Une méthode de compression sans perte pour les images de documents basée sur la séparation en couches

Abdelâali Hassaïne[1] Véronique Eglin[2] Stéphane Bres[2]

[1] CMM, Ecole des Mines de Paris
35 Rue Saint Honoré
77300 Fontainebleau

hassaine@cmm.ensmp.fr

[2] Laboratoire LIRIS-UMR 5205 CNRS
INSA de Lyon F-69621
Villeurbanne Cedex

{veronique.eglin,stephane.bres}@insa-lyon.fr

Résumé

Les usagers des bibliothèques numériques souhaitent avoir un accès rapide aux images de documents. La compression de ce type d'image est devenue donc un besoin incontournable. Bien que certaines méthodes de compression donnent de bons résultats sur des textes imprimés, aucune d'elles ne s'est montrée efficace sur les manuscrits anciens très dégradés. Nous proposons une méthode de compression structurant les informations en couches du plus au moins important et permettant ainsi une transmission progressive. Notre méthode exploite la similarité des formes dans le squelette du tracé. Toutes les formes sont générées à partir de certaines formes de base et d'une variété de transformations géométriques. Les données permettant de reconstruire l'image originale sont ensuite mémorisées 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 par similarité donnant des résultats très intéressants et pouvant être utiles pour une variété d'applications sur les images de documents.

Mots clefs

Compression sans perte des manuscrits du patrimoine, codage, similarités, séparation en couches d'informations.

1 Introduction

La numérisation des images de documents facilite l'accès à un large public et offre de nouveaux services tels que la consultation en ligne des documents rares, une duplication rapide et économique des ouvrages, une navigation simplifiée, une recherche efficace de l'information et la possibilité de partage de l'information [1]. Une méthode de compression efficace est nécessaire pour la consultation en ligne de documents. Une telle méthode de compression doit permettre une transmission progressive de l'information, de la plus à la moins importante [2].
Les algorithmes de compression génériques tels que JPEG ou les autres méthodes dédiées aux textes imprimés telles que DjVu [2] ou DEBORA [3] ont montré leurs limites sur les images de manuscrits dégradés. Ces limites sont dues à la présence d'irrégularités et de dégradations aussi bien sur la partie textuelle que sur l'arrière-plan. De plus, DjVu et DEBORA sont des méthodes de compression avec perte. Il n'existe pas à nos jours une méthode de compression sans perte complètement dédiée aux images de manuscrits. Le standard de compression JBIG2 [4] donne des taux de compression très hauts pour les images binaires de textes imprimés, mais ces taux diminuent rapidement dès que l'on passe aux manuscrits anciens.
Pour ces raisons, nous proposons ici une méthode de compression sans perte plus adaptée aux images de documents manuscrits. 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 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.

2 Proposition

Notre méthode de compression comporte les étapes suivantes :

- La séparation texte/non-texte par la création du masque binaire ;
- La squelettisation du masque binaire pour rendre l'analyse indépendante de l'épaisseur du tracé ;
- La décomposition du squelette en graphèmes et classification des motifs dans une table de similarité ;
- La restauration du squelette exploitant la redondance des formes ;
- La restauration du masque binaire à partir de la sauvegarde de l'épaisseur des traits ;
- La restauration de l'image originale à partir de la sauvegarde initiale de la couleur de l'arrière-plan et du tracé.

2.1 Séparation texte/non-texte

Cette séparation est réalisée à l'aide d'un k-means local initialisé avec les deux niveaux de gris les plus extrêmes. Cet algorithme donne de bons résultats sans être trop lourd en termes de temps de calculs.

