La phylogénie des images dans les réseaux sociaux

Étude bibliographique

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

Auteur
Noé Le Philippe

Superviseurs
William Puech
Christophe Fiorio

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

version 1 23 mars 2016

Résumé

Ce stage de master.

Abstract

This master thesis.

Table des matières

1

Introduction

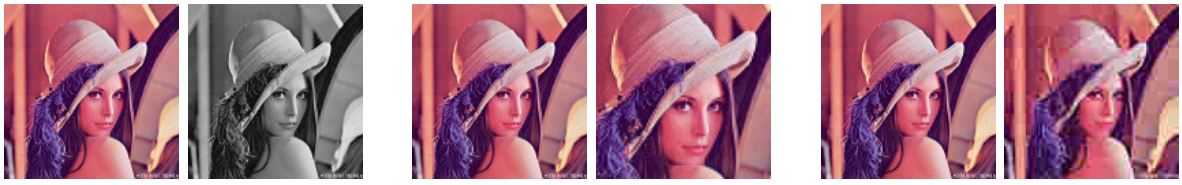
La phylogénie, en sciences naturelles, est définie comme l'étude des relations de parenté entre êtres vivants. Et c'est exactement de cela qu'il s'agit dans le cas des images, l'étude des relations de parenté entre images. [17]
À l'ère du numérique et des réseaux sociaux, il n'a jamais été aussi simple de partager des idées et du contenu. À chaque partage cependant, l'information peut être modifiée plus ou moins fortement. Les images, puisque c'est là notre sujet d'étude, peuvent avoir subi un certain nombre de transformations et modifications avant d'être publiées sur les réseaux sociaux. C'est dans ce contexte que nous allons intervenir afin tenter de reconstituer la phylogénie des images. Il peut être difficile de différentier une image modifiée de l'originale, et de savoir laquelle est l'originale. Alors que cette détection est cruciale, surtout dans un monde où l'information peut être facilement falsifiée par tout un chacun. Les applications sont multiples et variées, et ne se cantonnent pas à la détection et la discrimination d'images altérées. Il est également poissble de se servir de la phylogénie des images pour optimiser de l'espace de stockage en ne gardant que l'image originale ou encore suivre la diffusion et l'évolution des idées sur les réseaux sociaux.

Dans ce chapitre nous commencerons par définir ce que sont que les near-duplicates images, puis nous détaillerons la notion d'arbre phylogénétique et enfin nous expliquerons pourquoi nous nous sommes restreints à la compression JPEG dans le cadre de cette étude. Nous ferons un état de l'art dans le chapitre 2 où présenterons les différentes approches pour calculer l'arbre de phylogénie, nous y traiterons également la détection des recompressions ou l'estimation du nombre de recompressions. Nous présenterons ensuite notre approche dans le chapitre 3, où un algorithme de reconstruction d'arbre sera détaillé, en plus du théorème qui sera la base de la suite du stage. Nous finirons par conclure chapitre 4.

1.1 Near-duplicate images (NDI)

Nous travaillons sur un ensemble d'images, toutes similaires visuellement, et au milieu de cet ensemble 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. [12] définissent la notion de near-duplicate comme suit : $I_{n+1} = T(I_n), T \in \mathcal{T}$ où I_n est l'image parent, I_{n+1} est l'image à la génération suivante $n+1$, l'image enfant et $\mathcal{T}$ est un ensemble de transformations autorisées, I_{n+1} et I_n sont alors des NDI. Dans le cas général, $\mathcal{T} = \{$*resampling, cropping, affine warping, color changing, lossy compression*$\}$, la figure 1.1 montre un exemple de near-duplicates. Le *resampling*, ou rééchantillonage en français revient à changer le nombre de pixels de l'image, le *crop* est illustré figure 1.1b, *affine warping* englobe tout ce qui peut être translation ou rotation, *color changing* concerne tout ce qui va changer la couleur, comme le changement de contraste par exemple, ou le passage au noir et blanc (figure 1.1a), et enfin *lossy compression* est la compression avec pertes, une dégradation de l'image pour réduire son poids. 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 croppée 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. Ces transformations peuvent évidemment se composer, et une image enfant peut être le résultat de plusieurs transformations.

