La phylogénie des images dans les réseaux sociaux

Étude bibliographique

Master *Sciences et Technologies*,
Mention *Informatique*,
Parcours IMAGINA

Auteur
Noé Le Philippe

Superviseur
William Puech

Lieu de stage
Équipe ICAR - LIRMM UM5506 - CNRS, Université de Montpellier

version 1 15 mars 2016

Résumé

Ce stage de master.

Abstract

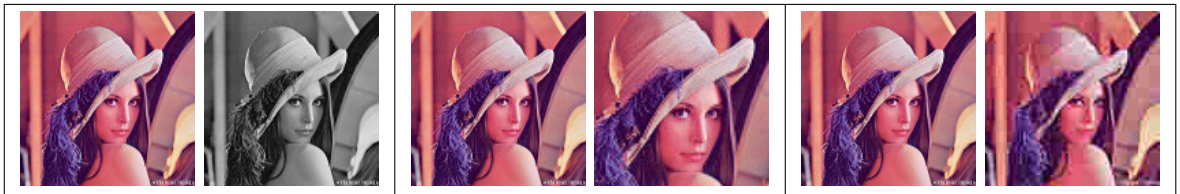
This master thesis.

Table des matières

1

Introduction

La phylogénie, en sciences naturelles, est définie[16] comme l'étude des relations de parenté entre êtres vivants. Et c'est exactement de cela qu'il s'agit dans le cas des images, l'étude des relations de parenté entre images.
À l'ère des réseaux sociaux, il n'a jamais été aussi simple de partager des idées et du contenu. À chaque partage cependant, l'information peut être amenée à être modifiée. Les images, puisque c'est là notre sujet d'étude, peuvent avoir subi un certain nombre de transformations et modifications avant de nous parvenir. C'est dans ce contexte que nous allons intervenir, et tenter de reconstituer la phylogénie de l'image. Il peut être difficile de différentier cette image de l'originale, et de savoir laquelle est l'originale, mais c'est pourtant crucial dans un monde où l'information peut être falsifiée par tout le monde et extrêmement facilement. Les applications sont variées, et ne se cantonnent pas à la détection et la discrimination d'images altérées, on peut également se servir de la phylogénie de l'image pour optimiser de l'espace de stockage en ne gardant que l'originale ou encore suivre la diffusion et l'évolution des idées sur les réseaux sociaux.



couleur → noir et blanc | image entière → crop | image → image compressée

FIGURE 1.1 – Exemples de near-duplicates

1.1 Near-duplicate images (NDI)

Nous travaillons sur un ensemble d'images, toutes similaires, et au milieu de cette jungle d'images, nous devons décider quelle image est le parent de quelle autre, ou autrement dit, quelles images sont des **near-duplicates**. Joly et al. [11] définit la notion de near-duplicate comme suit : $I_1 = T(I), T \in \mathcal{T}$ où I est l'image parent, I_1 est l'image enfant et $\mathcal{T}$ est un ensemble de transformations autorisées, I_1 et I sont alors des NDI. Dans le cas général, $\mathcal{T} = \{$*resampling, cropping, affine warping, color changing, lossy compression*$\}$, Fig. 1.1 montre un exemple de near-duplicates, dans le cadre de ce stage cependant, $\mathcal{T} = \{$*lossy compression*$\}$. Notons l'utilisation du terme *transformations autorisées*. Ce terme est double, d'une part, il place une limite arbitraire dans la force de la transformation, par exemple une image cropée à plus de 10% pourra ne pas pas être considérée comme un near-duplicate, et d'autre part, il permet de restreindre l'espace des transformations possibles. Ainsi, dans le cadre du stage, seules les images de la troisième case de Fig. 1.1 seront des NDI. Ces transformations peuvent évidemment se composer, et une image enfant peut être le résultat de plusieurs transformations.

