Stratégie d'Application de Traitement d'Image sur des Flux Compressés

C. Sibade[1,2] M. Akil[2] L. Perroton[2] S. Barizien[2]

[1] Département CCR, Océ Print Logic Technologies
1 rue J. Lemoine, 94015 Créteil Cedex, France

[2] Laboratoire A[2]SI, Groupe ESIEE,
Cité Descartes B.P. 99, 93162 Noisy-Le-Grand Cedex, France

{cesi}@ocegr.fr

Résumé

La quantité de données transmises, stockées et traitées dans les systèmes d'impression grand format, développés en particulier par la société Océ PLT, est aujourd'hui très importante. La représentation interne des images est de-facto non compressée, afin de faciliter les traitements et opérations qui doivent être pratiqués sur ces images. L'utilisation de la compression de données permet de diminuer fortement la masse de ces données numériques, mais son intégration dans une chaîne complète de traitement d'image n'est pas aisée. Nous étudions la méthodologie d'application de ces transformations sur un format compressé. Nous décrivons aussi les approches proposées par JPEG2000.

Mots clefs

Compression d'image, traitements sur flux compressés, impression grand format.

1 Introduction

Les systèmes actuels de traitement de données numériques évoluent pour proposer des solutions de plus en plus performantes en répondant en particulier à une demande de qualité accrue. Un meilleur rendu des données, qui nécessite une plus grande précision dans leurs représentations, est un des exemples dans le cas du traitement des images. Et c'est donc logiquement que l'on voit une augmentation des quantités brutes d'information qui sont transmises, sauvegardées et traitées au sein des systèmes. Cette constatation est clairement vérifiée dans notre domaine d'activité : les chaînes de traitement de documents grand format. Une résolution plus élevée, une meilleure qualité de rendu des couleurs et des vitesses de traitement toujours plus élevées sont demandées par les clients ; les nouveaux scanners ou imprimantes grands formats doivent donc s'y adapter. Une affiche d'une taille A0, c'est-à-dire d'une surface égale à 1 mètre carré, en vraies couleurs et avec une résolution de 600 points par pouce occupe un espace de 1,6 G Octets à stocker ou à transférer.
Face à cet enrichissement de l'information, une première réponse est de modifier les systèmes physiques pour satisfaire ces besoins : le développement de ces nouveaux produits capables de traiter ces tailles de données grandissantes requiert l'utilisation de plus grands espaces de stockage et de technologies de communication plus rapides. Cependant ces changements ont un coût non négligeable.
Nous avons proposé d'utiliser une autre solution en essayant d'optimiser non pas le matériel, mais plutôt les flux mêmes de données (cf. [1]). La Compression de Données apporte aujourd'hui des solutions face à cette inflation. Avec l'arrivée de nouveaux standards (tels que JPEG2000, décrit en particulier dans [2]), nous disposons d'outils performants pour coder efficacement nos importantes quantités de données (nous avons montré que JPEG2000 était l'algorithme de compression qui, aujourd'hui satisfait un grand nombre de nos contraintes : bonnes performances en terme de taille compressée, travail avec ou sans perte, prise en compte de différents contenus de couleurs, prédécoupage de l'image en bandes...). Et plutôt que de compresser ponctuellement les flux ou fichiers qui sont limitants, nous avons proposé dans [1] d'optimiser l'utilisation de ces compressions : nous cherchons à conserver un format compressé tout au long de notre chaîne de traitement. De cette manière, le format compressé est utilisé comme une *nouvelle représentation des données*. Nous étendons alors les bénéfices de l'utilisation de la compression sur une plus grande chaîne. La Figure 1 présente ce que serait un flot de données optimisé et idéal (compression **C** dès l'acquisition de l'image et décompression **D** avant l'impression) pour un processus de copie, du scanner vers l'imprimante.