2.2 Squelettisation du masque binaire

La squelettisation est une méthode appropriée parce qu'elle permet de rendre l'analyse indépendante du tracé et donc d'augmenter le taux de redondance. Le squelette que l'on cherche à obtenir doit avoir une épaisseur égale à l'unité pour être compressé efficacement, il doit aussi permettre la reconstruction du masque binaire. Plusieurs méthodes de squelettisation existent mais aucune d'entre elles ne possède de telles propriétés. Nous proposons alors une nouvelle méthode de squelettisation. Nous calculons horizontalement la carte des distances du masque binaire, nous déterminons ensuite les maxima locaux de cette carte. Le résultat ainsi obtenu peut avoir une épaisseur différente de l'unité. Pour cette raison, nous le squelettisons en utilisant l'élément structurant L (figure 1). La reconstruction du masque binaire sera abordée dans la section 2.6.

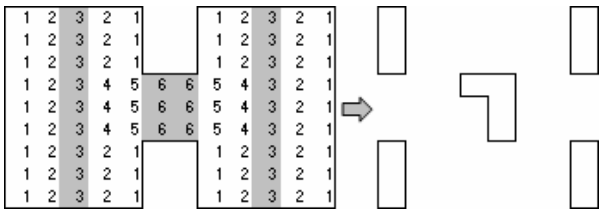


Figure 1 – *Etapes de notre méthode de squelettisation*

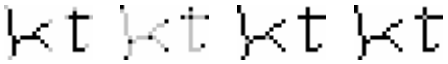
2.3 Décomposition du squelette

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 (figure 2).

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 (figures 3.c et 3.d)

Figure 2 – *Calcul du nombre de parties connexes*



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

Figure 3 – *Les quatre types des pixels du squelette*

Pour décomposer le squelette à partir de ses pixels de jonction, nous le considérons privé de ces pixels et de leurs 8 voisins respectifs (figure 4.b). Nous considérons ensuite chaque partie connexe du squelette obtenue comme un graphème indépendant (figure 4.c). Nous relions ensuite chacune des parties connexes aux graphèmes se trouvant dans leur voisinage (figure 4.d). Le pixel de jonction est relié à la première partie connexe en favorisant les voisins directs (dans le voisinage 4) (figure 4.e).

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

Figure 4 – *1ère étape de décomposition du squelette*

Des carrés de longueurs 2 pixels constituent un second type de pixels de jonction. Nous décomposons le squelette à partir de ces pixels de jonction de la même manière que précédemment (figure 5).

(a) (b) (c)

Figure 5 – *2ème étape de décomposition du squelette*

La figure 6 illustre un résultat complet de cette décomposition.

Figure 6 – *Exemple d'une décomposition du squelette*

2.4 Détection des similarités

Nous avons ensuite proposé trois méthodes pour l'extraction des formes similaires dans le squelette. Dans

la première méthode, les formes sont sous-échantillonnées sur une fenêtre d'une taille définie puis comparées (figure 7). Cette méthode donne de bons résultats quand le texte est très régulier, comme dans les documents du Moyen-Âge par exemple.

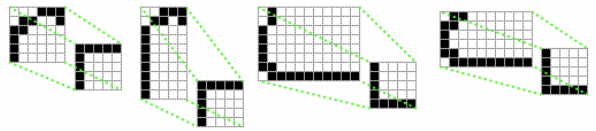


Figure 7 – *Sous-échantilonnage des formes*

Pour des formes moins régulières, la seconde méthode est plus adaptée : au lieu d'affecter une valeur binaire à chaque pixel dans la fenêtre de sous-échantillonnage, nous affectons la proportion des pixels qui lui correspondent, ceci génère la fenêtre de distribution (figure 8). Nous avons remarqué que chaque forme possède une distribution qui est assez proche de l'une des distributions de base illustrées dans la figure 9. Nous avons alors assigné chaque forme à la distribution de base qui lui est la plus proche en termes de somme des différences absolues. Cet assignement nous a permis de créer la table de similarité (figure 10.a). En affectant ensuite chaque pixel du texte à la forme du squelette la plus proche, on peut voir comment cette similarité se répercute sur l'image originale (figure 10.b).