Note. *Nous utiliserons relation parent-enfant et relation de parenté de manière interchangeable dans le reste de ce rapport, mais c'est bien d'une relation parent-enfant qu'il s'agit.*

1.2 Arbre phylogénétique (Image Phylogeny Tree - IPT)

C'est l'arbre représentant les relations de parenté entre les différentes images. Il sera extrait d'un ensemble de NDI, c'est l'objectif final de l'application. Un exemple est disponible Fig. 1.2. Le passage d'une génération à l'autre, autrement dit d'un noeud à son fils se fait à travers la transformation $I_1 = T(I)$, ainsi, une image et son parent sont des NDI, alors qu'une images et ses soeurs ne le sont pas (Fig 1.3).

La reconstruction de l'arbre se concentre autour de deux problèmes principaux. Le premier est l'identification de la racine, et le second et l'estimation du reste de l'arbre. Il est en effet critique d'identifier correctement la racine. Prenons par exemple un des cas d'utilisation de l'IPT, la détection d'altération d'images. L'idée est que pour un ensemble d'images, plus on est proche de la racine, moins l'image a subi de transformations, et donc moins elle est altérée, avec dans le meilleur des cas, la racine en image originale. On voit bien que si on identifie mal la racine, on déduira, à tort, qu'une image n'a pas été altérée. Il n'est pas toujours garanti que la totalité des

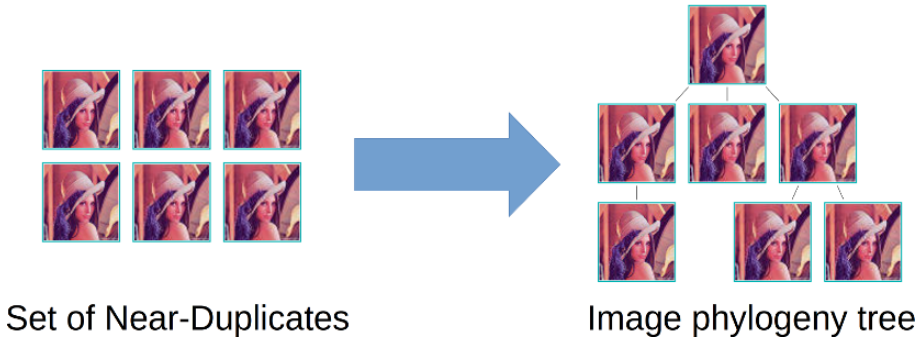


FIGURE 1.2 – Passage d'un ensemble de NDI à un arbre phylogénétique

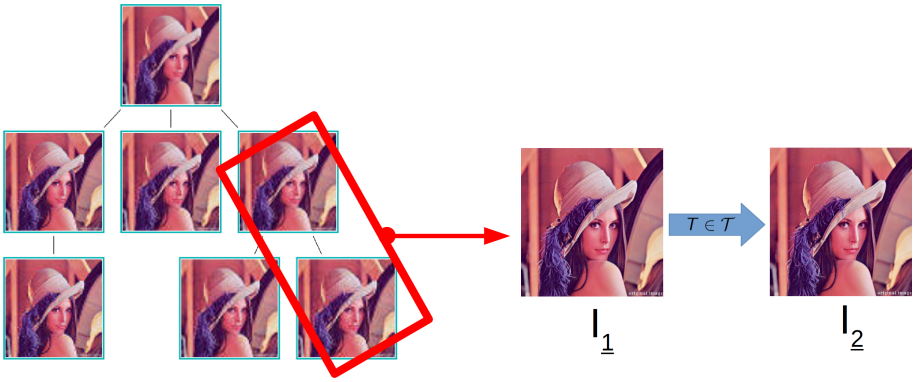


FIGURE 1.3 – Passage d'une image parent à l'image enfant

images de l'arbre original soit présente, de plus, certaines transformations peuvent être mineures, et difficile à détecter, c'est donc bien une estimation de l'arbre original qui sera faite.

1.3 Pourquoi se restreindre à la compression ?

Comme mentionné précédemment, nous ne considérons que le cas de la compression avec perte en laissant de coté les autres transformations. Nous avons décidé de ne nous concentrer que sur une seule transformation pour avoir la possibilité de la traiter en détail et en profondeur dans le cadre du stage et ne pas devoir survoler sans approfondir toutes les transformations. Le choix de la compression est assez naturel, cela n'a, à notre connaissance, pas encore été traité dans le cadre le la phylogénie, et c'est un domaine largement étudié en forensic.

