CODAGE DE GRAPHEMES ET COMPRESSION SANS PERTE D'IMAGES DE MANUSCRITS ANCIENS

Abdelâali HASSAINE

Master Recherche II
IGI : Spécialité Informatique Graphique et Image
abdelaali.hassaine@insa-lyon.fr

Encadrement :
Véronique EGLIN & Stéphane BRES
Affiliation : LIRIS Axe 2 (INSA de Lyon – Jules Verne)

Remerciements

Je tiens tout d'abord à remercier le Professeur Hubert EMTOZ de m'avoir accueilli au sein de son équipe.

Ce fut vraiment un grand plaisir et une grande chance d'avoir été encadré par Véronique EGLIN et Stéphane BRES. Qu'ils trouvent ici l'expression de ma profonde reconnaissance pour leur aide précieuse et leurs conseils constructifs. Je les remercie également pour la confiance qu'ils ont su m'accorder jusqu'à la dernière minute.

Je tiens à remercier tout particulièrement Frank LE BOURGEOIS d'avoir été très serviable. Son aide, ses conseils et ses discussions m'ont été d'une grande utilité.

Ce fut également un plaisir d'avoir connu Djamel GACEB, je le remercie pour ces idées, ces discussions et sa sympathie.

Merci à toute l'équipe du laboratoire LIRIS.

Jamais les mots ne me suffiront pour remercier papa et maman qui me manquent beaucoup.

Grand merci à Redouane, Yazid, Walid, Amina, Fawzi, Nadjima, Bachir, Mounia, Anouar, Yamina, Omar et à ceux que j'oublie encore… merci à vous tous.

Abdelâali

Sommaire

CODAGE DE GRAPHEMES ET COMPRESSION SANS PERTE D'IMAGES DE MANUSCRITS ANCIENS

Résumé

Les usagers des bibliothèques numériques souhaitent avoir un accès rapide aux images de documents. La compression de ce type d'image est devenue pour cette raison un besoin incontournable. Bien que certaines méthodes de compression donnent de bons résultats sur des textes imprimés, aucun des formats de compression existants ne s'est montré efficace sur les manuscrits anciens très dégradés. Nous proposons une nouvelle méthodologie de compression permettant de structurer les informations en couches du plus au moins important et permettant ainsi un affichage progressif des informations. Notre méthode est basée essentiellement sur l'extraction de formes similaires dans le squelette du tracé. A partir de certaines formes de base et d'une variété de transformations géométriques nous générons les autres formes. Les paramètres permettant la reconstruction de l'image originale sont ensuite mémorisés efficacement en exploitant les spécificités de ce type d'images. Les premiers résultats sont très encourageants et témoignent de l'efficacité de notre méthode. Nous proposons également des techniques de recherche de similarité donnant des résultats très intéressants et pouvant être utiles pour une variété d'applications sur les images de documents.

Mots clés : compression sans perte des manuscrits du patrimoine, codage, similarités, séparation en couches d'informations

Abstract

The users of the numerical libraries wish to have a fast access to the images of documents. Although some compression methods give good results on printed texts, the existing compression formats are not effective on much degraded old manuscripts. We propose here a new methodology of compression which allows to structure information in layers for a progressive information broadcast. Our method basically lies on the extraction of different similarities in the skeleton of handwriting shapes. From particular selected shapes (in the skeleton) and with a variety of geometrical transformations we generate the other shapes. The parameters which allow the original image rebuilding are then effectively stored by the exploitation of specificities of this type of images. The first results are very promising and testify to the effectiveness of our method. We also propose techniques for similarities retrieval which will be introduced to our compression system and which can be useful for a variety of applications on the handwriting documents images.

Keywords: lossless compression of manuscripts of our cultural inheritage, coding, similarity retrieval, multi- layers decomposition.

I. INTRODUCTION

I.1 Contexte de l'étude

A une époque où le papier n'est plus le support exclusif de l'écrit, où les coûts des numériseurs et des disques ne cessent de décroître et où les écrans d'ordinateurs sont de plus en plus privilégiés pour la consultation des documents, plusieurs besoins sont devenus incontournables : l'élaboration d'objets documentaires numériques, leur actualisation et leur exploitation, tout en permettant une navigation simplifiée, une recherche performante d'information et enfin un partage des connaissances avec les autres lecteurs [DEB00, BOT00, LEB04]. L'ensemble de ces besoins correspond en particulier aux objectifs de nombreux projets de numérisation de corpus de documents anciens du patrimoine qui émergent depuis une dizaine d'année et pour lesquels il n'existe à ce jour pas d'autres solutions pour la sauvegarde et la diffusion que l'exploitation numérique en mode image.

L'idée de mener à bien ces campagnes de numérisation a commencé à émerger dans les années 60. En effet, l'essor de « l'outil informatique » permettait d'entrevoir des perspectives attrayantes : sauvegarder les documents, constituer des bases d'images et de textes communes à plusieurs bibliothèques, concevoir des outils d'aide à la manipulation des données textuelles… Si l'objectif premier était avant tout de sauvegarder le patrimoine (dont une partie commençait à souffrir des outrages du temps) et de confectionner des catalogues d'images consultables, on est encore loin d'avoir trouvé des solutions satisfaisantes pour le stockage, la transmission et la mise en valeur de ce patrimoine. C'est dans un but de «développement des supports numériques permettant la libre manipulation de l'information » que l'ABU (Association des Bibliophiles Universels) a mis en place une bibliothèque d'un nouveau genre instaurant un accès à des milliers de documents en ligne désormais consultable par tous. Le succès de Gallica est la preuve de l'intérêt de ces campagnes de numérisations puisque quotidiennement plus de 20.000 connections sont référencées et qu'en 2002 plus d'1,4 TO d'images a été téléchargé.

La création des ces volumineux espaces numériques amène très vite à la question suivante : comment offrir aux usagers un accès facile et une consultation rapide de ces données, ainsi des possibilités d'interrogation à partir de sites distants du lieu de stockage des sources ? Cette question soulève naturellement le problème de l'accessibilité des données, la question de la transmission facilitée et ainsi la question de la compression des images sources.

L'ensemble de ces besoins, en nécessitant une compression plus efficace des images va conduire à une redéfinition de formats de données qui autoriseront des interrogations à distance de leurs contenus [LEB03].

De nombreux projets de numérisation se sont ainsi vus concerner par de tels besoins. Le travail de ce MASTER s'inscrit précisément dans le cadre du projet d'ACI (Action Concertée Incitative) MADONNE (MAsse de DONnées issues du PatrimoiNE) autour de la sauvegarde et de la valorisation de données patrimoniales, [MAD05]. L'aspect « masse de données » est considérée ici sous l'angle des collections d'ouvrages numérisés, qui constitueront à très court terme des entrepôts gigantesques de données, représentés sous forme d'images scannées. La génération de ces entrepôts de données, présentés sous forme de collections de documents hétérogènes faiblement structurés soulève le problème de l'accès, de la recherche d'information et de la navigation au sein de ces corpus.

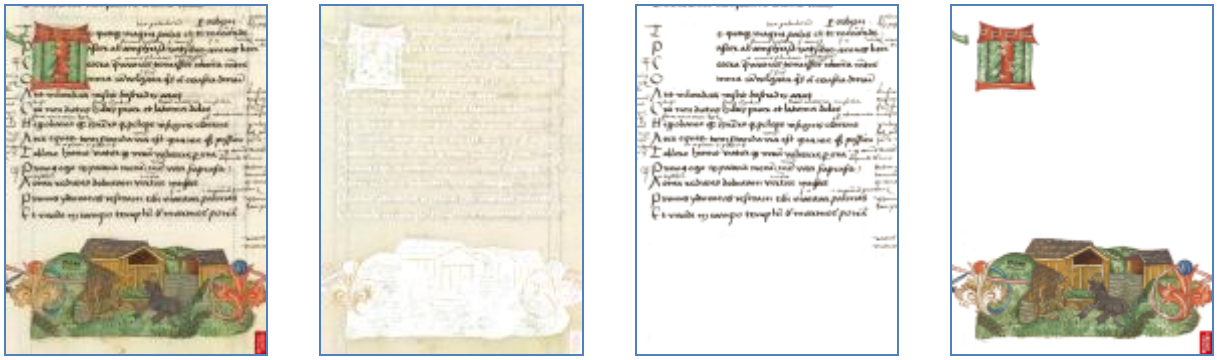
La consultation en mode image des documents patrimoniaux qui est visé par la plupart des projets de numérisation et de valorisation de corpus suppose dans un premier temps un archivage qui exige d'examiner de manière approfondie les possibilités spécifiques de compression de ces masses de documents. Il n'existe pas de travaux antérieurs relatifs à la compression des données du patrimoine écrit. Les seuls que l'on peut recenser et qui ont ouverts la

voie à ces nouvelles recherches concernent le patrimoine ancien imprimé, notamment autour des premiers ouvrages de la fin du Moyen Age et de la Renaissance [DEB00].

I.2 Transmission des grandes masses de données

A l'heure actuelle, il n'existe pas d'outils réellement adaptés à une exploitation à distance des collections numérisées. Jusqu'à maintenant, le besoin majeur était de stocker et d'archiver les données provenant généralement de collections et de corpus volumineux, sans se soucier réellement de la qualité de la transmission puisque les documents étaient généralement consultés à partir de CD Rom ou sur des réseaux locaux sans risque de pertes de paquets de données et garantissant des temps d'accès aux données très convenables, [DEB00].

Les méthodes usuelles de compression d'images de document ne sont pas satisfaisantes, soit parce qu'elles produisent des images difficilement lisibles, soit parce qu'elles produisent des fichiers très volumineux se traduisant par un temps de transfert insupportable par l'utilisateur final [BOT00]. Une bonne méthode de compression doit permettre non seulement de réduire la taille des fichiers et assurer une bonne lisibilité des images, mais en plus produire des fichiers structurés en plusieurs couches d'informations permettant ainsi un affichage successif du plus au moins important (voir figure 1). Aucune des méthodes de compression actuelles ne satisfait toutes ces contraintes.



(a) (b) (c) (d)

Figure 1 : Séparation en couche d'une image de document manuscrit (a : image originale, b : couche arrière plan, c : couche texte, d : couche dessin).

Les formats de compression existants ont montré leurs limites sur les images de documents très abîmés présentant de fortes irrégularités tant sur les zones de textes que sur l'arrière plan. Les documents manuscrits anciens, et plus particulièrement les brouillons d'auteurs illustrent très bien ce genre de difficultés, [EMP03, WES03, MAD05]. Les formats comme DjVu et DEBORA[1] sur lesquels nous reviendrons dans la suite du mémoire fonctionnent de manière très satisfaisante sur des documents imprimés pour lesquels il est possible de comprimer l'image en utilisant la redondance des formes de caractères. Sur les documents manuscrits de notre étude, ces approches ne sont pas satisfaisantes. Leur échec est principalement lié à la grande complexité des formes du tracé difficiles à localiser, à segmenter et à caractériser. La difficulté réside ainsi dans le repérage des similarités distribuées sur toute la surface de la page. La notion de similarité devient ainsi centrale.

Pour pallier ces problèmes, nous nous sommes orientés vers une nouvelle méthodologie de compression tout en essayant de satisfaire les différents besoins des utilisateurs : la qualité des images ne doit pas être dégradée et le format des fichiers produits doit être structuré en couches de telle sorte que l'on puisse transmettre et afficher les informations reçues successivement. Notre méthode commence par la compression de l'arrière-plan du document en exploitant notamment la redondance des formes. S'agissant de documents écrits sur lesquels nous travaillons, le codage des parties textuelles repose sur une technique basée sur la redondance des formes et qui est robuste aux différentes transformations géométriques pour maximiser le taux de similarité et mémoriser un

[1] DEBORA: Digital accEss to BOoks of the RenAissance. Projet européen dont l'objectif est de concevoir un ensemble d'outils permettant l'accès distant et collaborative à des livres numérisés du XVI^{ème} siècle.

minimum de formes de base. Les couleurs de l'avant et de l'arrière-plan sont ensuite compressées, et comme elles varient dans des intervalles assez différents elles seront traitées séparément.

Nous présentons tout d'abord l'état de l'art en matière de codage et de compression en mettant le point sur ce qu'apporte chacune de ces méthodes sur les images de documents. Nous décrivons ensuite notre méthode de compression en précisant les méthodes appliquées pour la compression de chaque couche d'information. Nous présentons ensuite nos expérimentations et nos analyses des résultats.

II. SYNTHESE DE L'EXISTANT : Codage et compression

Les méthodes de compression utilisées actuellement ont montré leurs limites sur les images de manuscrits présentant des dégradations de type : taches, trous du support, traces d'humidité, dégradation de l'encre et des traits fins, ou bien encore variations de la couleur du papier ou problèmes de transparence (apparition du verso sur le recto par exemple) [LEB04]. En raison de l'une, l'autre ou de plusieurs de ces dégradations cumulées, les algorithmes standards de compression donnent des résultats très limités. Nous présentons dans cette partie une vue globale des algorithmes de base de la compression, en précisant notamment leurs limites sur les images de document anciens.

II.1 Considérations générales autour du codage : les techniques et outils de base

La numérisation d'une image consiste en son échantillonnage spatial suivi d'une quantification des amplitudes des échantillons obtenus (pixels). Une telle opération met en jeu des quantités d'informations très importantes qui rendent souvent impératif l'application de techniques de compression (des données). L'opération de compression (d'une des données image) comporte trois phases : l'analyse, la quantification et l'encodage à la sortie du quantificateur.

L'analyse (décorrélation) est une opération inversible où il n'y a encore aucune compression. Son but est de fournir un nouvel espace de représentation du signal, capable d'en extraire les redondances spatio-temporelles ou spectrales. Pour rappel, les principales méthodes d'analyse sont :

- l'analyse par prédiction : ce codage prédictif se justifie par la très forte corrélation spatiale existant entre les intensités des pixels adjacents d'une image et une très forte corrélation temporelle entre les pixels des séquences d'images.
- la transformation orthogonale par bloc : le passage à un espace transformé contenant la même information répond au souci de mettre en évidence des caractéristiques spécifiques du signal en faisant appel à des transformations linéaires et orthogonales appliquées à des blocs de pixels.
- le codage sous bandes : il repose sur une décomposition fréquentielle du spectre de l'image, grâce à un banc de filtres.
- la transformation en ondelettes bidimensionnelles : elle permet de considérer le signal comme une superposition de signaux, obtenus par translation et dilatation d'une fonction particulière (l'ondelette mère).

La quantification est une opération de discrétisation qui permet de réduire le nombre moyen de bits par pixel à transmettre, au prix d'une distorsion irréversible de l'image reconstruite. Quand cette étape est omise, on parle de codage sans perte. On distingue deux sortes de quantifications : la quantification scalaire (quantification indépendante de chaque pixel de l'image) et la quantification vectorielle (quantification groupée d'un ensemble de pixels de l'image, ou « vecteur de pixels » de l'image).

L'encodage enfin consiste à transcrire les amplitudes discrètes de l'image quantifiée, sous forme de mots de code, souvent binaires, et permet de réduire la redondance restante.

Certaines techniques de compression de base peuvent aussi bien être appliquées aux images qu'aux autres types de données, nous donnons ici une description de ces techniques.

La figure 2 présente un bilan des méthodes de codage.