Note. *Nous utiliserons les notions 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.*



(a) couleur → noir et blanc (b) image entière → image croppée (c) image → image compressée

FIGURE 1.1 – Exemple de near-duplicates

1.2 Arbre phylogénétique (Image Phylogeny Tree - IPT)

L'arbre phylogénétique est l'arbre représentant les relations de parenté entre les différentes images. Il est extrait d'un ensemble de NDI, et constitue l'objectif final de l'application. La figure 1.2 illustre la construction de l'IPT à partir d'un ensemble de NDI. Le passage d'une génération à l'autre, autrement dit d'un noeud à son fils, se fait à travers la transformation $I_{n+1} = T(I_n)$, ainsi, une image I_{n+1} et son parent I_n sont des NDI, alors qu'une image $I_{n+1,i}$ et sa soeur $I_{n+1,j}$, $i \neq j$ ne le sont pas (figure 1.3).

La reconstruction de l'arbre se concentre autour de deux problèmes principaux. Le premier est l'identification de la racine ($n = 0$), et le second est l'estimation du reste de l'arbre, et en

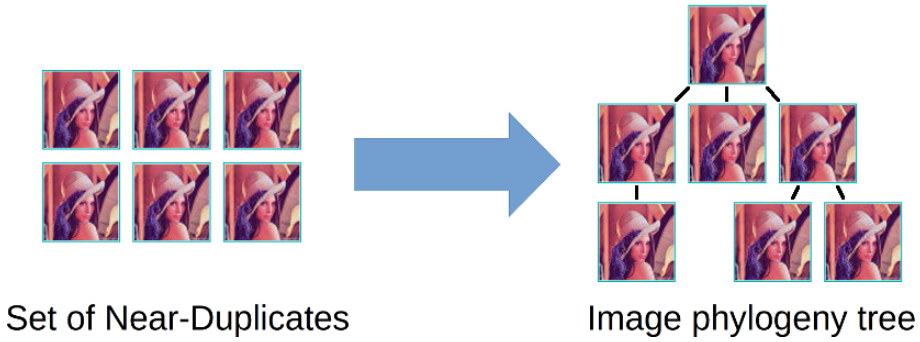


FIGURE 1.2 – Passage d'un ensemble de NDI à un arbre phylogénétique

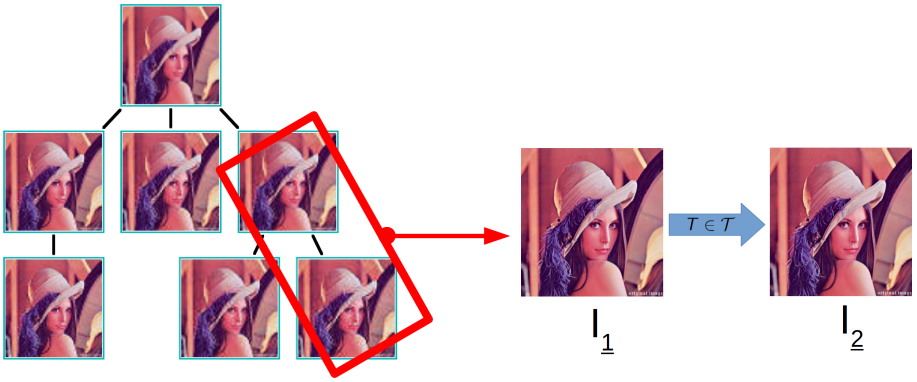


FIGURE 1.3 – Passage d'une image parent à l'image enfant

particulier positionner précisément chacune des images dans leur génération respective (valeur de n). 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 la génération est proche de la racine, c'est à dire plus n est petit, moins l'image a subi de transformations, et donc moins elle est altérée. Avec comme cas particulier $n = 0$, l'image est alors la génération 0 : la racine. Notons que si la racine est mal identifiée, une image pourra être à tort marquée comme altérée. Nous déduirons, à tort, qu'une image n'a pas été altérée. Il n'est pas toujours garanti que la totalité des images de l'arbre original soit présente, l'identification précise de la génération peut alors se révéler complexe. Certaines transformations peuvent être mineures, et difficile à détecter, dans ce cas deux images de deux générations proches n et $n+k$, avec k petit ($k = 1, 2$ ou 3 par exemple), peuvent être identifées à tort comme appartenant à la même génération. En conclusion, il est clair que l'arbre ne sera qu'une estimation, et sauf dans des cas idéaux, ne sera pas l'arbre original.