2

État de l'art

2.1 Étude de l'arbre phylogénétique

La Visual Migration Map (VMM)

C'est à notre connaissance, le premier article concernant vraiment notre sujet. Kennedy et al. [13] proposent une approche permettant d'automatiquement détecter la manière dont une image a été éditée ou manipulée, et d'en extraire des relations parent-enfant entre les images. Il vont construire à partir le l'estimation de ces transformations une Visual Migration Map (VMM) (voir Fig. 2.1) qui est en fait notre arbre de phylogénie.

Ils partent du principe que les transformations sont directionnelles, c'est à dire que l'on de ne peut passer que d'une image moins transformée à une image plus transformée. Ainsi, ils vont tenter d'estimer la direction de chaque transformation entre deux images I_1 et I_2 (sachant que $\mathcal{T} = \{$*scaling, cropping, grayscale, overlay, insertion*$\}$). Trois scénarios sont alors possibles : si toutes les transformations sont dans le même sens, l'image fille est alors celle vers qui pointent les transformations, si les transformations sont dans des sens contraires, les images sont sûrement des soeurs, elles n'ont en tous cas pas de relation parent-enfant, et enfin si aucune transformation n'a été détectée, c'est que soit les images sont identiques, soit qu'elles ne sont pas des near-duplicates. On peut en voir un exemple Fig. 2.2.

Un graphe va ensuite être construit à partir des couples d'images pour lesquels une relation parent-enfant a été détectée. À noter qu'une relation parent-enfant ne veut pas forcément dire que c'est le parent direct mais plutôt un ancêtre. Ainsi, un noeud du graphe (une image) peut avoir plusieurs noeuds parents, pour finalement obtenir l'arbre désiré, seuls les chemins les plus longs sont conservés, comme on peut le voir Fig. 2.3.

Image Phylogeny Tree (IPT)

Tout comme l'approche présentée précédemment, Dias et al. [6][7] proposent une approche basée sur le contenu de l'image. Cela consiste a mapper une image sur le domaine d'une autre, pour pouvoir les comparer, et ensuite estimer le coût de cette opération, cela fait comme hypothèse que si deux images sont dépendantes, alors il est possible d'obtenir l'une en appliquant une transformation à l'autre.

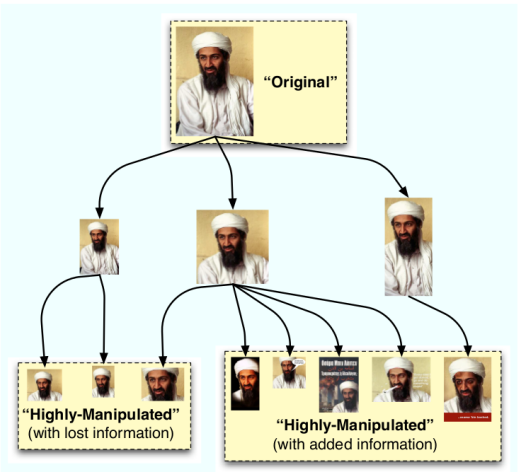


FIGURE 2.1 – Exemple de VMM, issu de [13]

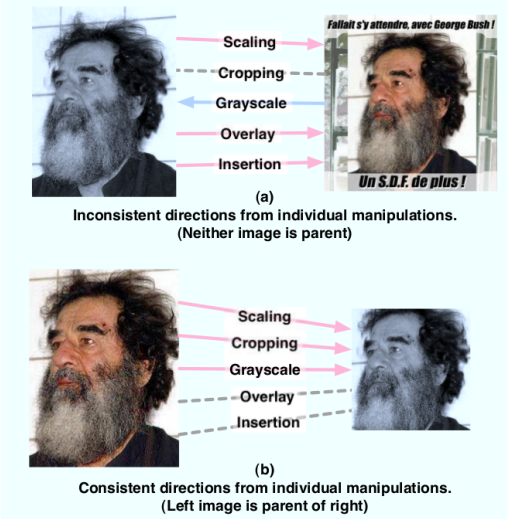


FIGURE 2.2 – Exemple de direction des transformations, issu de [13]