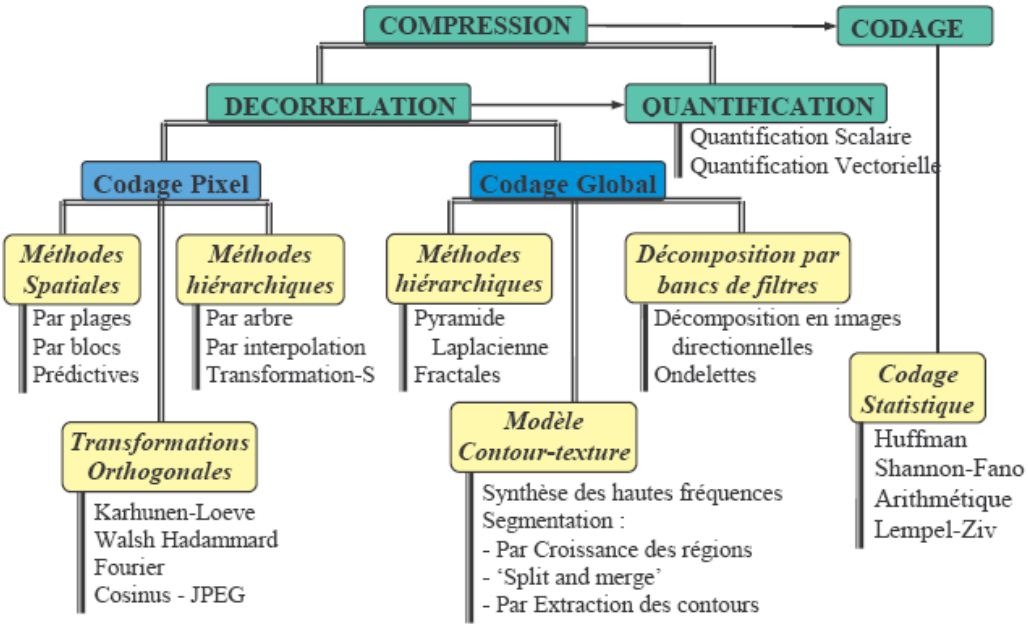


Figure 2 : Classification des méthodes de codage

Au sein de ces méthodes, on distingue les méthodes réversibles et les méthodes irréversibles.

Méthodes réversibles : Les techniques réversibles produisent un duplicata exact de l'image originale. Voici les principales approches correspondant à un codage statistique réversible.

Run Length Encoding (RLE) : Ce codage exploite la redondance sous sa forme la plus simple : si une succession de pixels a la même valeur, on transmet le nombre de ces pixels et leur valeur commune.

Les compressions entropiques : Dans plusieurs cas, certaines valeurs sont plus fréquentes que d'autres, le principe du codage entropique consiste à stocker les valeurs les plus fréquentes sur le moins de bits possible. Le codage de Shannon-Fano est une première méthode de compression entropique qui fut rapidement abandonnée avec l'apparition du codage de Huffman. Ce dernier [HUF52] affecte à chaque probabilité une séquence statique de bits. Le codage arithmétique quant à lui ne considère pas chaque probabilité indépendamment des autres, mais il s'intéresse à la combinaison de plusieurs probabilités pour affecter une seule valeur réelle correspondant à la probabilité de la séquence entière [JOR79].

Compressions à base de dictionnaire : Dans ce type de compression, toute nouvelle séquence est stockée dans un dictionnaire et une séquence déjà rencontrée est remplacée par son adresse dans le dictionnaire. L'algorithme le plus connu dans cette catégorie est sans doute le LZW (Lempel-Ziv-Welch) [ZIV77, ZIV78, WEL84]

Compression prédictive : Il s'agit de prédire la valeur de chaque pixel en fonction de la valeur des pixels précédents, les différences entre les valeurs prédites et les valeurs réelles varient dans un intervalle assez restreint et peuvent donc être efficacement compressées [BEL84].

Méthodes irréversibles : Les techniques irréversibles ne restituent qu'une approximation des données initiales. C'est l'étape de quantification qui rend le codage irréversible. Seule la transmission de l'erreur de quantification permet alors de rendre la compression réversible. Nous en reparlerons dans les sections suivantes avec les méthodes JPEG notamment.

En majorité, les algorithmes de compression sans perte d'informations sont généralistes : à la limite ils peuvent être adaptés à un type de fichier mais ils ne dépendent pas de la structure d'un fichier

en particulier. Les algorithmes de compression avec perte quant à eux, doivent effectuer une analyse individuelle pour connaître les informations qui pourront éventuellement être sacrifiées.

De façon évidente, les techniques de compression avec perte ne sont pas d'un emploi universel car de nombreuses applications n'admettent pas la perte d'informations en terme de couleurs ou de précision. Tel est en particulier notre cas.

Ainsi en analyse d'images de documents où les formes en présence sont de petites tailles (quelques pixels seulement pour former une lettre ou un graphème) ce type de compression n'est pas acceptable. Voyons à présent plus précisément les spécificités de ces deux types de compressions.

II.2 Compression générique avec perte : une solution peu adaptée aux documents

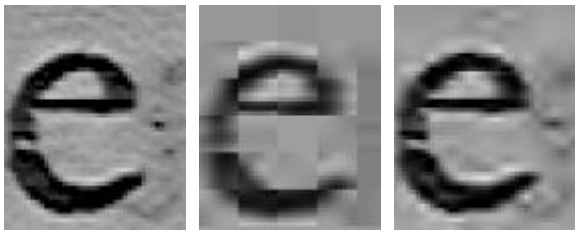
Bien que la numérisation d'un document constitue déjà une première perte d'informations à cause de la discrétisation, les méthodes de compressions avec perte d'information dégradent considérablement les images de document et font perdre des informations qui peuvent s'avérer utiles pour la suite des traitements.

Les méthodes génériques de compression avec perte sont basées sur les spécificités du système visuel humain, notamment le fait que l'œil il est plus sensible aux différences de luminosité qu'aux variations faibles de couleur, elles utilisent des transformations linéaires discrètes pour passer du domaine spatial vers le domaine fréquentiel. Parmi ces transformations, on peut citer : la transformée cosinus discrète (DCT) utilisée par le standard JPEG, Karhunen-Loeve, la transformation de Fourier discrète, les transformations de Hadamard et de Haar et enfin les transformées en ondelettes discrètes utilisées par le nouveau standard JPEG 2000 [TRE04].

Néanmoins aujourd'hui, trois méthodes de compression d'images se partagent l'essentiel du marché. Elles ont pour nom : JPEG (Joint Photographic Experts Group), RVQ (Recursive Vector Quantization) et compression fractale, qui reste cependant marginale.

II.2.1 La compression JPEG

Le format standard de compression JPEG (Joint Photographic Expert Group) offre une compression très efficace pour les images couleurs ou en nuances de gris. Il offre des taux de compression allant de 1:5 à 1:500, la qualité des images produites est bonne pour des images naturelles, mais cette qualité se dégrade rapidement pour des images particulières telles que les images de documents. En effet, ces images contiennent des traits noirs sur un fond uniforme et donc des transitions de couleur importantes sur les contours des formes. Toutes ces propriétés font que le filtrage des hautes fréquences pendant la quantification de la transformée cosinus discrète DCT ne passe pas inaperçu devant l'œil humain et nuit considérablement à lisibilité et aux traitements que l'on peut faire sur ce type d'image [LEB05]. Les différents inconvénients de la compression JPEG sur une image de document sont : la délocalisation des pixels, la réduction des nuances de gris, l'apparition de niveaux de gris qui n'existaient pas auparavant, les effets de blocs 8x8 pixels, la destruction des formes, la modification des contours, et enfin, la dégradation des performances de la reconnaissance des caractères pour les images des textes imprimés [LEB00], voir figure 3.



(a) (b) (c)

Figure 3 : Effets des compressions avec perte, a : image originale, b : compression JPEG avec un taux 12:1, c : compression JPEG 2000 avec un taux 12:1

II.2.2 La compression JPEG2000

La compression JPEG2000 à base d'ondelettes donne une meilleure qualité que la transformée cosinus discrète pour un même taux de compression de 1:12. Elle préserve mieux la complexité des formes. Cependant, dès que le taux de compression passe à 1:20 la déformation est facilement perceptible. Notons aussi que JPEG 2000 offre une version de compression sans perte que nous aborderons plus bas dans ce rapport.

II.2.3 La quantification du nombre des couleurs et la réduction de la résolution

La réduction du nombre des couleurs de l'image ou de sa résolution peut être considérée comme une méthode de compression avec perte. Son inconvénient est qu'elle élimine des informations qui pourront être utiles pour la suite des traitements tels que la séparation entre le recto et le verso. En plus, l'expérience acquise lors de la numérisation des collections pour le projet DEBORA a montré qu'une résolution minimale de 400 dpi est nécessaire. Cette résolution doit être optique et non interpolée, l'interpolation ne fait que répéter les données déjà présentes dans l'image. Enfin, l'acquisition doit se faire avec toutes les couleurs qu'offre le système informatique (« True Colors »). En effet, plus l'image est riche et plus ses utilisations ultérieures seront nombreuses et efficaces [LEB00].

La méthode RVQ est un exemple basé sur la quantification récursive des données. Non seulement elle exige une puissance de calcul considérable (même en comparaison des approches JPEG et fractales), mais de plus elle nécessite une intervention humaine à chaque étape afin de choisir les meilleurs paramètres de compression. Si la compression est un processus lent, la décompression en revanche est rapide. Cette rapidité rend RVQ utilisable pour la décompression d'informations en temps réel, comme des données audio ou vidéo.

II.2.4 Compression fractale

Les approches de compression fractales considèrent toute image comme un ensemble de minuscules motifs, chaque motif peut être obtenu à partir d'un autre. Ces méthodes commencent par décomposer l'image en un ensemble de motifs et recherchent une correspondance entre chaque couple de motifs. Deux motifs sont associés si l'un est obtenu en appliquant une transformation géométrique affine sur l'autre. Plus il y a des motifs vérifiant cette propriété, meilleur est le taux de compression. Plusieurs méthodes de décomposition peuvent être appliquées, la triangulation de Delaunay donne les meilleurs résultats. Il s'agit, après la décomposition, de détecter les redondances entre ces motifs à diverses résolutions en appliquant les différentes transformations fractales basées sur un opérateur contractant. Elles décrivent l'image de plus en plus finement. La compression consiste à ne mémoriser que les transformations mathématiques permettant de représenter le contenu de ces motifs et non pas le contenu du motif autant de fois qu'il a été rencontré dans l'image. Le procédé de reconstruction applique itérativement les transformations correspondant à chaque motif et garantit une convergence relative vers l'image originale. La qualité du résultat dépend fortement de la taille des motifs lors de la décomposition, plus ils sont petits, meilleur en est le résultat [ZHA95].

La compression fractale a l'avantage de ne pas présenter les effets de pixellisation contrairement à la compression JPEG, par contre elle introduit un flou dans l'image qui peut nuire à la lisibilité. Malgré toutes les améliorations elle reste très consommatrice en terme de temps de calcul. Appliquée aux images de documents (textes imprimés principalement), cette technique permet de trouver dans les imagettes (ou domaines qui sont des blocs de pixels qui partitionnent l'image) les éléments qui peuvent être retrouvés à des échelles différentes dans l'image. Avec l'ensemble des domaines extraits et utilisés dans la phase initiale de compression, on peut reconstruire une base de référence des traits pour l'alphabet utilisé sur ces images de textes. Cette base de référence représente les similarités internes contenues dans l'écriture, [SER04]. Il faut noter que la forme géométrique des imagettes des domaines dépend du type de partitionnement utilisé. Notons enfin que cette méthode ne détecte pas de similarité interne à grande échelle ce qui limite considérablement ses taux de compression.

II.3 Compression sans perte d'informations

La compression sans perte d'informations est sans doute l'alternative idéale dans le contexte des images de documents. Bien que les taux qu'offre ce type de compressions soient assez limités, ces méthodes garantissent la faisabilité de tout traitement ultérieur.

Les méthodes usuelles de compression sans perte ont la particularité d'être beaucoup plus efficaces sur les images binaires que sur les images en nuances de gris ou en couleur. En effet, sur des images binaires le taux de compression de ce genre de méthodes avoisine généralement les 1:5 alors qu'il reste aux environs de 1:1,5 pour les images en couleurs ou en nuances de gris. Pour cette raison, la plupart des bibliothèques numériques fournissent un accès aux documents par leurs versions binaires faciles à exploiter et pour lesquelles il existe plusieurs méthodes de compression très intéressantes.

II.3.1 Approches basées-pixel

La compression des images à deux niveaux a été l'objet de plusieurs recherches qui avaient pour objectif d'améliorer la vitesse de transmission des documents par fax. Les premières méthodes qui furent créées n'exploitaient pas la redondance au niveau des formes graphiques, elles sont, pour cette raison, appelées des méthodes basées-pixel (ou pixel-based approaches). Le CCITT (comité consultatif international télégraphique et téléphonique) propose des algorithmes de compression sans perte des images binaires de documents, ces algorithmes codent les séquences les plus fréquentes sur moins de bits (CCITT G3) [HUN80]. Une amélioration consiste à utiliser une connaissance a priori sur les séquences de pixels les plus fréquentes dans les images de documents (CCITT G4) [BOD85]. Ces deux formats sont incorporés dans le TIFF.

Le format JBIG (Joint Bi-level Image Experts Group) fournit un encodage prédictif des séquences et utilise un codeur arithmétique pour coder les prédictions, il surpasse le standard CCITT G4, mais il n'est pas adapté à une transmission progressive [KIA97].

II.3.2 Approches basées-forme

Une contribution importante dans le domaine de la compression des images de document eut lieu grâce aux travaux d'Asher et Nagy [ASH74] et fut formalisée ensuite par Witten [WIT94]. Il s'agit d'une similarité au niveau des formes et non pas au niveau d'une séquence linéaire de bits ou de pixels. Le nouveau standard JBIG2 tire parti de cette méthode pour donner de meilleurs résultats, la taille des fichiers qu'il produit est de 3 à 5 fois plus petite que le CCITT G4 et de 2 à 4 fois plus petite que le JBIG [ONO00]. JBIG2 a permis à plusieurs formats de voir le jour comme le XIFF (eXtended TIFF) de Xerox, le TIFF-FX (TIFF Fax eXtended) de Scansoft ou encore le DjVu d'AT&T. Ces formats utilisent des variantes du JBIG2 pour la compression de l'avant-plan des images de documents [LEB05].

II.3.3 Comparaison des méthodes de compression sans perte sur des images de document

Nous présentons dans cette section des comparaisons que l'on a faites entre les différentes méthodes de compression sans perte sur une variété d'images de documents. Les méthodes de compression comparées sont : PNG, TIFF (avec un codage LZW), JPEG-LS (qui est une version sans perte du standard JPEG) et JPEG2000 sans perte, nous avons aussi utilisé d'autres méthodes de compression qui ne sont pas appliquées uniquement aux images : ZIP, RAR, CAB et enfin le 7Z.

Les taux de compressions obtenus étaient assez différents ce qui ne nous a pas permis de calculer des moyennes significatives, par contre, nous avons remarqué que selon le type de document certaines méthodes de compression donnaient toujours le meilleur taux de compression, le tableau 1 illustre les taux de compression des différentes expérimentations que l'on a faite. Le taux de compression est défini comme étant le rapport entre la taille originale et la taille après compression.