1.3 Pourquoi se restreindre à la compression comme transformation ?

Dans le cadre du stage, nous proposons de réduire l'espace des transformations à $\mathcal{T} = \{lossycompression\}$ avec JPEG comme algorithme de compression. L'étude et la détection des recompressions JPEG est en effet un sujet suffisament vaste et complexe pour que cela soit la seule transformation étudiée dans le cadre de ce stage de recherche. C'est de plus un sujet qui à notre connaissance n'a pas été traité dans le cadre de la phylogénie des images, et qui est largement étudié dans le domaine du forensics (criminalistique en français). L'étude et la détection des recompressions est en effet un sujet vaste, la figure 1.4 détaille l'étendue du problème. La compression JPEG sera détaillée dans le chapitre 2 en section 2.2. Aussi la seul caractéristique dont nous parlerons dans cette section est le facteur de qualité Q. $Q \in [1..100]$ et plus Q est grand, plus l'image sera de bonne qualité, mais moins elle sera compressée. Nous sommes donc en présence de trois cas pour les recompressions successives : le premier est le cas où tous les Q successifs sont égaux, lors du second cas, l'image est plus dégradée à chaque recompression, et dans le troisième cas, nous n'avons aucune idée sur la manière dont l'image a été recompressée par rapport à son parent. Elle peut avoir été recompressée avec la même qualité, ou une meilleure qualité, ou encore une moins bonne qualité.

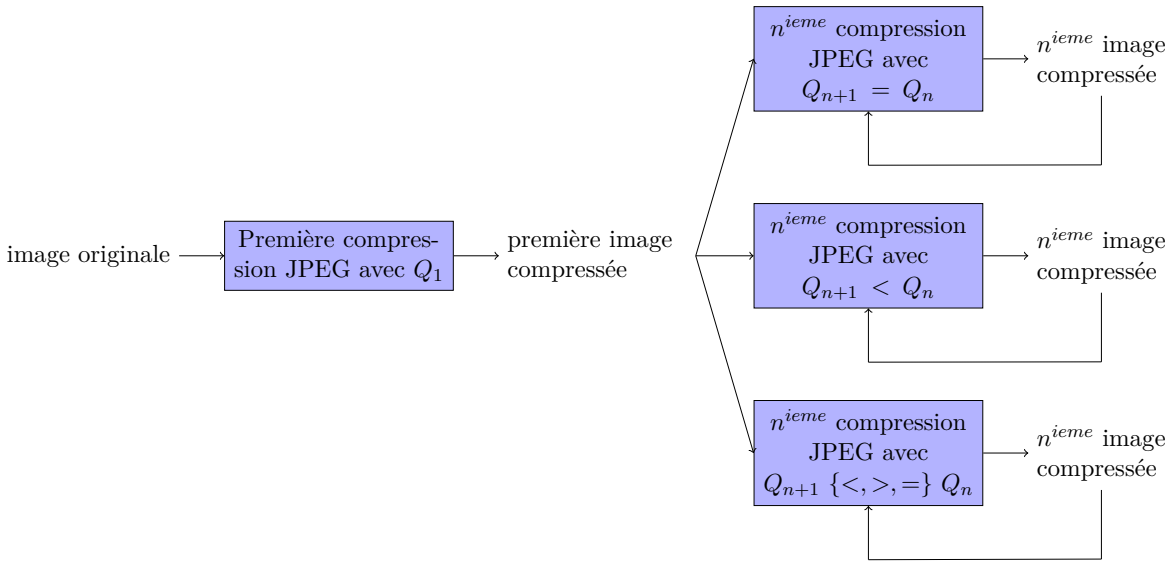


FIGURE 1.4 – Exemple des différents scénarios de recompression d'images

2

État de l'art

Ce chapitre tentera de faire un état de l'art des différentes techniques de calcul d'un arbre phylogénétique, puis traitera des différentes méthodes permettant d'extraire des informations de plusieurs compressions successives, et enfin présentera quelques calculs de distances.

2.1 Étude de l'arbre phylogénétique

La Visual Migration Map (VMM)

L'article de Kennedy et Chang est à notre connaissance le premier article concernant vraiment notre sujet [14]. Lesauteurs 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 figure 2.1) qui est en fait un arbre de phylogénie.

Kennedy et Chang partent du principe que les transformations sont directionnelles, c'est à dire qu'il n'est possible d'aller que d'une image moins transformée vers 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. La figure 2.2 illustre ce principe par un exemple.