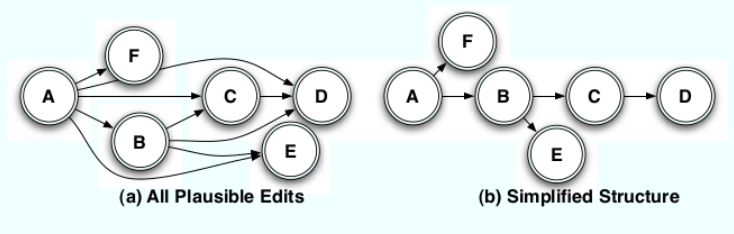


FIGURE 2.3 – Exemple de simplification de graphe, issu de [13]

Les images sont comparées à l'aide d'une *dissimilarity function d(.,.)* qui renvoie des petites valeurs lorsque les images sont proches (elles ont subi des transformations similaires). L'équation 2.1 détaille cette fonction. A et B sont les deux images qui vont être comparées, et $T_{\overrightarrow{\beta}} \in \mathcal{T}$ est la transformation en train d'être estimée. La transformation la plus faible est gardée comme résultat, c'est la transformation la plus probable qu'a pu subir l'image. À noter que $d(.,.)$ est asymétrique, c'est à dire que $d(A,B) \neq d(B,A)$, ce qui est parfaitement logique, en plus d'être nécessaire, sachant que les transformation sont directionnelles, comme expliqué précédemment.

$$d(A,B) = \underset{T_{\overrightarrow{\beta}}}{min} \left| B - T_{\overrightarrow{\beta}}(A) \right|_{comparison\ method} \tag{2.1}$$

On voit donc bien que la problématique principale de cette méthode est de trouver une bonne méthode de comparaison. Les auteurs procèdent de la manière suivante :

- Trouver des points caractéristiques (SURF [1])
- Filtrer les points et estimer les paramètres de transforamtion affines tels que les translations, rotations, rééchantillonages... avec RANSAC [9]
- Calculer la moyenne et la variance de chaque canal couleur de B pour normaliser les couleurs de A
- Compresser les résultats des deux étapes précédentes avec la table de quantification de B

La dissimilarité entre les deux images est enfin obtenue en utilisant la *minimum squared error* (MSE) entre les deux images dans le domaine spatial comme technique de comparaison pour Équation 2.1.

Ces quatre étapes ont servi à rendre les images comparables, en mappant l'une dans le domaine de l'autre, pour avoir des résultats pertinents.

$d(.,.)$ est donc appliquée à tous les couples d'images de l'ensemble, pour créer une *dissimilarity matrix*, une matrice de taille $n \times n$ qui sera ensuite donnée comme entrée à leur algorithme de reconstruction d'arbre, Oriented Kruskal. Nous n'allons pas entrer dans les détails de cet algorithme, il est expliqué de manière exhaustive dans [7], et ayant une approche différente (Chap. 3), la reconstruction sera également différente.

En plus de proposer une approche pour reconstituer l'arbre de phylogénie, ils proposent une approche pour comparer deux arbres, et donc évaluer notre arbre reconstruit si on le compare avec la vérité terrain. Cela consiste en quatre métriques :

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

Root est triviale, et renvoie si les racines sont identiques. **Edges** mesure le ratio de noeuds ayant le bon parent, **Leaves** est le ratio de feuilles correctes, et enfin **Ancestry** est le ratio d'ancêtres corrects jusqu'à la racine.

Ces métriques serviront à évaluer nos résultats dans la suite du stage.