	Manuscrits anciens					Textes imprimés					Manuscrits homogènes				
PNG	1,58	1,59	1,44	1,31	2,43	14	7,18	16,77	15,84	6,13	11,69	13,4	3,5	2,12	9,52
JPEG	1,33	1,32	1,37	1,3	1,75	2,6	3,49	4,27	3,3	1,93	4,64	3,46	2,61	2,14	3,6
TIFF	1,38	1,39	1,27	1,09	1,95	8,61	6,84	13,52	10,78	5,21	16,01	10,72	3,13	1,97	8,45
ZIP	1,27	1,23	1,41	1,22	1,69	16,6	7,46	17,86	21,44	7,66	14,61	14,92	5,05	2,44	9,96
RAR	1,82	1,58	1,68	1,27	2,05	25,6	8,28	21,32	30,77	9,29	19,48	17,65	5,5	2,56	12,1
CAB	1,43	1,45	1,67	1,3	2,21	28,9	9,29	23,82	35,98	11,2	21,44	19,82	6,93	2,81	13,3
JPEG2000	**2,43**	**2,49**	**2,06**	**1,55**	**3,09**	5,56	9,76	19,47	9,15	4,46	17,84	8,93	2,26	2,29	8,41
7Z	1,55	1,49	1,72	1,42	2,34	**32**	**10,12**	**32,92**	**40,06**	**11,62**	**29,22**	**20,67**	**7,32**	**3,05**	**17,9**

Tableau 1: Taux de compression des différentes méthodes de compression sans perte sur différents types d'images de documents

Nous avons distingué deux types d'images de documents :

Les textes imprimés ou les manuscrits ayant un arrière-plan et une écriture très homogène : pour ce type d'images la méthode de compression 7Z donne toujours le meilleur résultat. Le format 7Z utilise l'algorithme LZMA (Lempel-Ziv-Markov chain-Algorithm) qui est une version améliorée du Lempel-Ziv. Les taux de compression qu'offre cet algorithme pour ce type d'images vont de 1:3 à 1:40, la taille des fichiers qu'il produit est beaucoup plus petite qu'avec les autres formats de compression.

Les manuscrits anciens : pour ces images, JPEG2000 sans perte donne toujours les meilleurs résultats avec des taux allant de 1:1,4 à 1:3

II.4 Méthodes usuelles de compression des images de documents

Pour pallier les problèmes que cause la compression JPEG, plusieurs méthodes ont été proposées, nous présentons une description de ces méthodes.

II.4.1 DjVu

La technique de compression DjVu est spécialement conçue pour la compression de documents en couleurs numérisées à haute résolution. Elle offre un taux de compression très élevé (1:100). Cette technique commence par classer chaque pixel de l'image numérisée comme pixel d'avant-plan (texte, dessins au trait) ou pixel d'arrière-plan (images, photos, texture du papier) grâce à un apprentissage markovien. Cette classification forme une image bitonale (à deux couleurs) qui est compressée grâce à une technique appelée JB2 qui tire parti des similitudes de forme entre les divers caractères composant l'avant-plan. Les images d'avant-plan et d'arrière-plan sont ensuite compressées à l'aide d'un algorithme de compression par ondelettes avec une résolution réduite. Un algorithme de masquage minimise le nombre de bits utilisés pour coder les pixels d'avant-plan ou d'arrière-plan qui ne sont pas visibles dans l'image finale [BOT00].

Bien que ses taux de compression soient assez élevés, la technique de compression DjVu présente quelques inconvénients : la séparation entre le texte et l'arrière-plan se fait par apprentissage markovien sans aucune interprétation, l'image bitonale contiendra ainsi toute sorte d'éléments graphiques que le compresseur JB2 va tenter de regrouper sans y parvenir. De plus, l'appariement des formes se fait indistinctement sur des caractères comme sur des éléments graphiques ce qui rend difficile toute exploitation ultérieure de ces informations pour l'indexation et la construction automatique de liens. DjVu ne garantit pas une isolation complète de tous les éléments textuels ; les caractères classés avec l'arrière-plan seront détruits pendant la compression par ondelettes, de même, les éléments graphiques qui se retrouvent segmentés dans l'avant-plan surchargent la compression. Enfin DjVu n'utilise pas la redondance des formes sur tout un livre mais sur chaque page indépendamment de toutes les autres [LEB00].

Notons aussi qu'il n'existe pas de méthode de compression DjVu sans perte. L'appellation « DjVu lossless » est une version de DjVu qui donne un bon rendu visuel mais est loin d'être sans perte.

II.4.2 DEBORA

La Compression DEBORA est spécialement conçue pour les documents imprimés. Contrairement à DjVu, DEBORA distingue deux éléments dans l'avant-plan : les graphiques et le texte, et ce, pour exploiter la très forte redondance des formes dans le texte. Chaque document est séparé en quatre couches, une méthode spécifique est appliquée pour la compression de chaque couche. La première couche est une image bitonale contenant les éléments textuels et est donc compressée en utilisant la redondance des formes. La seconde couche est aussi une image bitonale, elle contient les éléments graphiques et est aussi compressée sans perte en utilisant le CCITT G4. La troisième couche est celle de l'arrière-plan, elle est compressée avec JPEG 70%. La dernière couche est la partie résiduelle, elle vient du fait que la compression appliquée au texte remplace toutes les formes similaires par leur forme prototypée, la correspondance entre la forme réelle et la forme prototypée qui lui correspond n'est en général pas exacte. Pour cette raison, DEBORA code aussi cette partie résiduelle en utilisant un RLE et un codage arithmétique [LEB05].

La figure suivante illustre les différentes couches de la méthode DEBORA :

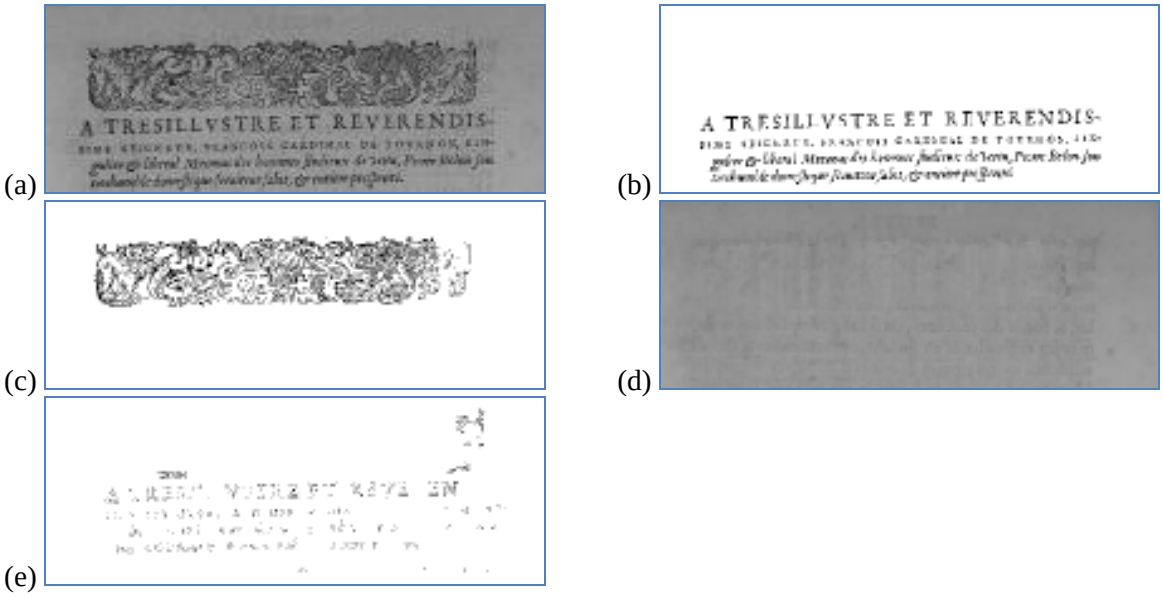


Figure 4 : Couches de la méthode DEBORA, a : image originale, b : couche textuelle, c : couche graphique, d : couche arrière-plan, e : couche résiduelle

La figure suivante illustre l'effet des différentes méthodes de compression sur une portion d'une image de document, remarquons que DEBORA donne des résultats comparables à « DjVu lossless ».

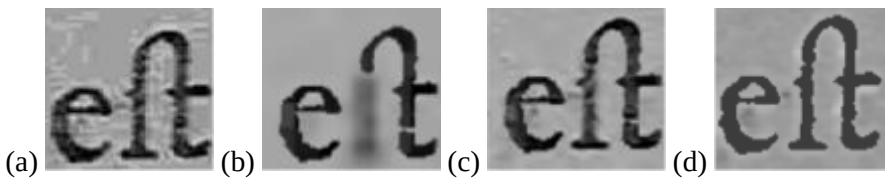


Figure 5 : Comparaison entre les différentes méthodes de compression, a : JPEG 1:10, b : DjVu avec perte 1:120, c : DjVu lossless 1:17, d : DEBORA 1:60

Enfin, notons que DEBORA utilise la redondance des formes pour la transcription des images de documents imprimés. En effet, la transcription d'une forme prototypée implique la transcription de toutes les formes qui lui sont affectées.

Nous concluons que tous les formats de compression, cités précédemment, sont performants sur des documents imprimés, mais les taux de compression se dégradent rapidement sur des documents dont le contenu est trop complexe à analyser et les formes ne sont pas suffisamment redondantes (documents manuscrits). Pour pouvoir comprimer de façon intelligente et efficace, il faut savoir segmenter correctement les images et reconnaître leur contenu.

Les images de documents manuscrits du corpus de notre étude sont fortement bruitées et contiennent des formes qui ne sont pas segmentables par les approches classiques de traitement d'images. La difficulté de la segmentation et l'absence de redondance de forme d'objets connexes rendent impossible la compression de ces images par les méthodes existantes. Pour remédier à cet obstacle majeur, il faut passer à d'autres approches exploitant cette fois des similarités de formes à des niveaux plus fins, au niveau du graphème, que nous redéfinirons ici.

Le tracé manuscrit est par définition très complexe à modéliser. Ceci est dû à la non régularité des formes, et leur importante variabilité que l'on peut constater d'une écriture à l'autre. Chez un même scripteur d'ailleurs on retrouve une hétérogénéité dans la forme des tracés, parfois même sur une même page d'un seul auteur. Tenant compte de cette forte variabilité propre aux tracés manuscrits, nous avons choisi de nous orienter vers une nouvelle méthodologie permettant de trouver localement des similarités partielles. Nous avons choisi ici de localiser les différentes zones d'intérêt reparties dans les images de traits. Ces zones d'intérêts vont nous permettre de définir les formes redondantes internes d'une écriture basées sur une décomposition en graphèmes. Ces formes redondantes ne sont pas nécessairement des mots ni des lettres : elles peuvent être simplement constituées des petites boucles des lettres qui se retrouvent localisés à d'autres endroits dans le tracé manuscrit et ceci à une transformation géométrique prés (un changement d'échelle, une rotation, une translation et bien d'autres transformations). L'échec de certaines méthodes de compression à ce stade (comme la compression fractale) est lié au mauvais partitionnement initial de l'image. Pour remédier à ce problème, il a fallu trouver un moyen intelligent et efficace de guider le partitionnement et de localiser les zones d'intérêts contenant les informations redondantes du tracé. La localisation de ces zones est une étape primordiale à la mise en place d'une méthode de compression efficace et adaptée aux contenus des images. Nous l'avons conçu par la séparation entre le texte et l'arrière plan.

Nous présentons dans la section suivante les détails de notre méthodologie.

III. NOTRE PROPOSITION

La figure 6 décrit le principe de notre méthode. Elle présente la succession des étapes de notre approche de la compression sans perte des images de documents. Dans l'ordre, on peut résumer notre méthode selon les six points suivants : 1/ La séparation entre fond et forme par la création du masque binaire de l'écriture. 2/ La squelettisation du masque binaire pour rendre l'analyse indépendante de l'épaisseur des traits de plume. 3/ La décomposition du squelette en graphèmes et classification des motifs dans une table de similarité. 4/ La restauration du squelette exploitant la redondance des formes. 5/ La restauration du masque binaire à partir de la sauvegarde de l'épaisseur des traits. 6/ La restauration de l'image originale à partir de la sauvegarde initiale de la couleur de l'arrière plan et du plan d'écriture.

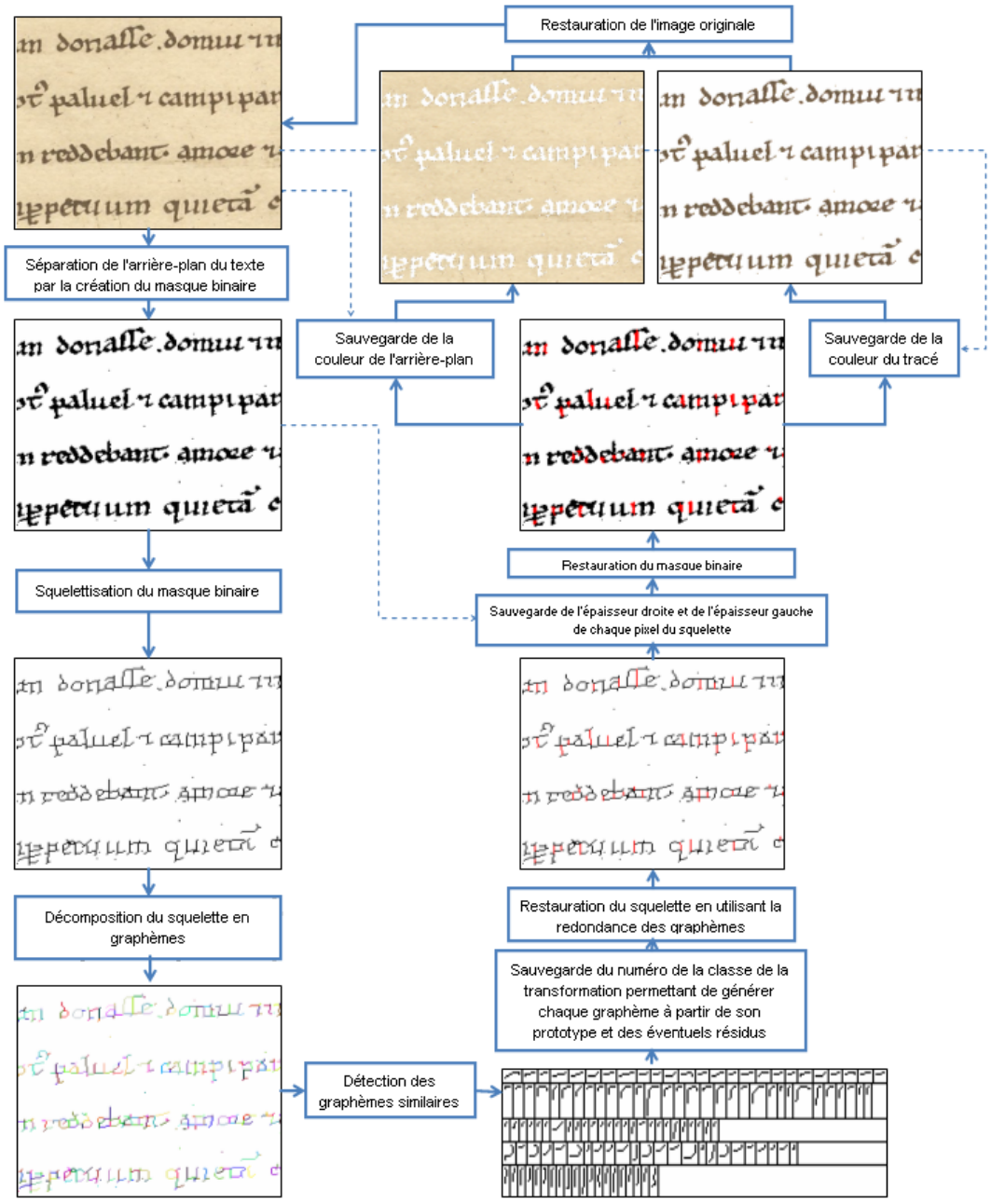
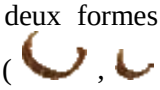
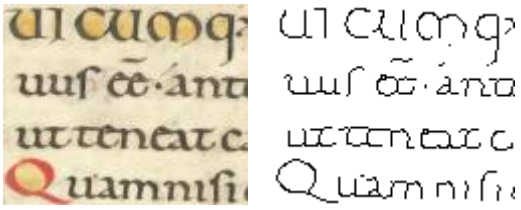


Figure 6 : Synoptique de notre approche de compression sans perte des images de documents

III.1 Segmentation de l'écriture en graphèmes à partir d'une décomposition du squelette

III.1.1 Pourquoi le squelette ?