Un graphe est ensuite 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 figure 2.3.

Image Phylogeny Tree (IPT)

Tout comme l'approche présentée précédemment, Dias et al. [7][8] proposent une approche basée sur le contenu de l'image. Cela consiste a mapper une image sur le domaine d'une autre, pour

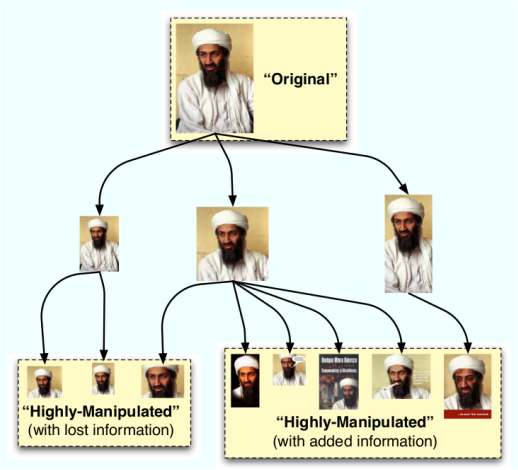


FIGURE 2.1 – Exemple de VMM, issu de [14].

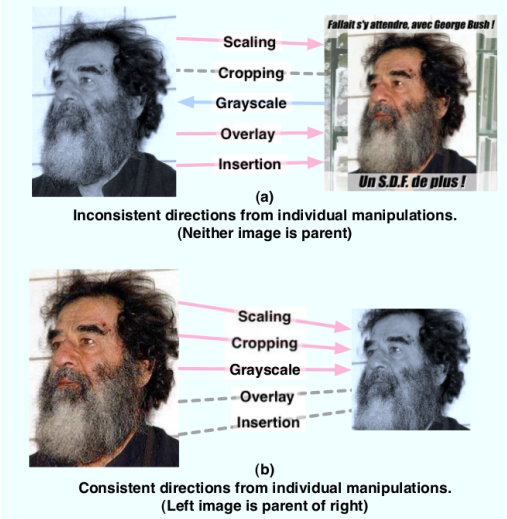


FIGURE 2.2 – Exemple de direction des transformations, issu de [14].

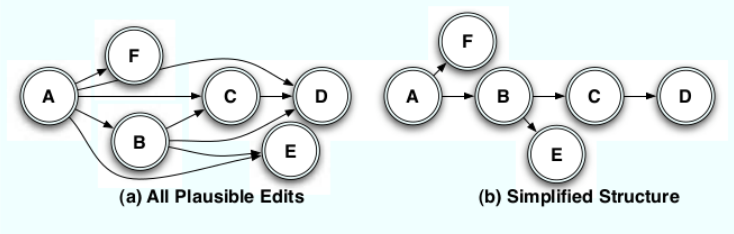


FIGURE 2.3 – Exemple de simplification de graphe, issu de [14].

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.

Les images sont comparées à l'aide d'une fonction de dissimilarité, ou *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. I_m et I_n 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 $d(I_m, I_n) \neq d(I_m, I_n)$, ce qui est parfaitement logique, en plus d'être nécessaire, sachant que les transformation sont directionnelles, comme expliqué précédemment.

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

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

1. Trouver des points caractéristiques (SURF [1]),
2. Filtrer les points et estimer les paramètres de transformation affines tels que les translations, rotations et rééchantillonages avec RANSAC [10],
3. Calculer la moyenne et la variance de chaque canal couleur de I_n pour normaliser les couleurs de I_m,
4. Compresser les résultats des deux étapes précédentes avec la table de quantification de I_n

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

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

La distance $d(.,.)$ est donc appliquée à tous les couples d'images de l'ensemble, pour créer une matrice de dissimilarité, ou *dissimilarity matrix*, une matrice de taille $n \times n$ qui sera ensuite donnée comme entrée à leur algorithme de reconstruction d'arbre, Oriented Kruskal. Cet algorithme est expliqué de manière exhaustive dans [8]. Dans le chapitre 3, nous proposons une approche différente, notre reconstruction sera donc également différente.