2.2 Analyse des recompression JPEG

Notre but n'est pas vraiment de détecter si une image a été compressée plusieurs fois, nous sommes en effet quasi-certains que nos images auront été recompressées, puisque c'est précisément sur cela que nous travaillons, un ensemble de NDI, où $\mathcal{T} = \{lossy\ compression\}$. L'important est plutôt de savoir combien de fois, et à partir de qui l'image a été compressée, et donc pouvoir en déduire sa distance à la racine. La majorité des articles dans le domaine du forensic ne se concentrent que sur une image, pour en extraire le maximum d'informations possible. Ce n'est pas exactement notre cas, puisque les informations pertinentes pour nous ne sont pas dans l'image directement, mais plutôt dans ses relations avec les autres. Il nous a cependant paru important de traiter l'aspect forensic du problème, et se renseigner sur les différentes techniques, qui bien que créées pour un autre cas d'utilisation, peuvent, sinon s'adapter, au moins nous donner des pistes.

Qu'est ce que JPEG ?

Une rapide introduction sur ce qu'est le JPEG et son fonctionnement s'impose, nous ne parlerons cependant que du mode de compression avec perte, en laissant de coté son mode de compression sans perte, peu intéressant en plus de n'être virtuellement jamais utilisé. Pour de plus amples détails, le lecteur se dirigera vers [17].

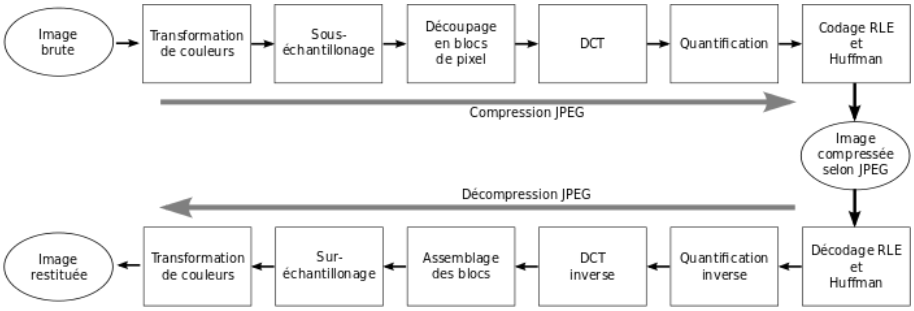


FIGURE 2.4 – Étapes de compression et décompression, issu de [12]

Fig. 2.4 liste toutes les étapes permettant de passer d'une image non compressée à une image compressée pour finir par la décompresser, mais les étapes sont sommairement :

- L'image est convertie dans l'espace YUV
- Les canaux U et V sont sous-échantillonés
- Chaque canal est découpé en blocs de 8×8

- Une DCT est appliquée à chacun de ces blocs
- Chaque coefficient est quantifié selon la table de quantification (voir Fig. 2.5) correspondant au facteur de qualité Q $\in \{1,2,3,...,100\}$ et arrondi
- Le tout est ensuite compressé à l'aide d'un codage entropique

C'est surtout l'étape de quantification qui va réduire la quantité d'information, et permettre de réduire la taille du fichier. Cette quantification, si elle est trop agressive va faire apparaitre des artefacts de bloc, des blocs 8×8 visibles (voir Fig. 2.6). C'est ce qui caractérise le JPEG, et qui permet de détecter un certain nombres de choses, comme l'altération d'image [3], les doubles compressions [2] ou encore dans notre cas les relations de parenté.

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

FIGURE 2.5 – Exemple de table de quantification, issu de [12]

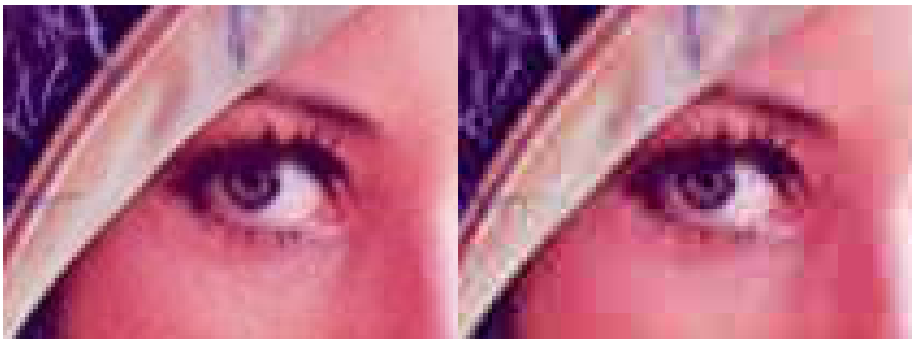


FIGURE 2.6 – Artefacts de blocs, à gauche avec Q=90, à droite avec Q=20

Détection des recompressions JPEG

[8]

