Un histogramme compact pour l'analyse d'images multi-composantes

Alain CLÉMENT, Bertrand VIGOUROUX

Laboratoire d'Ingénierie des Systèmes Automatisés
Institut Universitaire de Technologie d'Angers
BP 42018, 49016 ANGERS CEDEX

alain.clement@iut.univ-angers.fr, bertrand.vigouroux@univ-angers.fr

Résumé – Nous présentons l'algorithme de calcul d'un histogramme compact permettant de réduire de manière considérable et sans perte d'information, l'espace mémoire nécessaire au codage des histogrammes d'images multi-composantes. En résultats, nous vérifions sur des exemples les performances de l'algorithme à la fois en gain de mémoire et en rapidité de calcul.

Abstract – We present a compact histogram computation algorithm which considerably reduces, without loss, the amount of memory required to store multidimensional image histograms. As results, we verify on some examples the performances of the proposed algorithm both in memory gain and speed.

1. Introduction

L'histogramme d'une image est la représentation d'une fonction discrète qui à chaque valeur de l'image associe le nombre de pixels ayant cette valeur. Il peut être vu comme la densité de probabilité d'une variable aléatoire dont l'image constitue un ensemble de réalisations. En traitement d'images, les calculs d'histogrammes sont nombreux et fréquents : depuis la mise au point de la chaîne d'acquisition jusqu'à l'indexation d'images dans une base, de nombreuses méthodes de pré-traitements, de segmentation ou de mesure statistique nécessitent le calcul d'histogrammes.

L'histogramme P-dimensionnel d'une image de résolution M x N à P composantes codées chacune sur Q bits est constitué par un tableau P-dimensionnel comportant 2^{PQ} cellules. Chaque cellule de ce tableau doit pouvoir contenir un nombre au maximum égal à MN, donc codé sur un minimum de $\log_2(MN)$ bits. L'histogramme occupe alors

$$2^{PQ-3} \log_2(MN) \qquad (1)$$

octets de mémoire. Devant la difficulté à manipuler un tel volume de données, en pratique de l'ordre de plusieurs dizaines de méga-octets, il existe couramment deux stratégies :

- La première [1, 2] consiste à se limiter aux P histogrammes marginaux mono-dimensionnels, en n'exploitant pas la corrélation existant entre les différentes composantes de l'image. Ces P histogrammes n'occupent plus que

  $$2^{Q-3} P \log_2(MN) \qquad (2)$$

  octets, mais une part importante de l'information contenue dans l'image est perdue.
- La seconde stratégie [3, 4, 5] consiste à requantifier chaque composante sur q bits (q < Q), ce qui revient à effectuer une classification a priori de l'image. Cette démarche est contraire aux efforts généralement déployés pour obtenir une chaîne d'acquisition précise et calibrée avec un haut pouvoir de discrimination colorimétrique.

2. Histogramme compact

2.1 Définition

Soit C le nombre de cellules occupées dans l'histogramme (C < MN). En pratique, C est toujours très inférieur au nombre total 2^{PQ} de cellules disponibles. À nombre de pixels MN constant, plus le nombre P de composantes de l'image est important, plus le nombre de cellules inoccupées augmente. Le principe de l'histogramme P-dimensionnel compact consiste à ne coder que les C cellules effectivement occupées en triant ces cellules par ordre lexicographique sur les P composantes de l'image. L'espace mémoire occupé par l'histogramme compact d'une image de résolution M x N à P composantes codées chacune sur Q bits est alors réduit d'un facteur

$$\frac{2^{PQ}}{C\left\lceil \frac{PQ}{\log_2(MN)} + 1 \right\rceil} \qquad (3)$$

Dans le cas d'une image couleur (P=3) de résolution 512 x 512 et dont chaque composante est codée sur 256 niveaux (Q=8), l'histogramme classique occuperait 36 Mo de mémoire. Si, au pire des cas, chaque pixel de l'image présente une couleur différente (C = 512 x 512), l'histogramme compact n'occupe que 1,31 Mo de mémoire.

L'histogramme compact d'une image à P composantes codées sur Q bits et contenant C valeurs différentes sera représenté par un tableau de dimensions C x P pour stocker

les valeurs triées dans l'ordre lexicographique, et un tableau de dimension C x 1 pour les effectifs correspondants. Le parcours complet de l'histogramme compact ne nécessite qu'une boucle de C itérations, alors que l'histogramme classique ferait appel à P boucles imbriquées de taille 2^Q et à 2^{PQ} tests d'effectifs.

2.2 Principe algorithmique

Le calcul de l'histogramme compact repose sur un tri lexicographique des MN valeurs de l'image structurées en P sous-clés selon la relation d'ordre définie par l'ordre des P composantes de l'image. La méthode de tri utilisée est celle du tri rapide par dichotomie [6]. Pour MN > 20 cette méthode est plus rapide que toutes les méthodes itératives et pour MN > 1000 plus rapide d'un facteur 1.5 à 2 [7] que le tri par tas [8].

Le principe du tri rapide est le suivant : étant donné une liste de MN éléments à trier, on choisit un élément et on répartit les autres en deux sous-listes, ceux qui sont inférieurs ou égaux à l'élément choisi, et ceux qui lui sont supérieurs. On applique le même processus à ces deux sous-listes récursivement jusqu'à avoir des sous-listes réduites à un élément. Le choix de l'élément permettant la formation de deux sous-listes, le pivot, est fondamental [9]. Le graphe des appels récursifs de l'algorithme de tri est un arbre binaire. Si cet arbre est équilibré, donc de hauteur $\log_2(MN)$, on a Θ MN $\log_2(MN)$ comparaisons à effectuer. Dans le cas particulier où la liste est strictement monotone, le pivot est un extremum et le parcours effectué par l'algorithme devient un arbre en peigne de hauteur MN. La complexité en nombre de comparaisons ou de transferts est alors maximale soit $\Theta (MN)^2$. Afin d'éviter ce cas de complexité au pire, le pivot a été choisi comme l'élément médian parmi le premier élément, le dernier élément et l'élément du milieu de la liste.