En plus de proposer une approche pour reconstituer l'arbre de phylogénie, Dias et al. proposent une approche pour comparer deux arbres, et donc évaluer notre arbre reconstruit s'il est comparé

avec la vérité terrain. Il 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 recompressions 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 à partir d'un ensemble de NDI, où $\mathcal{T} = \{lossy\ compression\}$. L'important est plutôt de savoir combien de fois, et à partir de quelle image I_n a été compressée, et donc pouvoir en déduire sa distance avec la racine. La majorité des articles dans le domaine du forensics 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 forensics 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.

La compression JPEG

Une rapide introduction sur la compression 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 que rarement utilisé. Pour de plus amples détails, le lecteur se dirigera vers [18].

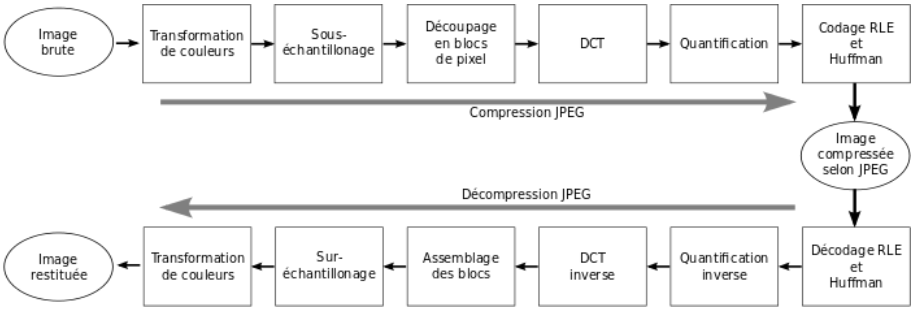


FIGURE 2.4 – Étapes de compression et décompression, issues de [13].

La figure 2.4 liste toutes les étapes permettant de passer d'une image non compressée à une image compressée ainsi que le processus inverse. 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 pixels,
- Une DCT est appliquée à chacun de ces blocs pour avoir une matrice de 8×8 coefficients,
- Chaque coefficient est quantifié selon une table de quantification (voir figure 2.5) correspondant au facteur de qualité Q $\in \{1,2,3,...,100\}$ et arrondi à l'entier le plus proche,
- 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 donc permettre de réduire la taille du fichier. Cette quantification, si elle est trop agressive va faire apparaître des artefacts de bloc, des blocs 8×8 visibles (voir figure 2.6). C'est ce qui caractérise le JPEG, et qui permet de détecter un certain nombres de choses, comme l'altération d'images [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 [13].

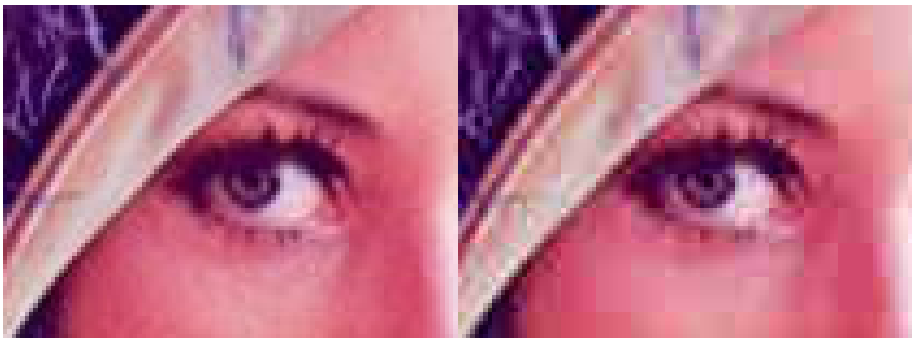


FIGURE 2.6 – Artefacts de blocs, à gauche avec Q=90, à droite avec Q=20

Estimation de la matrice de compression primaire dans les images JPEG doublement compressées

Lukáš et Fridrich [16] proposent dans leur approche non seulement de détecter si une image est doublement compressée, mais également de d'estimer les paramètres de la première compression. Pour bien comprendre la suite de leurs travaux, il convient d'expliquer plus en détail l'étape de quantification de JPEG.

La quantification se fait de la manière suivante :

$$F^*(u,v) = \left\lfloor \frac{F(u,v) + \left\lfloor \frac{Q(u,v)}{2} \right\rfloor}{Q(u,v)} \right\rfloor, \tag{2.2}$$

où $F(u,v)$ est la valeur aux indices u et v dans la matrice 8×8 de coefficients DCT et Q(u,v) et la valeur dans la table de quantification (figure 2.5) à ces même indices.