Détection des double compressions JPEG avec la même matrice de quantification

[10]

Convergence des blocs lors de compressions successives

L'intérêt de la méthode proposée par Carnein, Schöttle et Böhme [5] est l'estimation du nombre de compression JPEG qu'a pu subir l'image, une estimation qui va au-delà de deux ou trois compression successives [10][], il est en effet possible d'aller jusqu'à plusieurs centaines de compressions. Cette approche se base sur la notion de blocs cycliques.

Lors de compressions successives, une phénomène appelé *convergence de bloc* peut être observé. Un bloc converge lorsque la valeur des pixels du bloc à la compression t est égale à la valeurs des pixels à la compression $t+1$, ce bloc est alors appelé stable [14]. Certains blocs cependant échappent à cet effet et exhibe un phénomène de cycle. C'est à dire que $valeur\ des\ pixels(t) = valeur\ des\ pixels(t+n), n > 1$.

Ces blocs sont nommés blocs compteurs. Ainsi, si un bloc de l'image a un cycle de longueur $l = 9$, on pourra savoir, modulo l, combien de compressions a subi l'image. Et avec plus de blocs, ayant chacun des l différents, il est possible d'aller bien plus loin dans le nombre de compressions. Prenons l'exemple de trois blocs pour lesquels $l_1 = 2$, $l_2 = 5$ et $l_3 = 9$. Le nombre maximum de compressions successives que l'on pourra estimer est le *plus petit commun multiple* des ces longueurs, autrement dit, $ppcm(2, 5, 9) = 90$, soit 90 compressions successives, et pour l'exemple Fig. 2.7, $ppcm(6, 7, 10, 12) = 420$.

Les auteurs proposent deux approches pour les blocs cycliques. La première est de détecter les blocs cyliques à l'aide d'une recherche exhaustive sur l'ensemble des blocs et un certain nombre de compressions puis sélectionner les blocs intéressants. Cette méthode, bien qu'étant la plus simple, et ne modifiant pas l'image, peut cependant rencontrer des problèmes lorsqu'elle comporte un grand nombre de blocs plats, des blocs ayant tous les pixels de la même valeur, ces blocs ne peuvent en effet pas être des blocs cycliques, et il peut alors être difficile de trouver des blocs ayant des cycles de longueur intéressante. La deuxième approche proposée est donc d'insérer des blocs artificiellement créés dans l'image (voir Fig. 2.7).

Cette méthode nécessite de prendre un certain nombre de précaution et requiert une protection (padding) autour des blocs insérés. L'algorithme de sur-échantillonage est en effet parfois amené à utiliser les valeurs des pixels des blocs adjacents pour un meilleur rendu et provoquer un effet de débordement (spill over) des valeurs [4], ce qui modifie le bloc artificiellement créé (dans le vide) et donc modifie son cycle, rendant nulle son information.

Le principal inconvénient de cette méthode, si on utilise les blocs de l'image, et qu'on laisse de coté l'insertion de blocs, est qu'elle se restreint à des images ayant toutes le même facteur de qualité, et en particulier $Q = 100$, pour lequel les cycles sont les plus longs, et donc donnent le plus d'informations. C'est un cas trop spécifique pour notre cadre d'utilisation : les réseaux sociaux, où les facteurs de qualité de compression varient en fonction de l'application utilisée. Cette méthode est en effet plus orientée vers le tatouage et le tracage d'images, et est la plus simple à utiliser lorsque l'on a, à un moment, été en possession de l'image originale et pris soin de noter les blocs intéressants.