L'idée de la squelettisation vient du fait que le tracé est assez complexe à analyser et nécessite une certaine « simplification ». Le squelette conserve aussi bien les propriétés topologiques que les propriétés géométriques de la forme, il correspond donc bien à ce que l'on cherche. De plus, puisque l'épaisseur du tracé varie dans un intervalle assez restreint, deux formes ayant deux squelettes similaires () sont considérés comme étant similaires () selon le point de vue choisi dans notre application.



(a) (b)

Figure 7 : a : image originale, b : squelette du tracé

III.1.2 Squelettisation du tracé

Pour extraire le squelette de l'écriture nous devons tout d'abord la délimiter, c'est-à-dire classer chaque pixel de l'image comme un pixel du tracé ou un pixel de l'arrière-plan. Cette séparation est appelée binarisation. Il existe plusieurs méthodes de binarisation, la qualité des résultats varie d'une méthode à une autre, certaines donnent de bons résultats au prix d'un temps de calcul trop important, d'autres sont plus rapides mais spécifiques à un certains types de documents. Concernant les documents manuscrits très dégradés, nous avons choisi d'adapter une technique spécifique basée sur une classification de type k-means des pixels et compatible avec les environnements bruités et les images de traits sur lesquelles nous travaillons.

Méthodes de binarisation et adaptation au cas des images de manuscrits

Rappelons brièvement ici les techniques de base de binarisation des images nécessaires à l'extraction du squelette des formes.

Binarisation par seuillage **:** L'arrière-plan étant généralement plus clair que le tracé, la séparation la plus implicite consiste à considérer tous les pixels dont le niveau de gris est inférieur à un certain seuil comme faisant partie du tracé et vice-versa. La détermination du seuil peut être globale, c'est-à-dire que le seuil a la même valeur pour tous les pixels du document, ou locale, pour laquelle le seuil varie d'une position à une autre. Les méthodes de seuillage global ne sont pas trop consommatrices en terme de temps de calcul, par contre, elles ne donnent de bons résultats que si le document est uniformément éclairé. Les méthodes de seuillage local sont plus robustes à de telles dégradations mais leur temps de calcul est plus important. La méthode de seuillage local de base est celle de Niblack, elle fut ensuite améliorée par Sauvola [SAU97]. Ces méthodes calculent le seuil de chaque pixel en fonction de la moyenne et de la variance des niveaux de gris de ces pixels voisins.

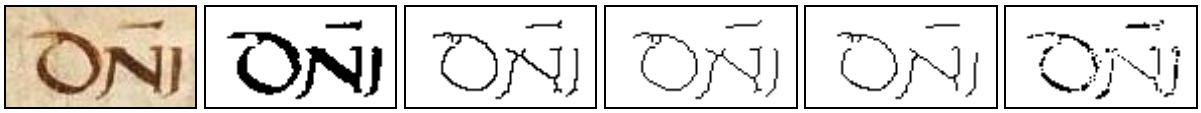
Binarisation par la méthode des k-means : Une autre méthode très utilisée est celle des nuées dynamiques (k-means), dans cet algorithme, on affecte à chaque classe un pixel qui constituera son centre de gravité initial. Chaque pixel de l'image est ensuite affecté à la classe dont le centre de gravité est le plus proche. Les centres de gravité sont de nouveau calculés et le processus continue itérativement jusqu'à ce qu'il ait convergence. Le k-means peut aussi être global appliqué directement à toute l'image ou local appliqué à chaque fenêtre de l'image dont on choisit la taille [TRE04]. La sérialisation du k-means consiste à initialiser les centres de gravité de chaque fenêtre avec les centres de gravité finaux de la fenêtre précédente. La sérialisation donne des résultats très intéressants mais est

très consommatrice en termes de temps de calcul et nécessite une initialisation des centres par l'utilisateur [LEY04].

Méthode de binarisation choisie : Nous avons décidé d'utiliser un k-means local ayant une fenêtre de 20x20 pixels et ayant pour centres initiaux le niveau de gris le plus bas et le plus haut de l'image. Ce choix est justifié par le fait que cet algorithme donne généralement de résultats satisfaisants et s'exécute plus rapidement que les autres algorithmes de binarisation locale tels que celui de Niblack ou de Sauvola.

Méthodes de squelettisation

Une fois le tracé délimité nous pouvons extraire son squelette, plusieurs méthodes de squelettisation existent, certaines suppriment itérativement les pixels dont le voisinage correspond à une configuration donnée jusqu'à ce qu'il y ait idempotence. Les configurations les plus connues sont les masques M et les masques L [BRE03]. D'autres méthodes suppriment parallèlement les pixels vérifiant plusieurs équations jusqu'à ce qu'il y ait convergence [PAV81, ZHA84]. L'extraction des maxima locaux de la carte des distances fournit l'axe médian qui a des propriétés proches de celles du squelette [BRE03]. Voir figure 8



(a) (b) (c) (d) (e) (f)

Figure 8 : Méthodes de squelettisation, a : image originale, b : image binarisée, c : méthode des masque L, d : méthode des masque M, e : méthode de Zhang, f : axe médian

Remarquons que la squelettisation par les masques L et M donne un squelette ayant plusieurs petits fragments aux extrémités des formes contrairement au squelette de Zhang. Remarquons aussi que le squelette obtenu par l'extraction des maxima locaux de la carte des distances donne l'axe médian qui n'est pas continu et qui n'est pas d'épaisseur unitaire.

Avec l'objectif précis de mettre en place une approche performante de compression, nous avons décidé de conserver dans un premier temps toutes ces méthodes de squelettisation et de les comparer pour finalement ne garder celle qui conduit aux meilleurs taux de compression.

III.1.3 Décomposition du squelette en graphèmes

Une fois le squelette obtenu, nous allons chercher à le décomposer en un ensemble de graphèmes élémentaires. Ces graphèmes sont considérés comme des unités graphiques stables pour un même scripteur associées à un mouvement de plume continu sans changement d'orientation ni reprise ni levé de plume. Au delà de ces unités, les similarités perceptibles deviennent plus incertaines, moins régulières et dans tous les cas très difficiles à évaluer.

Nous avons choisi d'extraire ces unités de formes à partir d'une décomposition des connexités du squelette précisément au niveau des points de jonctions observables en des points précis du squelette et mis en évidence par le protocole suivant :

- nous parcourons les 8 voisins de chaque pixel du squelette dans le sens indirect (inverse). Nous comptons le nombre de transitions pixel noir-pixel blanc. Ceci nous donne le nombre de parties connexes (voir figure 9). La plupart des pixels du squelette ont deux parties connexes. Ceux des extrémités en possèdent une seule. Nous nous intéressons aux pixels qui en possèdent trois ou quatre. Nous les appelons *pixels de jonction* et ils sont notés en *rouge* sur les figures 9.d et 9.e.

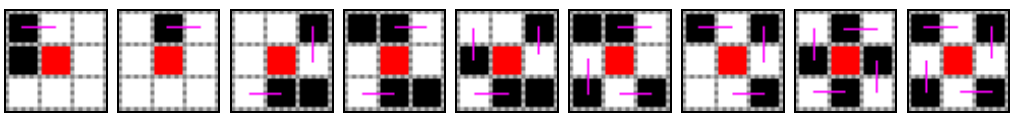
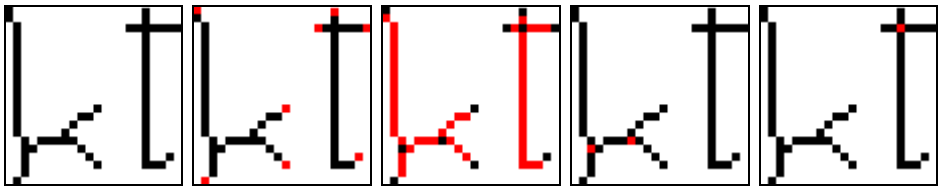


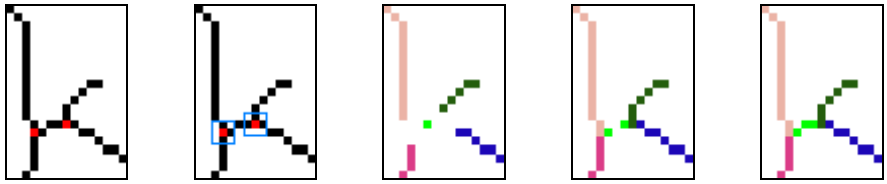
Figure 9 : Calcul du nombre de parties connexes



(a) (b) (c) (d) (e)

Figure 10 : Application à un cas réel. a. squelette du tracé. b. Pixels ayant une seule partie connexe. c. pixels ayant 2 parties connexes. d. pixels ayant 3 parties connexes. e. pixels ayant 4 parties connexes.

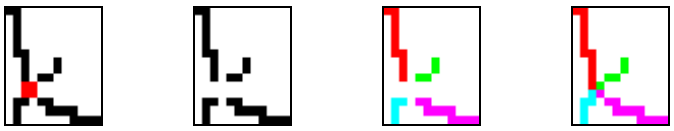
Points de jonctions simples. Pour décomposer le squelette à partir de ses pixels de jonction, deux cas de figures peuvent se présenter. Dans le premier cas (figure 11.a), nous considérons le squelette privé de ses pixels de jonctions et de leurs 8 voisins respectifs (figure 11.b). Nous considérons alors chaque partie connexe du squelette obtenue comme un graphème indépendant (figure 11.c). Nous relions ensuite chacune des parties connexes aux graphèmes se trouvant dans leur voisinage (figure 11.d). Le pixel de jonction est relié à la première partie connexe en favorisant les voisins directs (dans le voisinage 4) (figure 11.e). Dans l'exemple de la figure 11, le motif 'k' est formé de cinq graphèmes séparables.



(a) (b) (c) (d) (e)

Figure 11 : Etape de décomposition à partir des points de jonction simples

Points de jonctions multiples. Dans le second cas, nous nous sommes intéressés aux points de jonctions formés de plusieurs pixels contigus. Il s'agit précisément de régions carrées ayant une longueur de 2 pixels (figure 12.a). Comme pour le cas simple, le squelette est considéré comme étant privé de ses carrés (figure 12.b). Chaque partie connexe du squelette ainsi obtenu est considérée comme un graphème (figure 12.c). Les quatre pixels du carré sont ensuite affectés aux graphèmes se trouvant dans leur voisinage (figure 12.d).



(a) (b) (c) (d)

Figure 12 : Etape de décomposition à partir de points de jonction multiples

Cette décomposition est illustrée dans un cas réel à la figure 13 : on y observe ainsi l'ensemble des unités graphiques de base de l'écriture de l'extrait.

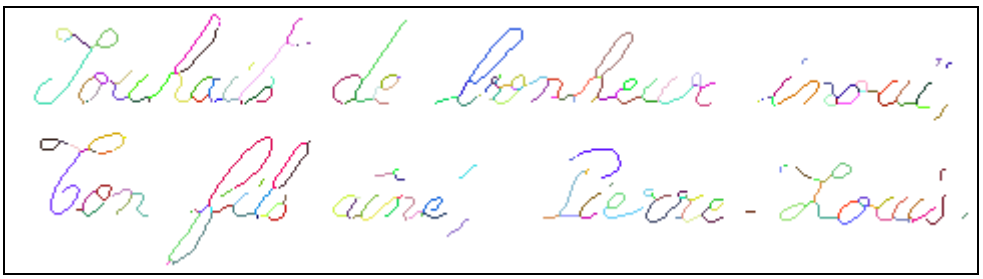


Figure 13 : Bilan d'une décomposition en graphèmes élémentaires de l'écriture

III.2 **Détection des similarités de formes**

L'objectif de cette partie est de définir une table de similarités de l'ensemble des graphèmes obtenus par décomposition du squelette. Il s'agit donc précisément de produire une classification des graphèmes selon des critères représentatifs et discriminants de formes impliquant notamment l'allongement et la distribution spatiale des points. La table de similarités produite est conçue pour être invariante à des transformations simples telles que le changement d'échelle (deux formes similaires de tailles différentes doivent pouvoir se ressembler), la rotation et la translation. Cette table servira ensuite d'indexes pour retrouver l'ensemble des formes redondantes dans une image de document. Dans la pratique, nous avons développé différentes stratégies d'extraction des similarités. Nous proposons donc deux tables de similarités différentes possédant chacune des propriétés d'invariance à des transformations géométriques simples. Elles seront choisies tout à tour en fonction de la nature de l'écriture impliquée dans les documents.

III.2.1 Détection des similarités par sous-échantillonnage des graphèmes

Une comparaison stricte des graphèmes pixel à pixel donnerait un taux de similarité quasi-nul. Une comparaison plus intuitive consisterait à mettre les graphèmes dans une même échelle avant de les comparer, d'où l'idée du sous-échantillonnage. Nous entendons par sous-échantillonnage un changement d'échelle qui met toutes les graphèmes dans une fenêtre de même taille. Soient par exemple L et H respectivement la largeur et la longueur d'une forme donnée et soient l et h respectivement la largeur et la longueur de la fenêtre de sous échantillonnage. Si x et y sont les coordonnées d'un pixel dans la fenêtre qui englobe le graphème, ses coordonnées dans la fenêtre sous-échantillonnée seront ($x.l/L$; $y.h/H$). Voir figure 14

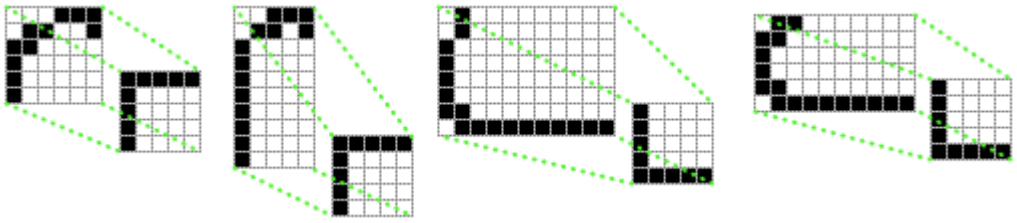


Figure 14 : Sous-échantillonnage des graphèmes

Une telle classification est invariante aux changements d'échelle, aux translations mais pas aux rotations. Elle s'applique essentiellement à des graphèmes très réguliers que l'on retrouve principalement dans les images numérisées d'ouvrages du Moyen-âge. Pour une plus grande robustesse aux variations de l'écriture, la technique suivante sera privilégiée.

III.2.2 Classification des graphèmes selon la distribution des points

Le sous-échantillonnage permet de regrouper des graphèmes similaires de tailles différentes, mais le taux de similarité reste globalement faible parce qu'un pixel de différence entre les fenêtres sous-échantillonnées produit une déformation telle qu'il arrive que deux formes quasi identiques se retrouvent dans deux classes différentes. De plus, un pixel dans la fenêtre sous-échantillonnée peut correspondre à un ou à plusieurs pixels de la forme initiale. Cela peut être préjudiciable dans le cas des manuscrits irréguliers, tels que les brouillons d'auteurs et les textes humanistes. Il s'agit dans cette version d'adapter le principe du sous-échantillonnage brut en respectant la distribution spatiale des points des graphèmes. Ainsi, au lieu d'affecter une valeur binaire à chaque pixel de la fenêtre sous-échantillonnée nous affectons un réel compris entre 0 et 1 indiquant la proportion des pixels de la forme initiale ayant pour image ce pixel. La figure 15 résume ce principe.