Le calcul inverse est :

$$F'(u,v) = F^*(u,v) * Q(u,v). \tag{2.3}$$

où $F(u,v)$ est la valeur aux indices u et v dans la matrice 8×8 de coefficients DCT et Q(u,v) et la valeur dans la table de quantification (figure 2.5) à ces même indices.

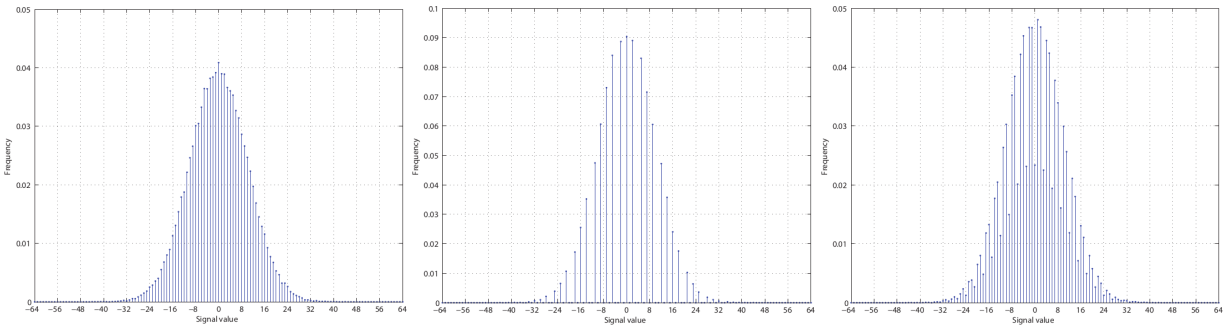
Nous voyons bien à partir des équations 2.2 et 2.3 que le résultat après décompression sera un multiple de $Q(u,v)$ et que toutes les valeurs n'étant pas des multiples seront absentes.

La double compression est donc le fait de compresser deux fois une image, et donc de lui faire subir deux fois toutes les étapes de compression et décompression, avec tous les arrondis et troncage que cela peut comporter. On notera, pour la double compression, Q^1 comme la matrice primaire, c'est à dire la table de quantification ayant servi à faire la première compression (inconnue donc), et Q^2 comme la matrice secondaire, ou la table de quantification actuelle, disponible dans l'en-tête du fichier JPEG. Les auteurs limitent la double compression aux cas où $Q^1 \neq Q^2$.

Lors de la première compression, les coefficients sont quantifiés avec Q^1, ce qui veut dire que les coefficients sont des multiples de Q^1. Cependant, lorsque l'image est repassée du domaine fréquentiel au domaine spatial grâce à la DCT inverse, un certain nombre d'arrondis et de troncages se produisent pour rester dans l'intervalle [0..255]. Lors de la seconde compression, les coefficients DCT sont calculés à partir des ces valeurs inexactes, et donc ne seront pas multiples de Q^1, mais se concentreront autour. Ces nouveaux coefficients DCT seront ensuite quantifiés pour former la nouvelle image, doublement compressée.

Les auteurs ayant laissé de coté le cas où $Q^1 = Q^2$, nous sommes en présence de deux cas : $Q^1 > Q^2$ et $Q^1 < Q^2$. Chacun des cas à des artefacts distincts et reconnaissables. La figure 2.7 illustre un exemple de distributions ayant subi des quantifications, la figure 2.7a n'a subi qu'une seule quantification, la figure 2.7b correspond au cas où $Q^1 > Q^2$ et la figure 2.7c correspond au deuxième cas. Un autre cas est délaissé, c'est celui où Q^1 est facteur Q^2, ce qui est assez logique, ils auront les même multiples, et il sera donc impossible de détecter quoi que ce soit, du moins en utilisant la méthode proposée ici.

L'estimation de la première table de quantification se limite aux basses fréquences, les hautes fréquences étant plus fortement quantifiés (souvent jusqu'à 0), elles contiennent moins d'informations et donc rendent leur estimation difficile.



(a) quantifié une fois (b) re-quantifié, $Q^1 > Q^2$ (c) re-quantifié, $Q^1 < Q^2$

FIGURE 2.7 – Exemple de quantifications sur l'histogramme d'une distribution normale avec des classes de largeur 1, issu de [9].