2.3 Distances entre distributions

[15]

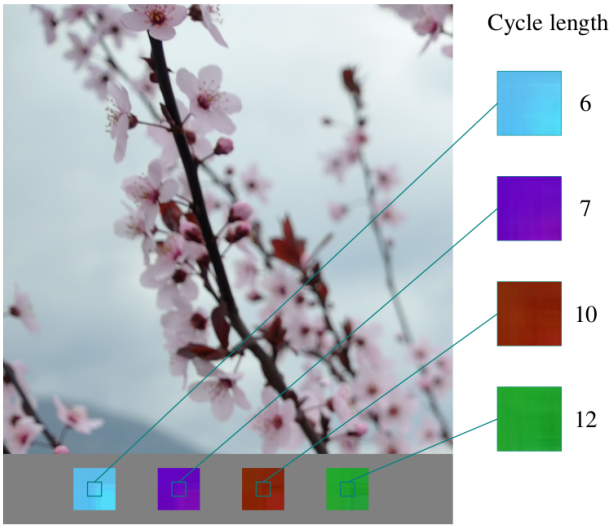


FIGURE 2.7 – Exemple d'insertion de blocs, issu de [5]

3

Notre approche

3.1 Principe

Théorème. *Pour tout couple d'image (A, B), s'il n'est pas possible de prouver que A n'est pas le parent de B, alors il y a une relation parent-enfant entre A et B, $A \rightarrow B$.*

Démonstration. Soit $f(A, B)$ une fonction qui pour tout couple d'image (A, B) détecte à chaque qu'il est présent un marqueur prouvant qu'il n'y a pas de relation de parenté entre A et B. Si $f(A, B)$ ne détecte rien alors c'est que A est le parent de B. □

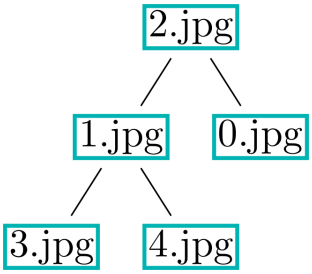
3.2 La construction de l'arbre

On va ainsi pour tout couple d'image de notre ensemble tenter de nier qu'il existe une relation de parenté, cela va permettre d'extraire une matrice binaire, de la taille de l'ensemble d'image, où une case à vrai indiquera une relation de parenté. À noter une fois de plus que ce n'est pas forcément un parent direct mais plutôt un ancêtre.

On peut noter plusieurs choses intéressantes de l'exemple Fig. 3.1. L'image I_2 a toute sa colonne marquée à vrai, c'est à dire que c'est un parent commun à toutes les autres images, et sa ligne n'a que des 0, elle n'a donc aucun parent. On se sert de ce principe pour la reconstruction de l'arbre à partir de la matrice : une image qui n'a pas de parent est la racine. On peut également remarquer les colonnes où toutes les cases sont à 0, cela veut dire que ces images ne sont le parent de personne, et donc sont des feuilles.

L'algorithme de reconstruction

De ces quelques principes nous avons extrait un algorithme.



(a) Arbre de phylogénie

-	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) Matrice de parenté

FIGURE 3.1 – Une arbre de phylogénie et sa matrice de parenté

Data: M a n×n parentage matrix
Result: the root of the tree

```
1  nextRoot ← row with min sum of elements;
2  treeRoot ← nextRoot;
3  forall the 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
```

Il prend en entrée la matrice de parenté détaillée précédemment et retourne la racine de l'arbre. La philosophie de l'algorithme est qu'une image n'ayant aucun parent est la racine, et ce de manière récursive grâce à la structure d'arbre.

À chaque tour de boucle, une image est sélectionnée comme la racine (lignes 1 et 11) et nommée *root*, elle est retirée (ligne 7) des ancêtres des autres images si c'est un ancêtre. Si ces autres images n'ont plus d'ancêtre (ligne 8), c'est que *root* était le parent direct de l'image en train d'être traitée, cette image est donc ajoutée comme enfant de *root* (ligne 9). La ligne 5 permet de ne traiter qu'une fois chaque image comme racine potentielle.

Cet algorithme a une complexité de $O(n^2)$. Il y a deux boucles imbriquées, et si les sommes sont calculées une une seule fois au début et mises à jour à chaque fois qu'un parent est enlevé, il n'y a pas de boucle supplémentaire augmentant la complexité.