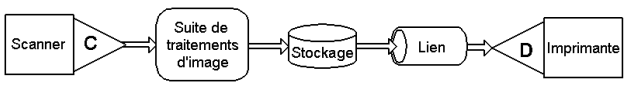

Figure 1 – *Exemple de chaîne de copie idéale utilisant une représentation compressée.*

Cette chaîne compressée nécessite de modifier la série de traitements qui y sont effectués. Nos systèmes d'im-

pression/copie grand format permettent d'appliquer une séquence de transformations géométriques et colorimétriques sur les images. Par exemple, la chaîne de traitement peut comprendre un changement de taille (soit un réechantillonnage), une rotation du mode paysage en mode portrait puis finir par une *rasterization* (qui permet de produire la bitmap prête à être imprimée).
Cet article présente donc les aspects fondamentaux de prise en compte de la compression pour une classe d'algorithmes de traitement d'image : plutôt que de décompresser, d'appliquer la transformation et de revenir à la représentation compressée, on cherche à opérer directement cette transformation sur les données compressées. Pour une multitude de formats de compression, une stratégie commune peut être adoptée : les données sont décodées –transformées d'un codage binaire vers une représentation intermédiaire. Il est ensuite possible de produire le traitement équivalent à celui effectué sur les données brutes, sur cette modélisation. Il reste alors à ré-encoder ces données transformées. Par ces approches, nous optimisons l'utilisation des ressources en évitant d'effectuer une décompression/re-compression globale.
La partie 2 décrit l'organisation de la majorité des algorithmes de compression de données. La 3ème partie expose la méthode d'application de ces transformées sur le flux compressé. La partie 4 énumère les possibilités d'utilisation de ces transformées en mode compressé et illustre cette stratégie en décrivant l'approche préconisée par le nouveau standard JPEG2000.

2 Structure d'algorithmes de compression

La majorité des algorithmes de compression peut se décomposer en trois opérations successives :

- **Modélisation des données :** ce premier processus produit une nouvelle représentation des données à partir de l'information brute de chaque pixel. Ce traitement ne dégrade typiquement pas l'information, et la représentation produite, la **modélisation**, met en évidence des propriétés statistiques qui seront prises en charge par le dernier étage du compresseur, le codeur.
- **Quantification :** cette deuxième opération introduit des pertes dans la nouvelle représentation de l'information produite par le modeleur ; ces distortions ont pour but de renforcer la distribution statistique, afin de mieux coder les informations. Cet étage de **quantification** permet donc une meilleure compression, au détriment d'une perte de qualité ; il est cependant optionnel si l'on veut retrouver l'information initiale exacte.
- **Codeur statistique :** ce dernier processus utilise le modèle de données (qui a pu être quantifié) pour créer un message binaire, de taille inférieure aux données initiales, en utilisant les propriétés statistiques du modèle. Chacun de ses symboles va ainsi produire un mot de code de taille variable, en fonction de sa probabilité d'occurrence dans l'image initiale. Par exemple, des symboles fréquemment présents recevront un code plus petit, tandis que les moins fréquents se verront attribuer un message codé plus long. Deux principales familles d'algorithmes sont utilisées de nos jours : les méthodes à base d'arbre de Huffman et les méthodes de codage arithmétique. C'est cette opération de **codage** qui produit donc un flux de taille réduite : un flux compressé.

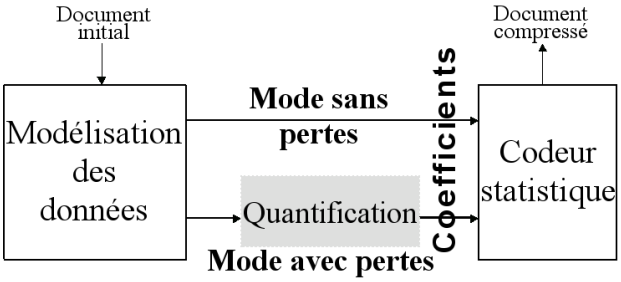

Figure 2 – *Organisation typique d'un compresseur*

Le processus de décompression suit l'ordre inverse : un **décodage** permet de retrouver les symboles quantifiés du modèle. Une **dé-quantification** (dans le cas d'une compression avec pertes) recrée le modèle ; la perte de précision obtenue à la quantification, ne permettra pas de recouvrer l'information exacte initiale, après application de la **modélisation inverse**. La Figure 2 illustre le processus de compression : Modélisation/(Quantification)/Codage (que nous noterons par $M \rightarrow (Q) \rightarrow C$).