Leur technique consiste à se munir d'une série de candidats pour la matrice primaire, notés $\{Q^{1,1}, ..., Q^{1,n}\}$. Ces tables de quantification viennent des tables utilisées dans les appareils photos, les téléphones portables ou les différentes implémentations de la compression JPEG. Parmi ces $\{Q^{1,1}, ..., Q^{1,n}\}$, le meilleur candidat, et donc la table la plus probable sera la table de quantification pour laquelle la différence entre l'histogramme de l'image et l'histogramme de l'image compressée avec $Q^{1,k}$ sera la plus faible.

Détection des doubles compressions JPEG avec la même matrice de quantification

Comme expliqué section 1.2, nous tentons dans un premier temps d'identifier la génération n d'une image I_n. Une tâche qui peut s'avérer difficile si les transformations sont minimes. En sachant que deux images I_m et I_n sont extrêmement similaire, sommes-nous en présence d'une double compression avec la même matrice de quantification, donc l'une est le parent de l'autre, ou sommes-nous dans le cas où les images sont soeurs, et une matrice de quantification identique a été utilisée pour les générer ?

Nous avons choisi d'inclure l'article de Huang et al. [11] car il traite précisément de ce problème.

Convergence des blocs lors de compressions successives

L'intérêt de la méthode proposée par Carnein et al. [5] est l'estimation du nombre de compressions JPEG qu'a pu subir l'image, une estimation qui va au-delà de deux ou trois compression successives [11][16], 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 [15]. Certains blocs cependant échappent à cet effet et exhibe un phénomène de cycle. C'est à dire que la valeur des pixels à la compression t est égale à la valeur des pixels à la compression $t + n, n > 1$. Le deuxième type de bloc à échapper à ce phénomène est les blocs plats, ayant tous les pixels de la

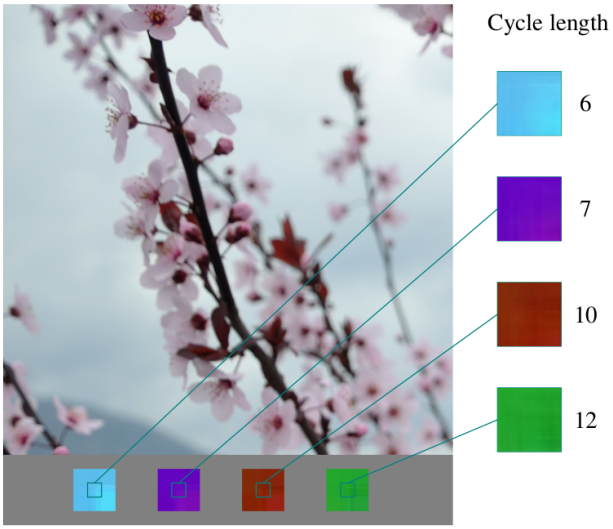


FIGURE 2.8 – Exemple d'insertion de blocs, issu de [5].

même valeur, ils ne peuvent pas être cycliques puisque tous les coefficients DCT sauf le premier sont nuls.

Ces blocs sont nommés blocs compteurs. Ainsi, si un bloc de l'image a un cycle de longueur $l = 9$, il sera possible de 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 qu'il sera possible d'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 illustré par la figure 2.8, $ppcm(6, 7, 10, 12) = 420$.

Les auteurs proposent deux approches pour les blocs cycliques. La première consiste à détecter les blocs cycliques 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, il peut en effet 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 figure 2.8).

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 ce sont les blocs présents dans l'image qui sont utilisés, et que la technique d'insertion de blocs est laissée de coté, 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 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 traçage d'images.

2.3 Distances entre distributions

Dans l'approche que nous proposons de développer lors de ce stage, il nous sera nécessaire de calculer des distances entre des distributions. C'est la raison pour laquelle nous avons décidé passer en revue un certain nombre de distance dans l'état de l'art.

Commençons par définir la différence entre une distance et une divergence. Une distance, contrairement à une divergence, est symétrique, c'est à dire que $distance(I_m, I_n) = distance(I_n, I_m)$, et elle vérifie l'inégalité triangulaire. Une distance est parfaite pour mesurer la proximité entre deux objets. Elle est cependant dans l'incapacité de nous donner des indications sur une direction.

Nous travaillons avec une relation asymétrique (relation parent-enfant), une divergence semble donc être mieux adaptée pour estimer la direction de cette relation (voir chapitre 3).