Avec cette version, nous avons jugé inutile de comparer des graphèmes très étendus en largeur à des graphèmes très étirés en hauteur. Pour cette raison, nous avons classé les graphèmes selon leur rapport hauteur/largeur en trois classes : les graphèmes étendus en largeur, les graphèmes étendus en hauteur et les graphèmes dont la largeur est proche de la hauteur. Cette classification est réalisée par un k-means.

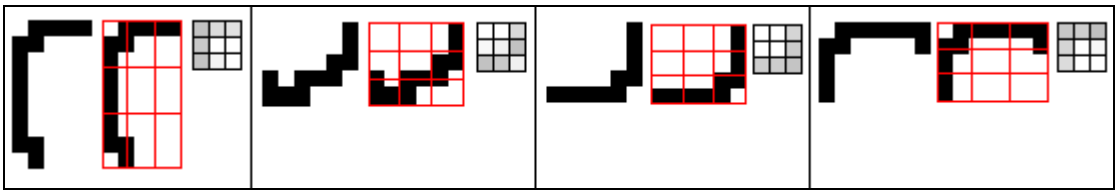


Figure 15 : Distribution des graphèmes sur une fenêtre 3x3. Les fenêtres de distributions contiennent des valeurs comprises entre 0 et 1, plus le niveau de gris est foncé, plus la proportion des pixels est importante.

Nous avons remarqué que les distributions de chaque classe de graphèmes étaient très proches de l'une des distributions de base indiquées dans la figure 16.

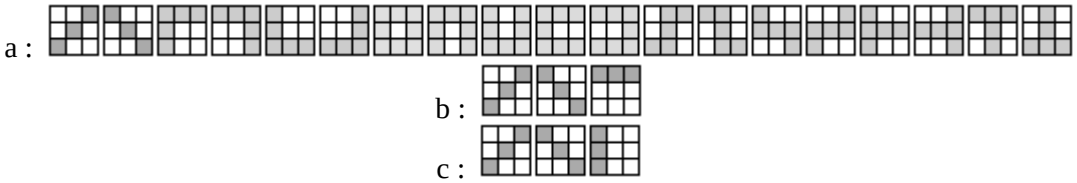
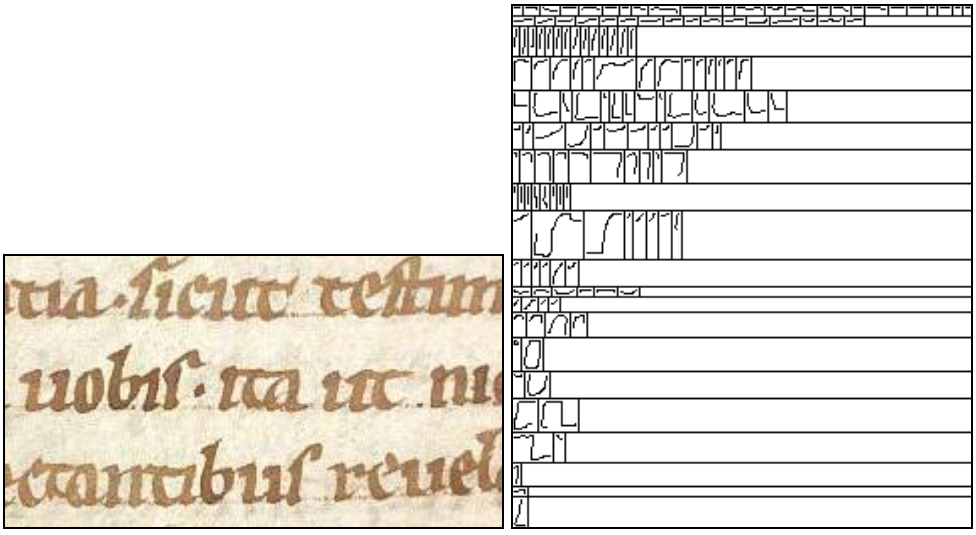


Figure 16 : Trois types de distributions de bases pour les graphèmes, a) dont la largeur est proche de la hauteur, b) étendus en largeur, c) étendus en hauteur

Nous affectons chaque graphème à la distribution de base la plus proche de sa distribution. La distance entre deux fenêtres de distribution étant la somme des différences absolues. Ainsi, chaque graphème est affecté à une classe selon son allongement, puis à l'une des distributions de base de cette classe. Nous obtenons donc au plus 25 types de graphèmes différents. Nous trions ensuite l'ensemble des groupes ainsi obtenus selon la fréquence des graphèmes pour réaliser la table des similarités (figure 17).

En affectant chaque pixel du tracé au graphème qui lui est le plus proche, nous pouvons voir comment cette similarité se répercute sur l'image originale (figure 18). Sur cette figure, la classe du graphème est numérotée 5. L'ensemble des graphèmes associés à cette classe est marqué de rouge sur l'image résultat. Nous pouvons généraliser cette visualisation aux cas des 25 classes référencées de l'étude.



(a) (b)

Figure 17 : a : Image d'origine, b : Table des similarités associée

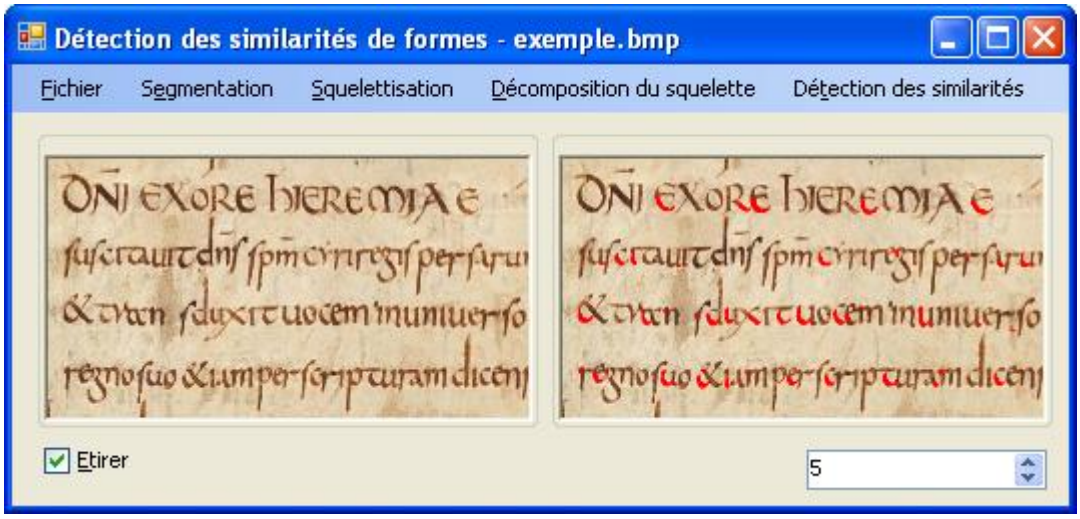


Figure 18 : Graphèmes similaires mis en évidence

Notons que comme pour le sous-échantillonnage, cette similarité est invariante aux changements d'échelles, aux translations mais ne l'est pas aux rotations. Bien que cette méthode regroupe bien des graphèmes similaires, son utilisation dans un but de compression n'est pas une tâche triviale. En effet, la compression nécessite une relation mathématique entre les graphèmes d'une même classe et un codage efficace de chaque graphème, ce que cette méthode ne garantit pas.

III.3 **Première étape de notre modèle : le codage du squelette**

Nous disposons désormais des informations de base nécessaires à la mise en place du codage des motifs : la liste des graphèmes et la table de similarités associée. Dans cette étape, nous proposons une approche visant à mémoriser les graphèmes du squelette le plus efficacement possible. Il s'agit de l'étape du *codage des graphèmes.*

III.3.1 Mémorisation des positions des graphèmes

Tout d'abord, il faut mémoriser la position des graphèmes, pour ce faire, nous trions les graphèmes selon leurs positions pour ne mémoriser que la première position et les différences entre les positions successives. Ces différences de positions sont mémorisées en utilisant le codage entropique de Huffman.

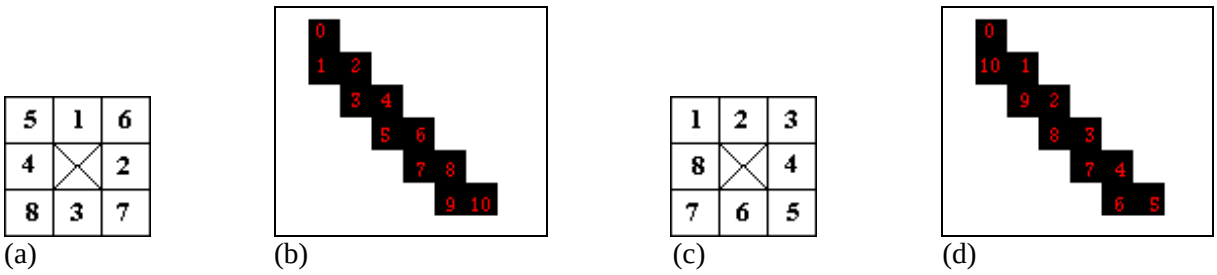
III.3.2 Codage des graphèmes par code Freeman

Le code de Freeman consiste à mémoriser pour chaque pixel une valeur entière comprise entre 0 et 7 indiquant sa position par rapport au pixel précédent. Notons que le problème qui se pose habituellement pour un codage de Freeman est le fait de rencontrer des pixels de jonction. Or dans notre cas, cela reste improbable puisque nous nous sommes justement basés sur ces pixels de jonction pour décomposer les graphèmes. Un même graphème peut avoir plusieurs codes de Freeman équivalents selon la méthode avec laquelle les pixels du graphème ont été parcourus. Nous devons donc définir une méthode pour parcourir les graphèmes avant de pouvoir générer leurs codes de Freeman. Le principe du parcours du graphème retenu est le suivant :

- Nous commençons le parcours à partir des pixels d'extrémité. Dans le cas de plusieurs pixels d'extrémité nous choisissons le premier rencontré. Si le graphème n'en contient pas, nous commençons le parcours à partir du premier pixel rencontré.

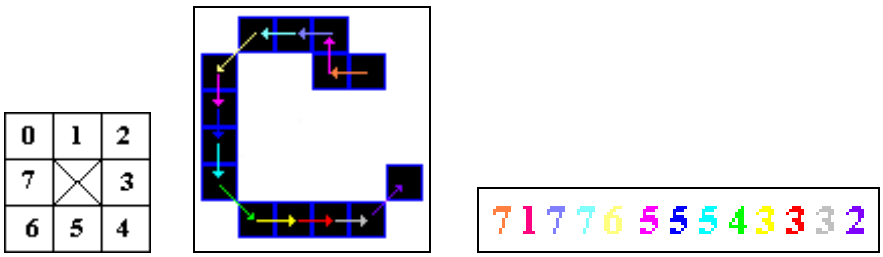
- Nous parcourons ensuite les voisins de ce pixel dans l'ordre indiqué dans la figure 19.a. Cet ordre favorise les voisins directs et rend le parcours des segments linéaires présents dans les graphèmes plus intuitif (voir figure 19.b). D'autres ordres tels que celui de la figure 19.c rend les parcours moins intuitifs (figure 19.d).

Grâce à cette procédure de parcours nous pouvons générer le code de Freeman de chaque graphème. En fonction de la position de chaque pixel par rapport à son précédent nous affectons l'une des 8 valeurs en suivant la configuration de la figure 20.a. Ainsi, le graphème de la figure 20.b aura pour code de Freeman la succession indiquée dans la figure 20.c.



(a) (b) (c) (d)

Figure 19 : Parcours des graphèmes



(a) (b) (c)

Figure 20 : Génération d'un code de Freeman, a : configuration à suivre pour générer le code de Freeman, b : parcours du graphème, c : code de Freeman correspondant

Un code de Freeman peut être codé efficacement sur 3 bits par valeur. Nous avons ajouté plusieurs optimisations :

- Quand la liste des valeurs est croissante ou décroissante, nous nous contentons de mémoriser la première valeur et la liste des différences sur le nombre de bits nécessaires (qui est en général moins de 3 bits).
- Quand la liste des valeurs n'est pas monotone, nous mémorisons la première valeur et la liste des différences signées, ceci nécessite un bit de plus pour la mémorisation du signe de la différence. Si les différences nécessitent 3 bits pour leur mémorisation, cela conduirait à 4 bits par valeurs à cause du bit de signe, si tel est le cas, nous utiliseront la mémorisation par défaut (3 bits par valeur).
- Nous avons aussi remarqué qu'avec notre méthode de parcours, une chaîne de Freeman de longueur 2 possède 20 configurations possibles. 94 configurations sont possibles pour une chaîne de longueur de 3, 428 pour une chaîne de longueur de 4 et 1929 pour une chaîne de longueur de 5. Pour simplifier le volume de stockage des motifs, il suffit dans ces cas de ne mémoriser que le numéro de la configuration sur le nombre nécessaire de bits.

III.3.3 Transformations réalisées sur la base du code de Freeman

Les familles de transformations

Transformation des codes par rotation. Nous partons du fait que des opérations arithmétiques simples sur le code de Freeman permettent de générer des transformations géométriques. Deux codes de Freeman identiques correspondent à deux graphèmes similaires (à une translation près). Sur la base de ce constat, nous définissons un ensemble de transformations géométriques de base sur le code de Freeman.

Soit *F* un code de Freeman, et *F[i]* la i^{ème} valeur de *F*. Si *F1* est un second code de Freeman défini par :

F1[i]= (F[i] + k) % 8

F1 est alors une rotation de $k \cdot 45°$ de *F* (Voir tableau 2).

11701233	22012344	33123455	44234566
55345677	66456700	77567011	00670122

Tableau 2 : Rotations de base d'un code de Freeman

Notons que cette rotation est composée avec un changement d'échelle de $\sqrt{2}$ dans le cas d'un *i* impair à cause de la discrétisation.

Nous avons ensuite défini trois jeux de 32 transformations, chaque transformation est composée avec les 8 rotations de base de Freeman. Nous obtenons alors trois jeux de 256 transformations. Ce qui nous permettra de les coder par la suite sur 8 bits, au-delà de cette taille, les temps de calcul augmentent considérablement.