3 Application d'opérateurs de traitement d'image

La représentation compressée, sous forme de message binaire est inadéquate pour les opérations de traitement d'image ; cette série de bits ne présente aucune organisation visible et les valeurs représentées ne sont pas corrélées à la teinte initiale du pixel. Le but est d'éviter de décompresser complètement le train binaire (en appliquant le décodage, puis la "dé-quantification" suivie de la modélisation inverse) pour revenir aux données brutes. Plutôt que d'appliquer la transformation "naturelle" de l'image T_{nc} (i.e. en mode non-compressé), on utilise une transformation T_c ($_c$ signifie "compressé") sur la représentation intermédiaire basée sur les coefficients du modèle. Le Tableau 1 décrit l'approche proposée.
Le gain de l'application de telles méthodes peut intervenir sur le temps et les ressources de calculs utilisées. En effet, la modélisation est souvent un processus "gourmand" en terme de calcul (filtrage, changement d'espace…).
Une propriété importante de l'application de transformations sur le flux compressé est la *localité* du traitement. Certains algorithmes de compression se basent sur des sous-parties de l'image (en prenant par exemple comme "unité" de traitement une ou deux lignes, un bloc de pixels, un bande horizontale de l'image…). Ce mode de parcours

Transformée naturelle	$C^{-1} \rightarrow (Q^{-1}) \rightarrow M^{-1} \rightarrow \mathbf{T_{nc}} \rightarrow M \rightarrow (Q) \rightarrow C$
Transformée en compressé	$C^{-1} \rightarrow (Q^{-1}) \rightarrow \mathbf{T_c} \rightarrow (Q) \rightarrow C$

Tableau 1 – *Comparaison entre l'application de transformations sur des données compressées T_c et sur des données non-compressées T_{nc}. On fera aussi la distinction entre la compression avec ou sans pertes (la quantification Q et la dé-quantification Q^{-1} sont alors optionnelles.)*

et d'utilisation de l'image originale est compatible avec les opérations de traitement d'image qui travaillent aussi en local, et plus particulièrement au niveau du pixel. Cette propriété sera illustrée dans la section suivante.
Dans ce cas-là, en appliquant la transformée en mode compressé T_c, ce n'est que la sous-partie de l'image qui se trouve transformée. Pour reconstituer l'image entière, on prendra soin de parcourir les sous-parties transformées dans l'ordre préconisé par la transformation. Par exemple, la rotation de 90^o en mode compressé R_c, appliquée sur des blocs de l'image (les blocs sont lus de gauche à droite, puis de haut en bas), effectue la rotation de chacun des blocs ; l'image globale transformée est constituée de chacun des blocs transformés, qu'il faut maintenant parcourir de haut en bas, puis de droite à gauche.

4 Compression et traitement d'image

Comme décrit précédemment, il est donc possible d'appliquer certains algorithmes de traitement d'image directement sur le flux décompressé – ou pour être plus précis sur le flux décodés. Cette section résume les possibilités existantes des transformations sur les images compressées.

4.1 Synthèse des opérations possibles

On se focalisera sur les algorithmes de traitement utilisés effectivement dans le domaine de l'impression de documents. Nous les classifions en trois catégories principales :
- Algorithmes travaillant au niveau du pixel : seule la valeur du pixel est utilisée. On utilise typiquement des traitements colorimétriques, de modification de *contraste* ou des *filtrages fréquentiels*
- Algorithmes de transformations géométriques : on s'attache à modifier l'apparence et la géométrie de l'image. On se base sur des opérations de *rotation* (par angles droits), de *symétrie*, de *translation*, de *découpage* ou d'extraction de régions de l'image, ainsi que les changements de taille (*agrandissement* ou *réduction*).
- Algorithmes haut niveau : les opérations effectuées sont des segmentations de régions, l'analyse du document...Nous n'étudierons pas sur ces types de transformations, qui opèrent sur des régions ou l'image entière.