Cependant, comme expliqué chapitre 2 section 2.1, Dias et al. mappent une image dans le domaine d'une autre, ce qui est en soit une opération asymétrique, et donc qui permettrait d'utiliser une distance. C'est pourquoi nous traiterons aussi bien les distances que les divergences.

Il existe un grand nombre de mesure de distances [6], aussi, nous ne parlerons que des distances entre distributions les plus utilisées. Nous n'entrerons cependant pas dans les détails de ces calculs de distance et quelle distance est la mieux adaptée à quel type de donnée, nous utiliserons ces distances comme des boites noires et choisirons de manière experimentale la plus adaptée. Nous allons donc donner pour chaque distance son nom et sa formule.

Dans la suite de cette section, P et Q seront des densité de probabilité.

Les distances

La distance euclidienne :

$$D_{euc}(P,Q) = \sqrt{\sum_{i=1}^{d} |P_i - Q_i|^2} \tag{2.4}$$

La distance de Bhattacharyya :

$$D_B(P,Q) = -ln \sum_{i=1}^{d} \sqrt{P_i Q_i} \tag{2.5}$$

L'index de Jaccard est donné comme :

$$S_{Jac}(P,Q) = \frac{\sum_{i=1}^{d} P_i Q_i}{\sum_{i=1}^{d} P_i^2 + \sum_{i=1}^{d} Q_i^2 - \sum_{i=1}^{d} P_i Q_i} \tag{2.6}$$

Il correspond à la taille de l'intersection entre deux distributions divisé par la taille de leur union. La distance de Jaccard est :

$$D_{Jac}(P,Q) = 1 - S_{Jac}(P,Q) = \frac{\sum_{i=1}^{d}(P_i - Q_i)^2}{\sum_{i=1}^{d} P_i^2 + \sum_{i=1}^{d} Q_i^2 - \sum_{i=1}^{d} P_i Q_i} \tag{2.7}$$

Les divergences

La divergence de Kullback–Leibler :

$$D_{KL}(P||Q) = \sum_{i=1}^{d} P_i \; log \; \frac{P_i}{Q_i} \tag{2.8}$$

La K divergence :

$$D_{kdiv}(P||Q) = \sum_{i=1}^{d} P_i \; log \; \frac{2P_i}{P_i + Q_i} \tag{2.9}$$

3

Notre approche

Ce chapitre sera consacré à présenter notre approche, nous y détaillerons notre approche, formalisée par un théorème, puis nous présenterons un algorithme permettant de reconstruire l'arbre de phylogénie.

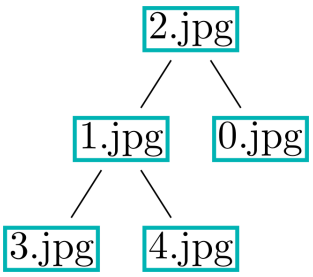
3.1 Principe et théorème

Pour tout couple d'images de notre ensemble, nous allons 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.

Nous pouvons noter plusieurs choses intéressantes de l'exemple figure 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 donc la racine. Nous pouvons également remarquer les colonnes où toutes les cases sont à 0, cela veut dire que ces images ne sont les parents de personne, et donc sont des feuilles.

Nous avons formalisé ce principe en un théorème :

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



(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.

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

3.2 L'algorithme de reconstruction

De ces quelques principes nous avons extrait un algorithme.

```
   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
```

Algorithm 1: Construction de l'arbre

L'algorithme 1 prend en entrée la matrice de parenté détaillée précédemment et retourne la racine de l'arbre. La philosophie de cet 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 itération, 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, en plus d'appliquer cet algorithme à plusieurs centaines d'exemples avec des configurations différentes pour pouvoir en tirer une analyse et une synthèse.

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 et al. “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 et al. “Telltale Watermarks for Counting JPEG Compressions”. In : *Proceedings of the Electronic Imaging 2016*. Publication status : Published. San Francisco, USA, 2016.

[6] Sung-Hyuk Cha. “Comprehensive survey on distance/similarity measures between probability density functions”. In : *City* 1.2 (2007), p. 1.

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

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

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

[10] 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.

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

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

[13] *JPEG*. URL : `https://fr.wikipedia.org/wiki/JPEG`.

[14] 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.

[15] 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.

[16] Jan Lukáš et Jessica Fridrich. “Estimation of primary quantization matrix in double compressed JPEG images”. In : *Proc. Digital Forensic Research Workshop*. 2003, p. 5–8.

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

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