*Transformations des codes par lissages***.** Théoriquement le lissage d'un code de Freeman est un filtre médian appliqué à la chaîne des valeurs [BRE03]. Pour accélérer la comparaison, nous proposons une variante : nous appelons lissage d'ordre *n* d'un code de Freeman *F*, le code de Freeman *F1* obtenu en affectant la première valeur de *F* aux *n* premières valeurs de F1, la (n+1)^{ème} valeur de F aux n secondes valeurs de F1 et ainsi de suite. Ceci revient à appliquer l'équation :

F1[i] = F[i – i % n] avec : $n \in [1; 32]$

La figure 21 illustre ce jeu de transformations appliquées à un graphème. Elle contient donc au total 32*8=256 transformations, car chaque jeu contient 32 transformations sur chacune de ces transformations, on applique les 8 rotations

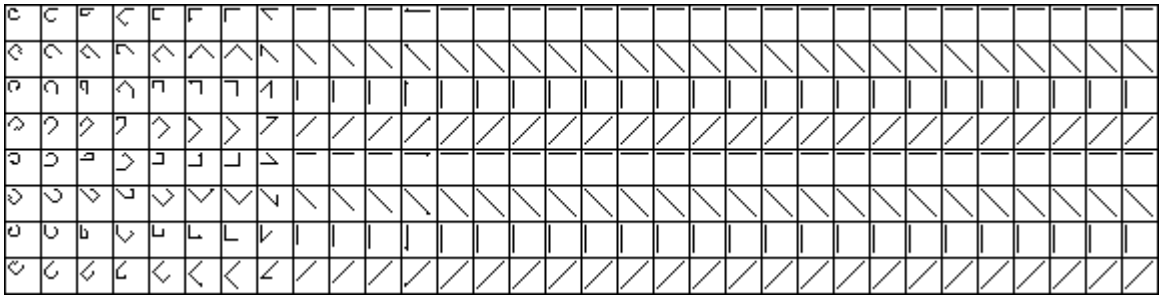


Figure 21 : Succession d'opérations de lissages appliquées à un graphème

Remarquons qu'un lissage d'ordre *n* converge dès que *n* atteint la taille de la chaîne de Freeman.

Généralisation autour des rotations. Pour générer une rotation de 45° en utilisant le code de Freeman, il suffit d'incrémenter toute la chaîne de Freeman. Pour générer une rotation d'un degré inférieur à 45° il faut altérer successivement entre 0 et 1. Nous avons généré 32 successions différentes de 0 et de 1 pour avoir différents degrés de rotation :

({0}, {0,1}, {1,0}, {1,0,1}, {0,1,0}, {1,1,0}, {0,1,1}, {0,0,0,1}, {0,0,1,0}, {0,0,1,1}, {0,1,0,0}, {0,1,1,0}, {0,1,1,1}, {1,0,0,0}, {1,0,0,1}, {1,0,1,1}, {1,1,0,0}, {1,1,0,1}, {1,1,1,0}, {0,0,0,0,1}, {0,0,0,1,0}, {0,0,0,1,1}, {0,0,1,0,0}, {0,0,1,0,1}, {0,0,1,1,0}, {0,0,1,1,1}, {0,1,0,0,0}, {0,1,0,0,1}, {0,1,0,1,0}, {0,1,0,1,1}, {0,1,1,0,0}, {0,1,1,0,1}).

En composant avec les 8 rotations de base nous obtenons un jeu de 256 rotations. La figure 22 illustre le résultat de ces 256 rotations sur un graphème.

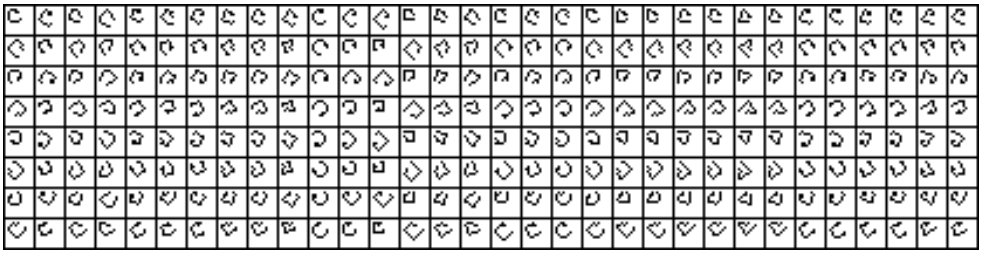


Figure 22 : Jeu de rotations appliquées à un graphème

Notons qu'il ne s'agit pas de rotations euclidiennes. En effet, celles-ci ne peuvent pas être générées rien qu'en utilisant des opérations arithmétiques sur le code de Freeman.

Transformations des codes par symétries. Nous obtenons ce jeu de transformations en remplaçant certaines valeurs par d'autres qui leur sont symétriques dans le code de Freeman. En remplaçant, par exemple, chaque 0 rencontré dans le code de Freeman (respectivement 7 et 6) par 2 (respectivement 3 et 4), ceci implique le remplacement de tout fragment du graphème dirigé vers la gauche par un fragment dirigé vers la droite (voir figure 23). Nous proposons un jeu de 32 remplacements différents pour générer le jeu des symétries (voir figure 24).

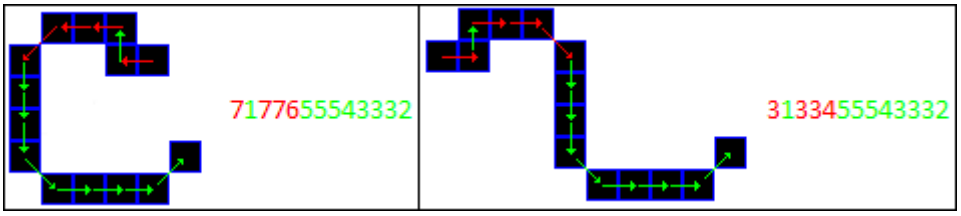


Figure 23 : Réalisation d'une symétrie en utilisant le code de Freeman

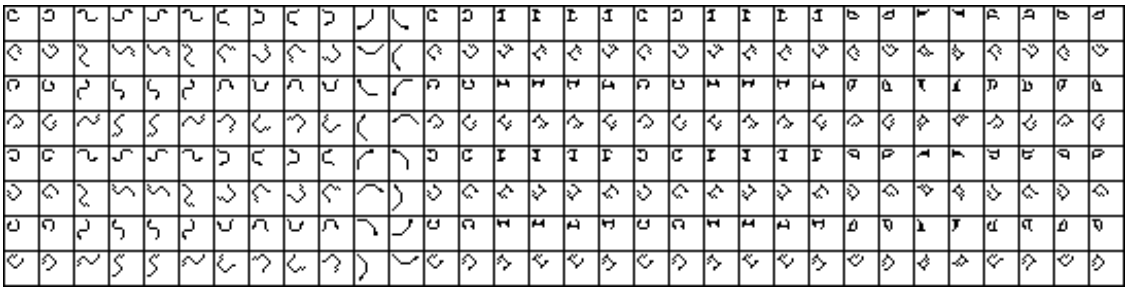


Figure 24 : Jeu de symétries appliquées à un graphème

Application au repérage des similarités.

Notons T_k le $k^{ème}$ jeu de transformation (le premier étant les lissages, le second étant les rotations et le troisième étant les symétries), notons $T_{k,i}$ la $i^{ème}$ transformation du $k^{ème}$ jeu de transformations.

Pour comparer deux codes de Freeman il faut qu'ils aient une même longueur, si ce n'est pas le cas, nous comparons le plus court avec le début du plus long. Soient F1 et F2 deux codes de Freeman que l'on souhaite comparer sachant qu'ils ont une même taille. Nous utilisons l'un des trois jeux de transformation T_k. L'algorithme de comparaison est le suivant :

Pour i=1 à 256 Si $T_{k,i}$[F1]=F2 alors F1 et F2 sont mis dans une même classe

Nous avons remarqué que cette comparaison qui nécessite une égalité stricte ne donne un taux de similarité élevé que dans le cas des textes imprimés. Pour cette raison nous avons mis au point une

autre méthode de comparaison moins rigoureuse : pour que F1 et F2 soient mis dans une même classe, il faut que toutes les différences entre les valeurs des deux codes de Freeman $T_{k,i}$[F1] et F2 soient inférieures ou égales à 1. La chaîne des différences doit être mémorisée. La figure 25 illustre un exemple d'une telle comparaison.

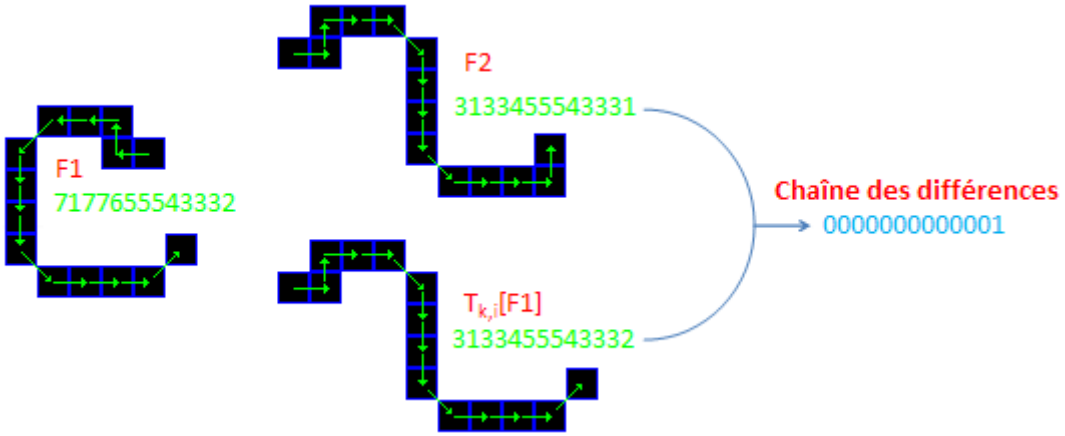


Figure 25 : Comparaison entre deux chaînes de Freeman et mémorisation de la chaîne des différences

Cette comparaison sera effectuée entre chaque couple de graphème et pour les trois jeux de transformations. Nous choisissons à chaque fois le jeu qui maximise la redondance. Nous mémorisons pour chaque graphème généré à partir d'un autre, le numéro du graphème l'ayant généré, le numéro de la transformation avec laquelle il a été généré et la chaîne des différences si elle n'est pas nulle partout. Notons aussi que pour une chaîne de Freeman « très petite » il est préférable de mémoriser la chaîne elle-même que de mémoriser tous les paramètres utiles pour sa reconstruction. Ceci étant, la détermination de la taille minimale à partir de laquelle la mémorisation des ces paramètres devient efficace n'est possible que par simulation.

Si dans une situation donnée, tous les graphèmes ont été générés à partir des n premières transformations, nous mémorisons le numéro de la transformation sur le nombre de bits nécessaires pour la mémorisation de n-1 et non pas sur 8 bits. Par exemple, si les numéros des transformations impliquées sont 0, 1, 2 et 3 nous mémorisons ces numéros sur 2 bits.

Une autre situation peut se présenter, par exemple, les numéros des transformations impliquées sont 2 et 250. La mémorisation de ces nombres nécessite 8 bits. La solution consiste à les indexer dans une table et mémoriser leurs index sur 1 bit.

Notons enfin que si certaines transformations ne sont utilisées que pour générer un nombre « très réduit » de graphèmes, il est parfois préférable de ne pas les utiliser parce qu'elle augmentent le nombre de bits nécessaires pour la mémorisation des numéros des transformations. La détermination du nombre de bits optimal pour la mémorisation des numéros de transformations ne peut être réalisée que par simulation. La figure 26 montre le résultat d'une telle simulation.

Finalement, à l'issue des transformations des graphèmes et de leur passage par les différents opérateurs de comparaison proposés, nous pouvons faire la distinction entre *trois* types de graphèmes :

- les graphèmes de base nécessaires à la reconstruction des autres graphèmes

- Les graphèmes générés qui sont reconstruits à partir des graphèmes de base

- les graphèmes restants qui ont été écartés, soit parce que leur taille est inférieure à la taille minimale à partir de laquelle la mémorisation des paramètres de reconstruction devient efficace, soit parce que les transformations qui les ont générés ne sont pas fréquentes, ou soit encore parce que la mémorisation des paramètres permettant leur reconstruction est plus coûteuse que la mémorisation directe de leurs chaînes de Freeman (bien que leurs tailles soient supérieures à la taille minimale à partir de laquelle la mémorisation des ces paramètres devient efficace).

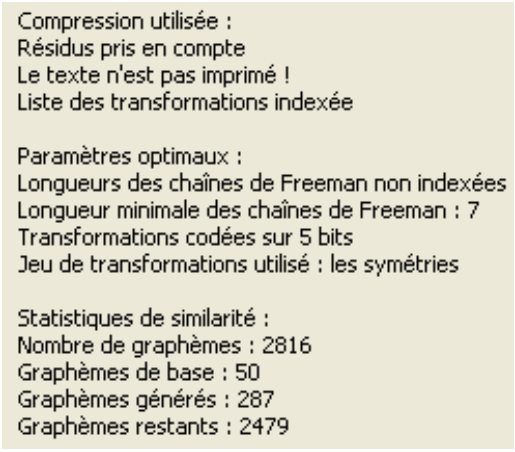


Figure 26 : paramètres optimaux de la compression

III.4 Reconstruction du masque binaire

La squelettisation n'est pas une transformation inversible. Cependant, l'axe médian permet la reconstruction de l'image binaire en effectuant une sauvegarde de l'épaisseur associée à chaque pixel du squelette (voir figure 27). Bien qu'il permette une reconstruction parfaite, l'axe médian présente quelques inconvénients comme énoncé précédemment : son épaisseur n'est pas unitaire et il présente plusieurs discontinuités (dans la majorité des cas).

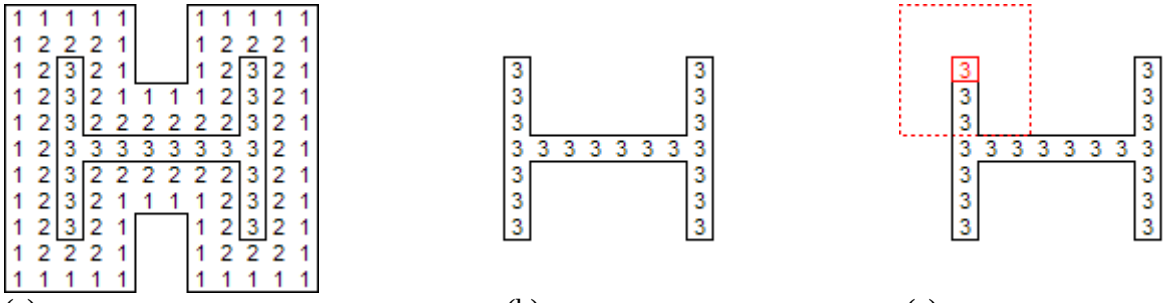


Figure 27 : a : Carte des distances, b : axe médian, c : reconstruction de l'image binaire à partir des valeurs sauvegardées

A cause de leurs structures, les méthodes de squelettisation par amincissements successifs ne permettent pas de telles sauvegardes. Nous proposons une méthode de sauvegarde permettant la reconstruction de la plupart des pixels de l'image binaire quelle que soit la méthode de squelettisation utilisée. Pour cela, nous sauvegardons pour chaque pixel du squelette l'épaisseur droite et l'épaisseur gauche (figure 28).