Le Tableau 2 résume les opérations possibles à appliquer directement sur le flux décodé. Les colonnes 2 à 4 décrivent quels sont les modèles, quantifications et codeurs, employés pour chacune des méthodes de compression choisies. Certains traitements sont décrits dans les standards ; d'autres ont été développés depuis leurs publications et sont indiqués dans le Tableau 2 en *italique*.

4.2 Exemple de traitements d'image appliqués à JPEG2000

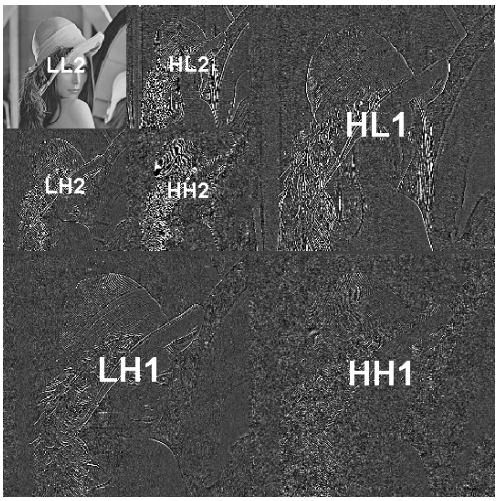

Figure 3 – *Décomposition dyadique en sous-bande par ondelettes. HH1 signifie filtrage passe-haut vertical et horizontal à la résolution 1.*

La définition du standard a été faite en prenant en compte les opérations à effectuer sur les données transformées (c'est-à-dire modélisées) par les ondelettes. Cette représentation en bandes de fréquence à différentes résolutions, permet cependant d'accéder aux données de façon spatiale et locale. La figure 3 présente un exemple d'une décomposition (sur deux niveaux) d'une image en niveau de gris. La bande basse LL2 (image de taille $\frac{1}{4}$), ainsi que les autres bandes fréquentielles contiennent les coefficients filtrés. Ils correspondent à l'emplacement spatial exact (à un facteur de résolution prêt) du pixel de l'image originale non-transformée.
A partir de cette représentation (image décomposée en bandes de fréquence), nous appliquons les transformations suivantes (les quatre premières transformations sont décrites dans le standard ; les suivantes peuvent être aisément implémentées connaissant les propriétés de la transformation en ondelettes et de l'organisation des bandes) :
- **Réduction de taille $\frac{1}{4}$ et $\frac{1}{2}$** : il suffit de ne conserver respectivement que la bande basse LL2 ou les quatre bandes basse de la résolution 2, soient HH2, HL2, LH2 et LL2.
- **Rotation de 90^o, 180^o ou 270^o**. Chaque bande doit être tournée, et il faut échanger les bandes HL et LH (le filtrage horizontal devient vertical et vice-versa). Cette méthode est illustrée sur le Figure 4.
- **Symétrie suivant l'axe vertical ou l'axe horizontal** :

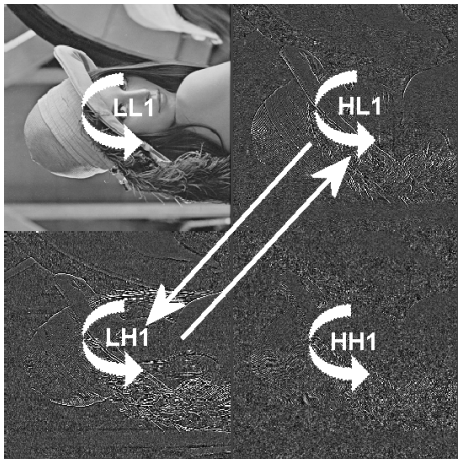

Figure 4 – *Rotation de 90 degrés par rotation et échange des sous-bandes*

cette transformation s'effectue de la même manière que la rotation (sans échange de sous-bandes) mais en utilisant la symétrie sur chacune des bandes.