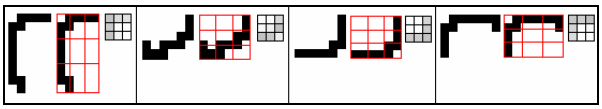


Figure 8 – *Fenêtres de distribution*

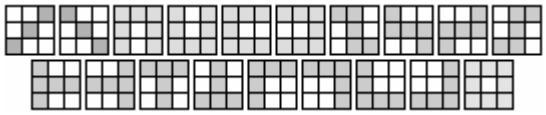
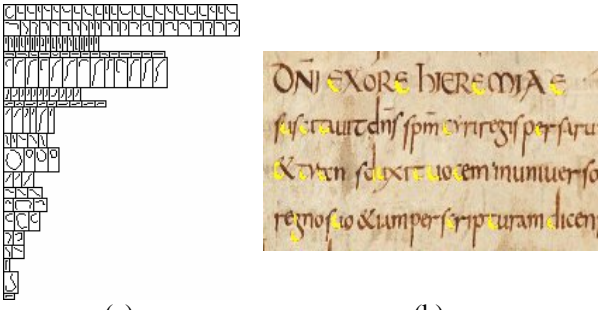


Figure 9 – *Distributions de base*



(a) (b)

Figure 10 – *a : table des similarités, b : formes similaires dans le texte*

Ces deux méthodes de détection de similarités ne sont pas utilisées pour le moment dans notre méthode de compression, mais elles illustrent la diversité des applications que l'on peut réaliser avec notre approche de décomposition du squelette.

La troisième méthode utilise le code de Freeman qui permet de coder efficacement de telles formes. Il permet aussi de définir facilement des transformations géométriques. La figure 11 montre une forme, son code de Freeman et la configuration utilisée pour la génération du code de Freeman.

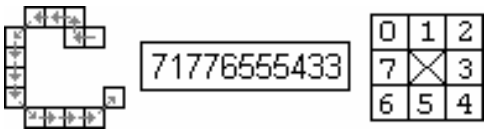


Figure 11 – *Génération d'un code de Freeman*

Après le calcul du code de Freeman de chaque forme, nous définissons des ensembles de transformations permettant de déterminer quelles formes de base et quelles transformations vont être utilisées pour générer toutes les autres formes.

Nous sélectionnons ensuite l'ensemble des transformations qui maximise le taux de redondance. Ces ensembles de transformations sont basés sur le code de Freeman et sont définis comme suit :

Notons tout d'abord qu'une rotation de i·45° (pour i=0..7) peut être obtenue en ajoutant i modulo 8 à chaque valeur du code de Freeman. Ces 8 rotations de bases vont être composées avec les trois ensembles suivants de 32 transformations pour obtenir trois ensembles de transformations de 256 transformations.

- L'ensemble des rotations : pour générer une rotation d'un degré inférieur à 45°, il faut ajouter à la chaîne de Freeman une succession de 0 et de 1. Nous définissons 32 différentes successions de 0 et de 1 : {(01),(001),(100),...} pour générer 32 différentes transformations. La figure 12.a montre l'application de ces rotations à une forme élémentaire. Notons qu'il ne s'agit pas de rotations euclidiennes exactes mais seulement d'approximations.
- L'ensemble des symétries : une symétrie peut être générée en remplaçant chaque valeur du code de Freeman par une autre qui lui est symétrique dans la configuration de base du code de Freeman. Par exemple, en remplaçant chaque 0 (respectivement 7 et 6) dans une chaîne de Freeman par 2 (respectivement 3 et 4), ceci implique le remplacement de chaque segment dirigé vers la droite par un autre dirigé vers la gauche. Nous proposons de la même manière un ensemble de 32 remplacements différents pour générer cet ensemble de transformations. La figure 12.b montre l'application de ces symétries à une forme élémentaire.
- L'ensemble des lissages : théoriquement, le lissage d'un code de Freeman est obtenu par un filtre médian. Pour accélérer les calculs, nous proposons une variante : nous appelons le lissage d'ordre n d'un code de Freeman *F*, le code de Freeman *Fs* obtenu en affectant la première valeur

de *F* aux *n* premières valeurs de *Fs* et la (*n+1*)ème valeur de *F* aux *n* valeurs suivantes de *Fs* et ainsi de suite. Ceci signifie que : *Fs[i]=F[i/n] (pour chaque i)*. La figure 12.c illustre le résultat de ces 32 lissages sur une forme élémentaire. Remarquons qu'un lissage d'ordre *n* d'un code de Freeman converge dés que *n* atteint la taille du code de Freeman.