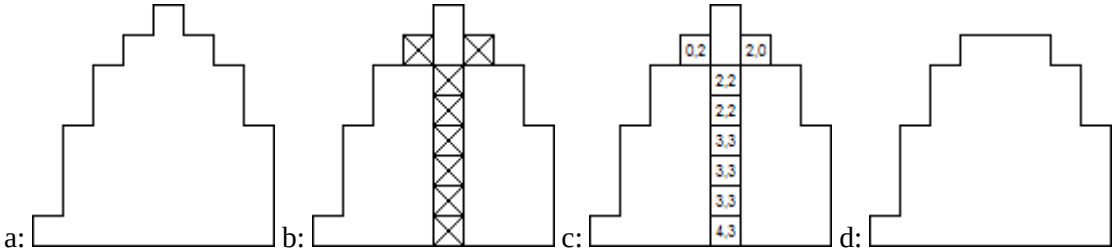


Figure 28 : Notre proposition pour la sauvegarde des épaisseurs, a : image binaire, b : squelette associé, c : sauvegarde de l'épaisseur droite et de l'épaisseur gauche pour chaque pixel du squelette, d : image binaire reconstruite

Remarquons qu'une reconstruction de tous les pixels de l'image binaire n'est possible que si le squelette a la même hauteur que cette image binaire. Pour cette raison nous avons développé une méthode de squelettisation dédiée qui vérifie cette propriété : nous calculons la carte des distances horizontalement (figure 29.a), nous considérons les maxima locaux comme des pixels du squelette

(figure 29.b), le squelette obtenu ainsi n'est pas toujours d'épaisseur unitaire, nous le squelettisons par la méthode des masques L (figure 29.c)

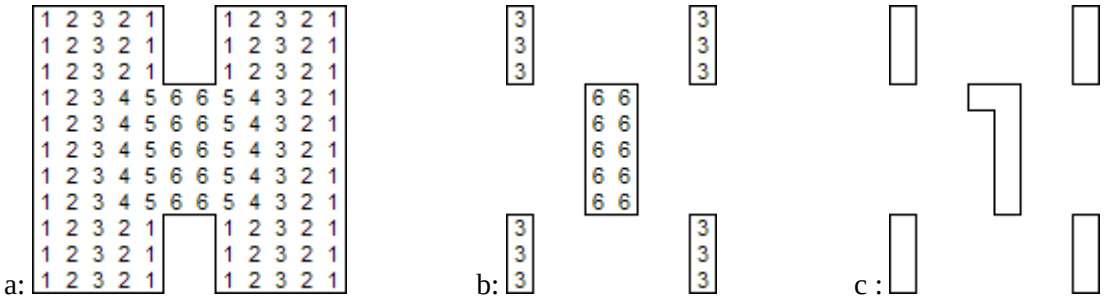


Figure 29 : Notre méthode de squelettisation, a : carte des distances calculée horizontalement, b : maxima locaux, c : squelettisation par la méthode des masques L des maxima locaux

Nous avons aussi remarqué que cette procédure pouvait s'appliquer sur les contours droits (ou gauches) et ne nécessitant ainsi que la sauvegarde d'une seule épaisseur (figure 30)

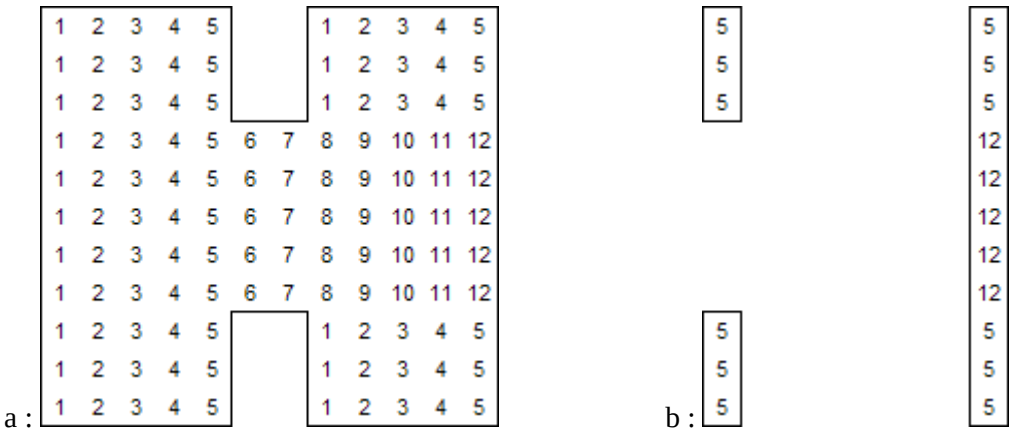


Figure 30 : a : variante de la carte des distances permettant l'extraction des contours droits, b : contours droits.

Pour les autres méthodes de squelettisation, nous mémorisons l'épaisseur droite et l'épaisseur gauche pour chaque pixel du squelette. Ceci étant, la reconstruction n'est pas complète, environs 90% des pixels de l'image binaire peuvent être reconstruits, les autres pixels sont classés avec le fond.

Nous avons retenu deux méthodes pour la compression de la succession des épaisseurs. Pour l'instant, nous choisissons à chaque fois celle qui donne le meilleur taux de compression. La première méthode est une compression *prédictive* puisque l'épaisseur d'un pixel du squelette est souvent très proche de l'épaisseur du pixel précédent. La deuxième méthode est le *codage de Huffman*, elle donne de bons résultats quand certaines valeurs des épaisseurs sont beaucoup plus fréquentes que d'autres.

III.5 **Codage des couleurs du plan et de l'arrière plan**

Une fois l'image binaire reconstruite, nous pouvons distinguer dans l'image originale les couleurs du tracé des couleurs de l'arrière-plan. Nous obtenons *deux* images contenant certains pixels (les pixels du tracé dans l'image de l'arrière-plan par exemple) dont la couleur n'a pas d'importance puisqu'elle ne sert pas à la reconstruction de l'image (figure 31).

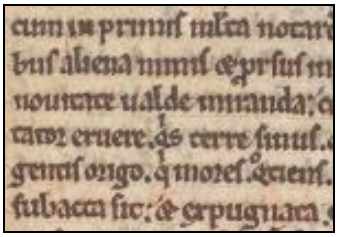
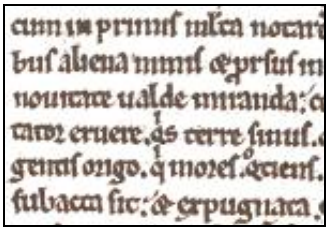
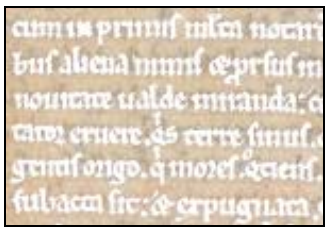
a :  b :  c : 

Figure 31 : a : image originale, b : couleurs du tracé, c : couleurs de l'arrière-plan

Pour ne pas mémoriser inutilement ces pixels sur des images de grandes tailles, nous avons créé deux images (de taille plus réduite) ne contenant que les pixels servant à la reconstruction. L'algorithme utilisé parcourt l'image d'origine et dirige les pixels du tracé vers une image et les pixels de l'arrière-plan vers l'autre. Chaque image ne contient donc que des pixels de même nature. Une fois cet algorithme exécuté, nous obtenons les résultats de la figure 32.

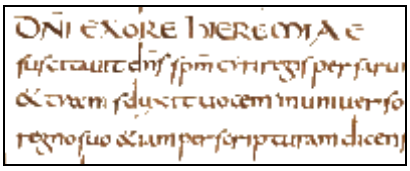
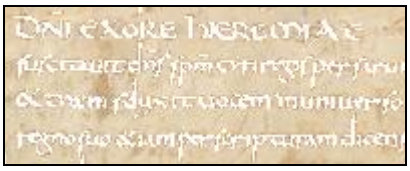
a:  b:  c:  d: 

Figure 32 : a : image du tracé, b : suppression des pixels inutiles de l'image du tracé, c : image du fond, d : suppression des pixels inutiles de l'image du fond

Pour réduire le bruit dans l'image du tracé et ainsi mieux la compresser, nous regroupons les pixels en fonction de leur distance au squelette. En effet, plus le pixel du tracé est proche du squelette plus la probabilité qu'il soit foncé est importante. La figure 33 illustre une comparaison entre trois images du tracé avec un parcours simple et un parcours allant du squelette vers le contour du tracé.

Dans le second cas, on observe un dégradé des couleurs (correspondant à la répartition des couleurs du squelette). La couleur du tracé étant généralement non uniforme, afin d'assurer un codage minimal, nous avons donc regoupé les pixels en fonction de leur distance au squelette, car nous avons pu constater que majoritairement, les pixels équidistants du squelette ont des couleurs assez proches.

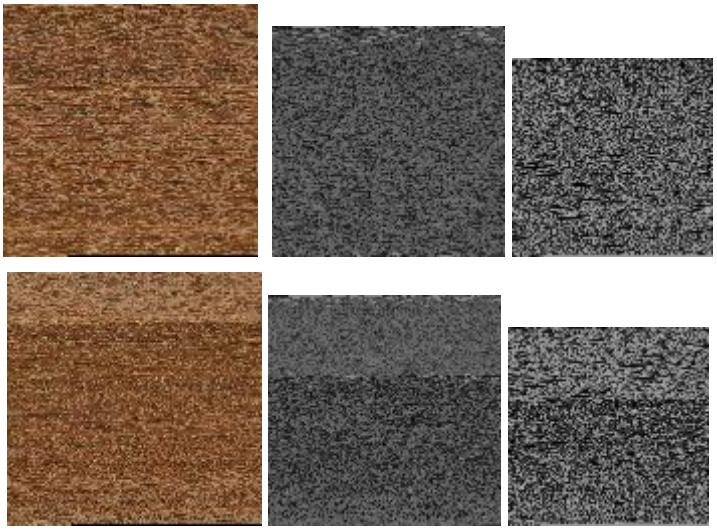


Figure 33 : Comparaison entre les résultats des deux méthodes de parcours, images en haut : parcours simple, image en bas : parcours du squelette vers les contours du tracé

Notons que les méthodes de squelettisation qui ne permettent pas de générer l'intégralité de l'image binaire (mais une reconstruction partielle seulement) font que certains pixels du tracé seront considérés comme faisant partie de l'arrière-plan ce qui introduit un bruit supplémentaire dans l'image de ce dernier et nuit considérablement à sa compression. Pour cette raison ces méthodes de squelettisation donnent toujours un taux de compression inférieur à celui des méthodes permettant une reconstruction complète de l'image binaire.

Avant de compresser les deux images (de couleurs du plan et de l'arrière plan) ainsi obtenues, nous dénombrons le nombre de couleurs et si ce dernier est inférieur ou égal à 2 (respectivement 4, 16 ou 256), nous codons alors ces couleurs sur 1 bit (respectivement 2, 4 ou 8 bits).

La méthode de compression qui donne les meilleurs résultats pour le codage de la couleur sur ces images est JPEG2000 sans perte si les couleurs sont codées sur 24 bits ou le 7Z si les couleurs sont indexées.

III.6 **Structures des fichiers**

La figure 34 illustre la structure des fichiers que produit notre méthode et permet ainsi de fournir une représentation synthétique du principe de compression retenu.

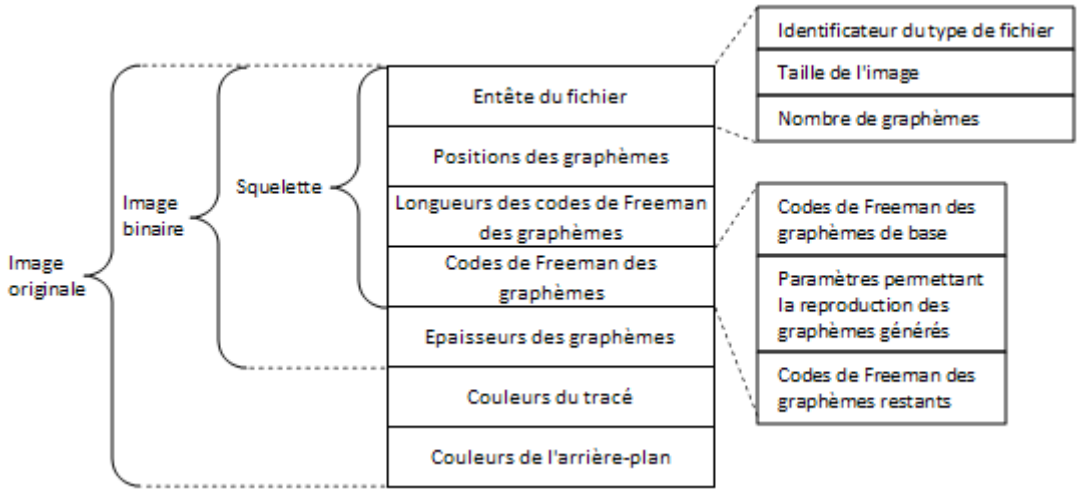


Figure 34 : Structure générale des fichiers que produit notre méthode

IV. EXPERIMENTATION ET ANALYSE DES RESULTATS

IV.1 Résultats de la compression des images binaires de documents

Dans cette partie nous allons comparer les résultats que donne notre méthode de compression sur les versions binaires des images de documents avec les méthodes de compression PNG, TIFF, ZIP, 7Z, CAB, RAR et JBIG2. Nous utilisons notre méthode de squelettisation proposée (celle de la figure 29) et les contours droits (ceux de la figure 30). Les autres méthodes de squelettisations ne permettent pas de générer toute l'image binaire, pour cette raison nous n'avons pas jugé utile de les comparer. Le tableau suivant illustre les taux de compression de ces différentes méthodes, les valeurs dans ce tableau correspondent à la taille du fichier BMP divisée par la taille du fichier compressé. La compression utilisant notre méthode de squelettisation proposée est étiquetée SKL sur le tableau, celle qui utilise les contours droits est étiquetée EDGE.

	Manuscrits anciens										Texte imprimé					Moyenne
TIFF	1,85	1,83	2,37	1,47	4,21	2,65	4,59	2,42	3,63	2,81	1,88	3,17	5,98	2,72	2,07	2,91
ZIP	3,24	2,78	3,23	2,13	5,48	3,62	7,17	3,73	5,05	3,76	3,02	5,89	10,80	4,29	2,93	4,47
PNG	3,51	2,91	3,26	2,16	6,08	3,98	7,99	4,09	5,79	4,18	3,21	6,43	12,26	4,62	3,13	4,91
RAR	3,64	3,02	3,23	2,21	6,26	4,00	8,33	4,23	5,84	4,14	3,31	6,65	12,87	4,80	3,22	5,05
CAB	3,79	3,22	3,41	2,28	6,74	4,14	8,61	4,46	6,09	4,34	3,40	6,98	12,91	4,92	3,40	5,25
7Z	3,93	3,65	3,63	2,45	8,08	4,49	9,67	4,81	6,70	4,74	3,61	7,42	14,49	5,26	3,66	5,77
JBIG2	**6,09**	**6,43**	**5,63**	**3,57**	**15,18**	**8,04**	**16,91**	**6,78**	**7,55**	**7,99**	**6,73**	**10,09**	**23,17**	**8,75**	**4,91**	**9,19**
SKL	3,71	2,95	3,59	2,04	7,41	4,82	10,15	4,02	13,27	4,61	3,19	6,96	11,62	3,81	2,90	5,67
EDGE	4,29	3,61	4,08	2,49	9,03	5,06	10,98	4,91	8,63	4,84	3,26	7,40	13,40	4,08	3,48	5,97

Tableau 3 : Comparaison des taux de compression des différentes méthodes sur des images binaires de document

Remarquons tout d'abord que notre méthode de compression utilisant les contours droits donne toujours un meilleur taux de compression par rapport à celle qui utilise la squelettisation proposée. Ceci est dû au fait que pour les contours droits, on ne mémorise qu'une seule épaisseur.

Remarquons ensuite que dans le cas des manuscrits anciens notre méthode surpasse généralement toutes les méthodes de compression excepté le JBIG2. Notre méthode est surpassée par le 7Z et le JBIG2 dans le cas des textes imprimés. Ceci est justifié par le fait que chacune des plages de données des fichiers que produit notre méthode admet plusieurs optimisations. Ajoutons que le format JBIG2 suit le formalisme de Witten [WIT94], une collaboration de ce formalisme avec celui de notre méthode mérite d'être exploitée pour optimiser les taux de compression.

IV.2 Résultats de la compression des images de documents en couleur

Nous reprenons dans cette partie les résultats du Tableau 1 illustrant la comparaison entre les différentes méthodes de compression sans perte sur des images de documents, nous illustrons dans le tableau 4 les résultats de toutes les méthodes de compression et ceux de notre méthode.