Le but est donc de trouver la fonction permettant de détecter le marqueur prouvant qu'il n'y a pas de parenté.

Il nous a semblé important, même dans le cadre de l'étude bibliographique, d'élaborer cet algorithme. Il est en effet assez simple mais néanmoins indispensable à notre méthode, et aurait pu orienter différemment nos recherche et notre méthode s'il n'avait pas été imaginé. Le reste du stage sera consacré à élaborer la fonction énoncée précédemment, et c'est là que la majorité du travail se concentrera.

Cette approche a pour avantage de réduire l'évaluation d'un arbre de phylogénie à partir d'un ensemble d'images à l'évaluation binaire de parenté entre deux images.

4

Conclusion

Bibliographie

[1] Herbert Bay et al. “Speeded-up robust features (SURF)”. In : *Computer vision and image understanding* 110.3 (2008), p. 346–359.

[2] Tiziano Bianchi et Alessandro Piva. “Detection of nonaligned double JPEG compression based on integer periodicity maps”. In : *Information Forensics and Security, IEEE Transactions on* 7.2 (2012), p. 842–848.

[3] Tiziano Bianchi et Alessandro Piva. “Image forgery localization via block-grained analysis of JPEG artifacts”. In : *Information Forensics and Security, IEEE Transactions on* 7.3 (2012), p. 1003–1017.

[4] Matthias Carnein, Pascal Schottle et Rainer Bohme. “Forensics of high-quality JPEG images with color subsampling”. In : *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on*. IEEE. 2015, p. 1–6.

[5] Matthias Carnein, Pascal Schöttle et Rainer Böhme. “Telltale Watermarks for Counting JPEG Compressions”. In : *Proceedings of the Electronic Imaging 2016*. Publication status : Published. San Francisco, USA, 2016.

[6] Zanoni Dias, Anderson Rocha et Siome Goldenstein. “First steps toward image phylogeny”. In : *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on*. IEEE. 2010, p. 1–6.

[7] Zanoni Dias, Anderson Rocha et Siome Goldenstein. “Image phylogeny by minimal spanning trees”. In : *Information Forensics and Security, IEEE Transactions on* 7.2 (2012), p. 774–788.

[8] Xiaoying Feng et Gwenaël Doërr. “JPEG recompression detection”. In : *IS&T/SPIE Electronic Imaging*. International Society for Optics et Photonics. 2010, 75410J–75410J.

[9] Martin A. Fischler et Robert C. Bolles. “Random Sample Consensus : A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In : *Commun. ACM* 24.6 (juin 1981), p. 381–395. issn : 0001-0782. doi : 10.1145/358669.358692. url : http://doi.acm.org/10.1145/358669.358692.

[10] Fangjun Huang, Jiwu Huang et Yun Qing Shi. “Detecting double JPEG compression with the same quantization matrix”. In : *Information Forensics and Security, IEEE Transactions on* 5.4 (2010), p. 848–856.

[11] Alexis Joly, Olivier Buisson et Carl Frélicot. “Content-based copy retrieval using distortion-based probabilistic similarity search”. In : *Multimedia, IEEE Transactions on* 9.2 (2007), p. 293–306.

[12] *JPEG*. url : `https://fr.wikipedia.org/wiki/JPEG`.

[13] Lyndon Kennedy et Shih-Fu Chang. “Internet image archaeology : automatically tracing the manipulation history of photographs on the web”. In : *Proceedings of the 16th ACM international conference on Multimedia*. ACM. 2008, p. 349–358.

[14] ShiYue Lai et Rainer 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, p. 3028–3032.

[15] Marina A Oikawa et al. “Distances in multimedia phylogeny”. In : *International Transactions in Operational Research* (2015).

[16] *Phylogénie*. url : `https://fr.wikipedia.org/wiki/Phylogen%C3%A8se`.

[17] Gregory K Wallace. “The JPEG still picture compression standard”. In : *Consumer Electronics, IEEE Transactions on* 38.1 (1992), p. xviii–xxxiv.