(a) (b) (c)

Figure 12 – *a : rotations, b : symmétries, c : lissages*

Pour comparer deux formes, on calcule leurs codes de Freeman. Pour que la comparaison puisse être effectuée, il faut que les deux formes aient la même longueur, si ce n'est pas le cas, nous comparons le début du plus long avec le plus petit.
Soient *F1* et *F2* deux codes de Freeman de même longueur et soit *T* une transformation géométrique. Nous disons que *F2* peut être généré à partir de *F1* par la transformation *T* si et seulement si toutes les différences entre *F2* et *T(F1)* sont inférieures ou égales à 1. Si tel est le cas, on peut reproduire *F2* sachant *F1*, la transformation *T* et la suite des différences. Nous verrons dans la prochaine section comment cette comparaison va être utilisée.

2.5 Compression du squelette

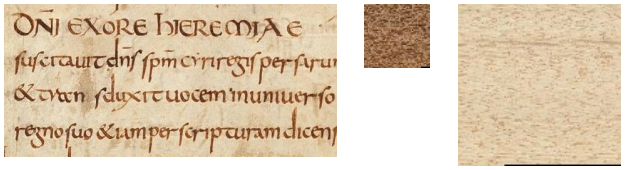
Pour pouvoir générer le squelette, les données suivantes doivent être stockées : tout d'abord, les positions des formes sont efficacement stockées en utilisant le code de Huffman [5]. Chaque forme est ensuite comparée à toutes les autres en essayant toutes les transformations d'un ensemble donné. Après ces comparaisons, les codes de Freeman des formes de base (qui permettent de générer toutes les autres formes) sont compressés en utilisant un codage prédictif [6]. Pour les autres formes qui peuvent être générées à partir des formes de base, il suffit de sauvegarder le numéro de la forme de base les ayant généré, le numéro de la transformation et la suite des différences. Cette opération est répétée avec tous les ensembles. On choisit à chaque fois l'ensemble qui maximise le taux de redondance.

2.6 Compression du masque binaire

On peut reconstruire le masque binaire en sauvegardant l'épaisseur droite et gauche de chaque pixel du squelette (figure 1). Ces différentes valeurs sont efficacement compressées avec un codage prédictif.
Notons que l'on pourra utiliser une seule épaisseur en travaillant avec les contours droits (ou gauches) au lieu du squelette.

2.7 Compression des couleurs

Une fois le masque binaire reconstruit, on peut distinguer les pixels du fond des pixels du texte. 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 13.



(a) (b) (c)

Figure 13 – *a : image originale, b : image du texte, c : image du fond.*

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 (Lempel-Ziv Markov chain Algorithm) LZMA [7] si les couleurs sont indexées.

3 Résultats de compression

Notre méthode de compression peut aussi bien être utilisée pour la compression des images binaires de documents (les couleurs dans ce cas ne sont pas stockées). Nous avons calculé les taux de compression de notre méthode et nous l'avons comparé aux autres méthodes.
La table 1 illustre cette comparaison. La ligne SKL montre le résultat obtenu avec notre méthode. Dans la ligne EDGE, les contours droits sont utilisés au lieu du squelette (la méthode de compression dans ce cas reste la même sauf qu'il ne faut enregistrer qu'une seule épaisseur). Bien que notre méthode admette plusieurs optimisations, on peut voir que les taux de compression sont comparables à ceux des autres méthodes.
La table 2 illustre les résultats sur des images de documents en couleurs. Notons que les algorithmes JPEG et JPEG2000 utilisés dans la comparaison sont sans perte.
Dans les manuscrits anciens, le standard JPEG2000 donne de meilleurs taux de compression mais il ne permet pas de compresser séparément le texte et l'arrière-plan, chose qui est généralement utile pour ce type d'images notamment

dans l'exploitation du contenu du document (i.e. pour une analyse ou un travail de transcription du texte seulement).
Le temps de compression dépend du nombre de formes contenues dans le texte. Il atteint plusieurs dizaines de secondes pour un manuscrit typique. Le temps de décompression quant à lui est d'environ une seconde.