Nous appelons manuscrit homogène tout manuscrit dans lequel le nombre des couleurs du tracé et de l'arrière-plan est inférieur ou égal à 256. La méthode que l'on utilise pour la compression de l'image du tracé et de l'arrière-plan est dans ce cas le 7Z.

Bien que JPEG 2000 sans perte soit utilisée pour la compression de l'image du tracé et de l'arrière-plan dans notre méthode, nous pouvons remarquer que JPEG2000 sans perte la surpasse dans le cas des manuscrits anciens. Ceci s'explique par le fait que JPEG2000 sans perte appliquée à toute l'image donne un meilleur taux que si on l'applique séparément aux images du tracé et de l'arrière-plan. En effet, ces deux images présentent (malgré notre méthode de parcours) un bruit qui gêne la compression JPEG2000.

Pour la même raison notre méthode est surpassée par le 7Z dans le cas des textes imprimés et des manuscrits homogènes.

	Manuscrits anciens					Texte imprimé					Manuscrits homogènes				
PNG	1,58	1,59	1,44	1,31	2,43	14	7,18	16,77	15,84	6,13	11,69	13,4	3,5	2,12	9,52
JPEG	1,33	1,32	1,37	1,3	1,75	2,6	3,49	4,27	3,3	1,93	4,64	3,46	2,61	2,14	3,6
TIFF	1,38	1,39	1,27	1,09	1,95	8,61	6,84	13,52	10,78	5,21	16,01	10,72	3,13	1,97	8,45
ZIP	1,27	1,23	1,41	1,22	1,69	16,6	7,46	17,86	21,44	7,66	14,61	14,92	5,05	2,44	9,96
RAR	1,82	1,58	1,68	1,27	2,05	25,6	8,28	21,32	30,77	9,29	19,48	17,65	5,5	2,56	12,1
CAB	1,43	1,45	1,67	1,3	2,21	28,9	9,29	23,82	35,98	11,2	21,44	19,82	6,93	2,81	13,3
JPEG2000	**2,43**	**2,49**	**2,06**	**1,55**	**3,09**	5,56	9,76	19,47	9,15	4,46	17,84	8,93	2,26	2,29	8,41
7Z	1,55	1,49	1,72	1,42	2,34	**32**	**10,12**	32,92	**40,06**	11,62	29,22	20,67	7,32	**3,05**	17,9
EDGE	1,58	1,78	1,83	1,53	2,07	24,21	9,19	**33,03**	31,68	**12,26**	**29,97**	**25,38**	**8,42**	3,03	**18,24**

Tableau 4 : Comparaison des taux de compression des différentes méthodes sur des images de document en couleur

IV.3 Evaluation de notre mesure de similarité sur les codes de Freeman

Nous évaluons dans cette partie la mesure de similarité qui nous a permis de compresser les différentes images de documents ainsi que les différents jeux de transformations proposés.

	Manuscrits anciens					Texte imprimé					Manuscrits homogènes				
Nombre de graphèmes de base	14	30	110	28	62	27	108	112	78	240	26	105	53	11	102
Nombre de graphèmes générés	80	114	223	215	415	1294	917	612	1599	1191	200	278	345	246	278
Nombre de graphèmes restant	618	1349	4822	1543	6359	2797	3474	4457	5293	6006	1924	3226	2137	2169	4093
Transformations codées sur	5bits	5bits	0bits	4bits	4bits	0bits	0bits	0bits	0bits	0bits	4bits	0bits	0bits	2bits	0bits
Jeu de transformations	sym.	sym.	X	sym.	sym.	X	X	X	X	X	sym.	X	X	lis.	X
Longueur minimale des chaînes de Freeman	10	12	6	6	13	6	5	7	5	7	8	5	7	9	5
Liste des transformations indexée	non	non	X	oui	oui	X	X	X	X	X	oui	X	X	oui	X

Tableau 5 : Statistiques de similarités (sym.=symétrie, lis.=lissage)

Remarquons tout d'abord que le nombre de graphèmes restants est toujours très important ce qui limite considérablement le taux de redondance. Ces graphèmes sont majoritairement de « très petite taille ». Ce nombre important de « petits graphèmes » est dû essentiellement à la méthode de squelettisation utilisée. L'élaboration qu'une méthode de squelettisation permettant une reconstruction complète de l'image binaire et ne générant pas autant de « petits fragments » améliorerait considérablement le taux de compression.

Remarquons ensuite que les transformations ne sont codées sur aucun bit dans le cas des textes imprimés, ceci signifie que la seule transformation utilisée dans ce cas est l'identité. Notons que le jeu de transformation n'a aucune importance dans de tels cas, puisque l'identité est la première transformation de chacun des trois jeux proposés.

Remarquons que le jeu de transformations le plus utilisé est celui des *symétries*, c'est le jeu qui maximise le taux de redondance des graphèmes. Il permet de générer des graphèmes très différents à partir d'un même graphème.

Les transformations sont codées dans le meilleur des cas sur 5 bits, ceci signifie qu'au plus 32 transformations sont utilisées sur 256, ceci implique une certaine répétition dans tous les jeux de transformations. Un nouveau jeu de transformation ne contenant qu'une sélection des transformations les plus fréquentes de chacun des trois jeux de transformations proposées donnerait un meilleur taux de redondance.

Ajoutons que la longueur minimale des chaînes de Freeman à partir de laquelle la sauvegarde des paramètres de reconstruction des graphèmes devient efficace est toujours supérieure ou égale à cinq : une amélioration consisterait à écarter, avant la simulation, tout graphème ayant une longueur inférieure à cinq.

Remarquons aussi que le rapport du nombre de graphèmes générés par rapport au nombre de graphèmes de base est très important dans le cas des textes imprimés. Grâce à cette propriété, nous pouvons détecter automatiquement si une image d'un document donné correspond à un texte imprimé ou à un manuscrit. Notons enfin que les temps de compression vont de quelques dizaines de secondes jusqu'à quelques minutes en raison du grand nombre de simulations qui testent tous les cas possibles. Ils dépendent essentiellement du nombre des graphèmes. Une amélioration immédiate consisterait à élaborer des heuristiques guidées par le type de document pour accélérer les calculs. Les temps de décompression sont par contre beaucoup plus réduits, ils ne dépassent pas quelques secondes.

IV.4 **Travaux futurs**

IV.4.1 Utilisation future des tables de similarités des graphèmes

Les tables de similarité créées en utilisant la distribution des graphèmes sont des données précieuses que nous n'avons pas encore intégrées dans notre système de compression. Elles n'en restent pas moins un outil de base très intéressant pour différentes applications basées sur l'analyse des écritures:

- Nous pouvons envisager des applications de *word-spotting* en cherchant les correspondances entre mots identiques à partir d'une recherche des occurrences des graphèmes similaires composant un même mot.
- De même, en regroupant les graphèmes similaires d'un même caractère, cette méthode pourrait également servir d'outil de *transcription* des images de documents.
- Enfin, deux écritures différentes se verront affecter deux tables de similarité différentes : une table de similarité constitue une signature individuelle d'un scripteur particulier et pourrait ainsi servir à l'identification d'une écriture, ou l'authentification d'un document.

IV.4.2 Optimisations de la méthode de compression

Nous pouvons envisager d'améliorer la méthode de compression proposée par les différents points suivants :

- L'optimisation de chaque plage de données des fichiers que produit notre méthode par une méthode compression adaptée.
- L'élaboration d'une méthode de squelettisation permettant une reconstruction complète de l'image binaire et ne générant pas des fragments de « petites tailles » qui limitent considérablement le taux de compression.
- La réalisation d'un ou de plusieurs jeux de transformations permettant de générer des graphèmes très différents à partir d'un même graphème maximisant ainsi le taux de redondance.
- L'optimisation des temps de calcul en remplaçant la simulation de tous les cas possibles par des heuristiques adaptées à chaque type de document.

Toutes ces optimisations visent à améliorer la compression de l'image binaire, cette partie est certes importante, mais la mesure de similarité ne doit pas s'arrêter à ce niveau, les prototypes de base ne doivent pas correspondre à de simples graphèmes bitonaux mais à des imagettes colorées sur lesquelles on applique des transformations de positions mais aussi de couleurs pour générer les autres graphèmes. En effet l'information la plus volumineuse dans une image de document est celle des couleurs.

V. CONCLUSION GENERALE

Face aux spécificités des images de documents, aux différentes dégradations qu'elles présentent, les méthodes de compression actuelles fonctionnent de manière très imparfaite. Nous nous sommes orientés vers une nouvelle méthodologie de compression permettant de générer des fichiers structurés en plusieurs couches parfaitement adaptés à une transmission progressive. Nous avons proposé plusieurs méthodes de mesures de similarité, certaines se sont avérées utiles pour effectuer plusieurs traitements sur les images de documents, d'autres permettent de générer plusieurs graphèmes à partir d'autres et servent ainsi à la compression.

Notre méthode passe par une décomposition du squelette du tracé en un ensemble de graphèmes. En se basant sur les codes de Freeman, elle cherche ensuite les graphèmes de base permettant d'en générer d'autres. La sauvegarde de l'épaisseur des graphèmes permet la reconstruction complète et sans perte de l'image binaire. Celle-ci sert à séparer les pixels du tracé de ceux de l'arrière-plan. Sont créées ensuite deux images (les images de couleurs) contenant l'ensemble des informations couleurs de ces deux ensembles de pixels. Chacune de ces deux images est compressées par un algorithme adapté aux variations de couleurs observées. Bien que notre méthode ne donne pas toujours les meilleurs taux de compression, elle ouvre des pistes à plusieurs optimisations pouvant l'améliorer considérablement. Elle constitue dans tous les cas une avancée très prometteuse dans un domaine où les travaux antérieurs restent rares.

VI. REFERENCES BIBLIOGRAPHIQUES

[ASH74] R. Ascher et G. Nagy. *A means for achieving a high degree of compaction on scan-digitized printed text*. IEEE transactions on Computers, 23:1174-1179, 1974

[BEL84] T. Bell, J. Cleary, et I. Witten (University de Waikato, Nouvelle Zélande) *Data compression using adaptive coding and partial string matching*. IEEE Transactions on Communications, Vol. 32 (4), p. 396-402. 1984

[BOD85] D. Bodson, S. Urban, A. Deutermann, C. Clarke, *Measurement of data compression in advanced group 4 facsimile system*, Proc. of the IEEE 73:731-739, 1985

[BOT00] Léon Bottou, Patrick Haffner, Yann LeCun, Paul Howard, Pascal Vincent, Bill Riemers, AT&T Labs, *DjVu: Un Système de Compression d'Images pour la Distribution Réticulaire de Documents Numérisés*.

[BRE03] S. Bres, J.M. Jolion, F. Lebourgeois. *Traitement et analyse des images numériques*, Hermes, 2003

[DEB00] Projet européen « DEBORA », livre en ligne : http://debora.enssib.fr, juin 2000.

[EMP03] Emptoz, H., Dalbera, J.P., Couasnon, B. « *Numérisation et patrimoine* », Document numérique, vol. 7, n°3-4, Hermes, 2003, 188p.

[HUF52] D. A. Huffman. *A method for the construction of minimum-redundancy codes*. Proceedings of the IEEE, 40(9) : 1098–1101, Septembre 1952.

[HUN80] R. Hunter, A. Robinson, *International Digital Facsimile Coding standards*, proc. IEEE,68:854-867, 1980

[KIA97] O. E. Kia, *Document Image Compression and Analysis*, PhD of the university of Maryland, 1997, 191 p.

[LEB00] F. Le Bourgeois, H. Emptoz, E. Trinh, F. Muge, C Pinto, I. Granado, *DEBORA Telematic Applications. Programme (project no.5608) WP 4.3 & WP 4.4 Description du matériel et logiciel de traitement d'image pour la numérisation des collections et leur interprétation*, 2000.

[LEB03] F. Le Bourgeois – H. Emptoz – E. Trinh, *Compression et accessibilité aux images de documents numérisés*, Document Numérique 7(3-4):103-125, Hermes Lavoisier, ISBN 2-7462-0845-8. 2003

[LEB04] F. Le Bourgeois, E. Trinh, B. Allier, V. Eglin, H. Emptoz, *Document Images Analysis Solutions for Digital libraries*, 1st International Workshop on Document Image Analysis for Libraries, 2004, Palo Alto, CA, USA. IEEE Computer Society 2004, pp. 2-24

[LEB05] Frank Lebourgeois, Hubert Emptoz, *DEBORA: Digital AccEss to BOoks of the RenAissance*, International Journal on Document Analysis and Recognition, Special Issue on Analysis of Historical Documents. 12/2005 (à paraître)

[LEY04] Y. Leydier, F. Le Bourgeois et H. Emptoz : *Sérialisation du k-means pour la segmentation des images en couleur*. Dans Actes du 8ième colloque international francophone sur l'écrit et le document, CIFED, 21-25 juin 2004, La Rochelle. pp. 191-196. 2004.

[MAD05] Projet MADONNE 2003-2006 Site Web : http://l3iexp.univ-lr.fr/madonne/

[ONO00] F. Ono, W. Rucklidge, R. Arps, and C. Constantinescu. *JBIG2-the ultimate bi-level image coding standard*, pp. 140–143, Proceedings, 2000 International Conference on Image Processing, (Vancouver, BC, Canada), vol. 1.

[PAV81] Pavlidis, T. *A Flexible Parallel Thinning Algorithm*. Proc. IEEE Comput. Soc. Conf. on Pattern Recognition and Image Processing. Août. 1981, pp. 162-167.

[JOR79] Jorma J. Rissanen et Glen G. Langdon. *Arithmetic coding*. IBM Journal of Research and Development, 23:149–162, 1979.

[SAU97] J. Sauvola, T. Sepplanen, S. Haapakoski, et M. Pietiklainen, *Adaptive Document binarization*, 4th International Conference on Document Analysis and Recognition, Ulm, Allemagne, Août, 1997.

[SER04] SEROPIAN, A., GRIMALDI, M., VINCENT,N. *Différenciation entre alphabets dans des textes manuscrits*, Colloque International Francophone sur l'Ecrit et le Document, La Rochelle, pp.134_144, 2004.

[TRE04] A. Trémeau, C. Fernandez-Maloigne et P. Bonton, *Image numérique couleur, de l'acquisition au traitement*, DUNOD, pp. 300

[WEL84] T. A. Welch, "*A technique for high-performance data compression*." Computer. Vol. 17, pp. 8-19. Juin 1984

[WES03] Westeel, I., Aubry, M., *La numérisation des textes et des images : Technique et Réalisations*, Presses de l'université Charles de Gaulle - Lille, 2003, 190 p.

[WIT94] S.Inlis et I.Witten. *Compression-based template matching*. In proceedings of the IEEE Data Compression Conference, pages 106-115, 1994.

[ZHA84] T. Y. Zhang et C. Y. Suen. *A Fast Parallel Algorithm for Thinning Digital Patterns*. Communications of the ACM. Mars 1984

[ZHA95] YING ZHANG, LAI-MAN PO, Fractal Color Image Compression using vector distortion measure, Proceeding of IEEE Inter. Conf. on Image Processing,, III – 276 to III – 279, D.C, U.S.A, October 1995.

[ZIV77] Jacob Ziv et Abraham Lempel; A Universal Algorithm for Sequential Data Compression, IEEE Transactions on Information Theory, Mai 1977.

[ZIV78] Jacob Ziv et Abraham Lempel; Compression of Individual Sequences Via Variable-Rate Coding, IEEE Transactions on Information Theory, Septembre 1978.