– **Extraction/découpage d'une région** : si on choisit une sous-région de l'image originelle, cela est équivalent à choisir une sous-région (à un facteur de résolution prêt) dans chaque bande. Il faut cependant effectuer un traitement local au niveau des "bords découpés" afin de retrouver l'information initiale (les coefficients des sous-bandes sont obtenus par filtrage *des* pixels originaux autour du pixel visé).
– **Décalage de l'image** : cette opération est une extension de la précédente méthode : extraction d'une région suivie d'un décalage de cette région découpée.
– **Augmentation/diminution de la teinte** : en ne changeant que les valeurs de la bande basse, on joue sur les informations basses fréquences (les teintes moyennes de l'image). Ajouter ou retrancher une même teinte à tous les coefficients de LL2 dans la figure 3) fait varier aussi la teinte globale de l'image reconstruite.
– **Filtrage fréquentiel** (passe-bas) de l'image : comme JPEG2000 effectue un filtrage récursif de l'image, on peut décider d'annuler les bandes hautes (HL1, HH1 et LH1 dans la figure 3) pour conserver seulement l'information basse fréquence.

5 Conclusions

Nous avons synthétisé les principes d'utilisation de transformations d'image pour des données compressées. Le décodage permet de revenir au modèle de la compression ; ensuite on peut appliquer certaines opérations directement en les adaptant à cette représentation interne de la compression. Ce principe devient alors d'une grande utilité, en particulier dans le domaine de l'impression de document grands formats. JPEG2000 propose d'ailleurs de nombreuses possibilités d'implémenter des transformations géométriques et colorimétriques directement sur la représentation sous forme de sous-bandes (obtenues après un filtrage à base d'ondelettes).
Notre étude se poursuit afin de vérifier et de quantifier le gain obtenu par ces transformations. Nous nous attachons aussi à proposer de nouvelles transformations, dédiées au monde de l'impression.

Références

[1] Cédric Sibade, Stéphane Barizien, Mohamed Akil, et Laurent Perroton. Wide format raster compression applied to a printing environment. Dans *Proceedings of NIP 18 conference*, Octobre 2002.

[2] M. Rabbani et R. Joshi. An overview of the JPEG2000 still image compression standard. *Signal Processing : Image Communication Journal*, 17(1), Janvier 2002.

[3] Stephen J. Urban. Review of standards for electronic imaging for facsimile systems. *Journal of Electronic Imaging*, 1(1) :5–21, Janvier 1992.

[4] ISO/IEC ITU-T. Information technology - coded representation of picture and audio information - progressive bi-level image compression. ITU-T Rec. IS 11544 / ITU-T T.82, ISO/IEC ITU-T, 1993.

[5] M. J. Weinberger, G. Seroussi, et G. Sapiro. The LOCO-I lossless image compression algorithm : Principles and standardization into JPEG-LS. *IEEE Trans. on Image Processing*, 9(8) :1309–1324, Aout 2000.

[6] Gregory K. Wallace. The JPEG still picture compression standard. *Comm. of the ACM*, 34(4), Avril 1991.

	Modélisation	Quantification	Codeur statistique	**Transformations en compression**
CCITT fax[3]	RLE	Non	Huffman	*Translation, découpage, symétrie, rotation*
JBIG[4]	Prédictions	Non	Arithm.	Réduction/agrandissement implicite
JPEG-LS[5]	Prédictions/RLE	Non(oui)	Arithm.	Aucunes
JPEG[6]	DCT	Oui	Huffman	*Constraste, rotation, symétrie, réduction/agrandissement, translation (8 par 8)*
JPEG2000[2]	Ondelettes	Oui(non)	Arithm.	Rotation, symétrie, réduction/agrandissement, découpage, extraction

Tableau 2 – *Comparaison entre les algorithmes de compression et la possibilité de transformation en mode compressé.*