	Manuscrits anciens			Textes imprimés		
TIFF	2.65	4.59	2.42	1.88	3.17	5.98
ZIP	3.62	7.17	3.73	3.02	5.89	10.80
PNG	3.98	7.99	4.09	3.21	6.43	12.26
RAR	4.00	8.33	4.23	3.31	6.65	12.87
CAB	4.14	8.61	4.46	3.40	6.98	12.91
7Z	4.49	9.67	4.81	3.61	7.42	14.49
JBIG2	8.04	16.91	6.78	6.73	10.09	23.17
SKL	4.82	10.15	4.02	3.19	6.96	11.62
EDGE	5.06	10.98	4.91	3.26	7.40	13.40

Table 1 – *Taux de compression sur des images binaires*

	Manus. anc.		Textes imp.		Manus. hom.	
PNG	1.58	1.59	14	7.18	11.69	13.4
JPEG	1.33	1.32	2.6	3.49	4.64	3.46
TIFF	1.38	1.39	8.61	6.84	16.01	10.72
ZIP	1.27	1.23	16.6	7.46	14.61	14.92
RAR	1.82	1.58	25.6	8.28	19.48	17.65
CAB	1.43	1.45	28.9	9.29	21.44	19.82
JP2	2.43	2.49	5.56	9.76	17.84	8.93
7Z	1.55	1.49	32	10.12	29.22	20.67
EDGE	1.58	1.78	24.21	9.19	29.97	25.38

Table 2 – *Taux de compression sur des images de documents en couleurs*

Manus. anc.=manuscrits anciens ; Textes imp.= textes imprimés ; Manus. hom.= manuscrits homogènes (avec très peu de couleurs).

4 Conclusion

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 ensuite créées deux images (les images de couleurs) contenant l'ensemble des informations couleurs de ces deux ensembles de pixels. Chacune de ces deux images est compressée par un algorithme adapté aux variations de couleurs observées.
Plusieurs pistes restent à exploiter notamment :
- L'optimisation plus adaptée pour chaque couche ;
- L'élaboration d'une méthode de squelettisation permettant une reconstruction complète de l'image binaire et ne générant pas des « petits » fragments 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.
Notre méthode ouvre donc des pistes à plusieurs optimisations pouvant l'améliorer considérablement et constitue dans tous les cas une avancée très prometteuse dans un domaine où les travaux antérieurs restent rares.

Références

[1] F. Lebourgeois, E. Trinh, B. Allier, V. Eglin, et H. Emptoz, "*Document images analysis solutions for digital libraries*", dans 1st International Workshop on Document Image Analysis for Libraries, 2004, pp. 2–24.

[2] P. Haffner, L. Bottou, P.G. Howard, et Y. LeCun, "*Analyzing and compressing scanned documents for internet distribution*," dans International Conference on Document Analysis and Recognition, 1999.

[3] F. Lebourgeois et H. Emptoz, "*Debora: Digital access to books of the renaissance*" International Journal on Document Analysis and Recognition, 2005.

[4] F. Ono, W. Rucklidge, R. Arps, et C. Constantinescu, "*Jbig2-the ultimate bi-level image coding standard*", dans International Conference on Image Processing, 2000, pp. 140–143.

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

[6] T. Bell, J. Cleary et I. Witten, "*Data compression using adaptive coding and partial string matching*". IEEE Transactions on Communications. 1984.

[7] http://tukaani.org/lzma/, 2005.