6.

[3] ——, “Image phylogeny by minimal spanning trees,” *Information Forensics and Security, IEEE Transactions on*, vol. 7, no. 2, pp. 774–788, 2012.

[4] A. Melloni, P. Bestagini, S. Milani, M. Tagliasacchi, A. Rocha, and S. Tubaro, "Image phylogeny through dissimilarity metrics fusion," in *Visual Information Processing (EUVIP), 2014 5th European Workshop on*, 2014, pp. 1–6.

[5] P. Bestagini, M. Tagliasacchi, and S. Tubaro, "Image phylogeny tree reconstruction based on region selection," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 2059–2063.

[6] F. Huang, J. Huang, and Y. Q. Shi, "Detecting double jpeg compression with the same quantization matrix," *Information Forensics and Security, IEEE Transactions on*, vol. 5, no. 4, pp. 848–856, 2010.

[7] J. Lukáš and J. Fridrich, "Estimation of primary quantization matrix in double compressed jpeg images," in *Proc. Digital Forensic Research Workshop*, 2003, pp. 5–8.

[8] T. Bianchi and A. Piva, "Detection of nonaligned double jpeg compression based on integer periodicity maps," *Information Forensics and Security, IEEE Transactions on*, vol. 7, no. 2, pp. 842–848, 2012.

[9] M. Carnein, P. Schöttle, and R. Böhme, "Telltale watermarks for counting jpeg compressions," in *Proceedings of the Electronic Imaging 2016*, Publicationt status: Published, San Francisco, USA, 2016.

[10] (2008). Bows 2 dataset, [Online]. Available: `http://bows2.ec-lille.fr/index.php`.

[11] S. Lai and R. Bohme, "Block convergence in repeated transform coding: Jpeg-100 forensics, carbon dating, and tamper detection," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, IEEE, 2013, pp. 3028–3032.