MN étant le nombre de pixels de l'image, la complexité en moyenne de l'algorithme de calcul de l'histogramme compact est donc Θ MN $\log_2(MN)$.

2.3 Implémentation

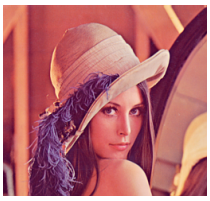
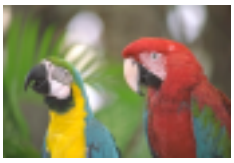
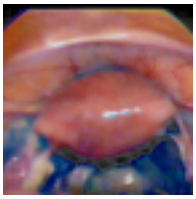
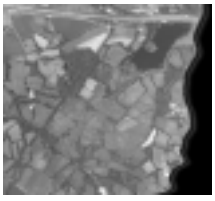
L'algorithme a été programmé en langage C en soignant tout particulièrement l'optimisation du code [10, 11] afin qu'un nombre minimal d'opérations soit effectué. Au niveau source, les calculs d'adresses redondants car implicites dans l'implantation du langage ont été supprimés et au niveau cible les registres de la machine ont été utilisés pour conserver les variables les plus souvent utilisées.

Le programme a ensuite été intégré sous forme de "Mex-File" dans une plate-forme de traitement d'images écrite sous MATLAB.

3. Résultats expérimentaux

Les exemples choisis (TAB. 1) sont des images appartenant à la base de données du Groupe de Recherche ISIS du CNRS. Ces images sont codées sur 8 bits par composante. Les temps de calculs présentés (TAB. 2) sont ceux obtenus par l'algorithme implanté dans l'environnement de traitement d'images avec un Macintosh G3 à 300 MHz. Les types de données MATLAB Double (64 bits) et Uint8 (8 bits) sont respectivement utilisés pour le codage des effectifs d'histogrammes et le codage des valeurs de chaque composante dans l'histogramme compact.

TAB. 1 : Caractéristiques des images.

Image	Nom et Résolution
	Clown 302 x 226 x 3
	House 256 x 256 x 3
	Lena 256 x 240 x 3
	Mandrill 256 x 256 x 3
	Parrots 384 x 256 x 3
	Utérus 512 x 512 x 3
	Multi-spectral M4 838 x 762 x 9

Tab. 2 : Volumes des histogrammes (en méga-octets) et temps de calcul de l'histogramme compact (en secondes).

Image	Volume Histo. classique	Volume Histo. compact	Temps de calcul
Clown	128	0.43	0.36
House	128	0.24	0.32
Lena	128	0.37	0.29
Mandrill	128	0.59	0.32
Parrots	128	0.41	0.49
Utérus	128	0.80	1.49
M4	$3.60\ 10^{16}$	1.89	13.55

Conclusion

Faible occupation mémoire, rapidité de calcul, rapidité d'exploitation et conservation complète de l'information sont les quatre avantages de l'histogramme multi-dimensionnel compact proposé ici. Conçu pour traiter un nombre quelconque de composantes, l'algorithme est capable de calculer tous les histogrammes de dimension inférieure ou égale au nombre de composantes de l'image. Les valeurs prises en compte sur chaque composante peuvent être limitées à un intervalle donné. L'algorithme s'adresse à tout type d'images multi-composantes, en particulier aux images en couleurs (P=3) ou multi-spectrales, et plus généralement à l'exploitation de tout ensemble de données corrélées.

Références

[1] P.A. Mlsna, J.J. Rodriguez. *A Multivariate Contrast Enhancement Technique for Multispectral Images*. IEEE Transactions on Geoscience and Remote Sensing, 33,1,212-216, 1995.

[2] M. Pietikäinen, S. Nieminen, E. Marszalec et T. Ojala. *Accurate Color Discrimination with Classification Based on Feature Distributions*. Proceedings of the 13th International Conference on Pattern Recognition, 833-838, 1996.

[3] A.L. Abbott, Y. Zhao. *Adaptive Quantization of Color Space for Recognition of Finished Wooden Components*. IEEE Workshop on Applications of Computer Vision, 252-257, 1996.

[4] D. Androutsos, K.N. Plataniotis et A.N. Venetsanopoulos. *Extraction of Detailed Image Regions for Content-Based Image Retrivial*. Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 6, 3713-3716, 1998.

[5] S.C. Cheng, C.K. Yang. *A fast and novel technique for color quantization using reduction of color space dimensionality*. Pattern Recognition Letters, 22, 845-856, 2001.

[6] C.A.R. Hoare. *Quicksort.* Computer Journal, 5:1, 10-15, 1962.

[7] W. Press, S. Teukolsky, W. Vetterling et B. Flannery. *Numerical Recipes in C. The Art of Scientific Computing*. Second Edition. Cambridge University Press, NY,1992.

[8] J. W. J. Williams. *Algorithm 232 : Heapsort.* Communication of the ACM, 7, 347-348, 1964.

[9] C. Froidevaux, M.C. Gaudel et M. Soria. *Types de données et Algorithmes*. McGraw-Hill, Paris, 1990.

[10] J.L. Bentley. *Writing Efficient Programs*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[11] A. Aho, R. Sethi et J. Ullman. *Compilateurs, Principes, techniques et outils*. InterÉditions, Paris, 1990.